VEHICLE ROUTING MODELS & APPLICATIONS

TA3: EV CHARGING LOGISTICS

Thursday 8:30 – 10:30 AM

Session Chair: Halit Uster

8:30 **Locating Refueling Points on Lines and Comb-Trees**
*Pitu Mirchandani, Yazhu Song**
Arizona State University

9:00 **Modeling Electric Vehicle Charging Demand**
*[1]Guus Berkelmans, [1]Wouter Berkelmans, [2]Nanda Piersma, [1]Rob van der Mei, [1]Elenna Dugundji**
[1]CWI, [2]HvA

9:30 **Electric Vehicle Routing with Uncertain Charging Station Availability & Dynamic Decision Making**
[1]Nicholas Kullman, [2]Justin Goodson, [1]Jorge Mendoza*
[1]Polytech Tours, [2]Saint Louis University

10:00 **Network Design for In-Motion Wireless Charging of Electric Vehicles in Urban Traffic Networks**
Mamdouh Mubarak, Halit Uster, Khaled Abdelghany, Mohammad Khodayar*
Southern Methodist University

Locating Refueling Points on Lines and Comb-trees

Pitu Mirchandani
School of Computing, Informatics and Decision Systems Engineering,
Arizona State University, Tempe, Arizona 85281 United States
Email: pitu@asue.edu

Yazhu Song
School of Computing, Informatics and Decision Systems Engineering,
Arizona State University, Tempe, Arizona 85281 United States
Email: ysong70@asu.edu

Due to environmental and geopolitical reasons, many countries are embracing *electric vehicles* as an alternative to gasoline powered automobiles. There are other alternative fuels such as Compressed Gas and Hydrogen Fuel Cells that have also been tested for replacing gasoline powered vehicles. However, since the associated refueling infrastructure of alternative fuel vehicles is sparse and is gradually being built, the distance between refueling points becomes a crucial attribute in attracting drivers to use such vehicles. Optimally locating refueling points (RPs) will both increase demand and help in developing a refueling infrastructure.

This paper introduces a new set of location problems related to locating refueling points on lines and tree networks. It first deals with the simplest case of locating refueling points on a line, where origins can be anywhere on the line and destinations can be anywhere on the line. First, problems of feasibility are studied. Given there are feasible locations, then the location problem becomes "where should RPs be located to minimize a given fuel-related objective". For example the objective of minimizing the maximum distance between RPs minimizes the anxiety for the drivers. Scenarios include single one-way Origin-Destination (O-D) pair, multiple one way O-D pairs, round trips, etc. Extensions to tree networks are discussed.

1. Background

Suppose we wish to locate n RPs in a place where there are none currently. The problem of optimally locating such refueling stations has been investigated by Kuby and collaborators [e.g. Kuby and Lim 2005, Kuby and Lim 2007, Upchurch, Kuby and Lim 2009, Lim and Kuby 2010, Capar, Kuby and Rao, 2012]. Typically, they use modifications of flow capturing or flow interception models [Hodgson 1990, Berman Larson and Fouska 1992, Rebello, Agnetis and Mirchandani 1995], to cover as many O-D routes as possible with a given number of stations. However, these models do not take into consideration the likely possibility of vehicles **making detours** to refuel.

Cabral et al. (2007) considered a network design problem with relays (NDPR) in the context of telecommunication network design and proposed a column generation scheme and four algorithms. Konak (2012) also studied NDPR and proposed a set covering formulation with a meta-heuristic algorithm. However, these models only choose vertices to locate relays, which normally will not be optimal in many cases as will be shown below.

Routing Issues

Taking a trip, especially one through sparsely populated areas, requires the driver to plan when the vehicle will need to be refueled. Given the abundance of gasoline stations for standard vehicles, a driver usually considers refueling only when the fuel tank is low. In the case of range-limited vehicles (RLV), planning when to refuel is important, since there are few places to refuel because, at least initially, RPs would be few and far in between. Therefore, one needs to develop models which look for the routes that include detouring RPs if necessary. Objectives for such models could be to (a) minimize the total detouring distances and (b) minimize the total number of refueling stops. It is surprising that detouring is not a consideration in these models. In fact, detouring plays a major role in the problems been analyzed in this research. Finding routes in a network considering refueling detours have been studied by, among others, Ichimori, 1981; Smith et al, 2012; Laporte and Pascoal, 2011; and Adler and Mirchandani, 2014.

2. Some Location Problems on a Line

A prototypical location problem that needs consider refueling detours is to locate RPs to minimize the total detour distance for given discrete O-D demands. The following models are trying to locate minimum number of RP stations and do not consider any stop limitation.

2.1 One-way problems

We begin by considering the simplest special case where we have RLVs for one-way trips between any two points along a road. Suppose that our RLV starts its trip with a full battery. Let r denote the maximum distance that our fully charged RLV can travel before refueling, and let L denote the length of the road between two endpoints, where $L > r$.

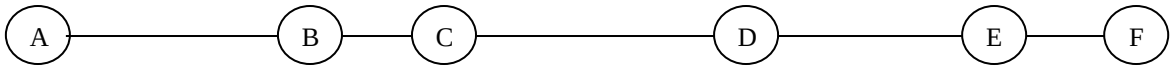


Figure 1 Simple illustration for line locations

Let's illustrate how to find feasible RPs' locations along a road with two points only. If $2r < L < 3r$, we need to locate two RPs, where the first one falls in the interval $[L-2r, r]$, and the second one falls in the interval $[L-r, 2r]$. However, this does not mean that choosing two points arbitrarily from the above two intervals will constitute a feasible solution since distance between two RPs has to be less than or equal to r.

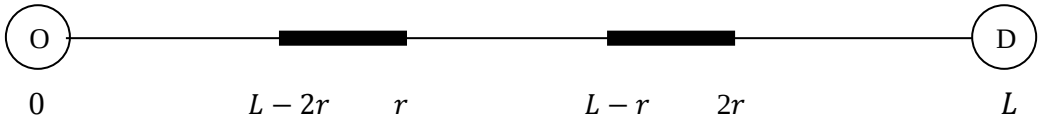


Figure 2 Refueling points localization intervals

If we need to locate n RPs, the location intervals should be:

$x_1 \in [L-nr, r]$

$x_2 \in [L-(n-1)r, 2r]$

$\vdots$

$x_n \in [L-r, nr]$.

Proposition 1: If one-way trips with the longest travel distance can be satisfied, then all O-D demands can be satisfied.

Proof: Provided in full paper.

Proposition 2: The minimum number of RP stations we need to locate is

$$p = \begin{cases} \left\lfloor \frac{L}{r} \right\rfloor & if\ mod(L,r) \neq 0 \\ \left\lfloor \frac{L}{r} \right\rfloor - 1 & if\ mod(L,r) \equiv 0 \end{cases}, \text{ equivalently, } p = \left\lceil \frac{L}{r} \right\rceil - 1.$$

Proof: Provided in full paper.

Proposition 3: Given any feasible solution, total detour distance is 0, that is, total weighted demand is constant. Therefore each feasible solution is optimal.

Proof: provided in full paper.

2.2 Round trip problems

We now consider the case where vehicles go for round-trips between any two points along a road. Let triple (v_i, v_j, v_i) represent a round-trip demand, that is, a vehicle starts at v_i, goes to v_j and goes back to v_i. Proofs for propositions below will be given in the full paper

Proposition 4: If the round-trip triples (v_1, v_n, v_1) and (v_n, v_1, v_n) can be refueled, then each round-trip triple (v_i, v_j, v_i) for $1 \leq i, j \leq n$ can be refueled with possible detours.

Proposition 5: The minimum number of refueling stations needed to serve all round-trip triples is $p = \left\lceil \frac{L}{r} \right\rceil$.

Locating stations to minimize total travel distanace can be formulated as a mixed integer quadratic mathematical program (not shown here), where the location varaibles are continuous on the line, the assignment of refuenig points to trips are 0-1 binary varaibles, and the objective is quadtratic because it has terms where continuous varaibles are multipled by 0-1 variables. The problem can be solved by MATLAB by using OPTI toolbox.

3. Some Location Problems on a Comb-Tree – the one-way cases

We consider a prototypical location problem on a small tree shown below. By a small tree we mean that one single RP is sufficient to serve all O-D demand pairs (see Fig 3). Consider the tree below: there are six ordered O-D pairs in total, and we consider one-way problem for these O-D pairs.

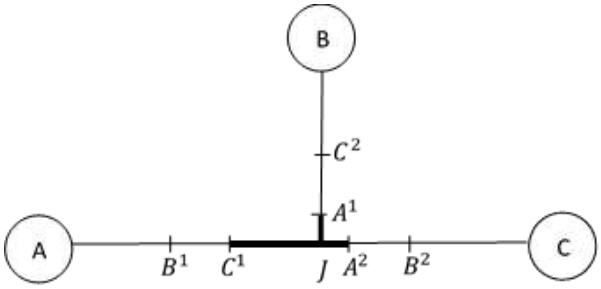


Figure 3 An example of a small tree problem

The breakpoints A^1, A^2, B^1, B^2, C^1 and C^2 are such that the distances AA^1, AA^2, BB^1, BB^2, CC^1 and CC^2 are all equal to the range limit r. The bold segments, which constitute a subtree, represent the intersection of all paths between these breakpoints. Note that the RP must be located on this subtree, and all O-D demand pairs can be served with possible detouring. Otherwise, if we locate the RP beyond this subtree, then only one RP cannot serve all O-D pairs. Specifically, if our objective is to minimize the total detouring, then the junction node J is the only optimal location to achieve zero detouring.

We consider a more general comb graph, where each leaf node serves as both origin and destination (see Fig 4).

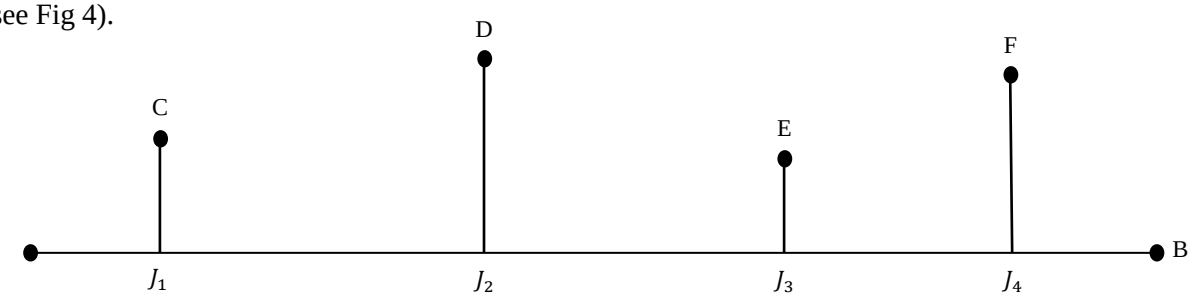


Figure 4 A comb graph

3.1 Minimum number of refueling stations needed

To find the minimum number of refueling stations needed, we first trim the graph: starting from each leaf node (e.g., A, B, C, …), we locate a refueling station every r distance until we are within r distance to the corresponding junction node. After trimming, we get a new graph, where the length of each branch is strictly less than r, and each leaf node may or may not have a refueling station (see Fig 5).

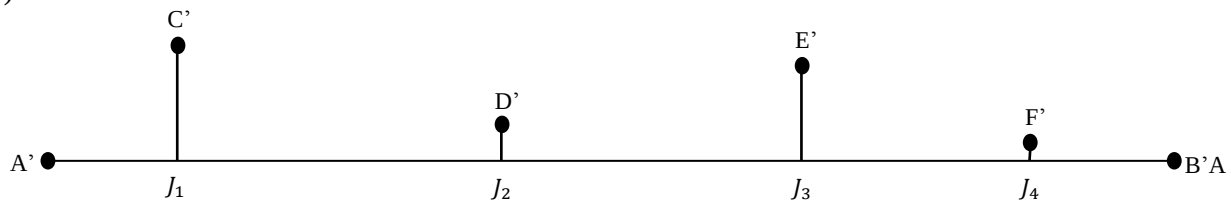


Figure 5 Localization comb subgraph after trimming

Then for this trimmed tree, we can proceed from leaf to right finding breakpoints (BPs) and can develop a *greedy algorithm* that locates the minimum number of refueling points. (The algorithm and proof will be avaiable in the full paper.)

3.2 Optimal locations of the minimum number of refueling stations

Although this approach provides the minimum number of refueling points for feasibility, locating these points optimally over the comb subgraph is not easy. Currently the research team is developing an algorithm to find these optimal locations.

4. Conclusions and Future Research

This extended abstract introduces some new problems of locating refueling points on a line and comb-tree networks where O's and D's could be anywhere on the network. In special cases, after locating optimally, no detouring is necessary. However, in general, detouring is necessary when one needs to consider round trips between the O's and the D's, and when there is an underlying tree structure. In

the full paper to be given at the conference, complexity results for problems on optimally locating on a tree will provided and exact and heuristic algorithms for these problems will be evaluated.

References

[1] Kuby, M., & Lim, S. (2005). The flow-refueling location problem for alternative-fuel vehicles. Socio-Economic Planning Sciences, 39(2), 125-145.

[2] Kuby, M., & Lim, S. (2007). Location of alternative-fuel stations using the flow-refueling location model and dispersion of candidate sites on arcs. Networks and Spatial Economics, 7(2), 129-152.

[3] Upchurch, C., Kuby, M., & Lim, S. (2009). A Model for Location of Capacitated Alternative-Fuel Stations. Geographical Analysis, 41(1), 85-106.

[4] Lim, S., & Kuby, M. (2010). Heuristic algorithms for siting alternative-fuel stations using the flow-refueling location model. European Journal of Operational Research, 204(1), 51-61.

[5] Kim, J. G., & Kuby, M. (2012). The deviation-flow refueling location model for optimizing a network of refueling stations. international journal of hydrogen energy, 37(6), 5406-5420.

[6] Capar, I., & Kuby, M. (2012). An efficient formulation of the flow refueling location model for alternative-fuel stations. IIE Transactions, 44(8), 622-636.

[7] Capar, I., Kuby, M., Leon, V. J., & Tsai, Y. J. (2013). An arc cover–path-cover formulation and strategic analysis of alternative-fuel station locations. European Journal of Operational Research, 227(1), 142-151.

[8] Hodgson, M. J. (1990). A Flow-Capturing Location-Allocation Model. Geographical Analysis, 22(3), 270-279.

[9] Berman, O., Larson, R. C., & Fouska, N. (1992). Optimal location of discretionary service facilities. Transportation Science, 26(3), 201-211.

[10] Mirchandani, P. B., Rebello, R., & Agnetis, A. (1995). THE INSPECTION STATION LOCATION PROBLEM IN HAZARDOUS MATERIAL TRANSPORTATION-SOME HEURISTICS AND BOUNDS. Infor, 33(2), 100-113.

[11] Cabral, E. A., Erkut, E., Laporte, G., & Patterson, R. A. (2007). The network design problem with relays. European Journal of Operational Research,180(2), 834-844.

[12] Konak, A. (2012). Network design problem with relays: A genetic algorithm with a path-based crossover and a set covering formulation. European Journal of Operational Research, 218(3), 829-837.

[13] Adler, J. D., & Mirchandani, P. B. (2014). Online routing and battery reservations for electric vehicles with swappable batteries. Transportation Research Part B: Methodological, 70, 285-302.

[14] Handler, G. Y., & Mirchandani, P. B. (1979). Location on networks: theory and algorithms (Vol. 979). Cambridge, MA: MIT press.

Modeling Electric Vehicle Charging Demand using Discrete Choice Models

Guus Berkelmans[1], Wouter Berkelmans[1], Nanda Piersma[2], Rob van der Mei[1], Elenna Dugundji[1]

[1] Centrum Wiskunde & Informatica (CWI)
[2] Hogeschool van Amsterdam (HvA)

Abstract. In the past 5 years Electric Car use has grown rapidly, almost doubling each year. To provide adequate charging infrastructure it is necessary to model the demand. In this paper we model the distribution of charging demand in the city of Amsterdam using a Cross-Nested Logit Model and sociodemographic statistics of neighborhoods.

Keywords: Electric Vehicle, Charging Demand, Discrete Choice Model, Cross-Nested Logit

1 Introduction

We tackle the question of estimating eletric vehicle (EV) charging demand on public charging stations using the multinomial, nested, and cross-nested logit models. These models use certain sociodemographic statistics of neighborhoods to estimate their share of the charging demand of the whole city. The multinomial logit model is a restricted version of the nested logit model, which in turn is a restricted version of the cross-nested logit model. The cross-nested logit model far outperforms the others.

2 Literature review

Many articles have been written about optimizing charging infrastructure based on demand, though this demand is often an unknown. Different researchers have dealt with estimating and measuring this in different ways. Dong, Liu and Lin[1] base their model on the multiday driving data collected from 445 instrumented gasoline vehicles in the Seattle metropolitan area. Tu, Li, Fang, Shaw, Zhou and Chang[2] use taxi GPS data to estimate demand. Liu[3] assesses the power grid impact if 10% of vehicles were EVs. Jung, Chow, Jayakrishnan and Park[4] model taxi service demand as a Poisson process based on an EMME/2 transportation planning model developed at the Korea Transportation Institute (KOTI). Wang, Wang and Lin[5] optimize charging strategies and charging station placements based on a randomly generated EV network. He, Kuo and Wu[6] optimize charging station location in Beijing, using three classic facility location models. EV demand is estimated using 6 socio-demographic attributes deemed important by the literature, which were then ranked by 11 interviewees. Van den Hoed, Helmus, de Vries and Bardok[7] analyse the data of charging behavior in Amsterdam, using the same data we use. We use a statistical model to estimate demand in Amsterdam based on a number of socio-demographic attributes.

3 Methods and Data

Our data consists of statistics about the neighborhoods of Amsterdam from the Central Bureau of Statistics (CBS) and records of every instance of an EV charging at a public station in Amsterdam from the Charge Infrastructure Efficiency Model (CHIEF) dataset. We used the program BIerlaires Optimization package for GEV Models Estimation (BIOGEME), a specialized log-likelihood maximizer,to estimate multinomial, nested, and cross nested logit models to predict the probability of an EV driver choosing to charge in a particular neighborhood.

3.1 Model description

The logit models are based on the principle of utility maximization, where the choicemaker simply chooses the alternative with the highest utility, the utility U_i of an alternative i is given by $V_i + \epsilon_i = \beta \cdot x_i + \epsilon_i$, where β is a vector of parameters to be estimated, x_i is a vector of properties of alternative i and ϵ_i is a random variable with a standard Gumbel distribution. In the case of the multinomial logit model (MNL), all ϵ are iid. So U_i are independently distributed Gumbel$(V_i, 1)$, which makes $max_{j \neq i} U_j$ distributed Gumbel$(ln(\sum_{j \neq i} e^{V_j}), 1)$. So the probability that $U_i > max_{j \neq i} U_j$ is:

$$P_i = \frac{e^{V_i}}{\sum e^{V_j}} \tag{1}$$

In the nested logit model (NL), there is a partition on the alternatives $\{N_1, ..., N_k\}$ the ϵ_i are independent if they are in different nests, but within the nest the shared cumulative distribution is given by $\exp\left(-\left(\sum_{j\in N_m} e^{-\epsilon_j \mu_m}\right)^{1/\mu_m}\right)$. This leads to probabilities

$$\tilde{V}_m = \frac{1}{\mu_m} \ln\left(\sum_{j\in N_m} e^{V_j \mu_m}\right) \tag{2}$$

$$P(N_m) = \frac{e^{\tilde{V}_m}}{\sum e^{\tilde{V}_n}} \tag{3}$$

$$P(i|N_m) = \frac{1_{i\in N_m} e^{V_i \mu_m}}{\sum_{j\in N_m} e^{V_j \mu_m}} \tag{4}$$

The cross nested logit (CNL) allows alternatives to lie in multiple nests at the same time with α_{im} denoting the degree of alternative i lying in nest m, with $\sum_m \alpha_{im} = 1$ for all i. This leads to probabilities

$$\tilde{V}_m = \frac{1}{\mu_m} \ln\left(\sum_j \alpha_{jm} e^{V_j \mu_m}\right) \tag{5}$$

$$P(N_m) = \frac{e^{\tilde{V}_m}}{\sum e^{\tilde{V}_n}} \tag{6}$$

$$P(i|N_m) = \frac{\alpha_{im} e^{V_i \mu_m}}{\sum_j \alpha_{jm} e^{V_j \mu_m}} \tag{7}$$

3.2 Data specification

The properties of the neighborhoods used were the number of charging stations, the total number of inhabitants, the percentages of inhabitants aged between 0 and 14, between 15 and 24, between 25 and 44, between 45 and 64, and 65 and older, the average income per inhabitant, the number of cars (electric or otherwise) per inhabitant, the percentage of homes built after the year 2000, and the number of homes per inhabitant. The charging sessions used were those between 2014 and 2015, properties of neighborhoods differed by year and were lagged variables (i.e. we used properties from 2013 to predict demand for 2014), except for the number of charging stations, which differed by month and were unlagged. In the nested logit, every district was a nest, in the cross nested logit we took the districts as nests, but allowed neighborhoods at the borders between districts to lie partially in the bordering district.

4 Results

The cross nested logit model proved superior to the multinomial and nest logit models. We obtained the following goodness of fit (McFadden's adjusted ρ^2[8]) and the following estimates for our β parameters and nest coefficents:

Table 1. Adjusted ρ^2 of the models for MNL, NL and CNL

Modeltype	Regular Users	Electric Carshare	Taxi
MNL	0.058	0.061	0.269
NL	0.063	0.073	0.296
CNL	0.07	0.08	0.30

Table 2. Estimates β Parameters for the CNL models, t-tests in brackets

Parameters	Regular Users	Electric Carshare	Taxi
Number of Charging Stations	N/A	0.683 (86.88)	0.477 (18.04)
Total number of inhabitants	0.586 (162.95)	0.193 (33.83)	0.436 (12.93)
Aged 0-14 (in %)	0.587 (66.45)	0.386 (20.17)	0.681 (7.90)
Aged 15-24 (in %)	0.520 (68.66)	0.00898 (0.63)	0.481 (7.46)
Aged 25-44 (in %)	0.647 (40.98)	0.237 (6.20)	0.776 (6.50)
Aged 45-64 (in %)	0.332 (27.41)	-0.312 (-12.95)	-0.679 (-7.84)
Aged 65+ (in %)	0.145 (23.28)	0.0329 (2.99)	0.414 (9.82)
Average income	1.05 (152.30)	-0.311 (-22.17)	-0.776 (12.93)
Average number of cars per inhabitant	0.809 (141.24)	0.388 (46.62)	-0.697 (-14.61)
Average number of homes per inhabitant	0.857 (67.10)	1.88 (56.68)	-2.91 (-9.42)
Percentage of homes built after 2000	-0.0399 (-30.92)	-0.0195 (-8.10)	0.145 (14.56)

Table 3. Estimates μ Nest Coefficients for the CNL models, t-tests in brackets

Nests	Regular Users	Electric Carshare	Taxi
Amsterdam City Centre	1 (fixed)	1.05 (152.82)	3.73 (8.45)
Amsterdam North	1.79 (149.52)	2.09 (82.32)	1.78 (35.28)
Amsterdam West	1.33 (314.61)	1.14 (241.08)	2.11 (24.42)
Amsterdam New-West	1.67 (167.98)	3.18 (68.14)	1 (fixed)
Amsterdam Westpoort	1 (fixed)	1.43 (23.04)	1 (fixed)
Amsterdam East	1.39 (213.40)	1 (fixed)	2.94 (30.02)
Amsterdam SouthEast	3.44 (99.43)	11.4 (26.62)	11.6 (5.83)
Amsterdam South	1.16 (463.53)	1.25 (201.53)	3.06 (25.64)

Table 4. Estimates α Nest coefficents for neighborhoods in more than 1 nest for the CNL models (reg=Regular users)

Neighborhood (Nest 1/Nest 2)	Reg 1	Reg 2	Carshare 1	Carshare 2	Taxi 1	Taxi 2
Haarlemmerbuurt (City Centre/West)	0.314	0.686	0	1	0.354	0.646
Jordaan (City Centre/West)	1	0	0.117	0.883	0	1
Weteringschans (City Centre/South)	0.868	0.132	0	1	0	1
Weesperbuurt en Plantage (City Centre/East)	0.9785	0.0215	1	0	0.272	0.728
Oostelijke Eilanden en Kadijken (City Centre/East)	0.582	0.418	0	1	0.130	0.870
Frederik Hendrikbuurt (West/City Centre)	1	0	0.970	0.03	1	0
Da Costabuurt (West/City Centre)	1	0	0.989	0.011	1	0
Overtoomse Sluis (West/South)	0.0117	0.9883	0.369	0.631	0	1
Vondelbuurt (West/South)	0.304	0.696	1	0	1	0
De Kolenkit (West/New-West)	0.978	0.022	0.911	0.089	0.174	0.826
Van Galenbuurt (West/New-West)	0.289	0.711	0.842	0.158	1	0
Hoofdweg en Omgeving (West/New-West)	0.313	0.687	1	0	1	0
Westindische buurt (West/New-West)	0.326	0.674	0.473	0.527	1	0
Slotermeer-Noordoost (New-West/West)	0.466	0.534	0.252	0.748	0.883	0.117
Overtoomse Veld (New-West/West)	0.723	0.277	0	1	0.804	0.196
Westlandgracht (New-West/South)	0.804	0.196	0	1	0	1
Oude Pijp (South/City Centre)	0.006	0.994	1	0	0.893	0.107
Diamantbuurt (South/East)	1	0	0	1	0.361	0.639
Hoofddorppleinbuurt (South/New-West)	1	0	0.019	0.981	1	0
Willemspark (South/West)	0	1	0.818	0.182	0.834	0.166
IJselbuurt (South/East)	0.253	0.747	0.649	0.351	0.366	0.634
Rijnbuurt (South/East)	0.361	0.639	1	0	0.388	0.612
Buitenveldert Oost (South/East)	1	0	0.997	0.003	1	0
Weesperzijde (East/South)	0.673	0.327	0.308	0.692	0.344	0.656
Oosterparkbuurt (East/City Centre)	0.631	0.369	0.829	0.171	0.697	0.303
Dapperbuurt (East/City Centre)	0.855	0.145	0	1	0.858	0.142
Oostelijk Havengebied (East/City Centre)	0.377	0.623	0	1	0.893	0.107
De Omval (East/South)	1	0	0.932	0.068	1	0

Table 5. Neighborhoods limited to being in a specific nest

Nests	Neighborhoods
Amsterdam City Centre	Burgwallen-Oude Zijde, Burgwallen-Nieuwe Zijde, Grachtengordel-West, Grachtengordel-Zuid, Nieuwmarkt en Lastage
Amsterdam North	Volewijck, IJplein en Vogelbuurt, Tuindorp Nieuwendam, Tuindorp Buiksloot, Nieuwendammerdijk en Buiksloterdijk, Tuindorp Oostzaan, Oostzanerwerf, Kadoelen, Nieuwendam-Noord, Buikslotermeer, Banne Buiksloot, Buiksloterham, Nieuwendammerham, Waterland
Amsterdam West	Houthavens, Spaarndammer- en Zeeheldenbuurt, Staatsliedenbuurt, Centrale Markt, Kinkerbuurt, Van Lennepbuurt, Helmersbuurt, Sloterdijk, Landlust, Erasmuspark, De Krommert
Amsterdam New-West	Slotermeer-Zuidwest, Geuzenveld, Eendracht, Lutkemeer en Ookmeer, Osdorp-Oost, Osdorp-Midden, De Punt, Middelveldsche Akerpolder en Sloten, Sloter-vaart, Sloten- en Riekerpolder
Amsterdam Westpoort	Westelijk Havengebied, Bedrijventerrein Sloterdijk
Amsterdam East	Indische Buurt West, Indische Buurt Oost, Zeeburgereiland en Nieuwe Diep, IJburg West, IJburg Zuid, Transvaalbuurt, Frankendael, Middenmeer, Betondorp,
Amsterdam SouthEast	Amstel III en Bullewijk, Bijlmer-Centrum D, F en H, Bijlmer-Oost E,G en K, Nellestein, Holendrecht en Reigersbos, Gein, Driemond
Amsterdam South	Nieuwe Pijp, Schinkelbuurt, Museumkwartier, Stadionbuurt, Apollobuurt, Duivelseiland, Scheldebuurt, Station-Zuid WTC en omgeving, Buitenveldert-West

5 Discussion of Results

From other similar discrete choice model estimations it has been observed that a McFadden's ρ^2 of around 0.1 indicates a good fit while a ρ^2 of between 0.2 and 0.4 indicates an excellent fit[8]. This would indicate that our models for Regular users and participants of an Electric Carshare scheme are reasonably good while our Taxi model is excellent. Indeed when we compared the probabilities given by our models to the actual frequencies of our data we observed a very good match. Our results also showed that some neighborhoods were far more correlated with other districts than the one they were located in and that this vastly differed for the different kinds of usertype (the Diamantbuurt being in the South nest entirely when considering Regular users but in the East district entirely when considering Carshare users for example). A possible explanation in the case of the Diamantbuurt is that the South district is a more expensive district than the East district, so the regular users (who tend to be more affluent) tend to be in the South district more than Electric Carshare participants (who tend to be less affluent).

6 Conclusion

We have generated models to estimate the distribution of charging demand for three different usertypes for a given year by using data of the previous year. Using these models we should be able to make reasonable predictions of the distribution of charging demand in the coming year, even in the case that the makeup of a neighborhood changes significantly (for example when the Jordaan went from a low/average income neighborhood to a high income one in a reasonably short time period) or in the case that new neighborhoods arise (for example through development projects). This way the city will be able to place charging infastructure to accomodate this demand.

7 Acknowledgements

We'd like to thank Merel Steenbrink and Tirza Jochemsen for being involved in the early stages of this research.

References

1. Dong, J.,Liu, C.,Lin, Z. Charging infrastructure planning for promoting battery electric vehicles: An activity-based approach using multiday travel data. Transportation Research Part C: Emerging Technologies Volume 38, January 2014, Pages 44-55
2. Tu, W., Li, Q., Fang, Z., Shaw, S. , Zhou, B., Chang, X., Optimizing the locations of electric taxi charging stations: A spatialtemporal demand coverage approach. Transportation Research Part C: Emerging Technologies Volume 65, April 2016, Pages 172-189

3. Liu, J. Electric vehicle charging infrastructure assignment and power grid impacts assessment in Beijing. Energy Policy Volume 51, December 2012, Pages 544-557 Renewable Energy in China
4. Jung, J., Chow, J.Y.J., Jayakrishnan, R., Park, J.Y. Stochastic dynamic itinerary interception refueling location problem with queue delay for electric taxi charging stations. Transportation Research Part C: Emerging Technologies Volume 40, March 2014, Pages 123-142
5. Wang, I.L., Wang, Y., Lin, P.C. Optimal recharging strategies for electric vehicle fleets with duration constraints. Transportation Research Part C: Emerging Technologies Volume 69, August 2016, Pages 242-254
6. He, S.Y., Kuo, Y.H., Wu, D. Incorporating institutional and spatial factors in the selection of the optimal locations of public electric vehicle charging facilities: A case study of Beijing, China. Transportation Research Part C: Emerging Technologies Volume 67, June 2016, Pages 131-148
7. van den Hoed, R., Helmus, J., de Vries, R., Bardok, D. Data analysis on the public charge infrastructure in the city of Amsterdam, 27th International Electric Vehicle Symposium & Exhibition, Barcelona, 2013.
8. D.A. Hensher, P.R. Stopher Behavioural Travel Modelling, Croom Helm, London, 1979

Electric Vehicle Routing with Uncertain Charging Station Availability & Dynamic Decision Making

Nicholas D. Kullman Justin C. Goodson Jorge E. Mendoza

1 Introduction

Motivated by environmental concerns and regulations, electric vehicles (EVs) are becoming more popular in supply chain distribution functions (e.g., La Poste [1]). However, EVs pose operational challenges to which their conventional petroleum-based counterparts are immune. For instance, EVs' driving ranges are often only 25 percent that of conventional petroleum-based vehicles' (CVs), charging infrastructure is still relatively sparse compared to the network of refueling stations for CVs, and the time required to charge an EV can range from 30 minutes to 12 hours depending on charging technology - orders of magnitude longer than the time needed to refuel a CV [3].

There are two general approaches to overcoming these operational challenges. The first is a simple approach in which routes are restricted to the vehicle's autonomy. That is, the EV is routed back to the depot when its battery nears depletion so it may charge overnight in preparation for the subsequent day's deliveries. In the second approach, the EV is allowed to perform mid-route recharging by taking advantage of charging infrastructure in the field.

Montoya showed that the second approach offers cost savings, because mid-route recharging allows for a decrease in the total distance traveled and an increase in the capacity of a single EV, thereby reducing the number of vehicles and drivers needed [5]. However, this study, like the others that consider mid-route recharging (e.g.,[8][2]), makes the assumption that the charging stations (CSs) are always available to the EV when it arrives to charge. In reality, this is often not the case. Because charging station infrastructure is limited and EVs require significant time to charge, charging stations will often be unavailable when an EV arrives and the EV may be forced to queue. This discrepancy between modeling assumptions and reality has thus far prohibited logistics companies from implementing mid-route recharging, despite the suggested cost savings [5].

Our research reduces this discrepancy by more realistically modeling both the uncertainty in availability and the queuing process at public charging infrastructure. We model the EV Routing Problem with Mid-route Recharging and Uncertain Availability (EVRP-MRUA) as a Markov decision process, and we provide a stochastic dynamic programming solution with three different policies. This work aims to enable logistics companies to take advantage of the increases in capacity offered by mid-route recharging, thus extending the utility of EVs as delivery vehicles.

1.1 Related Literature

The literature reports on only two attempts to address charging station availability in EV routing problems. In a recent study, Sassi et al. address an EV routing problem in a semi-public infrastructure context [7]. In their application the infrastructure is owned by several companies. Each company is assigned time slots during which it is allowed to use a given station. The decision

maker must then take into account these time windows when designing the routes. Sweda et al. study a shortest path problem in which a vehicle must travel from an origin to a destination on a network with charging stations positioned at every node [9]. Each station has a probability of being available and expected waiting time until becoming available (known a priori to the planner). In this context, the decision maker must not only select which path to take to arrive at the destination as quickly as possible, but also decide where to recharge and what to do in case a desired charging station is unavailable (i.e., wait or seek an alternative station). The authors propose an approach to determine an adaptive routing and recharging policy that minimizes the sum of all traveling, waiting, and recharging costs. Our work builds on these studies by dynamically routing an EV over a network of customers and charging stations, assuming public infrastructure with unknown availability.

2 Problem Statement

The EVRP-MRUA consists of a set of known customers $\mathcal{N}$ and charging stations $\mathcal{C}$ and a single electric vehicle. At time 0, the EV begins at the depot, which we refer to as node $0 \in \mathcal{C}$. It then traverses the complete graph on $\mathcal{N} \cup \mathcal{C}$.

We assume there exists a Hamiltonian path among the set of CSs such that the edge connecting two charging stations can be traversed with a fully charged vehicle. Further, we assume that the edge from each customer node $i \in \mathcal{N}$ to the nearest CS can be traversed by a half-charged vehicle. These two assumptions guarantee each customer in $\mathcal{N}$ can be serviced by the vehicle. Without loss of generality, we assume travel times are whole numbers and time periods can be indexed via the nonnegative integers.

If the EV elects to visit a CS $c \in \mathcal{C}$, the vehicle may charge if there are available charging terminals ("chargers"), or it may elect to join the queue if all chargers are in use. Let the number of chargers at a CS c be ψ_c. We assume that the ψ_c chargers at the CS are identical, although the charging technology may differ between charging stations. We further assume that the depot is always available for charging and that other charging station queue lengths are unknown prior to arrival.

We model waiting line dynamics at a CS c as a pooled first-come-first-served queue with a system capacity of $\ell_c \geq \psi_c$, where ℓ_c is chosen such that the system capacity is practically infinite. We consider a discrete-time Markov model on the state-space $\{0, 1, \ldots, \ell_c\}$ and assume that the the random inter-arrival time of vehicles to the station and the random service time of a single charger are geometric random variables with known parameters p_x and p_y, respectively. After the vehicle joins the queue, it may continue to wait, or it may leave. When a station is available, the vehicle may restore its charge to full capacity Q or to an intermediate capacity.

The problem terminates when the EV has visited all customers and returns to the depot. The goal of the EVRP-MRUA is to find a routing policy that minimizes the total expected time of the EV to visit each customer in $\mathcal{N}$, including travel time, charging time, and queuing time.

3 Problem Formulation

We model the EVRP-MRUA as a Markov decision process and solve it using an approximate stochastic dynamic program (SDP).

State We denote the state of the system at decision epoch k by s_k. s_k is the vector containing all information necessary to make a routing decision at epoch k and consists of the EV's current charge level in kWh $q_k \in [0, Q]$, the time $t_k \in \mathbb{N}$, the EV's current location $i_k \in \mathcal{N} \cup \mathcal{C}$, the set of customers that the EV has not yet visited $\bar{\mathcal{N}}_k \subseteq \mathcal{N}$, and the vehicle's position at the current node z_k. We assume that at customer nodes and at the depot, the EV is always in the first position $z_k = 1$, while at public charging stations, the position depends on demand at the CS, so $z_k \in \{0, \ldots, \ell_{i_k}\}$. Thus, $s_k = \big(q_k, t_k, i_k, \bar{\mathcal{N}}_k, z_k\big)$. The initial state is $s_0 = (Q, 0, 0, \mathcal{N}, 1)$, and the problem terminates at some decision epoch K with $s_K \in \{\, (q_K, t_K, 0, \emptyset, 1) \mid q_K \in [0, Q], t_K \in \mathbb{N} \,\}$.

Action space At each decision epoch $k \in \{0, 1, \ldots, K\}$, we begin in a pre-decision state s_k and select an action x from the action space $\mathcal{X}(s_k)$. Actions are location-charge pairs, $x = (i', q')$. The action space consists generally of queuing, moving, and charging decisions. We impose the following restrictions on the action space: the vehicle may only queue if it resides at a CS with no available chargers; the vehicle may only make moves to nodes $i' \in \bar{\mathcal{N}}_k \cup \mathcal{C}$ that are energy-feasible, with the additional requirement that if $i' \in \bar{\mathcal{N}}_k$ it must have sufficient charge to subsequently reach a CS from i'; finally, the vehicle may only charge if it resides at a CS with available chargers, it may not charge in two consecutive epochs, and it must at least charge to an energy level sufficient to reach the nearest CS.

Transition to post-decision state Following the selection of an action $x = (i', q')$, we transition to the post-decision state s_k^x. In this transition, we update the location $i_k^x = i'$ and charge $q_k^x = q'$. We also update the set of unvisited customers: $\bar{\mathcal{N}}_k^x = \bar{\mathcal{N}}_k \setminus \{i'\}$ if $i' \in \bar{\mathcal{N}}_k$, and $\bar{\mathcal{N}}_k^x = \bar{\mathcal{N}}_k$ otherwise.

Transition to pre-decision state From the post-decision state s_k^x, we transition to the subsequent pre-decision state s_{k+1}. This transition involves updating the remaining components of s_k: the EV's position at the current location z_{k+1} and the time at which the next epoch occurs t_{k+1}.

Position: If the EV now resides at the depot or a customer, then $z_{k+1} = 1$; however, if the EV resides at a CS $c \neq 0$, then z_{k+1} is probabilistic. Let $G_{c,k+1}$ be a random variable that represents the queue length of CS c at time $k+1$, and let $g_{c,k+1}$ be a realization of $G_{c,k+1}$. Then $z_{k+1} = g_{c,k+1} + 1$.

Time: If the action x selected in decision epoch k was a charging or moving action, then the time of decision epoch $k+1$ is deterministic. Specifically, $t_{k+1} = t_k + \tau_{ii'}$ for moving actions and $t_{k+1} = t_k + \bar{u}(q, q')$ for charging actions, where $\tau_{ii'}$ is the time required to travel between locations i and i', and $\bar{u}(q, q')$ is the time required to charge from charge level q to q'. For queuing actions, the time of the next epoch is probabilistic and equal to either the time of the next departure from the queue or after a predetermined amount of time δ has elapsed, whichever comes first.

Costs When we are in state s_k and choose action $x = (i', q')$ we incur a cost of $C(s_k, x)$, measured in time. The cost for moving and charging actions is $\tau_{ii'}$ and $\bar{u}(q, q')$, respectively, the time required to perform these actions. If the vehicle elects to queue at a charging station, the cost is the expected waiting time until the next decision epoch. Let ψ_c be the number of chargers at the CS c, κ be a realization of the waiting time until a departure, and γ be a realization of the number of departures

that occur at time $t_{k+1} = t_k + \kappa$. Then the expected waiting time is

$$C(s_k, x) = \delta \Pr(\text{no departures}) + \sum_{\kappa=1}^{\delta} \sum_{\gamma=1}^{\psi_c} \kappa \Pr(\gamma \text{ departures at time } \kappa). \tag{1}$$

Objective A decision rule at epoch k is the function X_k^π that selects an action x from the action space $\mathcal{X}(s_k)$. A policy π is a sequence of decision rules. Letting Π denote the set of Markovian deterministic policies, we seek to find an optimal policy $\pi^* \in \Pi$ that minimizes the expected total cost (duration) of the tour, conditional on our initial state. The value of this policy is measured by the expected duration of the resulting tour T^*

$$T^* = \min_{\pi \in \Pi} \mathbb{E}\left[\sum_{k=0}^{K} C(s_k, X_k^\pi(s_k)) \middle| s_0 \right] = \mathbb{E}\left[\sum_{k=0}^{K} C(s_k, X_k^{\pi^*}(s_k)) \middle| s_0 \right]. \tag{2}$$

4 Solution Methods and Preliminary Results

To date, we have implemented three policies for the SDP: a simple myopic policy, a one-step rollout of the myopic policy, and a fixed-route policy. We ran each policy on a set of 22 instances that were generated by Montoya et al. [6] to emulate three different types of customer distributions in urban environments: customers randomly scattered throughout the service area ("Random"); customers that form clusters throughout the service area ("Cluster"); and a combination of these two in which most customers are in clusters, but some have been randomly scattered throughout ("Random+Cluster"). For each instance, we ran each policy under nine different assumptions of the average utilization of the charging stations (5% average utilization, 15%, 25%, ..., 85%). We performed 200 simulations for each policy-instance-utilization combination. For the one-step myopic rollout, we simulated the myopic base policy 25 times for each reachable state s_{k+1} from s_k (see Goodson for more on rollout policies [4]). Our results are summarized in Figure 1.

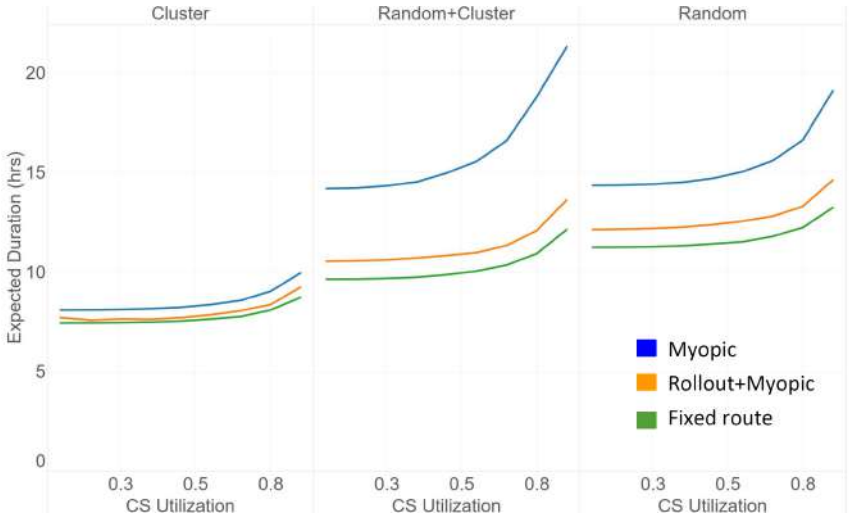


Figure 1: Expected total tour duration across policies, customer distributions, and CS utilizations.

We find that the rollout of the myopic policy performs better than the myopic policy alone, and the fixed-route policy performs better still than both of these, although the extent by which varies by customer distribution and CS utilization. The biggest differences in policy performance are for the Random+Cluster customer distribution at high CS utilizations where the policies differ by more than 75%, while the smallest differences are for the Cluster customer distributions at low CS utilizations where all policies perform within 9% of one another on average. The Cluster customer distribution lends itself to the myopic approach where the EV simply performs the cheapest immediate action, thus resulting in the similarity in performance across policies that we observe. For the other customer distributions, the myopic policy has greater potential to make costly decisions, but these are largely avoided by rolling out the myopic policy.

The improvement in performance that the rollout of the myopic policy offers over the myopic policy alone is encouraging. We are currently in the process of implementing a rollout for the fixed-route policy as well and hope to be able to report on similar improvements.

References

[1] La Poste completes electric vehicle fleet with the addition of three-wheelers. `http://www.postaltechnologyinternational.com/news.php?NewsID=62409`. Accessed: 2016-12-07.

[2] R. G. Conrad and M. A. Figliozzi. The recharging vehicle routing problem. In *Proceedings of the 2011 industrial engineering research conference*, 2011.

[3] D. U. Eberle and D. R. von Helmolt. Sustainable transportation based on electric vehicle concepts: a brief overview. *Energy Environ. Sci.*, 3:689–699, 2010.

[4] J. C. Goodson, B. W. Thomas, and J. W. Ohlmann. A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. *European Journal of Operational Research*, 258(1):216 – 229, 2017.

[5] A. Montoya. *Electric Vehicle Routing Problems: models and solution approaches.* PhD thesis, Université de Nantes Angers Le Mans, 2016. Chapter 5: The technician routing problem with conventional and electric vehicles.

[6] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas. The electric vehicle routing problem with nonlinear charging function. working paper or preprint, Nov. 2016.

[7] O. Sassi, W. R. Cherif-Khettaf, and A. Oulamara. *Iterated Tabu Search for the Mix Fleet Vehicle Routing Problem with Heterogenous Electric Vehicles*, pages 57–68. Springer International Publishing, Cham, 2015.

[8] M. Schneider, A. Stenger, and D. Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014.

[9] T. M. Sweda, I. S. Dolinskaya, and D. Klabjan. Adaptive routing and recharging policies for electric vehicles. *Transportation Science*, 2017. Forthcoming.

Network Design for In-Motion Wireless Charging of Electric Vehicles in Urban Traffic Networks*

Mamdouh Mubarak[1], Halit Üster†[1], Khaled Abdelghany[2], and Mohammad Khodayar[3]

[1]Department of Engineering Management, Information, and Systems
[2]Department of Civil and Environmental Engineering
[3]Department of Electrical Engineering
Lyle School of Engineering
Southern Methodist University
Dallas, TX 75275-0123

June 19, 2017

1 Introduction

Electric Vehicle (EV) adoption has been dramatically lagging behind anticipation. While the U.S. goal was to have 1 million EVs on the road by the end of 2015, it was not until September 2016 that the size of the EV fleet has reached half a million units [5, 14]. The hesitation of many drivers to switch to EV is based on two key reasons: The limited driving range of EVs and the long charging times of the batteries. To overcome these drawbacks, dynamic charging was proposed and tested Onar et al. [11], Jang et al. [8] as a promising solution. This pioneering technology allows EVs to charge wirelessly from roadbed transmitters while the vehicle is moving. However, the shortage of charging facilities in the urban traffic network continues to hamper the growth of the EV market [7]. While many potential consumers are shying away from buying into the EV technology due to this particular limitation, both public and private sectors seem reluctant to invest in charging infrastructure. The reason behind this hesitation is the insufficient number of EVs on the road. Solving this (chicken-and-egg) dilemma depends strongly on a strategic deployment of wireless charging stations (WCS) that optimizes both locations and capacities of these stations in urban areas [13]. In this paper, we address this problem via an analytical approach in which the new wireless charging technology is optimally deployed to minimize the investment cost. In doing so, to facilitate EV adoption, we capture the existing traffic pattern on the road network via prior

*This research was supported by the National Science Foundation under grant CMMI-1550448.
†Corresponding Author, E-mail: uster@smu.edu, Phone: (214) 768 3575.

user equilibrium (UE) traffic assignment solution and devise a solution so that the drivers do not need to deviate from their usual routes.

2 Related Literature and Contribution

Studies on wireless EV charging networks only recently started to appear in the literature. Ko and Jang [9] present a mathematical formulation to optimize the design of the dynamic-charging-based mass transportation system, addressing the trade-off between the number of deployed power transmitters and the size of EV battery. Riemann et al. [12] propose a flow-capturing location model to maximize the captured EV flow by locating a fixed number of WCSs without cost and capacity considerations. They also assumed that an EV is always fully charged when traveling over a charging lane. Fuller [6] considers investment cost minimization for a WCS infrastructure that allows EV travel between 39 key origin-destination pairs in California. Chen et al. [3] study the optimal deployment of charging lanes under a limited budget and a fixed charging rate of the charging lanes.

Our study contributes to the emerging literature of EV dynamic charging with a new mathematical formulation that optimally locate WCSs and decide on the optimal power allocation for each WCS. The model embraces the user equilibrium traffic pattern and imposes in-motion charging requirements so that traveling EVs are ensured to reach their destinations without being stranded on their routes. This study also offers a Benders Decomposition based algorithm to solve the large instances of the formulated problem. Both combinatorial and classical Benders cuts are utilized in the proposed approach.

3 Problem Definition

We approach the problem from the *perspective of a city as the decision maker* whose aim, for societal benefits, is to satisfy the charging demands of all EVs in its urban network at the minimum investment cost. We represent the underlying road network as a directed graph with nodes and arcs representing specific locations including intersections and road segments, respectively. Given a set of origin-destination (OD) node pairs with a certain traffic flow demand, as an input to our model, we first calculate a user equilibrium (UE) traffic assignment which provide travel volumes and travel times on links. As noted by Mahmassani [10], UE offers a suitable system representation for long term planning purposes as in our case.

Letting road segments (links) represent potential locations for WCS, our problem is formulated as a mixed integer program where the decision variables include locations and power capacities

of the WCSs as well as the wireless charging amounts on links. Inputs to the model, other than UE results, include electric parameters such as the charging power of the system, the charging efficiency, and the battery size of EVs. The objective of our model is to minimize the installation (investment) cost which is composed of a fixed cost for base power capacity and a stepwise capacity expansion cost, as well as the total charging cost subject to the main constraints representing

1. energy conservation (balance) constraints on links,
2. relationship between the amount of energy that EVs receive on links and the power capacities of WCSs and travel time on these links,
3. relationship between the required power capacities of WCSs and the travel demands their links,
4. limitations on the maximum power capacity that can be installed at WCSs,
5. limitations on the amount of charge an EV can receive based on its battery size, and
6. initial and ending states of charge of EVs in the network.

4 Solution Methodology

Based on our initial computational experiments we observed that the Branch & Cut algorithm as implemented in CPLEX was not efficient, especially due to excessive run times needed to generate good bounds. Therefore, in this study, we combine features from classical Benders decomposition (BD) [2] and combinatorial BD [4] to devise an efficient exact algorithm as a solution methodology for the problem in hand. Specifically, we decompose the formulated MIP into two problems: A master problem (MP) that includes the location decisions, and a subproblem (SP) including the capacity decisions and the charging amounts decisions. In an iterative fashion, the locations obtained by solving MP are passed to SP which is solved for the capacities of WCSs and the amounts of charge on links. At each iteration, the MP solution provides a lower bound on the overall problem. If a set of locations generated by MP is found infeasible when solving SP, then a Benders combinatorial cut is generated and augmented to MP in the next iteration to force MP to change its solution. This framework is strengthened by considering, in addition to the combinatorial cuts, Benders classical cuts obtained by solving the dual of the linear relaxation of SP. Our approach also employs three sets of surrogate constraints to further improve the solution quality of the Benders master problem. In addition, by exploring the inherent trade-off specific to our problem, we devise an effective heuristic algorithm to obtain good upper bounds for the WCS network design problem. This heuristic algorithm also guarantee the feasibility of Benders subproblems, and thus, to generate good Benders cuts.

5 Computational Study on Algorithmic Performance

The effectiveness of the proposed solution methodology is assessed based on comparisons to Branch & Cut (B&C) approach implemented by CPLEX. Random data sets are generated to represent different transportation networks in the state of traffic UE. Due to excessive solution times with the B&C method, for a fair comparison, we solve each instance first by using the proposed BD approach with an optimality gap of 2.0% and record the runtime. The BD runtime employed as a stopping criterion when solving the same instance using the B&C approach and the optimality gap upon termination is recorded.

The results show superior performance for the proposed methodology in comparison to the B&C approach. For all tested data classes, under the same runtime, our suggested BD approach was able to solve our model to optimality gaps significantly less than 2%. On the other hand, B&C produced low-quality solutions, specifically with poor upper bound values and, most of the time, without lower bound values.

6 A Case Study: Chicago Sketch Network

We present a case study on Chicago Sketch Network [1] (with 933 nodes, 2,950 links, and 378 zones) to demonstrate the applicability of our model and the proposed algorithm on real traffic network. We also provide a sensitivity analyses on some of the system parameters that affect the system cost. The results indicate that the model captures key dynamics among input parameters including battery capacity, charging power, and system efficiency and thus provide the means of analyzing trade-offs involved in different real networks.

References

[1] Bar-Gera, H. 2016. Transportation test problems. Website: `https://github.com/bstabler/TransportationNetworks`. Accessed July 5 2016.

[2] Benders, J. F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* **4** 238–252.

[3] Chen, Zhibin, Fang He, Yafeng Yin. 2016. Optimal deployment of charging lanes for electric vehicles in transportation networks. *Transportation Research Part B: Methodological* **91** 344–365.

[4] Codato, G., M. Fischetti. 2006. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research* **54**(4) 756–766.

[5] energy.gov. 2016. September 2015 dashboard. `https://energy.gov/eere/vehicles/fact-947-october-17-2016-over-half-million-plug-vehicles-have-been-sold-united-states`.

[6] Fuller, M. 2016. Wireless charging in california: Range, recharge, and vehicle electrification. *Transportation Research Part C: Emerging Technologies* **67** 343–356.

[7] Haddadian, G., M. Khodayar, M. Shahidehpour. 2015. Accelerating the global adoption of electric vehicles: Barriers and drivers. *The Electricity Journal* **28**(10) 53–68.

[8] Jang, Y. J., Y. D. Ko, S. Jeong. 2012. Optimal design of the wireless charging electric vehicle. *Electric Vehicle Conference (IEVC)*. IEEE, 1–5.

[9] Ko, Y. D., Y. J. Jang. 2013. The optimal system design of the online electric vehicle utilizing wireless power transmission technology. *Intelligent Transportation Systems, IEEE Transactions on* **14**(3) 1255–1265.

[10] Mahmassani, H. S. 1994. Development and testing of dynamic traffic assignment and simulation procedures for ATIS / ATMS applications. Technical Report DTFH6 1-90-R-00074-FG, Center for Transportation Research, The University of Texas at Austin.

[11] Onar, O. C., J. M. Miller, S. L. Campbell, C. Coomer, C. White, L. E. Seiber, et al. 2013. A novel wireless power transfer for in-motion ev/phev charging. *Applied Power Electronics Conference and Exposition (APEC), Twenty-Eighth Annual IEEE*. 3073–3080.

[12] Riemann, R., D. Z. Wang, F. Busch. 2015. Optimal location of wireless charging facilities for electric vehicles: flow-capturing location model with stochastic user equilibrium. *Transportation Research Part C: Emerging Technologies* **58** 1–12.

[13] Trigg, T., P. Telleen, et al. 2013. Global EV outlook: Understanding the electric vehicle landscape to 2020. *Int. Energy Agency* 1–40.

[14] U.S. Department of Energy. 2011. One million electric vehicles by 2015. `https://www1.eere.energy.gov/vehiclesandfuels/pdfs/1_million_electric_vehicles_rpt.pdf`.

VEHICLE ROUTING MODELS & APPLICATIONS

TB3: VRP EXACT METHODS

Thursday 1:00 – 2:30 PM

Session Chair: Michel Gendreau

1:00 **An Integer Programming Approach for the Time-Dependent Traveling Salesman Problem with Time Windows**

[1]*Agustin Montero,* [2]*Isabel Mendez-Diaz,* [3]*Juan Jose Miranda Bront**

[1]*FCEyN-Universidad de Buenos Aires,* [2]*Universidad de Buenos Aires,* [3]*Universidad Torcuato Di Tella/Consejo Nacional de Investigaciones Científicas y Técnicas*

1:30 **A Mixed-Integer Linear Program for the Traveling Salesman Problem with Structured Time Windows**

[1]*Philipp Hungerlaender,* [2]*Christian Truden**

[1]*Laboratory for Information & Decision Systems-MIT,* [2]*Alpen-Adria Universität Klagenfurt, Austria*

2:00 **A Branch-and-Price Algorithms for a Multi-Attribute Technician Routing and Scheduling Problem**

[1]*Michel Gendreau**, [2]*Ines Mathlouthi,* [2]*Jean-Yves Potvin*

[1]*CIRRELT and MAGI-École Polytechnique de Montréal,* [2]*CIRRELT and DIRO-Université de Montréal*

An Integer Programming approach for the Time-Dependent Traveling Salesman Problem with Time Windows

Agustín Montero*[1], Isabel Méndez-Díaz†[1], and Juan José Miranda-Bront‡[2,3]

[1]Departmento de Computación, FCEyN, Universidad de Buenos Aires
[2]Universidad Torcuato Di Tella
[3]Consejo Nacional de Investigaciones Científicas y Técnicas

1 Introduction and literature review

The efficient use of the transportation infrastructure and the impact of congestion have become one of the major issues in city planning and urban logistics due to an increase of the complexity of operations, specially in highly populated areas (see, e.g., Savelsbergh and Van Woensel [8]). Therefore, the current traffic situation as well as the projected traffic scenarios are likely to have, if not addressed correctly, a negative impact from a social, economic and a environmental standpoint.

Most of the research related to the Vehicle Routing Problem (VRP) considers that the travel time between two locations are fixed along the time horizon. An updated description of variants and methods can be found in Toth and Vigo [10]. In the last few years, there has been a trend to enrich these models by incorporating more complex travel time functions, aiming to obtain solutions that are closer to real-world operations. These models are particularly useful for urban logistics, where congestion may produce significant variations in travel times during different moments of the day. For instance, last mile deliveries, which are estimated to account of an important percentage of the total delivery costs, could be significantly improved by more realistic approaches, translating into a better service and a more efficient use of the resources.

Time-Dependent Vehicle Routing Problems (TDVRPs) is the name given to a family of problems that generalize the classical VRPs by considering more complex travel time and cost functions, generally by incorporating some variability depending on the moment of the day an arc is traversed. A recent survey on TDVRP variants is available in Gendreau et al. [5], covering exact and heuristic algorithms.

The approach suggested by Ichoua et al. [7] to model congestion has recently caught the attention of many researchers. Cordeau et al. [3] tackle the TDTSP with the objective to minimize the makespan. They study some of the properties of the travel time function, including the computation of a lower bound obtained by solving an auxiliary TSP with constant travel

*aimontero@dc.uba.ar
†imendez@dc.uba.ar
‡jmiranda@utdt.edu

times. They show that the bound is tight depending on some parameters related to the travel speed definitions and that, under some particular settings, the solution of the auxiliary TSP is indeed optimal. They also propose a Branch and Cut (BC) algorithm and are able to solve instances with up to 40 vertices. Ghiani and Guerriero [6] further exploit some of the properties of the travel time function, and study its generality. In a follow up paper, Arigliano et al. [1] extend the ideas proposed in Cordeau et al. [3] to the TDTSP with Time Windows (TDTSP-TW). However, the results obtained are not as good as for the TDTSP. A BC algorithm is evaluated on instances with up to 40 clients, obtaining mixed results.

Multi-vehicle versions of the TDVRP have also been tackled by exact algorithms that consider the model proposed in Ichoua et al. [7]. Dabia et al. [4] consider the TDVRP with time windows with the objective of minimizing the overall duration instead of the makespan. They propose a set partitioning model and develop a Branch and Price (BP) algorithm, where the column generation subproblem is solved by means of a tailored labeling algorithm. Related to this research is the work by Sun et al. [9], where a profitable TDTSP with time windows and precedence constraints are considered. Indeed, this particular variant arises as the column generation subproblem of a TDVRP with time-windows and precedence constraints. They propose an Integer Linear Programming (ILP) model for the problem, which is not studied in detail due to its poor performance when tackled with standard commercial solvers, and resort to dynamic programming techniques.

In this research we tackle the version TDTSP-TW considered also in Arigliano et al. [1]. The contributions of this paper are the following: *1) we propose an alternative approach for the TDTSP-TW that builds on the ILP formulation proposed by Sun et al. [9]; 2) we develop a BC algorithm, including several initial heuristics, preprocessing rules and a cutting plane algorithm that incorporates several families of valid inequalities; 3) we evaluate our approach and compare the results with the ones reported in by Arigliano et al. [1].* To the best of our knowledge, this is the first comparison of two exact approaches for the TDTSP-TW, establishing a baseline for future approaches for the TDTSP-TW and other related problems.

2 Problem definition

The network is defined over a digraph $D = (V, A)$, with $V = \{0, 1, \ldots, n, n+1\}$ the set of vertices and A the set of arcs. Vertices 0 and $n + 1$ represent the depot, for which we do not consider the incoming and outgoing arcs, respectively. There is a time horizon $[0, T]$ in which vehicles move along the network. For each vertex $i \in V$, we denote by p_i to its processing time and $W_i = [r_i, d_i]$ the corresponding (hard) time window, where r_i and d_i are the release and deadline times, respectively. In particular, we set $W_0 = W_{n+1} = [0, T]$. We allow waiting times when arriving at a vertex before its release time r_i, but the vehicle must wait until r_i before starting to process it. In addition, each arc $(i, j) \in A$ has an associated travel distance L_{ij}. Without loss of generality, $d_i + p_i \leq T$ for all $i \in V$. In addition, to simplify the notation in the manuscript, we slightly modify the standard definition and assume that $p_i = 0$ for $i \in V$. However, the models and formulae present in this paper can be easily adapted to consider processing times.

Ichoua et al. [7] propose to partition the planning horizon into M intervals $[T_h, T_{h+1}]$, $h = 0, \ldots, M - 1$. For each arc $(i, j) \in A$, the average value of the travel speed during the time interval $[T_h, T_{h+1}]$, denoted by v_{ijh}, for $h = 0, \ldots, M-1$, is known. We refer to this partition as *speed profiles*. It is important to remark that the speed profiles may differ among arcs. Based

on this definition, the travel times are computed using the information of the distance to be traveled, i.e. L_{ij}, combined with the travel speeds v_{ijh} defined for the arc. However, it is not assumed that the travel speed remains fixed during the trip and it may change whenever the boundaries of a time interval are crossed. We denote by $\tau_{ij}(t)$ to the time-dependent travel time value on arc $(i,j) \in A$ if departing from i at time $t \in [0,T]$, and it can be computed following Algorithm 1 from Ichoua et al. [7]. This model is able to capture the time dependency while satisfying the FIFO condition on the travel times.

The TDTSP-TW involves finding a tour that visits each vertex exactly once with the objective of minimizing the makespan of the route. The route starts at vertex 0 and ends at vertex $n+1$, while processing each vertex within its defined time window and computing the travel times following the speed model proposed in Ichoua et al. [7].

Cordeau et al. [3] propose expressing the travel speeds $v_{ijh} = \delta_{ijh} b_h u_{ij}$, where u_{ij} represents the maximum speed for arc $(i,j) \in A$ during the planning horizon, $b_h \in [0,1]$ is the best congestion factor during interval $[T_h, T_{h+1}]$ and $\delta_{ijh} \in [0,1]$ represents the heaviest degradation of the congestion factor of $(i,j) \in A$ in interval $[T_h, T_{h+1}]$ with respect to the less congested arc in $[T_h, T_{h+1}]$. From a practical standpoint, this decomposition allows the authors to formulate alternative scenarios that can be used to compute lower bounds for the problem by solving an auxiliary problem as mentioned in the introduction. Our approach, however, does not rely on exploiting this decomposition and builds directly upon the average travel speeds v_{ijh}.

3 ILP formulation and Branch and Cut algorithm

An alternative formulation for the TDTSP-TW can be obtained from the model proposed by Sun et al. [9] for the Profitable TDTSP with Time Windows and Pickup and Delivery. Sun et al. [9] report that for this particular problem the model does not produce good results when solved by a commercial solver.

One of the interesting features of this ILP formulation is that, for each edge, it redefines the partitions of the time horizon in order to obtain a linear travel time function within each of them. The limits defining this new partition are referred as *time breakpoints* and allow to easily embed the piece-wise linear time function within an ILP formulation. Formally, let $T^{ij} = \{T_1^{ij}, \ldots, T_M^{ij}\}$ be the new partition of the time horizon into time intervals (also called time zones) for arc $(i,j) \in A$. We denote the consecutive time breakpoints defining $T_m^{ij} \in T^{ij}$ as $T_m^{ij} = [w_m, w_{m+1}]$. At this point, we abuse notation and refer to each time zone T_m^{ij} as $m \in T^{ij}$, with $m = 1, \ldots, |T^{ij}|$. By definition, $\tau_{ij}(t)$ becomes a linear function within each time zone. Therefore, in order to obtain the linear function describing the travel time when traversing the arc starting within the time zone can be calculated using w_m, w_{m+1} and the proper evaluations of $\tau_{ij}(t)$. We denote by θ_{ij}^m and η_{ij}^m to the coefficients of the linear function, such that

$$\tau_{ij}(t_i) = \theta_{ij}^m t_i + \eta_{ij}^m, \qquad \forall\, t_i \in T_m^{ij}. \tag{1}$$

Let binary variables x_{ij}^m take value 1 iff the vehicle traverses arc $(i,j) \in A$ starting from i within time zone $T_m^{ij} \in T^{ij}$. For each vertex $i \in V$, a continuous nonnegative variable t_i accounts for the time that i is visited in the tour. The value of t_i is decomposed to indicate time for a given arc in a given zone. This is achieved by introducing continuous variables t_{ij}^m defined in the following fashion such that $t_{ij}^m = t_i$ if $x_{ij}^m = 1$, and 0 otherwise. The travel time function for arc

(i,j), $\tau_{ij}(t_i)$, can be defined in an aggregated as

$$\tau_{ij}(t_i) = \sum_{T_m^{ij} \in T^{ij}} \theta_{ij}^m t_{ij}^m + \eta_{ij}^m x_{ij}^m. \tag{2}$$

Due to space limitations, we do not include the ILP formulation explicitly. We remark that it can be obtained by adapting the one presented in Sun et al. [9] to the TDTSP-TW. We name the resulting ILP formulation as TTBF. A BC algorithm based on the TTBF with the characteristics described below has been developed.

- *Preprocessing rules:* we extended and adapted to the time-dependent case some of the rules proposed in Ascheuer et al. [2], including arc removal, tightening of time windows and precedence inference between vertices based on the information provided by their time windows.
- *Initial heuristic:* A sorting heuristic, a nearest neighbor and an insertion heuristic combined with a local search procedure are considered to construct an initial feasible solution, which is then transferred to the BC algorithm as an initial incumbent solution.
- *Cutting planes:* we included the classical Subtour Elimination Constraints (SEC) and the $\pi-$, $\sigma-$ and $(\pi,\sigma)-$inequalities from the Precedence Constrained TSP which exploit the precedences identified in the preprocessing step.

4 Summary of computational results

We conducted computational experiments in order to evaluate the performance of the BC algorithm described in the previous section, named TTBF-BC. We compare our results to the other exact approach for the TDTSP-TW, proposed by Arigliano et al. [1] and referred as LBF-BC. The algorithms are coded in C++, using CPLEX 12.5 Callable Library as LP and MILP solver. The experiments are run on a Workstation with an Intel Core i7-2600 3.4GHz CPU and 16GB of RAM. The algorithms are evaluated on the instances considered in Arigliano et al. [1], which are constructed by extending the instances generated in Cordeau et al. [3] to the case with time windows. The congestion is partially captured by the values δ_{ijh} and $\Delta = \min_{i,j,h} \delta_{ijh}$. They consider values of $n = 15, 20, 30, 40$, $\Delta = 0.7, 0.8, 0.9, 0.98$ and two different traffic patterns, with 30 instances for each combination of these parameters.

We report the number of instances solved to optimality over 30 instances (OPT), computational times in seconds (Time), number of nodes explored in the BB tree (Nodes), the % gap at the root node (%rG) and at the end of the execution (%fG). We impose a limit of 3600 seconds for the execution time, the results are disaggregated between solved and unsolved instances. Each metric is averaged over the corresponding subset of instances, depending on the case. Gaps %rG and %fG are computed as $(z_{\text{best}} - z)/z$, where z_{best} represents the objective function of the best feasible solution for the instance and z the value being considered.

The results obtained suggest that TTBF-BC outperforms LBF-BC, produces better results in terms of number of instances solved, the average computing time and number of nodes explored during the enumeration. Due to space limitations, only the results for $n = 40$ are presented in Table 1[1] The behavior exhibited regarding the relation between %rG and %fG indicates that

[1]We believe the %rG of LBF-BC for $\Delta = 0.7$, Traffic Pattern B is indeed a typo and should be multiplied by 100. There are two missing files in the set of instances, which we consider as *unsolved* for our algorithm.

Pattern	Δ	LBF-BC					TTBF-BC				
		OPT	%rG	%fG	Nodes	Time	OPT	%rG	%fG	Nodes	Time
A	0.98	18	0.06	0.00	3	1.76	26	46.22	0.00	1918	471.36
			2.51	2.35	-	-	4	67.38	3.39	5749	-
	0.90	10	0.22	0.00	300	40.73	25	42.58	0.00	1759	389.56
			2.14	1.88	-	-	5	65.48	2.81	19161	-
	0.80	11	0.53	0.00	615	47.66	30	41.66	0.00	1891	382.74
			2.10	1.25	-	-					-
	0.70	12	1.00	0.00	6	2.40	22	33.44	0.00	535	385.11
			3.11	1.73	-	-	6	32.15	0.56	34645	-
B	0.98	9	0.13	0.00	0	0.78	27	121.38	0.00	1536	426.12
			1.00	0.73	-	-	3	204.63	3.22	17890	-
	0.90	10	0.70	0.00	205	25.21	25	78.91	0.00	2576	480.51
			1.93	0.70	-	-	5	135.28	2.73	18.833	-
	0.80	11	1.48	0.00	26	8.03	29	86.25	0.00	1090	293.45
			3.40	1.72	-	-	1	187.90	1.73	3331	-
	0.70	3	0.99	0.00	0	1.08	28	70.48	0.00	1022	381.54
			0.03	2.80	-	-	2	179.30	0.98	4094	-

Table 1: Results for traffic pattern A and B, $n = 40$.

the lower bound considered in LBF-BC is very tight in general, but in terms of a BC algorithm the formulation finds difficulties to improve this bound and close the gap to prove optimality. Indeed, the authors report that the cutting plane algorithm is not able to improve this bound at the root node, and we observed that in many of the instances solved, the optimality is proved before starting the enumeration.

References

[1] A. Arigliano, G. Ghiani, A. Grieco, and E. Guerriero. Time dependent traveling salesman problem with time windows: Properties and an exact algorithm. Technical report.

[2] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, 90(3):475–506, 2001.

[3] J.-F. Cordeau, G. Ghiani, and E. Guerriero. Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. *Transportation Science*, 48(1):46–58, 2012.

[4] S. Dabia, S. Ropke, T. van Woensel, and T. D. Kok. Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science*, 47(3):380–396, 2013.

[5] M. Gendreau, G. Ghiani, and E. Guerriero. Time-dependent routing problems: A review. *Computers & Operations Research*, 64:189–197, 2015.

[6] G. Ghiani and E. Guerriero. A note on the ichoua, gendreau, and potvin (2003) travel time model. *Transportation Science*, 48(3):458–462, 2014.

[7] S. Ichoua, M. Gendreau, and J.-Y. Potvin. Vehicle dispatching with time-dependent travel times. *European journal of operational research*, 144(2):379–396, 2003.

[8] M. Savelsbergh and T. Van Woensel. 50th Anniversary Invited Article City Logistics : Challenges and Opportunities. *Transportation Science*, (March), 2016.

[9] P. Sun, S. Dabia, L. P. Veelenturf, and T. Van Woensel. The time-dependent profitable pickup and delivery traveling salesman problem with time windows. Technical report, Eindhoven University of Technology, 2015.

[10] P. Toth and D. Vigo, editors. *Vehicle Routing: Problem, Methods and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2014.

A Mixed-Integer Linear Program for the Traveling Salesman Problem with Structured Time Windows

Philipp Hungerländer* Christian Truden †

5th January 2017

Abstract

In this extended abstract we introduce the Traveling Salesman Problem with structured Time Windows (TSPsTW) that is motivated by an online shopping application of an international supermarket chain. We suggest an efficient and easy to implement mixed-integer linear programm (MILP) of the TSPsTW and in a computational study compare it to related MILPs from the literature. Finally we analyze the relation of TSPsTW, TSPTW and TSP with the help of dependency graphs and give an outlook on further planned extensions of the applicability of our MILP.

Key words. Traveling salesman problem, time windows, exploiting structure, mixed-integer linear programming.

1 Introduction

The NP-hard [12] Traveling Salesman Problem with Time Windows (TSPTW) is concerned with visiting a set of n customers within their assigned time windows such that a given objective function is minimized. For the TSPTW there are two main objective functions considered in literature:

- *Makespan*: Minimize the total tour duration, i.e. the completion time of the tour,
- *Travel time*: Minimize the sum of travel times between all pairs of subsequent customers in the tour.

The TSPTW has several applications in its own right [4] and additionally also appears as subproblem within the more general capacitated Vehicle Routing Problem with Time Windows (cVRPTW), see e.g. [2, 5] for excellent survey articles. In their recent paper Kara and Derya [11] give a comprehensive review of exact approaches to both the symmetric and the asymmetric TSPTW and survey available benchmark instances for both problem variants.

In our forthcoming conference paper [9] (see also the technical report [10] for an extended version) we have already defined the Traveling Salesman Problem with structured Time Windows (TSPsTW) as a subproblem of a capacitated Vehicle Routing Problem with structured Time Windows (cVRPsTW). The difference between the TSPsTW and the standard TSPTW is the special structure of the time windows: structured time windows can hold several customers and are non-overlapping. Note that similarly the TSPTW was introduced as a subproblem of the VRPTW by Savelsbergh [15] in 1992.

This special structure of the time windows is motivated by an application in the context of a large international supermarket chain that builds their vans' tours as new customer orders come in. Nowadays, all main supermarket chains provide online shopping services, where customers select groceries on the supermarket's website, as well as a delivery time window. Then the supermarket distributes the groceries to the customers within the time window that the customer selected.

For this online shopping application the structure of the time windows is set by the supplier who provides the customer with a selection of time windows to choose from. Hence in our application providing these special structural features does neither impose substantial restrictions to the supplier nor to the customers. Due to the strict time limits of our application we strive to develop fast methods that are able to computationally exploit this additional structure of the time windows efficiently. The main contributions of this extended abstract are:

*Laboratory for Information & Decision Systems, MIT, USA, philipp.hungerlaender@aau.at
†Department of Mathematics, Alpen-Adria Universität Klagenfurt, Austria, christian.truden@aau.at

1. Defining the TSPsTW that is motivated by an online shopping application.
2. Suggesting a simple and nonetheless efficient mixed-integer linear program (MILP) for the TSPsTW that can be easily and quickly implemented by practitioners and comparing it to related MILPs from the literature in a computational study.
3. Analyzing the relation of TSPsTW, TSPTW and TSP with the help of dependency graphs.

The extended abstract is organized as follows. We give a formal problem description of the TSPsTW in Section 2 and state a corresponding MILP in Section 3. In Section 4 we demonstrate the efficiency of our MILP on benchmark instances related to our online shopping application in a computational study. In Section 5 we motivate the concept of dependency graphs and Section 6 concludes the paper and gives an outlook on planned future extensions.

2 Formal Definitions for the TSPsTW

A TSPsTW instance consists of the following input data:

- A set of *customers* $\mathscr{C}$.
- A *travel time* function $t : \mathscr{C} \times \mathscr{C} \to \mathbb{R}^+$.
- A set of *time windows* $\mathscr{W} = \{w_1, \ldots, w_q\}$, where each time window $w \in \mathscr{W}$ is defined through its *start time* s_w and its *end time* e_w.
- A function $w : \mathscr{C} \to \mathscr{W}$ that assigns a time window to each customer during which the delivery van has to arrive at the customer.

We speak of *structured* time windows, if the number of customers $|\mathscr{C}| = n$ is much larger than the number of time windows $|\mathscr{W}| = q$, i.e. $n >> q$, and therefore typically several customers are assigned to the same time window. We further denote the set of customers assigned to time window $w \in \mathscr{W}$ as $[n_w]$. We assume that all time windows are non-overlapping, i.e. $e_w \leq s_{w+1},\ w \in [q-1]$, holds, where the sets $[u],\ u \in \mathbb{N}$, and $[u_0],\ u \in \mathbb{N}$, contain the elements $\{1,2,\ldots,u-1,u\}$ and $\{0,1,2,\ldots,u-1,u\}$ respectively.

A *tour* $\mathscr{A} = \{a_0, a_1, a_2, \ldots, a_n, a_{n+1}\}$ contains n customers, where the indices of the customers display the sequence in which the customers are visited. A tour starts at the depot a_0 and ends at the depot a_{n+1}. Typically $a_0 = a_{n+1}$ is assumed. Furthermore our start and end depot get assigned the time windows w_0 and w_{q+1} with start time 0 and the end time set to ∞.

Note that most TSPTW models from literature do not explicitly contain service times at the customers. If a *service time* function $s : \mathscr{C} \to \mathbb{R}^+$ is given, then the service times are usually added to the travel times, i.e. $t_{ij} = t(i,j) + s(i),\ i,j \in \mathscr{C}$, resulting in an asymmetric travel time function. Hence a symmetric travel time function is only possible if all service times are equal which makes the symmetric TSPTW not usable for many applications.

In our online shopping application, we typically deal with asymmetric travel time functions for which the triangle inequalities $t(u,v) \leq t(u,w) + t(w,v),\ u,v,w \in \mathscr{C}$, hold in theory. However in practice the travel times are usually rounded to integers, which typically destroys the correctness of some triangle inequalities [7]. Thus it is quite convenient that our MILP presented in the following section is applicable to arbitrary asymmetric travel time functions.

3 A MILP for the TSPsTW

In order to formulate a MILP for the TSPsTW, let us introduce the following decision variables:

- The binary variables $x_{ij} \in \{0,1\},\ i \in [n_k],\ j \in [n_\ell],\ k \in [q_0],\ \ell \in [q+1],\ k \leq \ell \leq k+1,\ i \neq j$, with the interpretation:

$$x_{ij} = \begin{cases} 1, & \text{if customer } a_j \text{ is visited directly after customer } a_i, \\ 0, & \text{otherwise.} \end{cases}$$

- The non-negative variables $w_i,\ i \in [q_0]$, giving the wait time during time window i.

Furthermore we use the non-negative parameters α_1 and α_2 with $\alpha_1 + \alpha_2 = 1$ that define a convex combination of *travel time* and *wait time* in our objective function. If we set $\alpha_1 = \alpha_2 = 0.5$, then our MILP determines a tour with minimal *makespan*.

Note that the ability to consider different objective functions is important in our online shopping application. While new customer orders come in and are integrated into the delivery schedule, we minimize the sum of travel time and then after closing the delivery schedule for new customer orders the goal is to determine tours with minimal makespan.

Now we can formulate the TSPsTW as the following MILP:

$$\min \quad \alpha_1 \sum_{\substack{i\in[n_k],\ j\in[n_\ell],\ k\in[q_0] \\ \ell\in[q+1],\ k\le\ell\le k+1,\ i\neq j}} t_{ij}x_{ij} + \alpha_2 \sum_{i\in[q_0]} w_i \tag{1a}$$

$$\text{s.t.} \quad \sum_{\substack{i\in[n_k],\ k\in[q_0] \\ i\neq j,\ k\le\ell\le k+1}} x_{ij} = 1, \quad j \in [n_\ell],\ \ell \in [q+1], \tag{1b}$$

$$\sum_{\substack{j\in[n_\ell],\ \ell\in[q+1] \\ j\neq i,\ k\le\ell\le k+1}} x_{ij} = 1, \quad i \in [n_k],\ k \in [q_0], \tag{1c}$$

$$\sum_{\substack{i,j\in S, \\ i\neq j}} x_{ij} \le |S| - 1, \quad S \subset [n_k],\ k \in [q],\ |S| \ge 2, \tag{1d}$$

$$\sum_{\substack{i\in[n_k],\ j\in[n_\ell] \\ k\le\ell\le k+1,\ k<h,\ i\neq j}} t_{ij}x_{ij} + \sum_{i\in[h-1]} w_i \ge s_h, \quad h \in [q], \tag{1e}$$

$$\sum_{\substack{i\in[n_k],\ j\in[n_\ell] \\ k\le\ell\le k+1,\ k,\ell\le h,\ i\neq j}} t_{ij}x_{ij} + \sum_{i\in[h]} w_i \le e_h, \quad h \in [q_0], \tag{1f}$$

$$x_{ij} \in \{0,1\}, \quad i \in [n_k],\ j \in [n_\ell],\ k \in [q_0],\ \ell \in [q+1],\ k \le \ell \le k+1,\ i \neq j, \tag{1g}$$

$$w_i \ge 0, \quad i \in [q_0]. \tag{1h}$$

The objective function (1a) ensures minimization of a weighted sum of travel and wait time. Equalities (1b) guarantee that we visit all customers and the end depot exactly once and equalities (1c) ensure that we leave the start depot and all customers exactly once. Inequalities (1d) are the well-known subtour elimination constraints. Finally inequalities (1e) and (1f) guarantee that the arrival times at all customers are neither before the start nor after the end of their assigned time window.

4 Computational Experiments

All experiments were performed on a Ubuntu 14.04 machine equipped with an Intel Xeon E5-2630V3 @ 2.4 GHz 8 core processor and 132 GB RAM. We use Gurobi 6.5.1 as an IP-solver in single thread mode. We implemented the two MILPs for the symmetric and the asymmetric TSPTW suggested by Kara and Derya [11], where we set $M = \max_{i\in[n]} e_i + \max_{i\in[n]} t_{i0}$ in the symmetric TSPTW MILP as it was not specified in their paper.

We generated symmetric TSPsTW instances in order to compare all three MILPs. The coordinates of the customers are sampled from a two-dimensional uniform distribution and the travel times are calculated as the Euclidean distance between customers. The instances are generated by a simple heuristic that assigns customers to time windows randomly and checks the feasibility of the assignment.

In Table 1 we compare the run times of our TSPsTW MILP with the symmetric and the asymmetric TSPTW MILPs proposed by Kara and Derya [11]. We solve all models to optimality and set the time out *TO* to 20 minutes. By n_q^- and n_q^+ we denote the minimal and maximal number of customers assigned to one of time windows in the respective benchmark instance. We use symmetric instances and the makespan objective function such that we are able to apply all three models.

n	q	n_q^-	n_q^+	Symmetric TSPTW[11]	Asymmetric TSPTW[11]	TSPsTW
12	5	1	4	9 ms	4 ms	1 ms
39	10	2	6	11 ms	4.57 sec	43 ms
35	15	1	4	12 ms	1.88 sec	17 ms
38	20	1	4	7 ms	38 ms	10 ms
83	30	1	6	55 ms	21.03 sec	255 ms
135	40	1	6	122 ms	TO	1.46 sec
133	50	1	6	107 ms	1.45 min	704 ms
209	60	1	8	482 ms	TO	1.94 sec
169	70	1	6	185 ms	46.49 sec	815 ms
194	80	1	7	208 ms	1.62 min	1.45 sec
246	90	1	6	406 ms	1.07 min	2.35 sec
327	100	1	7	855 ms	TO	5.62 sec

n	q	n_q^-	n_q^+	Symmetric TSPTW[11]	Asymmetric TSPTW[11]	TSPsTW
31	5	5	8	54 ms	10.42 min	152 ms
35	10	2	5	12 ms	4.33 sec	139 ms
51	15	1	6	40 ms	1.64 min	104 ms
95	20	2	9	76 ms	TO	661 ms
151	30	2	9	215 ms	TO	6.12 sec
184	40	1	9	275 ms	TO	5.30 sec
219	50	1	11	474 ms	TO	10.34 sec
269	60	1	9	497 ms	TO	7.04 sec
300	70	2	11	926 ms	TO	12.55 sec
355	80	1	8	935 ms	TO	12.55 sec
415	90	2	11	1.39 sec	TO	18.99 sec
447	100	1	10	1.67 sec	TO	1.03 min

Table 1: Results of our computational experiments comparing our TSPsTW MILP with two MILPs from the literature on instances with up to 100 time windows and up to 447 customers. The time out *TO* is set to 20 minutes.

While the symmetric TSPTW MILP is clearly the fastest one, it also has the most narrow applicability. On the one hand it is not possible to model an objective function considering the sum of travel times and on the other hand the symmetric TSPTW MILP cannot handle asymmetric instances or instances for which the triangle inequalities do not hold. These are a quite severe restrictions, see the last two paragraphs of Section 2 for further details.

The asymmetric TSPTW MILP is clearly outperformed by our TSPsTW MILP on all considered benchmark instances. For the larger instances it often even does not find a feasible start solution until the time out. Hence in summary the TSPsTW MILP is clearly the most appropriate exact approach for our shopping application, where offering structured time windows instead of arbitrary ones is not a significant restriction for the supplier.

Further computational experiments show that the run times of our TSPsTW MILP solely depend on n and n_q^+. Hence our MILP shows the same performance on benchmark instances with asymmetric travel times, independent of the correctness of the triangle inequalities. We omit detailed computational results showcasing these properties due to the space restrictions of this extended abstract.

5 Extension: Dependency Graphs

In this section we outline a concept for analyzing and illustrating the connection between the TSP, the TSPTW and the TSPsTW. When considering structured time windows, it is easy to determine which edges between customers are needed in the TSPsTW model. On the one hand we have to include all edges between ordered pairs of customers assigned to the same time window and on the other hand we have to consider all directed edges going from customers assigned to time window i to customers assigned to time window $i+1$ for $i \in [q_0]$.

For formalizing and generalizing the above description we introduce so-called dependency graphs $G = (V, D)$, where the set of vertices is equal to the set of time windows of the corresponding TSPTW instances, i.e. $V = [(q+1)_0]$. A directed edge between an ordered pair (a,b) of time windows is included in the edge set D, if there is a customer from time window b who can be visited after a customer that is serviced during time window a.

Applying this definition the dependency graph of a TSPsTW instance is a directed line. Hence the TSPsTW can be viewed as consisting of q connected TSPs (one for each time window) and therefore as an in-between of TSPTW and TSP. In Figure 1 we illustrate the dependency graphs for three toy instances. Note that the dependency graphs become quite dense as soon as there are only a few overlapping time windows.

Gendreau et al. [7] and Ascheuer et al. [1] propose preprocessing approaches using the start and end times of time windows and the travel times for determining unnecessary edges of TSPTW models. Contrary to that our dependency graphs display all edges that have to be contained in a TSPTW model:

1. All edges between pairs of customers assigned to the same time window.

2. All directed edges between customers assigned to time windows that are connected by a directed edge in the dependency graph. Note that as a further refinement we can additionally apply the preprocessing approaches mentioned above to this edge set.

In summary dependency graphs can be used for easy identification of:

- Edges that must be included in the TSPTW model.
- Possible subtours, indicated by cycles in the dependency graph, which have to be prohibited by adding respective subtour elimination constraints.
- The practical complexity of the TSPTW instance that is related to number of edges contained in the corresponding dependency graph.

For determining dependency graphs of general, large-scale TSPTW instances we will propose a polynomial-time algorithm for building them in our forthcoming paper.

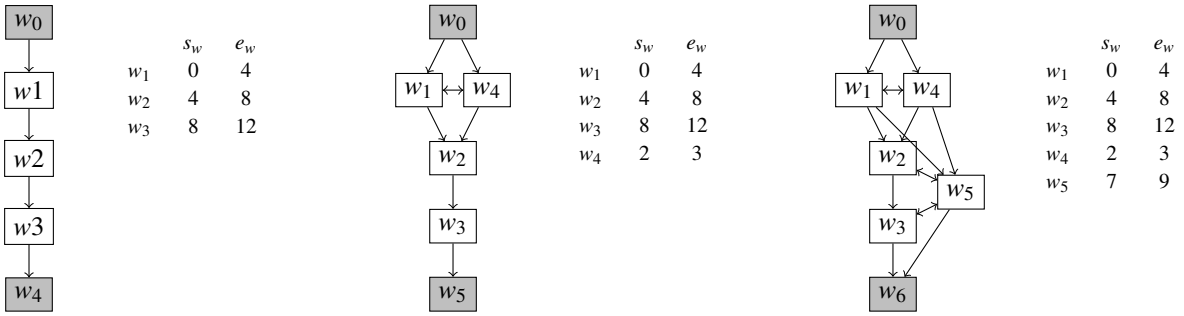


Figure 1: Dependency graphs for 3 instances with non-overlapping respectively partly overlapping time windows.

6 Conclusion & Outlook

In this extended abstract we presented first research findings derived within a common project with an international supermarket chain dealing with determining efficient tours for supplying customers that are assigned to structured time windows. We defined the Traveling Salesman Problem with structured Time Windows (TSPsTW) and suggested an easy to implement and nonetheless efficient mixed-integer linear program (MILP) for solving it. Additionally we analyzed the relation of TSPsTW, TSPTW and TSP with the help of dependency graphs.

Over the following months we plan to build a journal paper on the material presented in this extended abstract. In particular we plan the extend the applicability of our MILP in the following three directions:

1. Consider the on-line version of the TSPsTW, where customer requests are placed and processed by our MILP in real-time.

2. Design both cluster-first route-second and route-first cluster-second approaches for the cVRPsTW, where our MILP is used for determining efficient TSPsTW tours. Corresponding approaches for the standard VRP are e.g. discussed in [6, 8, 14, 16] and [3, 13] respectively.

3. Integrate constraints for modeling multiple salesmen with assigned capacity restrictions into our MILP in order to obtain a simple and efficient exact approach for the cVRPsTW.

Note that these extensions will also lead to a more comprehensive computational study of our TSPsTW MILP and related approaches.

References

[1] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving the Asymmetric Travelling Salesman Problem with time windows by branch-and-cut. *Mathematical Programming*, 90(3):475–506, 2001.

[2] R. Baldacci, A. Mingozzi, and R. Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6, 2012.

[3] J. E. Beasley. Route first - cluster second methods for vehicle routing. *Omega*, 11(4):403–408, 1983.

[4] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 35–139. Elsevier, 1995.

[5] N. A. El-Sherbeny. Vehicle routing with time windows: An overview of exact heuristic and metaheuristic methods. *Journal of King Saud University*, 22:123–131, 2010.

[6] M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2): 109–124, 1981.

[7] M. Gendreau, A. Hertz, G. Laporte, and M. Stan. A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows. *Operations Research*, 46(3):330–335, 1998.

[8] B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349, 1974.

[9] P. Hungerländer, K. Maier, J. Pöcher, A. Rendl, and C. Truden. Solving an on-line capacitated vehicle routing problem with structured time windows. In *Operations Research Proceedings 2016*, 2016. Accepted, a preprint is available from `http://www.optimization-online.org/DB_HTML/2016/12/5764.html`.

[10] P. Hungerländer, K. Maier, J. Pöcher, A. Rendl, and C. Truden. Solving an on-line capacitated vehicle routing problem with structured time windows. Technical Report TR-AAUK-M-O-16-12-30, Alpen-Adria Universität Klagenfurt, Mathematics, Optimization Group, 2016.

[11] I. Kara and T. Derya. Formulations for Minimizing Tour Duration of the Traveling Salesman Problem with Time Windows. *Proceedia Economics and Finance*, 26:1026–1034, 2015.

[12] R. M. Karp. Reducibility among Combinatorial Problems. In R. E. Miller and J. W. Thather, editors, *Complexity of Computer Computation*, pages 85–103. Plenum Press, 1972.

[13] C. Prins, P. Lacomme, and C. Prodhon. Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C: Emerging Technologies*, 40:179–200, 2014.

[14] D. M. Ryan, C. Hjorring, and F. Glover. Extensions of the petal method for vehicle routing. *Journal of the Operational Research Society*, 44:289–296, 1993.

[15] M. W. P. Savelsbergh. The Vehicle Routing Problem with Time Windows: Minimizing Route Duration. *ORSA Journal on Computing*, 4(2):146–154, 1992.

[16] E. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673, 1993.

A Branch-and-Price Algorithms for a Multi-Attribute Technician Routing and Scheduling Problem

Michel Gendreau, Ines Mathlouthi, Jean-Yves Potvin

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport
Montréal, Québec, Canada

1 Introduction and Problem Statement

The Technician Routing and Scheduling Problem (TRSP) is a common problem for many companies that provide a variety of services to other companies or private customers. Despite its complexities and its challenges, it has received a limited attention from the research community. TRSPs consists in routing staff to a set of locations in order to perform different tasks while satisfying resource and skill constraints.

In this paper, we tackle a problem already presented in [2]. It is motivated from an application in the maintenance and repair of electronic transaction equipments. The problem can be defined as follows. There is a set of technicians with different skill levels available to carry out different tasks. A task is created when a customer calls for maintenance of its equipment. Each task has a priority level depending on its emergency and the importance of the customer. One must then then assign the tasks to the technicians and build a route for each technician, starting and ending at his home base location, so as to minimize a given objective with terms for overtime, total traveled distance, and total gain over the performed tasks. The solution must also satisfy different constraints related to the required skills of the technicians to perform their assigned tasks, working hours, multiple time windows, total distance traveled, as well as availability of spare parts and special parts.

Two particular features distinguish this specific application from more generic routing problems. First, the technicians' schedules must account for complex break constraints: during a workday, which normally lasts from 9 AM to 5 PM, a technician performs some tasks and takes three breaks, a 15-minute break in the morning and in the afternoon, and a mid-day break of 30 minutes, with

suitable time windows. The second special feature is that some tasks require so-called *special parts*, i.e., parts that are carried in a technician's vehicle only if one or more tasks along his route need them. Technicians must retrieve these parts from a depot before performing these tasks. Furthermore, the solution must also account for the inventory of spare parts carried by each technician. Each technician starts his route with an initial inventory. If there are not enough parts to serve all tasks, the technician can replenish this inventory once along the route by going through a depot that has previously been assigned to him.

In [2], this problem was defined and solved with a commercial solver. This allowed us to tackle routinely instances with 10 tasks. Larger instances, with up to 20 or 25 tasks were also considered, but they could not usually be solved to optimality even in computational times of several hours. The main contribution of this work is to propose a specialized branch-and-price algorithm aimed at solving exactly instances with a larger number of tasks in reasonable CPU times.

The remainder of this paper is organized as follows. In section 2, we briefly review some related work. In section 3, we present the solution method. Some computational results are reported in section 4.

2 Related Work

There are two broad categories of TRSPs: *static TRSPs*, in which all service requests are known in advance, before routes are constructed, and *dynamic TRSPs*, in which a fraction of the service requests occur dynamically. Several authors (see, for instance, [5, 6]) have tackled static TRSP's derived from a variety of real-life settings in the late '90's and early 2000's. In most cases, they proposed solution methods based on local search and metaheuristics. More recently, Tang et al. [4] addressed a periodic maintenance operations scheduling problem and solved it with a tabu search heuristic. As for dynamic TRSPs, they are beyond the scope of this paper.

3 Solution method

In [2], we proposed a direct mixed-integer programming (MIP) formulation for the problem. Here, since we are interested in developing a branch-and-price approach, we proceed differently and derive a set packing formulation together with a route (column) generation subproblem.

The master (set packing) problem formulation is very simple: the decision variables correspond to the possible (feasible) routes for each technician (we must define a specific route set for each technician, since they operate from their own home location and have different skills). The master problem is a set packing one, because we do not need to cover all tasks: each task can

be performed at most once and each technician can perform at most one of his routes. To each route is associated a cost, which corresponds to the total disutility associated with performing it (including a negative component for the gain received for the tasks performed), and we minimize the total cost of all routes. As in other column generation schemes for routing problems, we consider at each iteration a restricted master problem (RMP), in which only a (small) subset of routes are available for each technician, and solve its linear relaxation RLMP. New routes for technicians are obtained through route generation subproblems (one per technician)

We first include in RLMP a set of initial routes. It consists of one route per technician. Each route starts and ends in the technician's home location; we then insert tasks one by one in the route according to their priority and taking into account skills and time windows. In case of shortage of parts or if a task needs a special tool, we plan a detour to the depot. The technician's route ends when we cannot insert any other task.

The route generation (pricing) problem associated with each technician aims at the identification of feasible routes (for this specific technician), given the current dual values associated with the constraints of the linear relaxation of the RMP (RLMP). This problem can modeled as a shortest path problem with various constraints pertaining to the tasks, the visits to the depot, the breaks, time windows, workday duration, and total distance traveled. This subproblem can be formulated as a MIP, which corresponds to a 1-technician version of the model presented in [2] (if one drops the dual variables). This MIP formulation is not practical for algorithmic purposes. We thus transform the problem into the usual form of an Elementary Shortest Path Problem with Resource Constraints(ESPPRC). Here, paths start and finish at the technician's location and visit nodes associated with the depot and nodes. As one proceeds, information associated with resources is stored in the labels. Each label stores the following information: C, the reduced cost of the partial path; T, the time spent to execute the partial path; D, the remaining distance available; SP, the quantity of spare parts available; F, a Boolean variable stating if the depot has been visited or not for replenishment in this partial path. Labels also include information on each task i stating whether or not this node is *unreachable*, i.e., if i has already been visited on the partial path, or if traveling from the current position (depot or another task) would violate its time windows, or the total distance permitted per day, or if the technician does not have the required spare parts and has already passed to the depot.

To solve this pricing problem, which is strongly NP-hard, we use two methods. The first one is based on the ESPPRC algorithm proposed by Feillet et al. [1], which implements an exact dynamic programming procedure enhanced by dominance rules to reduce the number of possible labels. The second approach relies on the Decremental State-Space Relaxation (DSSR) proposed by [3]. DSSR considers a relaxation of the ESPPRC in which multiple visits to nodes are possible. If cycles are found, the relaxation is tightened by consider-

ing some tasks as critical and forbidding multiple visits to them.

To enforce integrality of the route selection variables in RLMP, we must implement a suitable branching strategy. In fact, branching must be performed at each node where the optimal solution of the linear relaxation includes fractional route variables. We have decided to use a branching strategy that exploits particular features of our problem, namely the skill constraints. The basic idea in our branching scheme is to assign tasks to specific technicians, based on their skills. More precisely, when we encounter a fractional optimal solution in a node, we branch on the (technician, task) pair with *flow* closest to 0.5. This *flow* is obtained by summing the basic variables of the master problem associated with the routes containing a given task i and a given technician k. The removal of a task from the subproblem associated with a technician is much more restrictive than the elimination of a single arc in standard branching shemes for routing problems. We thus expect this strategy to be more effective than standard ones. We have implemented two variants of this branching strategy: a binary one, in which two nodes are created, one in which technician k is no longer allowed to perform task i, and one in which task i must be performed by technician k; and a ternary one, in which three nodes are created (here, the situation where technician k cannot to perform task i is split into two subcases: one in which task i is not performed, and one in which task i is performed by another technician).

4 Computational results

To test the efficiency of our branch-and-price algorithm, we use the same instances created by [2]. The main parameters of this instance generator are the width of the time windows (Narrow or Wide), the size of the service area (a square of 40 or 50 kms side), and the number of tasks.

We now report the results obtained on some of these instances. Branch-and-Price was given a maximum of 24 hours of computation time on a 3.07GHz Intel Xeon X5675 processor. We recall that each subset is made of 5 different instances.

Table 1 presents a comparison between solving instances with up to 25 tasks with CPLEX [2] and using the Branch-and-price algorithm. We limit ourselves to these instances, because CPLEX cannot handle larger instances within 24 hours of computation. It is obvious that our specialized algorithm is much more effective.

Detailed computational results for instances with up to 45 tasks will be presented at the conference.

Name	CPLEX			DSSR-Branch 2	
	# opt	CPU	# GAP	# Opt	CPU
N-40-10	5	839	0%	5	0.37
N-40-15	2(3)	59875	5%	5	0.87
N-40-20	0(5)	-	10%	5	2.84
N-40-25	0(5)	-	12%	5	61.43
N-50-10	5	542	0%	5	0.34
N-50-15	3(2)	29429	10%	5	0.53
N-50-20	1(4)	55029	17%	5	0.44
N-50-25	0(5)	-	18%	5	0.3
W-40-10	5	745	0%	5	0.34
W-40-15	1(4)	46142	3%	5	1.2
W-40-20	0(5)	-	9%	5	4.23
W-40-25	0(5)	-	9%	5	153.73
W-50-10	5	633	0%	5	0.29
W-50-15	3(2)	25531	10%	5	0.52
W-50-20	1(4)	25370	17%	5	1.03
W-50-25	0(3)	-	12%	5	7.86

Table 1: CPLEX vs B&P

References

[1] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[2] I. Mathlouthi, J.Y. Potvin, and M.Gendreau. Mixed integer programming for a multi-attribute technician routing and scheduling problem. Technical Report CIRRELT-2016-23, CIRRELT, 2016.

[3] G. Righini and M.Salani. Dynamic programming algorithms for the elementary shortest path problem with resource constraints. *Electronic Notes in Discrete Mathematics*, 17:247 – 249, 2004. Workshop on Graphs and Combinatorial Optimization.

[4] H. Tang, E. Miller-Hooks, and R. Tomastik. Scheduling technicians for planned maintenance of geographically distributed equipment. *Transportation Research Part E: Logistics and Transportation Review*, 43(5):591 – 609, 2007.

[5] E. Tsang and C. Voudouris. Fast local search and guided local search and their application to british telecom's workforce scheduling problem. *Operations Research Letters*, 20(3):119 – 127, 1997.

[6] J. Xu and S.Y. Chiu. Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7:495–509, 2001.

Vehicle Routing Models & Applications

TC3: VRP Extensions

Thursday 2:45 – 4:15 PM

Session Chair: Daniele Vigo

2:45 **Multi-Modal Variations of the Vehicle Routing Problem**

[1]*Marc-Antoine Coindreau,* [1]*Olivier Gallay*,* [2]*Nicolas Zufferey*

[1]*University of Lausanne,* [2]*University of Geneva*

3:15 **Multi-Commodity Two-Echelon Vehicle Routing Problem with Time Windows**

*Tom Van Woensel**

Eindhoven University of Technology

3:45 **The Vehicle Routing Problem with Private Fleet and Common Carrier: Extension and Exact Algorithm**

[1]*Said Dabia,* [1]*David Lai,* [2]*Daniele Vigo**

[1]*Dept. of Information, Logistics and Innovation-VU Amsterdam,* [2]*D.E.I. - Universita' di Bologna*

Multi-Modal Variations of the Vehicle Routing Problem

Marc-Antoine Coindreau[a], Olivier Gallay[a,*], Nicolas Zufferey[b]

[a] *Department of Operations, HEC, University of Lausanne, Switzerland*

[b] *Institute of Management, GSEM, University of Geneva, Switzerland*

Abstract

In this work, we extend the Vehicle Routing Problem formulation by proposing multi-modal variations of this well-established problem. For these new formulations, we empirically show that a streamline metaheuristic is able to highlight the potential benefit offered by the introduction of multi-modality.

Key words: Vehicle Routing, Multi-Modality, Dial-A-Ride-Problem.

1. Introduction

This study proposes a generalization of the static day-ahead *Vehicle Routing Problem with Time Windows*, which consists in finding the set of routes of minimum cost that visit a set of jobs spread on a territory. Each job has a given duration and a given time window. In the *Vehicle Routing Problem* (VRP) formulation, each worker moves from one job to another by driving her/his assigned car. In this work, multi-modality (MM) is possible as walking between jobs is allowed, and a car can transport multiple workers. These new features are expected to help reducing both the driving distance and the number of cars used, which are the two main objective functions minimized in the VRP literature [4], where usually, minimizing the number of vehicles has the largest priority. This generalization is called the *Multi-Modal Vehicle Routing Problem* (MMVRP).

This project – with a preliminary version exposed in [1] – is motivated by the network of a large energy provider (denoted as *ABC*: it cannot be named because of a non-disclosure agreement) in which the workers have to visit clients to achieve various tasks (e.g., evaluate consumptions, upgrade consumer settings). *ABC* has observed that their workers often leave their vehicle and perform clustered jobs on foot even if their given planning would recommend to drive to their next jobs. This situation is enhanced in an urban context, where distances between jobs can allow walking. Furthermore, when the parking spots are limited and when the traffic jam is dense, walking could help reducing the high uncertainty affecting the car travel times.

2. Literature Review

In the context of home-care staff scheduling, a formulation that shows some similarities with the introduced MMVRP is presented in [3], where nurses can walk between the patients locations. The main discrepancy with our approach lies in the fact that nurses are not allowed driving a car, and their transportation between jobs is ensured by an independent transportation system. Also, the main focus is put on the minimization of the number of vehicles without considering any tradeoff with the overall driving distance. In the present approach, the tradeoff between the reduction of vehicle and the total driving distance is analyzed.

Transporting the same worker along her/his schedule, with different vehicles, leads to a transportation problem of persons. This is tackled in the literature as the *Dial-A-Ride-Problem* (DARP) [6]. MM raises

*Corresponding author

Email addresses: marc-antoine.coindreau@unil.ch (Marc-Antoine Coindreau), olivier.gallay@unil.ch (Olivier Gallay), n.zufferey@unige.ch (Nicolas Zufferey)

additional complexity to the DARP. First, all the routes that transport the same worker along her/his path of jobs are interdependent, meaning that a delay in a route can potentially be propagated to all other routes. This extends the DARP with a temporal precedence constraint between pick-up and delivery pairs. Second, the set of pick-up and delivery locations is not given as an input but is part of the optimization process. The only dependencies in the DARP context can be found in the DARP with transfers, where transportation requests can be split into multiple routes. Dependencies appear explicitly at transfer points, as the pick-up time at transfers depends on the previous associated deposit time.

A need for synchronization also appears in the so-called *Vehicle Routing Problem with Trailers and Transshipment* (VRPTT) [2], where both autonomous (trucks) and non-autonomous (trailers) vehicles have to be synchronized in order to complete the routing needs. In the MMVRP framework, while workers driving their own car could be modeled as being autonomous, the workers without an assigned car cannot be considered as non-autonomous. Indeed, these workers are semi-autonomous, as they need to be transported for longer distances, whereas walking allows walking locally. Because of this important discrepancy, we cannot take advantage of the VRPTT literature for tackling our problems.

3. Problem Formulation

Consider a set J of n jobs, a set W of workers, and a set K of homogeneous vehicles of capacity Q (the number of workers that can be transported in the same car). Both workers and vehicles start and end their working day at a central depot 0. Like the VRP, the MMVRP is defined on a complete graph $G = (V, A)$, where $V = J \cup \{0\}$ represents the set of nodes and $A = \{(i, j) \mid i, j \in V, i \neq j\}$ represents the set of arcs. With each arc $(i, j) \in A$ is associated a driving time $\tau^d_{i,j}$, a walking time $\tau^w_{i,j}$ and a distance $d_{i,j}$. With each job $j \in J$ is associated a processing time p_j and a time window $[e_j, l_j]$. A worker is allowed waiting at a node in order to serve the associated job within its time window, and the daily walking distance of each worker is upper-bounded.

In a MM context, the number $|K|$ of cars can be lower than the number $|W|$ of workers. Therefore, the workers can be split into drivers and passengers. Drivers have to *(1)* perform their assigned jobs and *(2)* fulfill transportation requests of passengers. Both driver and passenger workers can walk to reach a job, but in the driver case, the return path to the car is mandatory. A walking path between consecutive jobs is called a *walking route* (WR). In a MM solution, the jobs can be partitioned into WRs, and the WRs must be partitioned in the workers planning. A WR is seen from the transportation point of view as a delivery (resp. pick-up) point where the WR starts (resp. ends). In the driver case, the pick-up and delivery points are the same. A WR can thus be agglomerated in two aggregated nodes (delivery and pick-up), the characteristics of which (aggregated time window and total duration) are computed according to the ordered set of jobs contained in the WR. The vehicles only visit these aggregated nodes, and the sequence of WRs in the workers planning gives the set of all pick-ups and deliveries to be satisfied.

Three degrees of multi-modality are considered, ranging from (F1) to (F3). The goal is to first minimize the amount of used resource (i.e., workers and vehicles), and next the total routing cost, which is directly proportional to the driving distance. For each instance, anytime a feasible solution is found, a new instance is generated by reducing either K or W by one unit (i.e., removing a car or a worker). The formulations are the following. **(F1)** VRP: each worker is a driver, and a worker can only move using her/his assigned car. **(F2)** MMVRP with $|K| = |W|$: while each worker has her/his assigned car, a worker can visit a set of jobs by walking as long as her/his maximal walking distance is not reached. **(F3)** MMVRP with $|K| < |W|$: we allow for the possibility to have fewer cars than workers, and thus a vehicle can transport multiples workers. Each worker can walk to reach a job as long as her/his maximal walking distance is not reached.

4. Solution Method: Large Neighborhood Search and Job Best-Insertion Algorithm

LNS [7] aims at improving the current solution s by iteratively unbuild and rebuild it. At each step, q (parameter) jobs are randomly removed from s and are then sequentially reinserted (in a best-insertion fashion, see below) in order to get a new solution s'. LNS is embedded into a simulated annealing algorithm

to decide with a certain probability whether to move (or not) the search from s to s'. In our implementation, W and K are given for each instance. At each step of the LNS, q is an integer randomly selected in $\{1, \ldots, 0.2 \cdot n\}$. In the simulated annealing, the initial temperature allows for a deterioration of 20% with a probability 0.5, and the cooling is set such that the final temperature does not allow any deterioration.

To find the best position for inserting a job, all the eligible insertion positions are exhaustively tested. When a job is inserted, different cases must be considered depending on the type of insertion performed (i.e., either in a driver or in a passenger planning). Each insertion type leads to an associated method. In a MM context, we first check whether the insertion can be performed by extending an existing WR, or if a new WR must be created for the job to be inserted. The feasibility of an extension can be easily tested by updating the involved aggregated nodes at their specific positions in the route. But when a new WR is created, the number of potential insertion positions quickly increases. Indeed, the new WR must first be inserted in a worker planning (driver or passenger), and then this insertion must be evaluated within the chosen route. In the driver case, the insertion is similar to the insertion in a VRP case (i.e., insert a node in a route). The number of tests to perform is in $O(n)$. In the passenger case, the insertion turns out to be more complicated. Assume that the new WR (denoted as ω_j) is introduced between WRs ω_i and ω_{i+1} in a passenger planning. The transportation between the pick-up $P(\omega_i)$ at the end of ω_i and the delivery $D(\omega_{i+1})$ at the beginning of ω_{i+1} becomes obsolete and must thus be removed from the partial solution. Consequently, two new transportation requests must be satisfied: $(P(\omega_i) \rightarrow D(\omega_j))$ and $(P(\omega_j) \rightarrow D(\omega_{i+1}))$. In this case, the number of tests to be performed is in $O(n^4)$. To tackle this large complexity, we adapt the fast feasibility-check procedure introduced in [5], which allows verifying in constant time if inserting two pick-up and delivery pairs is feasible. To further reduce the number of feasible insertions to test, necessary conditions and filters are used to focus on the most promising insertions. For instance, we only test the five positions i that minimize $d_{P(\omega_i),D(\omega_j)} + d_{P(\omega_j),D(\omega_{i+1})}$.

5. Computational Results

The instances are generated based on the real cases faced by *ABC*. The considered urban territory is modeled by a square grid of 10km $\times$ 10km, where the random locations of the n jobs are uniformly distributed. This situation obviously allows walking between some pairs of jobs. The duration of each job is randomly generated between 15 and 34 minutes, and the same time window [8am, 5:30pm] is considered for each job (in order to enrich the resulting solution space). In an urban context, the average vehicle speed is set to 30km/h, and the walking speed to 4km/h. The upper bound on the daily walking distance is set to 5km, and each vehicle can transport $Q = 2$ workers. The time limit of the LNS is n minutes (i.e., it is proportional to the instance size). For each considered $n \in \{25, 45, 65, 85\}$, five instances are generated (with different job configurations on the grid), on which 10 LNS runs are performed. The used computer is a 3.4 GHz Intel Quad-core i7 with 8 GB DDR3 of RAM memory.

The results are presented in Table 1, where the above (resp. below) part represents formulation (F2) (resp. (F3)). First, the instance characteristics are given, namely $n, |W|$ and $|K|$. It means that for each instance, the available resource W and K is known. Therefore, only the transportation costs have to be minimized. The value of $|W|$ is the smallest obtained feasible value for (F1) with respect to n. Next, the average (over the five configurations associated with each triplet $(n, |W|, |K|)$, and over the ten runs) percentage gap is given with respect to the (F1) solution values. For instance, in the case $(n, |W|) = (25, 2)$: *(1)* no feasible solution was found for (F1) with $|K| = |W| = 1$; *(2)* formulation (F2) reduces the solution values by 6.5%; *(3)* formulation (F3) leads to a 2.6%-augmentation of the transportation costs but one vehicle is saved. Finally, in the last column, the same information (i.e., gaps) is provided, but for the most clustered configuration (as it allows better highlighting the MM potential benefit). Consider the fifth column involving all the configurations. On the one hand, for formulation (F2), the transportation cost improvement (versus the classical VRP formulation (F1)) increases with n, which confirms the benefit of allowing walking routes. On the other hand, considering formulation (F3) (again versus (F1)), one vehicle can be saved if additional transportation costs are encountered. The transportation costs degradation augments with n, which probably indicates the increasing complexity of larger problems. It however opens the door to interesting tradeoffs

between the transportation costs and the total used resource. Indeed, company ABC is likely to accept a reasonable augmentation of the transportation costs if a vehicle can be saved. When considering the sixth column, the same trend is observed for formulation (F2) versus (F1). Interestingly, it appears that formulation (F3) can simultaneously reduce the number of vehicles as well as the transportation costs! The sixth column of Table 1 gives the average value obtained for the 10 runs performed on the most clustered configuration. For formulation (F3), we observe a decreasing quality of the solutions when the instance size increases. We conjecture that this is more a consequence of the limited performance of the used basic metaheuristic than of the capacity of multi-modality to reduce the driving costs. Indeed, the complexity of managing passenger workers increases drastically with the instance size, which directly impacts the efficiency of the solution method and its robustness.

Table 1: VRP versus MMVRP

Formulation	n	$\|W\|$	$\|K\|$	% gap (5 configurations)	% gap (most clustered configuration)
F2	25	2	2	-6.5%	-7.4%
	45	3	3	-8.7%	-7.5%
	65	4	4	-13%	-12.5%
	85	5	5	-11.2%	-9.3%
F3	25	2	1	2.6%	-2.3%
	45	3	2	14.9%	-4.7%
	65	4	3	18.9%	-0.2%
	85	5	4	42.9%	22%

Table 2 compares the average value, the best value and the standard deviation σ obtained over the 10 runs, for the most clustered configuration with $n = 45$. The σ-value is low for (F1) and (F2), which is likely to indicate efficient and robust solutions. The larger σ-value for (F3) highlights the additional complexity and the need for a more specialized metaheuristic. Considering the best solutions (column 6), we have observed that increasing multi-modality can lead to both a shorter driving distance (the improvement gap of (F3) can be up to 10.8% with respect to (F1) and up to 3.6% with respect to (F2)) and a reduced fleet of vehicles (one car is saved). These three best solutions are shown in Figure 1. We can graphically observe that (F2) allows decreasing the driving distance when compared to (F1), as some trips are walked instead of driven. The driving distance can be further decreased in formulation (F3). Indeed, the detours generated to drop (resp. pick-up) the passenger worker at job 15 (resp. job 5) are overbalanced by the gain obtained by the simultaneous presence of two workers in the same car for part of the routing.

Table 2: Detailed results for the most clustered configuration when $n = 45$

Formulation	n	$\|W\|$	$\|K\|$	average distance (km)	best found (km)	σ
F1	45	3	3	61.79	61.71	0.19
F2	45	3	3	57.15	57.08	0.09
F3	45	3	2	58.91	55.04	3.31

6. Conclusions and Future Works

In this study, a first step is performed in multi-modal vehicle routing, motivated by a real situation. It was numerically shown that using two travel modes (namely by car and on foot) instead of only one can be beneficial in terms of the total transportation costs and according to the employed resource (for instance, a reduced fleet of vehicles). A conservative case was however considered here, as congestion and parking spots limitation are not taken into account, and as the density of jobs is rather small (less than 1 job per km^2). Our observations might thus be amplified in such situations. Following these encouraging results, upcoming effort will be devoted to the development of a set of efficient metaheuristics for larger instances. Another avenue of research consists in considering additional transportation modes for the involved workers, for example taxi service or an external transportation company, particularly if it allows reducing the fleet of vehicles at a reasonable additional cost.

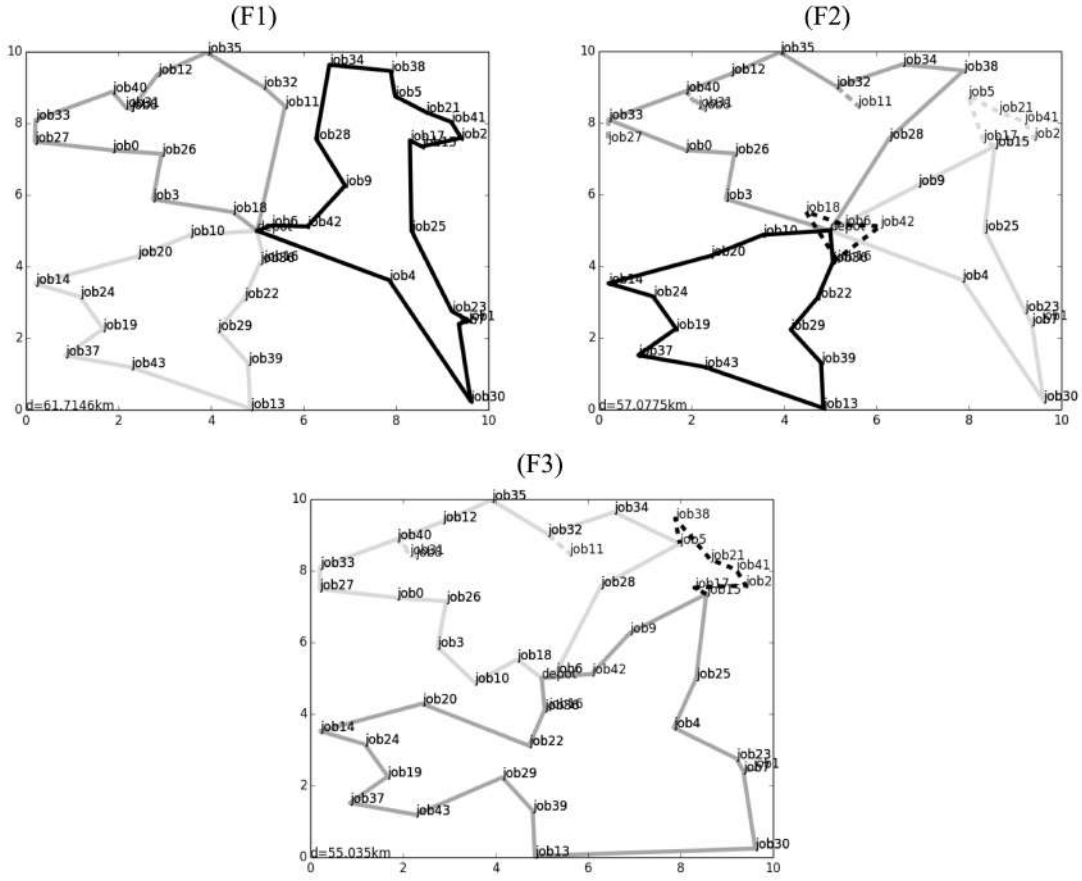


Figure 1: Best solutions for formulations (F1), (F2) and (F3) for the most clustered configuration with $n = 45$. Dashed lines correspond to walking paths, whereas solid lines indicate travels by cars.

References

[1] M.-A. Coindreau, O. Gallay, and N. Zufferey. Vehicle Routing with Multi-Modality: A Practical Application. In *Proceedings of the 18th Annual Congress of the French Operations Research Society (ROADEF 2017)*, Metz, France, February 2017.

[2] M. Drexl. Synchronization in vehicle routing - a survey of vrps with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012.

[3] C. Fikar and P. Hirsch. A matheuristic for routing real-world home service transport systems facilitating walking. *Journal of Cleaner Production*, 105:300–310, 2015.

[4] C. Koc, T. Bektas, O. Jabali, and G. Laporte. Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research*, 249(1):1–21, 2016.

[5] R. Masson, F. Lehuédé, and O. Péton. Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. *Operations Research Letters*, 41(3):211–215, 2013.

[6] R. Masson, F. Lehuédé, and O. Péton. The dial-a-ride problem with transfers. *Computers and Operations Research*, 41(1):12–23, 2014.

[7] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *Lecture Notes in Computer Science*, 1520:417–431, 1998.

Multi-Commodity Two-Echelon Vehicle Routing Problem with Time Windows

F. Dashti Saridarq, N. Dellaert, T. Van Woensel and T.G. Crainic

The two-echelon vehicle routing problem (2E-VRP) is a two-echelon distribution system where goods are transferred to customers by using intermediate facilities (called satellites). There are three types of facilities that are located in a hierarchical structure: (i) Depots act as the sources of goods and are located at the first level; (ii) Satellites are used to consolidate goods and are located at the second level; (iii) Final customers are located at the last level and are destinations of the goods. The first echelon of the 2E-VRP consists of vehicle routes where each route starts (and ends) at a depot and delivers a subset of goods to a subset of satellites. The second echelon consists of vehicle routes where each route starts and ends at the same satellite and delivers goods to their associated customers.

Most of the existing studies in the field of 2E-VRP have focused on the basic variant of the problem without origin-destination structure for demands, i.e, all demands are considered fully substitutable. In practice, this assumption is not realistic. Clearly, solutions derived from existing models to the 2E-VRP could be infeasible when implemented in real life. For example in parcel delivery application, each parcel should be picked up from a specific origin and be delivered to a specific customer destination and thus the parcels are non-substitutable.

The problem under study is particularly motivated by two-echelon city logistics systems where the loads of different shippers are consolidated and distributed to customers through intermediate facilities. Therefore, the decision makers usually deal with origin-destination demands where each demand starts from a specific depot and is supposed to be delivered to a specific customer. A demand is a unique load (a specific parcel or package). We take non-substitutable demands into account by introducing commodities: a *commodity* consists of the destination customer, origin depot, a specific volume, and a time window which the delivery should take place within. The described problem is referred as the *multi-commodity two-echelon vehicle routing problem with time windows* (MC-2E-VRPTW). Although some papers introduced the concept of non-substitutable demand [1], this is the first paper that considers non-substitutable origin-destination demands for the 2E-VRP.

The objective of this paper is to introduce the MC-2E-VRPTW and develop exact solution methods. The problem is defined and mathematically formulated as a mixed integer programming problem. Specifically, a combined arc-and-path based formulation is presented which employs arc-flow variables for the first

echelon and path variables for the second echelon. A branch-and-price algorithm is presented based on this specific formulation. Throughout this paper, we refer to the first and second echelon vehicles as urban vehicles and city freighters respectively using the terminology proposed by [2].

To sum up, the *contributions* of this paper are as follows:

1. The literature on 2E-VRP has only focused on the single commodity variant of the problem which does not represent the non-substitutable demands arises in city logistics. We introduce the MC-2E-VRPTW which considers origin-destination structure for demands by employing multiple commodities. We investigate the problem from an exact optimization point of view and propose mathematical formulations and a branch-and-price algorithm. Specifically:
   (a) A combined arc-and-path based formulation is designed where arc-flow variables are used for the first echelon and path variables are used for the second echelon (denoted as the multi-commodity two-echelon combined arc-and-path based formulation, or *MC2E-A-P*).
   (b) A branch-and-price algorithm is proposed which works on this formulation. A column generation method and specific branching strategies are introduced for the proposed algorithm.
2. The numerical evaluation through a large set of instances demonstrates the power of the newly developed model and the respective solution method. We study how large instances can be solved using the proposed branch-and-price algorithm for the arc-path-based formulation.

1 Problem description

The MC-2E-VRPTW deals with multiple commodities. Each commodity (a demand request of a specific customer which should be transfered from a specific depot) should be shipped employing an urban vehicle (first echelon vehicle) and a city freighter (second echelon vehicle) which are connected through a satellite. Without loss of generality, we assume that each commodity is coupled with a customer. Note that, a customer demanding multiple commodities could be represented by multiple copies of the same customer as we do not consider split delivery. Therefore, each customer has a demand of one commodity which should be transfered from a particular depot without any preference for the satellite. Such a demand request consists of the size of the demand and also a hard time windows (the delivery of the commodity is not allowed either before the start of the window or after the end of this window).

We assume that these commodities are available at the start of the planning horizon. Thus, no service time is considered for depots. Satellites can handle any commodity without any constraint on the number of arriving urban vehicles or departing city freighters or the amount of the freight. There is no explicit time window for satellites. Each satellite is open and ready to service during the

planning horizon without any fixed or variable cost. A service time is considered for each satellite where the de-consolidation and consolidation should occur within this period. In a similar way, a service time is required to unload the commodity from a city freighter and deliver it to a customer.

De-consolidation and consolidation activities requires a connection and synchronization between the urban vehicle and the city freighter which are used to transfer the same commodity: (i) Connection: the urban vehicle should deliver the commodity to the starting satellite of the the city freighter; and (ii) Synchronization: the departure time of the city freighter should be greater than the arrival time of the urban vehicle to the satellite (clearly, a city freighter which is transferring a subset of commodities should wait for the arrival of all urban vehicles which brings at least on of these commodities).

A solution for the problem consists of the first echelon routing decisions and the second echelon routing decisions such that these two echelon routings are connected and synchronized. Therefore, an optimal solution of the problem consists of the vehicle tours of both echelons such that each commodity is delivered to its associated customer through a satellite, while the time windows of customers are respected and the total transportation costs are minimized. Figure 1 shows an example of the problem with 3 depots $\{A, B, C\}$, 4 satellites $\{i, ii, iii, iv\}$ and 8 customers $\{1, ..., 8\}$. A possible feasible solution is shown, which consists of three urban vehicle tours and five city freighter tours. An example of the de-consolidation and consolidation activities can be observed at satellite (ii) where the commodities of customers $\{2, 3, 4, 5\}$ delivered by two urban vehicle tours, originating from depots A and B, are de-consolidated and consolidated to two city freighter tours to finally deliver them to the customers.

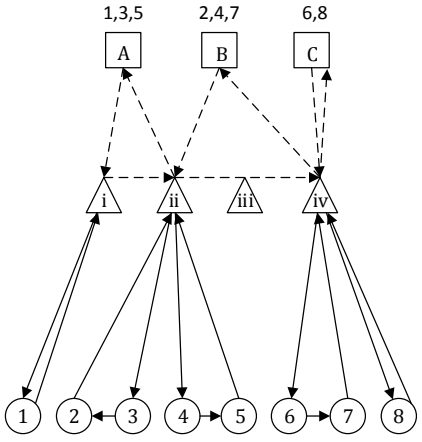


Figure 1: A MC2E-CVRPTW instance

2 Mathematical formulation

There are two extremes in modeling routing problems: (i) The arc-based models use arc-flow variables to present the flow on each arc and employs constraints to build route; (ii) The path-based models use path-flow variables to present the routing. We use a middle-ground approach to propose a combined arc-path-based model which benefits from advantages of both extremes. Particularly, it employs arc-flow variables for the first echelon routing problem and path variables for the second echelon routing problem. Note that, in practice, the number of satellites and depots is usually less than the number of customers. Therefore, the number of constraints to represent urban vehicle (first echelon vehicles) tours is not very huge. The full formulation will be discussed in the presentation and the full paper.

3 Branch-and-price

The proposed algorithm is based on a column generation method (see [4] for an overview) which computes the lower bounds. At each node of a branch-and-bound tree, a restricted master problem (RMP) is achieved by replacing the set of city freighter tours L by a subset $L' \subset L$ at the LP-relaxation of the MC2E-A-P formulation. However, we insert all first echelon routing arc variables, second echelon timing variables and the commodity-satellite assignment variables to the model. Two heuristics and one exact algorithm are used to solve a pricing problem to generate negative reduced cost city freighter tours.

We use multiple branching strategies to deal with fractional values of the decision variables. These branching strategies can be categorized in three main categories:

- First echelon routing variables,
- Commodity-satellite assignment variables,
- Second echelon routing variables (number of selected city freighter tours and second echelon arc)

4 Computational study

Detailed results and insights based on an extensive instance set (inspired based on [3]) will be provided on the conference and in the full paper.

References

[1] T. G. Crainic, N. Ricciardi, and G. Storchi. Models for evaluating and planning city logistics systems. *Transportation Sci.*, 43(4):432–454, 2009.

[2] T. G. Crainic, A. Sforza, and C. Sterle. *Location-routing models for two-echelon freight distribution system design.* Technical report CIRRELT-2011-40, CIRRELT, Montréal, 2011.

[3] N. Dellaert, F. D. Saridarq, T. Van Woensel, and T. G. Crainic. *Branch & Price Based Algorithms for the Two-Echelon Vehicle Routing Problem with Time Windows.* Technical report CIRRELT 2016 45, CIRRELT, Montreal, 2016.

[4] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Oper. Res.*, 53(6):1007–1023, 2005.

An Exact Algorithm for the Vehicle Routing Problem with Private Fleet and Common Carrier

Said Dabia[1], David Lai[1] and Daniele Vigo[1,2]

[1] Faculty of Economics and Business Administration
Vrije Universiteit Amsterdam, Netherlands
[2] Department of Electric, Electronic and Information Engineering
Alma Mater University of Bologna, Italy

Abstract

The Vehicle Routing Problem with Private Fleet and Common Carrier (VRPPC) is a generalization of the classical Vehicle Routing Problem where the owner of a private fleet can either visit a customer with one of his vehicles or assign the customer to a common carrier. The latter case occurs if the demand exceeds the total capacity of the private fleet or if it is more economically convenient to do so. The owners objective is to minimize the variable and fixed costs for operating his fleet plus the total costs charged by the common carrier. This family of problems has many practical applications, particularly in the design of last mile distribution services, and has received some attention in the literature, where some heuristics were proposed. We present an exact approach based on a branch-and-cut-and-price algorithm for the VRPPC and for a more general and practical case where the cost charged by the external common carrier is based on cost structures inspired from practice.

Keywords: Vehicle Routing, Exact Algorithms, Private Fleet and Common Carriers.

1 Introduction

We consider the Vehicle Routing Problem with Private Fleet and Common Carrier (VRPPC) which is a generalization of the classical Vehicle Routing Problem where the dispatcher may either serve the customers by using the vehicles of the owned fleet (called the *private fleet*) or assign them to a common carrier, e.g. a third-party logistics provider. The latter case occurs when either the total customers' demand exceeds the capacity of the private fleet or if it is more economically convenient to do so, e.g. because the customer is isolated and far from the private fleet depot.

This type of problems has many practical applications, particularly in the design of last-mile distribution services where outsourcing of unprofitable services are frequently considered options, and has received some attention in the recent literature. Chu [2005] considered a single depot routing problem with outsourcing options which can be considered the first paper on VRPPC. The problem considers a private fleet of vehicles with limited capacity and fixed costs associated with their use. A set of customers with known demand can be served by the private fleet which incurs travel costs as in the standard VRP. In addition, customers may be outsourced to a common carrier for which only a fixed cost per customer has to be paid. The objective is to minimize the total costs of the carrier involving fixed costs for vehicles, variable travel costs and fixed costs for orders performed by the common carrier. Chu [2005] proposed a heuristic solution method, consisting of a modified savings algorithm [see Clarke and Wright, 1964] and a simple improvement phase, that was tested on 5 instances. Bolduc et al. [2008] showed that the VRPPC can be modeled as an heterogeneous VRP, and presented a metaheuristic based on a perturbation procedure, substantially improving the results of Chu [2005]. In addition, they provided results of their algorithm for two new benchmark sets, one with homogeneous and one with heterogeneous vehicle fleet, with up to 480 customers. A tabu search heuristic for the VRPPC, which outperformed the approach of Bolduc et al. [2008] for the case of homogeneous vehicles was presented in Côté and Potvin [2009]. Furthermore, Potvin and Naud [2010] proposed a tabu search heuristic with ejection chains that further improved results on both homogeneous and heterogeneous instances, at the expense of significantly larger total computing times. More recently, Stenger et al. [2013] introduced a multiple-depot version of the problem denoted as MDVRPPC, for which they defined a variable neighborhood search algorithm. The resulting algorithm was tested on a benchmark set of instances derived from *Multi-Depot VRP* (MDVRP) ones showing the potential benefits associated with subcontracting. The algorithm is also capable of obtaining state-of-the-art results on both the single depot VRPPC. To the best of our knowledge no exact algorithm has been proposed so far for the VRPPC or its variants. The problem under study is also related to the wide family of VRP with Profits: we refer the interested reader to the survey of **?** for an overview of the literature in this field.

We present here an exact approach based on a branch-and-cut-and-price algorithm for the VRPPC and for a more general and practical case where time windows and heterogeneous fleet are present and the cost charged by the external common carrier is based on cost structures inspired from practice. We call for short our problem *Rich* VRPPC (RVRPPC) because it contains as special cases the known variants of VRPPC with single depot. In the following we outline the solution approach and the results of a computational testing.

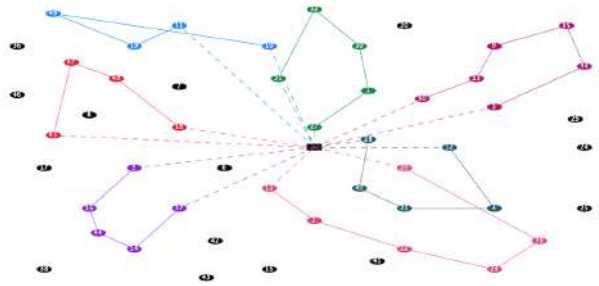
2 Problem Definition and Solution Approach

We consider a directed graph $G(V', A)$, where V' is a set of vertices corresponding to the customers and the depot (vertex 0), and A is the set of all feasible arcs. For all $i \in V'$ let d_i, TW_i and s_i denote the demand, time window and service time, respectively. For all $(i, j) \in A$: c_{ij} and τ_{ij} are traveling cost and time from i to j. We also consider a limited heterogeneous private fleet of vehicles in which each vehicle $k \in K$ has finite capacity Q_k and fixed cost f_k, whereas the common carrier charges a cost based on the total quantity subcontracted. More precisely, the subcontracting cost is expressed by a piecewise linear function $c(d)$ of the total demand d of the customers to be subcontracted, hence the cost of outsourcing an order is not known in advance.

We define a Branch-and-Cut-and-Price approach based on a set-partitioning formulation as commonly done for the VRP. Because of the special structure of the problem the algorithm incorporates separate dedicate pricing procedures for the private fleet routes and for the subcontracted service. The first one is based on the solution of Resource-Constrained Shortest Paths with the ng-path relaxation, while the second pricing problem calls for the solution of Generalized Knapsack Problems with variable capacity. The formulation is strengthened through valid inequalities and specialized branching strategies are developed. Finally, a second set-partitioning formulation is developed in which the outsourcing option is implicitly modeled so as to reduce the symmetry of the formulation.

3 Computational Testing

Figure 1: Solution of instance R102.50



The algorithm is tested on a set of instances derived from the classical ones

from the literature both uniformly distributed and with clustered structures. As expected the results show a positive impact of the introduction of the outsourcing option, both in terms of overall cost savings and of the productivity of the private fleet. Such impact is larger on randomly distributed instances but is interesting also in clustered ones. An example of solution for a 50 customers instance is depicted in Figure 1 where it is clearly visible that the subcontracted customers are mostly peripheral and relatively isolated. An extensive testing of the sensitivity with respect to the relative ratio between the private fleet service costs and the subcontracting costs.

References

M.-C. Bolduc, J. Renaud, F. Boctor, and G. Laporte. A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers. *International Journal of the Operational Research Society*, 59(6):776 – 787, 2008.

C.-W. Chu. A heuristic algorithm for the truckload and less-than-truckload problem. *European Journal of Operational Research*, 165(3):657 – 667, 2005.

G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568 – 581, 1964.

J.-F. Côté and J.-Y. Potvin. A tabu search heuristic for the vehicle routing problem with private fleet and common carrier. *European Journal of Operational Research*, 198(2):464 – 469, 2009.

J.-Y. Potvin and M.-A. Naud. Tabu search with ejection chains for the vehicle routing problem with private fleet and common carrier. *to appear in: International Journal of the Operational Research Society*, 2010. doi: doi:10.1057/jors.2010.102.

A. Stenger, D. Vigo, S. Enz, and M. Schwind. A variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. *Transportation Science*, 47:64–80, 2013.

Vehicle Routing Models & Applications

TD3: Dynamic Routing

Thursday 4:30 – 6:00 PM

Session Chair: Barrett Thomas

4:30 **Route-Based Markov Decision Processes for Dynamic Vehicle Routing Problems**

[1]*Justin Goodson**, [2]*Marlin Ulmer*, [2]*Dirk Mattfeld*, [3]*Barrett Thomas*

[1]*Saint Louis University*, [2]*Technische Universität Braunschweig*, [3]*Department of Management Sciences-University of Iowa*

5:00 **Scalable Anticipatory Policies for the Dynamic and Stochastic Pickup and Delivery Problem**

*Gianpaolo Ghiani, Emanuele Manni**, *Alessandro Romano*

Department of Engineering - University of Salento

5:30 **Enough Waiting for the Cable Guy - Estimating Arrival Times for Service Vehicle Routing**

[1]*Barrett Thomas**, [2]*Marlin Ulmer*

[1]*Department of Management Sciences-University of Iowa*, [2]*Business Information Systems-Technische Universitat Braunschweig*

Route-Based Markov Decision Processes for Dynamic Vehicle Routing Problems

Justin C. Goodson
St. Louis University, St. Louis, MO, USA

Marlin W. Ulmer, Dirk C. Mattfeld
Technische Universität Braunschweig, Braunschweig, Germany

Barrett W. Thomas
University of Iowa, Iowa City, IA, USA

1 Introduction

Dynamic routing problems are problems in which a set of geographically dispersed customers is visited by one or more travelers or vehicles. The problems are dynamic in that problem information changes over the horizon of the problem and there exist opportunities to make decisions in response to the revealed information. While dynamic vehicle routing has a 30 year history in the operations research literature, the majority of research in the field has been published in only the last few years. For example, 51 out of a total of 117 dynamic routing papers cited in Psaraftis et al. (2016) have been published since 2012. This trend is likely to accelerate. Recent advances in data availability and computing power offer increasing opportunities as well as accompanying challenges for integrating predictive tools with prescriptive optimization methods to anticipate and dynamically respond to uncertain events.

A good starting point is a unified framework for modeling dynamic routing problems. A unified modeling framework allows researchers to express their problems in a common language that allows for better identifying contrasts with existing work and also opportunities to derive new work from the existing. In many disciplines, Markov decision processes (MDPs) are the natural modeling language for stochastic and dynamic problems. Powell et al. (2012) argues that MDPs also offer a natural modeling framework for such problems in transportation, including routing problems.

Traditionally, however, MDPs model decision making that is made sequentially in response to new information. Rather, MDPs for dynamic routing problems tend to model actions in terms of the next customer to visit. Yet, while routes are executed sequentially, or customer by customer, the vast majority of the routing literature develops plans for future visits and uses the plans to guide the execution. While such solution approaches are natural for deterministic problems, many approaches to dynamic problems carry the concept of a tentative *route plan* into the dynamic problem, even though such route plans are not necessarily fixed.

In part, maintaining route plan in approaches to dynamic routing problems is natural. For one, the

research community is conditioned to think about the solutions to even dynamic routing problems as routes, even if such routes will change as new information becomes available. Second, with the rich literature devoted to deterministic routing, it is natural that we would want to use the solution approaches originally developed for deterministic problems, which operate on routes, in the solution of dynamic problems. Thus, in the same way that a set covering formulation supports a column generation approach for a deterministic vehicle routing problem, our models for dynamic problems should support the chosen solution approach as well. Yet, conventional MDPs do not naturally model the evolution of a route. Thus, there is a disconnect between our models for dynamic routing problems and the solutions approaches.

In this talk, we present a unified modeling framework for dynamic and stochastic vehicle routing problems (DVRP) that allows authors to model the evolution of routes. The framework transforms traditional MDPs into route-based MDPs by storing information about the planned routes. The route-based MDP model augments the MDP actions to the selection of updated routes. Given the notion of planned routes, the current period reward (cost) is redefined as a marginal reward to account for changes in the plan. The route-based formulation allows the MDP model to reflect that solution approaches often make use of planned routes.

This talk makes the following contributions. First, it presents an MDP model for dynamic routing that is based on traditional routing plans. Second, we prove the equivalence of the two formulations. Third, we present an example of a DVRPs from the literature and shows how it can be modeled using the route-based formulation. We also demonstrate how the route-based formulation aligns the solution approach to the route-based MDP.

2 Illustrative Example

For this purposes of this abstract, we use illustrations to demonstrate the differences between conventional and route-based MDP models. The purpose of the illustrations is to give readers, particularly those unfamiliar with MDPs, a conceptual understanding of the two models. In subsequent sections we will provide formal notation for both conventional MDPs and route-based MDPs.

For concreteness, we use the *vehicle routing problem with stochastic service requests* (VRPSSR) as a framework for our illustration. The VRPSSR is a DVRP, and variants of the VRPSSR are often considered in the literature. Our VRPSSR is characterized by the need to dynamically route one uncapacitated vehicle to meet service calls arriving randomly over a working day of duration T and from a set of potential customers. The vehicle serves customers as long as time permits. We denote the known customer locations by the set $\mathcal{N} = \{0, 1, \ldots, N\}$, where 0 represents a depot and the remaining locations represent customers. Although the location of each customer in $\mathcal{N}$ is known, whether or not a customer requests service is uncertain. The known travel time between two locations n and n' in $\mathcal{N}$ is denoted $d(n, n')$. For each customer served, we earn a reward of 1. Requests not serviced by the vehicle are visited via a third party. The objective is to maximize the expected number of serviced customers.

The following subsections illustrate the conventional MDP for the VRPSSR and the route-based MDP, respectively. Each illustration has three parts and illustrates a single decision making episode using each MDP model. The first part, the left-hand side of each illustration, illustrates the state of the system

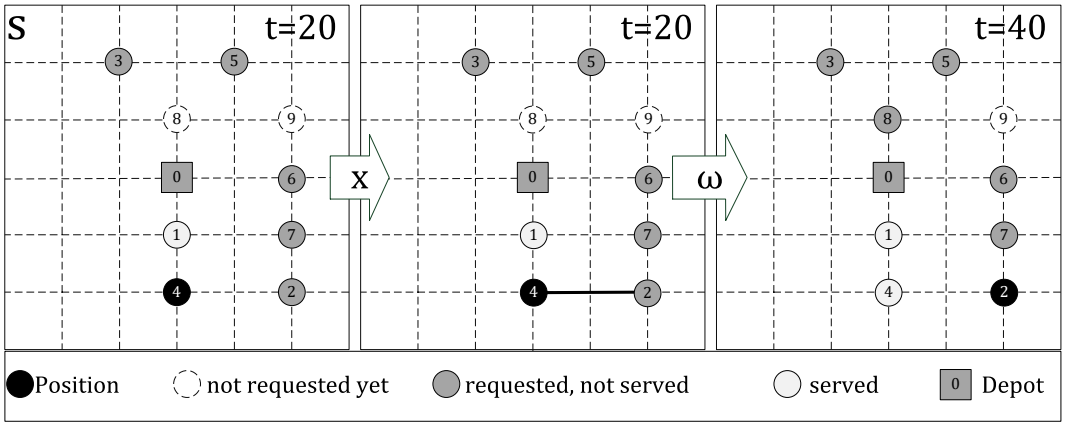


Figure 1: Illustration of a Conventional MDP

at a point in time in which a decision is to be made. The state of an MDP is a set of information that represents all of the information that is necessary for the decision maker to make an optimal decision. In the second panel of each illustration, we illustrate the decision that is made given the state of the system shown in the first panel. Finally, the panel on the right-hand side of each illustration demonstrates a new state of the system that results from the observation of new exogenous information, which in this case is the realization of new customer requests.

2.1 Conventional MDP Illustration

Figure 1 illustrates the conventional MDP model. We consider a situation in which there are nine known customers. The customers are depicted as 1 through 9 in each of the three panels of Figure 1. The left-most panel of Figure 1 shows the situation at time 20. At this time, the vehicle is located at Customer 4, customers 8 and 9 have not requested service, customers 2, 3, 5, 6, and 7 have requested service but have not yet been visited, and Customer 1 has already received service. The time, the vehicle location, and the customer statuses constitute the state of the system. With the information given in the state, we make a decision. For the purposes of this example, we assume that we choose to travel to Customer 2, a customer who has requested service but who has not yet been visited. We will refer to this decision as $x = 2$. We represent this decision by a solid line connecting the current vehicle location at Customer 4 and the next location at Customer 2. To execute the decision, the vehicle travels to Customer 2. We assume that the vehicle traverses a Manhattan-style grid where each edge requires 10 time units. We illustrate the arrival to Customer 2 at time 40 in the panel of the right-hand side of Figure 1. At this time, we also observe any new requests, the random information ω, that occurred between the departure form Customer 4 at time 20 and the arrival to Customer 2 at time 40. At this point, we have a new state where the vehicle is located at Customer 2, Customer 4 has now been visited, and Customer 8 has just requested service but has not yet been served. Customers 3, 5, 6, and 7 have also requested service but have not yet been visited, and Customer 1 has already received service.

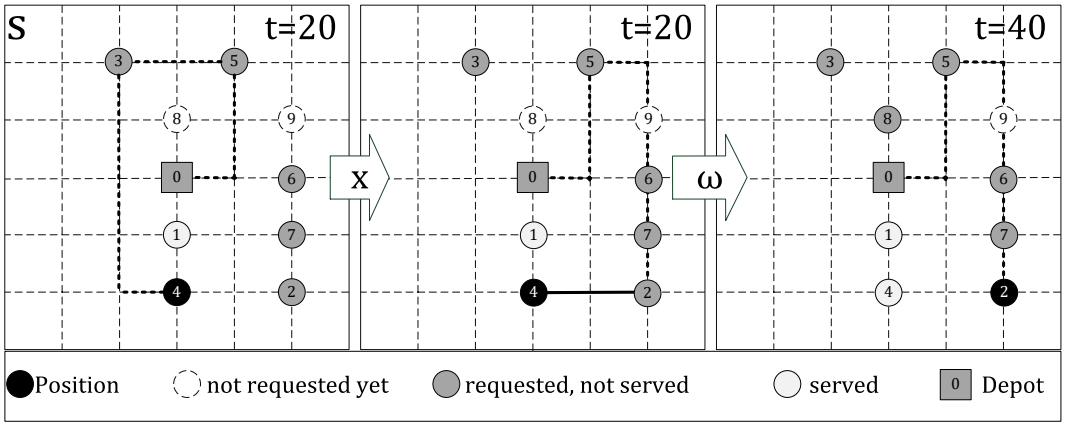


Figure 2: Illustration of a Route-Based MDP

2.2 Route-based MDP Illustration

Looking at the illustration given in Figure 1, we can observe a gap between the problem formulation and the way that most practitioners as well as almost all of the solution approaches in the literature view the problem. Notably, Figure 1 does not present a route, even only a planned route through known customers. Almost all of the literature for the VRPSSR apply approaches that, at each decision epoch do not merely decide about the next customer to visit, but also determine a set planned of routes. These plans reflect planned future steps. For the VRPSSR, a plan usually contains a sequence of unserved customers ending in the depot. The plan then induces the actual routing action. In a subsequent decision epoch, the remainder of the former plan is updated.

The goal of the route-based MDP model is to close the gap. Figure 2 presents the analogous illustration as Figure 1, but using a route-based MDP. The left-most panel of Figure 2 shows the same situation at time 20 as Figure 1. However, it also illustrates the additional piece of information, the route, associated with the state of the route-based MDP model. At the current state at time 20, the proposed route travels from the current location of the vehicle at Customer 4 to Customer 3, then Customer 5, and back to the depot. Despite the planned tour depicted in the leftmost panel of Figure 2, the decision in the route-based case is also to travel to Customer 2. This decision is depicted by the solid line from Customer 4 to Customer 2 in the middle panel of Figure 2. In contrast to the analogous panel in Figure 1, the middle panel of Figure 2 also depicts the updated routes associated with the route-based MDP. With the choice to go from Customer 4 to Customer 2, the planned route continues from Customer 2 to Customers 7, 6, and 5 before returning to the depot. While Customer 9 has not yet requested service, we note that this new planned route puts the vehicle in a position to serve Customer 9 if Customer 9 does request service. Finally, the panel on the right-hand side of Figure 2 shows the state of the route-based MDP at time 40 when the vehicle arrives at Customer 2. At this point, Customer 8 has requested service and the planned route through customers 7, 6, and 5 before returning to the depot is shown.

In both Figures 1 and 2, the immediate action, that of traveling to Customer 2, is the same. However, Figure 2 combines the action with a route plan. This route plan allows the route-based formulation to connect with the solution approaches in much of the related literature. Further, as demonstrated by the

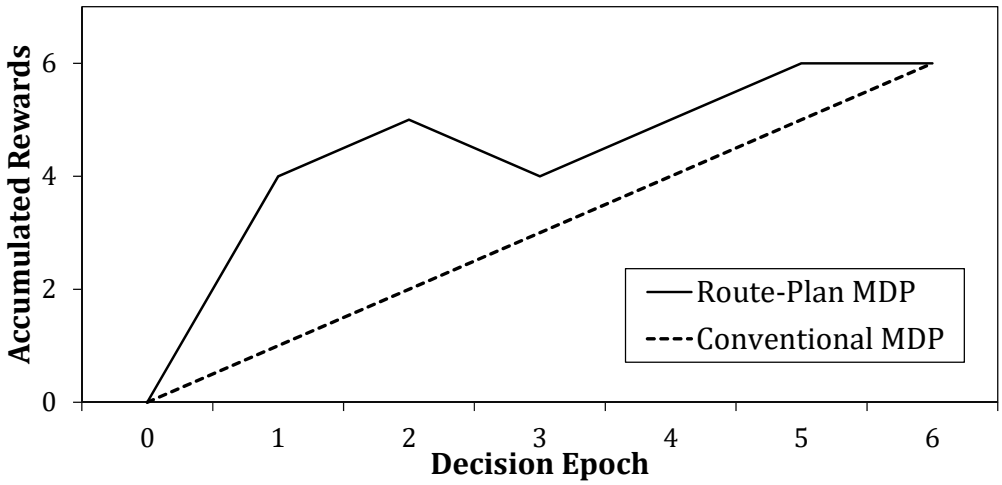


Figure 3: Accumulated Reward Over Time for Route-Plan versus Conventional MDP

route plan and Customer 9 in Figure 2, the route plans can be an indicator of the potential impact of future realizations.

A key feature of the route-based model is the definition of the reward. We can incorporate the value of the route plans into the reward by redefining the reward as a marginal reward. This marginal reward accounts for the difference in value between the previous plan and the new plan as well as immediate contributions by the current period action. Importantly, this redefinition of the reward allows us to maintain an equivalence between the two models. Figure 3 shows that the accumulated rewards between the two models are the same at the end of the horizon. Thus, the two models are equivalent. The talk will present a formal proof of this result.

References

Warren B Powell, Hugo P Simao, and Belgacem Bouzaiene-Ayari. Approximate dynamic programming in transportation and logistics: a unified framework. *EURO Journal on Transportation and Logistics*, 1(3): 237–284, 2012.

Harilaos N Psaraftis, Min Wen, and Christos A Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016.

Scalable anticipatory policies for the dynamic and stochastic pickup and delivery problem

Gianpaolo Ghiani[1], Emanuele Manni[1,*], Alessandro Romano[1]

[1]Dipartimento di Ingegneria dell'Innovazione, Università del Salento
via per Monteroni, 73100 Lecce, Italy
{gianpaolo.ghiani, emanuele.manni, alessandro.romano}@unisalento.it

*Corresponding author

1. Introduction

In this paper, we deal with the dynamic and stochastic Pickup and Delivery Problem (PDP) in which a fleet of vehicles must service a set of customers' requests characterized by a pickup and a delivery location, as well as by a priority class. The goal is to maximize the overall customer service level or, equivalently, to minimize customer inconvenience. The problem is dynamic in that customers' requests are disclosed during the planning horizon, whereas it is stochastic because we assume that the requests arrive according to a known stochastic process. This type of problem occurs in several sectors (e.g., in the couriers industry or in emergency systems). Nowadays, the huge advances in communication and information technologies allow to obtain in real-time data on vehicles locations and customers requests. This continuous flow of data allows the dispatchers to modify the vehicle routes in real time. At each stage it is important to make accurate decisions, since a bad choice in the present might affect the ability to make good decisions in the future. For this class of problems two kinds of policies are common in the literature: (i) reactive policies which manage new requests only once they have occurred, neglecting any available stochastic information; (ii) anticipatory algorithms which exploit the stochastic characterization of future demand, in an attempt of anticipating it, to provide the highest possible quality of service. Reactive policies are typically characterized by extremely fast running times, which are obtained at the expenses of solution quality, because of the myopic choices made at each stage. On the other hand, anticipatory procedures generally achieve better results than reactive algorithms, but with longer running times because of the need to simulate multiple scenarios. Detailed surveys can be found in [4] and [5]. In this paper we propose two dispatching policies that have a twofold goal. Trading off between the two extremes previously described, the first aim is to match the quality of anticipatory algorithms with a computational effort comparable to that of reactive approaches. The second goal is to achieve scalable performances, meant as the ability to easily deal with instances of increasing size.

The remainder of the paper is organized as follows. In Section 2 we describe the problem and in Section 3 we present our dispatching policies, whereas in Section 4 we illustrate our computational experiments.

2. Problem description

In our formulation of the dynamic and stochastic PDP, we assume that the entire territory is represented by a rectangular area with given height h and width w, so that a generic point a (pickup, delivery, depots and vehicles' positions) is identified by means of its Cartesian coordinates (x_a, y_a). The time t_{ab} needed to

go from an origin a to a destination b is derived from some metrics, e.g. is obtained by using the Euclidean distance between the two points and considering a given fixed speed. We assume there is a fleet of m vehicles, each of which is located at a depot (not necessarily the same for all the vehicles) at the beginning of the planning horizon. In addition, vehicle routes must meet the following constraints: (a) each route starts and ends at a depot; (b) the pickup and the delivery points of a request must be visited by the same vehicle; (c) a pickup point must precede its associated delivery point; (d) a vehicle cannot be diverted from its next destination to service a new request. The customers (and therefore their service requests) are classified into a number C of classes, according to their "importance". Without any loss of generality, we assume that classes are ordered according to a decreasing priority. In general, the k-th request ($k = 1, 2, \dots$) is characterized by $(\boldsymbol{i_k^+}, \boldsymbol{i_k^-}, c_k, t_k)$, where $\boldsymbol{i_k^+} = (x_k^+, y_k^+)$ are the coordinates of the pickup point, $\boldsymbol{i_k^-} = (x_k^-, y_k^-)$ are the coordinates of the delivery point, $c_k \in \{1, \dots, C\}$ is the class of the request, $t_k \geq 0$ is the occurrence time of the request. We assume that the requests are independent and uniformly distributed over the service territory and arrive according to a known stochastic process. Our goal is to maximize the customer satisfaction by minimizing the sum of the inconveniences of the requests, that are calculated differently according to the class the request belongs to. More specifically, we denote by $f(\tau_k, \boldsymbol{w_k})$ the penalty function measuring the inconvenience associated with the k-th request, defined as:

$$f(\tau_k, \boldsymbol{w_k}) = \begin{cases} 0 & t_k \leq \tau_k < D_k \\ p_{c_k}(\tau_k - D_k) & \tau_k \geq D_k. \end{cases}$$

Here, τ_k is the delivery instant and $\boldsymbol{w_k} = (l_{c_k}, p_{c_k})$. In particular, $l_{c_k} \geq 0$ is the amount of time (after t_k) after which the penalty starts to be counted, determining a *soft deadline* $D_k = t_k + l_{c_k}$. Finally, $p_{c_k} \geq 0$ is the slope of the penalty function. Then, the objective function is: min $z = \sum_k f(\tau_k, \boldsymbol{w_k})$.

3. Dispatching policies

The fact of considering several classes of customers characterized by different penalty functions is a common practice, for instance, in same-day delivery services. To give an idea, let us consider the case with $C = 2$, where: $c = 1$ refers to top-priority customers (that typically sign Service Level Agreements with the delivery companies), whose service must be performed within a very short amount of time from the instant of booking to avoid incurring in high penalties; $c = 2$ denotes spot customers, for which time windows are quite large and penalties are not significant. Top-priority customers are typically a small fraction of the total demand, but may have a huge impact on the total inconvenience. Our observation is that, for a dispatching policy to be effective, a portion of the fleet capacity should be reserved to the top-priority customers. The goal is to avoid that, upon the arrival of a class 1 request, all the vehicles are already traveling to service spot requests that: (i) could potentially be far to reach (recall that a vehicle cannot be diverted while en-route); (ii) could indeed be delayed without determining high penalties. Thus, moving on from this idea we develop the two policies described in the following.

3.1 Adaptive capacity-reserve policy

In this policy, a fraction α_c of the fleet of m vehicles is reserved to service the requests belonging to class $1, \dots, c$ (with $c = 1, \dots, C$). Thus, α_c denotes the fraction of the fleet that can be used to service the requests belonging to class c and to the more prioritized classes. In this way, when taking the dispatching decision we employ a cheapest-insertion approach, but the number of alternatives is limited by the fact that not all the vehicles can service all the requests. As an additional dispatching rule, when evaluating the different alternatives we allow an insertion if the delivery instants of the requests of each class c ($c = 1, \dots, C$) that

are already allocated on a route are not delayed more than a given value $\epsilon_c \geq 0$. As pointed out before, we also want to get advantage of the available stochastic knowledge of the problem. As a consequence, the values of the vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_C)$ are determined by solving off-line a training problem on a sample that is representative of the instances to be solved. Of course, the solution must be such that $\sum_{i=1}^{C} \alpha_i = 1$. The training procedure starts with a given value for $\boldsymbol{\alpha}$. Then, we try to repeatedly modify each component α_c by adding or subtracting a value δ and evaluating the (possible) objective function improvement. In case of an improvement, the value of α_c is updated and the procedure is iterated. The procedure terminates when no further improvements are possible.

Under certain circumstances, it could be beneficial to relax the capacity-restriction constraint (i.e., a request of class c can be serviced by a vehicle reserved for one of the classes $1, \ldots, c-1$). In particular, let Δ_k^R be the increase in the objective function caused by the insertion of request k using the capacity-reserve policy, and Δ_k be the increase in the objective function when the capacity-restriction constraint is relaxed. Then, for a given value of a parameter $\beta > 1$, if $\Delta_k^R > \beta\Delta_k$ then the request is allocated without considering the capacity-restriction constraint. The aim of β is to allow such an insertion only if the value of Δ_k^R is considerably greater than Δ_k (e.g., in our experiments we test various values for β, ranging from 1.5 to 5).

3.2 Adaptive capacity-reserve policy with relocation

An important aspect to deal with when trying to anticipate future demand is the relocation of idle vehicles. In this policy, we combine the characteristics of the previous adaptive capacity-reserve policy with a strategy to reposition idle vehicles to a number of home positions. The resulting policy is termed adaptive capacity-reserve policy with relocation. In particular, let r be the number of idle vehicles among those that are devoted to servicing requests belonging to the top V classes, where V is a parameter to be tuned. The home locations where such vehicles are repositioned are determined by solving an *r-median problem* ([1]). The solution approach we employ is taken from [2], in which r points are first chosen at random to form an initial solution. Then, the algorithm checks whether a new point could replace one of the points in order to produce a new solution with a better objective function value. If so, the substitution is performed and the current solution is updated. The algorithm stops when substitutions do not lead to improving solutions.

The optimal assignment of idle vehicles to the home locations is determined considering the distances between the current positions of the vehicles and the home locations as assignment costs. More precisely, let I be the set of current positions of idle vehicles and J the set of home locations ($|I| = |J| = r$). The allocation problem is modeled as a *linear assignment problem* ([1]): {Minimize $\sum_{i \in I} \sum_{j \in J} d_{ij} x_{ij} \ : \ \sum_{j \in J} x_{ij} = 1, \ i \in I; \ \sum_{i \in I} x_{ij} = 1, \ j \in J; \ x_{ij} = \{0, 1\}, \ i \in I, j \in J\}$. Here, x_{ij} is 1 if and only if idle vehicle $i \in I$ is assigned to home location $j \in J$. This problem can be solved in polynomial time with several assignment algorithms.

To speed up the execution, the r possible home locations are computed off-line for each $r = 1, \ldots, m$ and the allocation is solved greedily. The candidate points among which to choose the home locations are obtained by sampling the probability distributions of the requests of the top V classes. On the other hand, the assignment is solved in real time, given that it depends on the current positions of the idle vehicles.

4. Computational experiments

To test our policies, we compare them with two policies: a reactive policy that allocates each new request by using a cheapest-insertion approach and neglecting all the available stochastic information; the anticipatory policy described in [3]. Our goal is to show that our policy can match the quality of a complex anticipatory algorithm, with a computational effort that is comparable to that of a reactive procedure. Moreover, we want to demonstrate that our policies are scalable, i.e., they are able to handle instances of growing size. Thus, we perform a number of computational experiments on randomly generated instances of varying size. All the

Table 1: Results for $N_1 = 40$ and $N_2 = 160$

Dataset	m	RES				RES_REL				ANT	
$(l_1, l_2, \epsilon_1, \epsilon_2)$		$\boldsymbol{\alpha}^*$	β^*	DEV	T_s	$\boldsymbol{\alpha}^*$	β^*	DEV	T_s	DEV	T_s
$(0, 0, 0, 0)$	20	$(0.20, 0.80)$	1.5	-0.48%	0.01	$(0.30, 0.70)$	3.5	-0.26%	0.01	4.92%	29.36
	40	$(0.20, 0.80)$	3.5	-1.09%	0.01	$(0.50, 0.50)$	4.0	-7.53%	0.01	-4.43%	76.97
	60	$(0.20, 0.80)$	3.5	-1.12%	0.01	$(0.70, 0.30)$	1.5	-9.12%	0.04	-4.84%	144.21
$(15, 15, 0, 0)$	20	$(0.20, 0.80)$	1.5	-1.00%	0.01	$(0.10, 0.90)$	4.0	2.74%	0.01	10.24%	28.14
	40	$(0.20, 0.80)$	3.5	-2.54%	0.01	$(0.80, 0.20)$	1.5	-14.02%	0.04	-8.03%	71.18
	60	$(0.20, 0.80)$	3.5	-2.58%	0.01	$(0.50, 0.50)$	2.5	-15.98%	0.03	-10.57%	134.68
$(15, 15, 15, 15)$	20	$(0.20, 0.80)$	1.5	-1.35%	0.01	$(0.10, 0.90)$	4.0	2.60%	0.01	13.32%	24.45
	40	$(0.20, 0.80)$	1.5	-1.78%	0.01	$(0.70, 0.30)$	2.0	-13.86%	0.03	-7.36%	70.41
	60	$(0.20, 0.80)$	1.5	-1.80%	0.01	$(0.50, 0.50)$	3.0	-15.37%	0.04	-7.91%	133.42
$(15, 15, 0, 15)$	20	$(0.20, 0.80)$	1.5	-1.32%	0.01	$(0.10, 0.90)$	4.0	2.78%	0.01	13.74%	30.74
	40	$(0.20, 0.80)$	1.5	-1.63%	0.01	$(0.70, 0.30)$	2.0	-13.85%	0.03	-6.87%	72.66
	60	$(0.20, 0.80)$	1.5	-1.65%	0.01	$(0.50, 0.50)$	3.0	-15.36%	0.01	-7.56%	132.14

datasets are characterized by several experimental factors, some of which have the same value across all the datasets, whereas the others have different values. In particular, in the former set of experimental factors we consider: $C = 2$, a planning horizon of 480 minutes, an average speed of the vehicles of 40 km/h, $h = 20$ km, $w = 20$, $V = 1$, $p_1 = 10$ and $p_2 = 1$. The remaining experimental factors are:

- expected number of requests per class N_c ($c = 1, \ldots, C$): we have tested various values for N_1 and N_2, such that N_1 is equal to the 20% of the overall expected number of requests and $N_1 + N_2$ is equal to 200, 500, and 1500;
- number m of vehicles: we have considered various values that depend on the instance size;
- parameters l_c and ϵ_c ($c = 1, \ldots, C$): we have tested various combinations, obtaining four datasets.

The performance of our dispatching policies are assessed by using two indicators. First, we consider the average time T_s (in seconds) needed by the procedure to allocate a single request. We highlight that, to allow a particular heuristic to be useful in the real world, the value of T_s must be lower than the inter-arrival time between two consecutive requests. Second, we compute the average percentage objective function deviation of the anticipatory algorithm (ANT, in the following) and of our two policies (RES for the adaptive capacity-reserve policy, and RES_REL for the adaptive capacity-reserve policy with relocation, in the following) with respect to the reactive procedure. These deviations (reported under the column heading DEV in the tables) are obtained as $100(\text{obj}_H - \text{obj}_R)/\text{obj}_R$, where obj_R represents the objective function value of the reactive procedure, whereas obj_H is the objective function value of the heuristic we want to evaluate. Thus, a negative value indicates that the considered heuristic achieves an improvement over the reactive procedure. For all the datasets, we have first determined the best values for $\boldsymbol{\alpha}$ by employing the procedure illustrated in Section 3.1 with $\delta = 0.1$. Such values are reported in the tables under the column $\boldsymbol{\alpha}^*$. Analogously we have determined the best values for β (β^* in the tables) in the set $\{1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0\}$. For each dataset, the results are averaged over 30 instances.

The results contained in Table 1 show that the RES and RES_REL policies obtain a maximum improvement of about 2.5% and 16%, respectively, whereas the maximum improvement of the anticipatory procedure is about 10.5%. On the other hand, we observe that our policies are extremely fast, with an average time to allocate a single request that ranges between 0.01 and 0.04 seconds. As expected, the ANT procedure is much slower, taking up to 144 seconds to take a dispatching decision. This is a border-line result with respect to the requirement that the value of T_s must be lower than the inter-arrival time between two consecutive requests (about 144 seconds in this case). Tables 2 and 3 contain the results for the case with an expected overall number of requests equal to 500 and 1500, respectively. For these tables, we report only the results for our policies, because the ANT procedure has always obtained a value for T_s consistently higher than the

Table 2: Results for $N_1 = 100$ and $N_2 = 400$

Dataset $(l_1, l_2, \epsilon_1, \epsilon_2)$	m	RES $\boldsymbol{\alpha}^*$	RES β^*	RES DEV	RES T_s	RES_REL $\boldsymbol{\alpha}^*$	RES_REL β^*	RES_REL DEV	RES_REL T_s
$(0, 0, 0, 0)$	60	$(0.20, 0.80)$	3.5	-1.17%	0.01	$(0.50, 0.50)$	3.0	-4.76%	0.02
	80	$(0.20, 0.80)$	3.5	-1.20%	0.01	$(0.40, 0.60)$	2.5	-6.24%	0.03
	100	$(0.10, 0.90)$	3.5	-1.21%	0.01	$(0.60, 0.40)$	2.5	-8.41%	0.11
$(15, 15, 0, 0)$	60	$(0.10, 0.90)$	5.0	-3.19%	0.01	$(0.50, 0.50)$	4.5	-5.19%	0.04
	80	$(0.10, 0.90)$	5.0	-2.75%	0.01	$(0.50, 0.50)$	2.0	-7.66%	0.09
	100	$(0.20, 0.80)$	5.0	-2.72%	0.01	$(0.70, 0.30)$	2.0	-12.52%	0.35
$(15, 15, 15, 15)$	60	$(0.20, 0.80)$	5.0	-2.12%	0.01	$(0.70, 0.30)$	1.5	-3.80%	0.12
	80	$(0.10, 0.90)$	5.0	-2.17%	0.01	$(0.50, 0.50)$	4.5	-6.73%	0.10
	100	$(0.10, 0.90)$	5.0	-2.17%	0.01	$(0.60, 0.40)$	3.0	-11.35%	0.20
$(15, 15, 0, 15)$	60	$(0.20, 0.80)$	5.0	-1.96%	0.01	$(0.70, 0.30)$	1.5	-4.71%	0.11
	80	$(0.10, 0.90)$	5.0	-2.05%	0.01	$(0.50, 0.50)$	2.0	-6.57%	0.10
	100	$(0.10, 0.90)$	5.0	-1.97%	0.01	$(0.60, 0.40)$	3.0	-11.39%	0.18

Table 3: Results for $N_1 = 300$ and $N_2 = 1200$

Dataset $(l_1, l_2, \epsilon_1, \epsilon_2)$	m	RES $\boldsymbol{\alpha}^*$	RES β^*	RES DEV	RES T_s	RES_REL $\boldsymbol{\alpha}^*$	RES_REL β^*	RES_REL DEV	RES_REL T_s
$(0, 0, 0, 0)$	180	$(0.20, 0.80)$	2.0	-1.36%	0.02	$(0.50, 0.50)$	3.5	-4.36%	1.73
	220	$(0.10, 0.90)$	2.5	-1.37%	0.02	$(0.50, 0.50)$	4.5	-5.56%	2.51
	260	$(0.20, 0.80)$	2.0	-1.36%	0.02	$(0.70, 0.30)$	1.5	-6.12%	4.96
$(15, 15, 0, 0)$	180	$(0.20, 0.80)$	4.5	-3.30%	0.02	$(0.70, 0.30)$	3.0	-2.46%	2.02
	220	$(0.10, 0.90)$	3.0	-3.31%	0.03	$(0.60, 0.40)$	4.5	-4.84%	2.11
	260	$(0.10, 0.90)$	4.5	-3.30%	0.02	$(0.80, 0.20)$	3.0	-7.65%	5.64
$(15, 15, 15, 15)$	180	$(0.10, 0.90)$	5.0	-2.03%	0.02	$(0.70, 0.30)$	3.0	-0.18%	3.23
	220	$(0.20, 0.80)$	2.0	-2.03%	0.03	$(0.60, 0.40)$	4.0	-1.40%	6.53
	260	$(0.10, 0.90)$	2.0	-2.03%	0.02	$(0.80, 0.20)$	4.0	-3.60%	16.53
$(15, 15, 0, 15)$	180	$(0.10, 0.90)$	3.0	-1.75%	0.03	$(0.70, 0.30)$	3.0	0.86%	2.72
	220	$(0.20, 0.80)$	3.0	-1.74%	0.03	$(0.60, 0.40)$	4.0	-1.84%	2.99
	260	$(0.10, 0.90)$	3.0	-1.74%	0.03	$(0.90, 0.10)$	1.5	-3.74%	9.39

inter-arrival time between two consecutive requests (about 57 seconds for 500 requests, and about 20 seconds for 1500 requests). The results of these tables show that our policies are consistently better than the reactive procedure, with maximum improvements of about 3% and 12.5% when the overall expected number of daily requests is 500, and about 3% and 8% in the case of 1500 requests. With respect to the computing times, their values are a little bit higher, especially for RES_REL, but still lower than the inter-arrival times.

References

[1] Brandeau, M. L., Chiu, S. S. *An Overview of Representative Problems in Location Research.* Management Science, 1989, 35(6), 645–674.

[2] Bruno, G., Ghiani, G., Improta, G. *Dynamic positioning of idle automated guided vehicles.* Journal of Intelligent Manufacturing, 2000, 11(2), 209–215.

[3] Ghiani, G., Manni, E., Quaranta, A., Triki, C. *Anticipatory algorithms for same-day courier dispatching.* Transportation Research Part E: Logistics and Transportation Review, 2009, 45(1), 96–106.

[4] Psaraftis, H. N., Wen, M., Kontovas, C. A. *Dynamic vehicle routing problems: three decades and counting.* Networks, 2016, 67(1), 3–31.

[5] Ritzinger, U., Puchinger, J., Hartl, R. F. *A survey on dynamic and stochastic vehicle routing problems.* International Journal of Production Research, 2016, 54(1), 215–231.

Enough Waiting for the Cable Guy - Estimating Arrival Times for Service Vehicle Routing

Barrett W. Thomas
Department of Management Sciences
University of Iowa, Iowa City, IA, USA
Email: barrett-thomas@uiowa.edu

Marlin Ulmer
Carl-Friedrich Gauss Department Business Information Systems
Decision Support Group
Technische Universität Braunschweig, Braunschweig, Germany
Email: m.ulmer@tu-braunschweig.de

1 Introduction

Everyone hates waiting for the cable guy. In fact, people dislike "waiting for the cable guy" so much that it has spawned its own meme with over two million Google search results. More generally, customers dislike the wide time windows (TWs), the earliest and latest times at which a service will begin, that service providers and home-attended delivery companies provide customers. While customers are willing to tolerate some amount of wait (Zeithaml et al., 1993), these wide TWs are frustrating for customers who often must take at least a part of a day off work to stay at home and wait for the technician (Ragsdale, 2012). For example, one of the co-authors was recently told that the technician flipping the switch for internet service would arrive between 8am and 2pm. While these wide TWs offer home service companies better scheduling and routing flexibility, one study estimated that the economic loss resulting from people waiting for service amounted to $38 billion in 2011 in the United States alone (Ellis, 2011).

Ideally, companies would offer narrower TWs. However, shrinking TWs is challenging for several reasons. First, travel and service times are uncertain when the time windows are communicated. Perhaps more importantly, TWs must often be communicated at the time that the customer makes the service request, before all of the requests that will be served on that day are known. The integration of new customers into the planned tours shifts the arrival times of already assigned customers. Hence, dispatchers need to anticipate future requests in the determination of suitable arrival times.

In this work, we seek to improve the customer experience by improving the width of the

TWs that are offered without sacrificing reliability. The key to the problem is to estimate a state-dependent arrival time of a technician at the time of the service request when we do not know all of the requests that the technician will be asked to serve on that day. We call the problem the time window assignment problem with dynamic requests (TWAP). Over the course of the day, customers request service for some future day. We call this the capture phase. The execution of the route takes place during a future execution phase.

For the purposes of this study, we assume that the day and the driver to which the request is allocated are determined exogeneously. With the day and driver chosen, the decision maker must communicate to each requesting customer a time window in which the technician will arrive to begin the service. We assume that the requests as well as the travel and service times are stochastic. The firm follows a given routing strategy. To facilitate the use of multiple time-window widths, the objective of the problem is to minimize the absolute difference between the actual arrival time and the communicated arrival time.

The optimal solution for the TWAP minimizes the expected absolute deviation is to determine the median arrival time for each individual. We demonstrate that estimates of the median are unstable and instead focus on estimating mean arrival times. To estimate the expected arrival time of a technician at a customer, we make use of techniques familiar in approximate dynamic programming. First, we use the concept of a state to describe the status of the system at the time of each request. In this way, we can make each arrival time estimate state-dependent. Because of the large number of potential states, we aggregate the state space based on important temporal parameters. This aggregation is motivated by the success of temporally-based aggregation schemes in the dynamic routing literature. With the aggregated state space, we then execute a series of offline simulations that allow us to learn the expected arrival times for a state. To achieve a balance between simulation runs and solution quality, we rely on a dynamic aggregation of the state space. By performing the computation offline, the proposed method is particularly amenable to the need to real-time communication with requesting customers.

This work makes several important contributions to the literature. First, we introduce a new model and method valid for a number of applications related to completion time estimation in both dynamic routing and scheduling. Second, we introduce a temporal-aggregation scheme for the state space. This aggregation allows us to develop high-quality state-dependent estimates of the arrival times for a given request and thus provide high quality state-dependent TWs. Given the challenges associated with estimating the median of the arrival-time distributions in the case of a large state space, we instead estimate mean arrival times and show that our estimates of the mean lead to superior TWs. We present an offline approach for developing these estimates, giving us the ability to communicate TWs in real time. An extensive computational study demonstrates that the proposed method significantly outperforms the benchmarks.

The work presented in this talk is the only work that considers uncertainty in requests, travel times, and service times in providing state-dependent arrival time estimates for routing problems. Further, the method presented in this work can be extended to scheduling applications in which release or leadtime estimates are required. For both routing and related scheduling problems, this work is the only work that presents an offline estimation approach offering the opportunity to provide customers with real-time estimates.

2 Solution Approach

Because of the spatial and temporal interaction between requests, an analytical approach is not possible, and we turn to approximation. We propose an approach in which we use simulation to derive estimates for values of aggregated states. This method can be viewed as non-parametric as the mapping from the state to the value need not take any particular functional form and is state dependent. Our choice of a non-parametric approach is based on the results reported by state-of-the-art methods in the dynamic vehicle routing literature, all of which use non-parametric approaches to estimate the value of post-decision states (the cost-to-go) and the fact that a dynamic vehicle routing problem drives our estimates.

Ideally, we would approximate median arrival times for every state. However, as the state space is essentially infinite, we could not possibly store, let alone, calculate values for each state. Thus, we operate on an aggregated state space. Our solution approach makes two additional assumptions. First, a complication in solving the TWAP arises in trying to approximate medians. Empirical estimations of median values often exhibit significant instability when only a few observations are available. The obvious solution is to increase the number of simulation runs to increase the number of samples per state. With millions of aggregated states, however, even an offline approach coupled with efficient methods for computing running medians or methods that do not require storing all observed values are computationally intractable. Interestingly, our results show that the approximation of mean values does not suffer as greatly from the challenges of small sample sizes, and fortunately, median and mean values are often similar, differing significantly only for skewed distribution with long tails. Second, because we are estimating means rather than medians, we base our approximations on mean deterministic travel and service times. The mean value is invariant to random service and travel times. We call our approach the anticipatory time window assignment approach (ATW).

3 Results

Table 1 presents results comparing the proposed approach to a myopic estimate of arrival time and to an approach from the literature called *linear*. For the ATW, the linear, and the myopic approaches, the rows of the table represent the average difference from the estimated arrival time, the maximum difference, the percentage of customers served within a 30-minute TW, the percentage of customers served within a 60-minute TW, and the

Table 1: Solution Quality

Measure	ATW	linear	myopic
$\mathcal{Q}$	19.1	26.3	81.0
$\mathcal{Q}_{\max}$	74.1	105.0	261.5
$\mathcal{Q}_{\text{TW}}^{30}$	58.8	48.5	17.9
$\mathcal{Q}_{\text{TW}}^{60}$	80.3	70.5	32.7
$\mathcal{Q}_{\text{TW}}^{120}$	94.6	88.9	53.6

percentage of customers served within a 120-minute TW, respectively. On average, for the solution returned by the ATW, the estimated arrival time and the realized arrival times differ by 19.1 minutes. The average difference for linear is less than a half an hour, and for myopic, it exceeds 80 minutes. The results for the myopic approach demonstrate the value of anticipating future requests when estimating arrival times. The maximum difference for ATW is 74.1, more than 30 minutes less than linear. Looking at the percentage of customers served within a TW, we can see that 58.8% of the customers are served within a 30-minute TW centered at the estimated arrival time given by the ATW method. Further, for one- and two-hour TWs common in many practical applications, the ATW meets on average 80.3% and 94.6% customers within the TWs, respectively. Neither the linear nor myopic approaches are capable of such performance. However, the linear approach does close the gap as the TW widens. This result suggests that the simple linear approach may be amenable in cases in which a provider offers wide TWs.

Examining a disaggregation of the results, we observe a general pattern. The differences between the estimated arrival time and actual arrival times increases with the expected number of customers and the service time. An increase in expected number of customers and service time also leads to higher differences. In the worst case, the difference reaches half an hour for ATW and two hours for the myopic approach.

The development can be explained by the individual impact of both the service time and the number of customers. The greater the number of potential customers on a route, the more likely a customer is to be shifted in the route from when the customer's arrival time was first estimated to when the route is executed. That is, for a given customer request, the arrival time distribution is more variable as more customer requests are expected. Further, if the service time is high, every newly assigned customer results in a significant shift of subsequent existing customers in the tour. Again, the range in the arrival time distribution is high, making accurate prediction more difficult. Still, ATW is able to anticipate the shifts induced by the new customer requests, particularly, for a high number of requests and lower service times. For low service times, the average difference does not exceed 18.4 minutes, while for the highest service time, the difference is always higher than 26.8 minutes. This result suggests that applications with lower service times, such as home

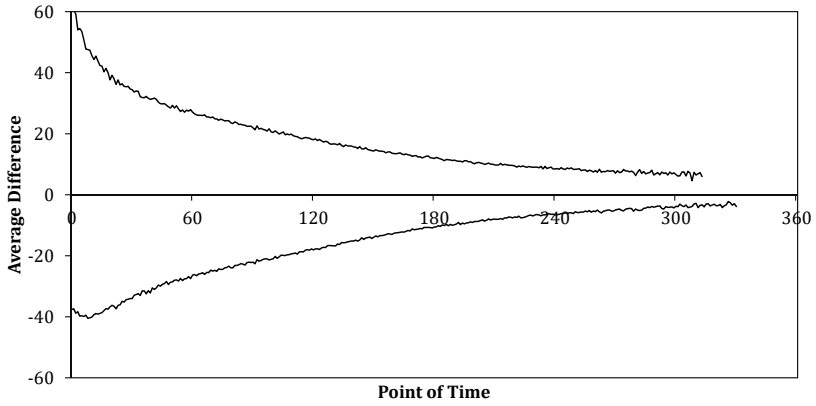


Figure 1: Divergence from Estimated and Realized Arrival Times

attended delivery, may be more more suited for narrow TWs compared to applications with high service times such as complex repair-services.

The point of time at which a customer requests service also affects the accruacy of the arrival time prediction. For a particular instance setting, Figure 1 depicts the average difference of the realized arrival times from the estimated arrival times with respect to the point of time in the capture phase at which the request occurs. On the x-axis, we plot time. On the y-axis, we plot the positive differences from the estimate arrival times and the negative divergences. Figure 1 exhibits decreasing values for positive and negative differences over time. Early requests experience a greater difference between the estimated and realized arrival time. Customers requesting service early in the horizon of the capture phase encounter a tour that is not yet established and to which many requests are still to be assigned. Over time, the tour becomes more established and there are fewer opportunities for customers to be inserted. .

References

Blake Ellis. Waiting for the cable guy is costing us $38 billion, 2011. URL `http://money.cnn.com/2011/11/03/pf/cost_of_waiting/`. [Online; accessed 12-December-2015].

John Ragsdale. Improve service provider efficiency with gps-enabled dispatch software, 2012. URL `https://jragsdale.wordpress.com/2012/09/11/what-is-your-biggest-field-service-pet-peeve/`. [Online; accessed 8-September-2016].

Valarie A. Zeithaml, Leonard L. Berry, and Arantharanthan Parasuraman. The nature and determinants of customer expectations of service. *Journal of the Academy of Marketing Science*, 21(1):1–12, 1993.

VEHICLE ROUTING MODELS & APPLICATIONS

FA3: DYNAMIC PICK UP AND DELIVERY

Friday 8:30 – 10:30 AM

Session Chair: Martin Savelsbergh

8:30 **Dynamic Pickup and Delivery Problems with Transfers**

[1]*Afonso Sampaio,* [2]*Martin Savelsbergh,* [3]*Lucas Veelenturf*,* [4]*Tom Van Woensel*
[1]*Eindhoven University of Technology,* [2]*Georgia Institute of Technology,* [3]*Eindhoven University of Technology,* [4]*Technische Universiteit Eindhoven*

9:00 **How Many Vehicles Do We Need? Pickup and Delivery Problem with Synchronized Tasks and Transfers**

[1]*Monirehalsadat Mahmoudi*,* [2]*Junhua Chen,* [1]*Xuesong Zhou*
[1]*Arizona State University,* [2]*Beijing Jiaotong University*

9:30 **Order Acceptance Mechanisms for Same-Day Delivery**

[1]*Mathias Klapp*,* [2]*Alan Erera,* [2]*Alejandro Toriello*
[1]*Pontificia Universdad Catolica de Chile,* [2]*Georgia Institute of Technology*

10:00 **Optimization Algorithms for Meal Delivery Operations**

Damian Reyes, Alan Erera, Martin Savelsbergh*
Georgia Institute of Technology

Dynamic Pickup and Delivery Problems with Transfers

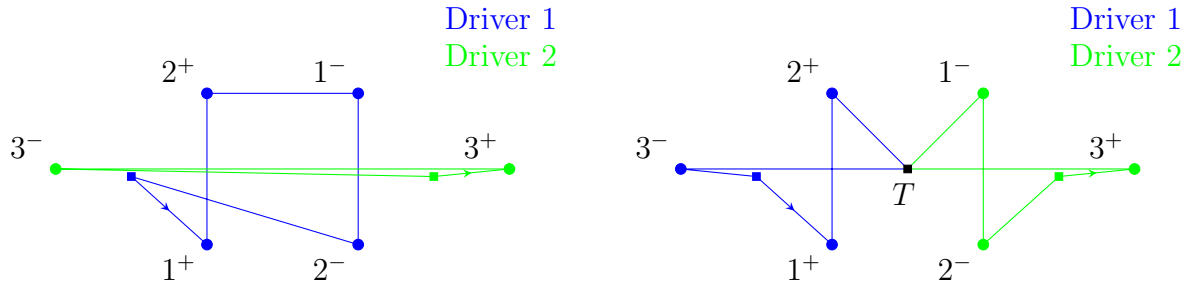
Afonso Sampaio, Luuk Veelenturf, Tom Van Woensel
Eindhoven University of Technology

Martin Savelsbergh
Georgia Institute of Technology

1 Introduction

In on-demand transportation systems, a decision-maker has to decide on actions using little or no knowledge of future requests and, as importantly, these decisions have to be made quickly. This prospective study aims to provide new insights on addressing such problems. In particular, we consider pickup and delivery systems in which each transportation request is associated with an origin, destination and specific time windows for service. In the literature this problem gets much attention and is referred as the Dynamic Pickup and Delivery Problem. However, we include transfer opportunities to facilitate constructing and maintaining more cost-effective and robust transportation plans. Transfer points are locations in the network where requests can be transferred between vehicles and temporarily stored. Hence, more than one vehicle (type) can be used to serve a request, e.g., a request may be picked up at its origin by one vehicle, then dropped off at a transfer point where another vehicle (with other characteristics) will pick it up and drop it off at its destination. The introduction of transfer opportunities allows serving more requests with a given set of vehicles and/or serving a given set of requests with fewer vehicles. An illustration of the benefits of transfer points can be found in Figure 1. As transfers require time synchronization, developing decision technology that effectively exploits transfer opportunities is challenging, more so in a dynamic setting where future requests need to be anticipated and the time to make decisions is limited.

The presence of transfer locations within transportation networks is not new and is encountered, for example, in less-than-truckload transportation, where freight is transferred at breakbulk terminals to increase utilization. In the context of City Logistics, however, transfers have received less attention, albeit they will have to play a more prominent role in the future in order to handle vehicle type and time access restrictions in urban areas, for example. Transfers allow freight delivery by a mixed fleet of vehicles (truck, van, tricycle), but also allow integration of freight and passenger transport. Transfers may also overcome challenges related to driver availability arising in the context of crowdshipping. Consider, for example, a situation in which an individual is able (and willing) to perform a pickup,



(a) Without transfers

(b) With transfers

Figure 1: A set of three requests, $(i^+, i^-)_{i=1,2,3}$, served by two vehicles. Transfers (T) allow more balanced routes and avoid long travel distances (e.g., going from 3^+ to 3^-).

as it is on the way to work, but unable to do the delivery, as it is out of the way and would take too much time. The ability to transfer may allow another individual to complete the delivery. Coordinating such multi-leg deliveries, in real-time, is challenging, and developing the technology to do so effectively is one goal of the research.

2 Literature

The static Pickup and Delivery Problem (PDP) and its variants have given rise to a substantial amount of research. Only recently, the possibility of allowing transfers of goods (or people) is addressed in the literature. In [7], the authors evaluate the usefulness of transfers motivated by the operations of a large courier company. They propose a two-phase heuristic where an insertion procedure is used to construct a solution and an improvement method is applied to the best solution found. A first mathematical formulation and a branch-and-cut approach were proposed in [2], but only small instances could be solved. Motivated by an air cargo carrier application, [9] introduced an insertion heuristic to identify profitable circumstances to exploit the transshipment option. The authors developed a GRASP algorithm and proposed a set of randomly generated instances. More recently, [10] proposed a new model for the problem, distinguishing between vehicle (routes) and request flows and using multi-commodity flows to match these two. [5] proposed an adaptive large neighbourhood (ALNS) algorithm for the problem, and tackled the Dial-a-Ride Problem (DARP) with transfers in [6]. A similar problem was proposed in [3], where requests are allowed to be transferred to/from scheduled lines such as bus, train and metro, operating between two terminals. In a recent survey [4], pickup and delivery problems with transfers are cited as one of the extensions of the PDP with cross-docking.

Recently, dynamic vehicle routing problems have been receiving a great deal of attention from the community [8]. To the best of our knowledge, only [1] tackled a dynamic PDP with transfers. The authors consider transfers occurring at predetermined (e.g., depots)

or arbitrary locations, provided that both vehicles are close enough to each other at some point in time. To satisfy a request, they propose solving a shortest path problem in a special graph and show that this graph does not exhibit the principle of optimality. Thus, instead of applying classical shortest path algorithms (e.g., Bellman-Ford, Dijkstra), the authors introduce a label-setting algorithm to speed up the enumeration of all paths from the pickup to the delivery location of the request. The proposed method does not consider reassignment of requests (i.e., removing a request from one route and insert into another) and, since a request is only accepted if there exists a route sufficiently close to the pickup location, this might lead to many requests being refused. Moreover, no insight is given on the gains that the transfer opportunities yield.

3 d-PDP-T: Formalization

Let $\mathcal{R}$ be the set of transportation *requests*. A request $r \in \mathcal{R}$ specifies a demand $d_r \in \mathbb{N}_+$ to be collected at location r^+ after time e_r^+ and to be delivered at location r^- before time l_r^-. A vehicle fleet $\mathcal{K}$ is used to serve the requests. Each vehicle $k \in \mathcal{K}$ has capacity Q_k^{veh}, is initially positioned at depot $s_k \in \mathcal{S}$, and is available for service between e_k^{veh} and l_k^{veh}. Furthermore, at each depot $s \in \mathcal{S}$ an extra vehicle with capacity Q^+ can be brought into service at any time for a fixed cost of c^+, and it will be available for a maximum duration of L time units.

We consider a set $\mathcal{H}$ of *transfer locations* where requests can be transferred from one vehicle to another. A transfer operation (i.e. a vehicle visits a transfer location) has a fixed service fee of c^{tr} and each transfer location $h \in \mathcal{H}$ has a fixed capacity Q_h^{tr} to temporary store items. Let $\mathcal{N} = \mathcal{S} \cup \mathcal{R}^+ \cup \mathcal{R}^- \cup \mathcal{H}$, where $\mathcal{R}^+ = \{r^+ | r \in \mathcal{R}\}$ is the set of pickup locations and $\mathcal{R}^- = \{r^- | r \in \mathcal{R}\}$ the set of delivery locations. Without loss of generality, we assume that the intersections of any combination of $\mathcal{S}$, $\mathcal{R}^+$, $\mathcal{R}^-$ and $\mathcal{H}$ is empty. Define a network $\mathcal{G}(\mathcal{N}, \mathcal{A})$, with arc set $\mathcal{A}$ representing links connecting locations. For all $(i, j) \in \mathcal{A}$, let $c_{i,j}$ be the cost and $\tau_{i,j}$ the travel time (including service time) of traversing arc (i, j). Waiting at a location is possible

Customers call in for transportation requests during the working day, i.e., during the time interval $[0, T]$. At time 0, let $\mathcal{R}_0$ be the requests already known to the system before operation starts. At any time $t \in [0, T]$ a customer can call-in for service and the locations, demand and time restrictions of the new request are then revealed. Let t_r be the call-in time of request r and $\mathcal{R}_t$ the requests known to the system at time t. The routing plan should be modified to include the new request. The assignment of requests to vehicles and the order that locations are visited can be modified, as well as new vehicles may be dispatched from their initial locations. Once a vehicle is instructed to visit the next location of its route, this location must be visited immediately, i.e., deviation is not allowed. The goal is to minimize the transportation costs.

4 Methodology

At the moment of writing, the methodology has focused on the static variant of the problem (i.e. all request are known at time 0). It is expected that we learn from the methodology to solve the static problem and gain insights to apply in the dynamic variant.

An adaptive large neighborhood search heuristic is introduced consisting of two parts: the generation of initial feasible solutions and improvement strategies. Especially the improvement strategies are valuable for the dynamic setting, as in this case there will be a current routing plan to which in each stage one or more new requests need to be added.

4.1 Options for initial feasible solutions

A routing plan in which no transfers are used is clearly a solution to the problem. However, defining neighborhoods that can exploit the introduction of transfers is quite challenging as it has to involve rerouting of at least two vehicles (to visit a transfer location). Therefore, it might be beneficial to create an initial feasible solution in which some of the requests are transferred. From there, in the improvement phase, other requests could also make use of these transfer opportunities, if doing so leads to a better solution. To identify requests that seem good candidates to be transferred, we introduce the following notion of regions.

We first divide the whole area underlying the transportation network in regions. Vehicles need to stay within their region and if the pickup and delivery of one request are not in the same region, transfers are required. For each of these requests, we find the shortest path via the transfer points and split the path in requests per region with appropriate time windows. For example, if the path traverses two regions via transfer point h, the request r is split up into a request from r^+ to h and a request from h to r^-. In the first region the deadline l_h^- for reaching h must be set such that it is lower or equal than the time e_h^+ it becomes available in region 2. Afterwards, each region is solved as a standard pickup and delivery problem.

4.2 Improvement strategies

One of the important operators for the ALNS is the insertion operator. This one is also of special interest for the dynamic setting of the problem as new requests pop up all the time. We can differentiate between insertions of transfer points into request routes (interesting for the first mentioned initial solution) or the insertion of requests into vehicle routes. In both cases one insertion can change multiple vehicle routes at the same time which is different from the standard insertions in the Pickup and Delivery literature.

5 Current stage of the research

At the moment we have only results for so-called "toy-problems", in which we investigated situations where it is worthwhile to include transfers and situations where it is not. How-

ever, the framework of our methodology is ready and at the TSL conference we will be able to present results of the static and dynamic variant of this problem. We will also present the interesting methodological challenges we have faced by introducing transfers. It is not obvious to come up with valuable starting solutions (e.g. in the regions variant, defining the regions is already interesting), insertion methods (e.g. while inserting a new request or transfer in a route the time synchronization of vehicles at transfer location is a challenging problem in itself) and objective functions for the dynamic case (i.e. optimizing the system at each stage will not automatically lead to minimum operational cost).

References

[1] P. Bouros, D. Sacharidis, T. Dalamagas, and T. Sellis. Dynamic pickup and delivery with transfers. In *Advances in Spatial and Temporal Databases: 12th International Symposium*, pages 112–129. 2011.

[2] C. E. Cortés, M. Matamala, and C. Contardo. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200(3):711 – 724, 2010.

[3] V. Ghilas, E. Demir, and T. Van Woensel. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research*, 72:12 – 30, 2016.

[4] G. Guastaroba, M. G. Speranza, and D. Vigo. Intermediate facilities in freight transportation planning: A survey. *Transportation Science*, 50(3):763–789, 2016.

[5] R. Masson, F. Lehuédé, and O. Péton. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47(3):344–355, 2013.

[6] R. Masson, F. Lehuédé, and O. Péton. The dial-a-ride problem with transfers. *Computers & Operations Research*, 41:12 – 23, 2014.

[7] S. Mitrović and G. Laporte. The pickup and delivery problem with time windows and transshipment. *INFOR: Information Systems and Operational Research*, 44(3):217–227, 2006.

[8] H. N. Psaraftis, M. Wen, and C. A. Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016.

[9] Y. Qu and J. F. Bard. A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research*, 39(10):2439 – 2456, 2012.

[10] A. Rais, F. Alvelos, and M. Carvalho. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research*, 235(3):530 – 539, 2014.

How many vehicles do we need? Pickup and delivery problem with synchronized tasks and transfers

Monirehalsadat Mahmoudi[1], Junhua Chen[2], Xuesong Zhou[1]
[1]School of Sustainable Engineering and the Built Environment, Arizona State University, Tempe, Arizona 85281,
[2]School of Traffic and Transportation, Beijing Jiaotong University, Beijing, P. R. China 100044

Keywords: pickup and delivery problem with transfers, ride-sharing, ride-hailing, forward dynamic programming, state-space-time network, clustering algorithm, Lagrangian heuristic.

Abstract

The most complicated level of coordinated transportation services is ride-sharing with synchronized transfers. In general, transfers are used to provide more efficient transportation networks by reducing the operational costs, as well as making more flexible routes available for the customers. A large number of our daily trips is classified in this category. For instance, in multi-modal transports, we use two or more transport modes for our trips (e.g. train and bus). Multi-modal transit provides convenient and economical connection of various modes to make complete journey from origin to destination. Another example of this type of transportation service can be found in the first mile/last mile transport of the commuters who need to go from origin to a transit station and then from the station at the other end of the trip to a final destination. Ride-sharing between households or fellow workers is another example of ride-sharing with synchronized transfers. In this case, members of a family or any other social group informally arrange their trips and share the travel information such as departure time, stops, and transfer points among themselves.

In general, ride-hailing and ride-sharing with/without transfers can be mathematically modeled by the vehicle routing problem with pickup and delivery with time windows (VRPPDTW). The vehicle routing problem with pickup and delivery with time windows (VRPPDTW) is a combinatorial optimization problem that searches for an optimal set of routes for a fleet of vehicles to serve a set of transportation requests. Each request is a combination of pickup at the origin and drop-off at the destination within particular time windows.

Although several algorithms have been proposed to solve the VRPPDTW, this problem even for single vehicle cases is still classified as one of the toughest problems of combinatorial optimization. Generally, in the most commonly used exact approaches for solving the VRPPDTW, column generation and branch-and-cut, generating additional columns and cuts for real-world transportation networks is a computationally-challenging task. Moreover, dealing with several constraints especially non-linear constraints related to the validity of the time and load variables in the classical VRPPDTW model, prompted us to look at this challenging problem from a different angle.

In this research, we intend to embed the VRPPDTW constraints, i.e. passengers' preferred departure/arrival time windows and vehicles' capacity constraints, on a three-dimensional state-space-time network in which time and load are explicitly added as new dimensions to the physical transportation network. We will further show that our time-expanded network structure in comparison to the classical network representation performs better in terms of (i) handling passengers' desired pickup and drop-off time windows and (ii) fulfilling the demands with less number of vehicles. We will also show that our multi-dimensional network structure not only handles large-scale transportation networks with links whose routing cost (travel time) may vary over the time of day (based on the real-time traffic conditions), but also performs on the networks in which routing cost of the links is load dependent (e.g. HOV or HOT lanes).

In addition, the distinctive structure of our proposed multi-dimensional network allows us to mathematically model different forms of coordinated transportation services. In fact, introducing passengers' cumulative service state as an independent dimension to the space-time network enables us to distinguish the ways by which a passenger can be served (i.e. ride-hailing, ride-sharing without transfer, or ride-sharing with synchronized transfers). We will further apply a Lagrangian Relaxation (LR) framework to determine the price of each trip request considering the way by which it is supposed to be fulfilled (e.g. through Lagrangian multipliers).

Note that our proposed multi-vehicle passengers' cumulative service state-space-time network representation is able to solve the VRPPDTW to optimality for a limited number of passengers due to the exponential order of passengers' cumulative service state; therefore, in order to handle a real-world transportation network with a large set

of customers, we must split the large-sized primary VRPPDTW into a number of small-sized sub-problems in which trips with the most compatibility are clustered together. In order to find well-matched customers, we utilize the three-dimensional space (XY plane)-time network representation and apply a rational rule to explore all potential matchings. To define passengers' cumulative service patterns within each cluster, we utilize the path representation schema for the Traveling Salesman Problem (TSP) proposed by Bellman (1962) and Held and Karp (1962). We further relax the group of constraints by which we guarantee that each passenger is served by a single way (i.e. ride-hailing, ride-sharing without transfer, or ride-sharing with synchronized transfers). As a result, the problem is converted to a state-dependent time-dependent least cost path problem which can be solved by various algorithms already proposed for solving the shortest path problem efficiently. Here, we develop a forward dynamic programming (DP) solution-based approach across multiple vehicles to reach the optimality within the cluster.

As a final point, by introducing passengers' cumulative service patterns in the VRPPDTW, we are able to tackle the symmetry issue which is a common issue in the combinatorial problems. To explain this issue, suppose vehicles u and u' are identical in terms of starting and ending depots, work shift, and capacity. Despite the fact that from the practical point of view, it does not matter passenger j is served by vehicle u or u', the computational procedure spends plenty of time exploring the vertexes of these two vehicles' network separately. As a result, many regions which are symmetric to the parts that have been already examined are scanned. One of the common and effective methods of handling symmetries is to introduce symmetry breaking constraints to the main problem to impose the system, not to search within symmetric solutions (Raviv, Tzur, and Forma 2013). In this paper, by the aid of our passengers' cumulative service patterns, we are able to impose the symmetry breaking constraints implicitly to assignment-routing paths in a well-structured state-space-time network. To sum up, our major contributions in this research are as follows:

1. Providing a mathematical framework for pickup and delivery problem with two types of synchronization: tasks and transfers.
2. Embedding the vehicle-to-task assignment constraints by a DP solution algorithm on a state-space-time network. This provides an exact solution for small scale problem and overcomes the infeasibility, as well as symmetry issue in the lower bound estimator.
3. Handling large-scale instances by the aid of a Lagrangian heuristic to evaluate the price of synchronized transfers and guide a fast search.
4. Testing large scale real-world instances considering road capacity constraints to encourage synchronized transfers and reduce the number of vehicles and corresponding traffic congestion.

Computational experiments

The time-dependent DP described in this paper was coded in C++ platforms, and passengers' grouping problems and vehicles' performance improvement procedure were solved from GAMS Distribution 23.00. The experiments were performed on an Intel Workstation running two Xeon E5-2680 processors clocked at 2.80 GHz with 20 cores and 192GB RAM running Windows Server 2008 x64 Edition. In this section, we initially examine our proposed model on instances proposed by Ropke and Cordeau (2009) which is publicly available at http://www.diku.dk/~sropke/ followed by the randomly generated instances on the real-world City of Tempe transportation network to demonstrate the computational efficiency, as well as, solution optimality of our developed algorithm.

Ropke and Cordeau (2009) data set is the modified version of instances employed by Ropke et al. (2007) initially introduced by Savelsbergh and Sol (1998). In this data set, passenger p's origin and destination are denoted by node p and node $n + p$, respectively. In addition, the coordinates (x and y) of passengers' origin and destination are randomly generated and uniformly distributed over a $[0,50] \times [0,50]$ square. A single depot is located in $[25,25]$. The load of each passenger is randomly generated from $[5, Cap_v]$, where Cap_v is the maximum capacity of the vehicles (in these instances the vehicles' capacity is assumed to be the same). Moreover, $[0,1000]$ is considered as the vehicles' time horizon (vehicles' time horizon is assumed to be identical). Feasible departure/arrival time windows are also randomly generated for each passenger.

Six groups of instances are examined by considering different values of vehicle' capacity, different length of departure/arrival time windows, and different passengers' load. The values of vehicles' capacity in instances AA,

BB, CC, and DD are 15, 20, 15, and 20; and the length of passengers' departure/arrival time windows are 60, 60, 120, and 120, respectively. In addition, as we mentioned before, in these four instances, the load of each passenger is randomly generated from $[5, Cap_v]$. In instances XX and YY, the value of vehicles' capacity is 15, while the length of passengers' departure/arrival time windows are 60 and 120, respectively. In addition, the load of each passenger is assumed to be 1. In instances XX and YY, due to the large value of vehicles' capacity (i.e. 15) in comparison to the load of each passenger (i.e. 1), more levels of complexity are expected. To the best of our knowledge, very few papers have published the results of instances XX and YY. Our proposed model can handle these two instances though. Table 1 presents the results obtained from running our algorithm on Ropke and Cordeau (2009) instances. The two alphabetical letters in the instances names are representative of vehicles' capacity, length of passengers' time windows, and load of passengers, while the double-digit number after alphabetical letters demonstrates the total number of passengers in that data set. According to Table 1, in most instances, our heuristic-based algorithm in comparison to the heuristic proposed by Ropke et al. (2007) performs better in terms of number of vehicles (as the primary objective) and routing cost (as the secondary objective); however, from computation time perspective, it seems that the heuristic by Ropke et al. (2007) performs slightly better than ours. Figure 1 illustrates the position of passengers' origin and destination and the routes of vehicles v_1-v_4 in data set AA30. The ratio of $\frac{value\ of\ time\ (\$/min)}{value\ of\ distance\ (\$/mile)}$ is supposed to be 1.

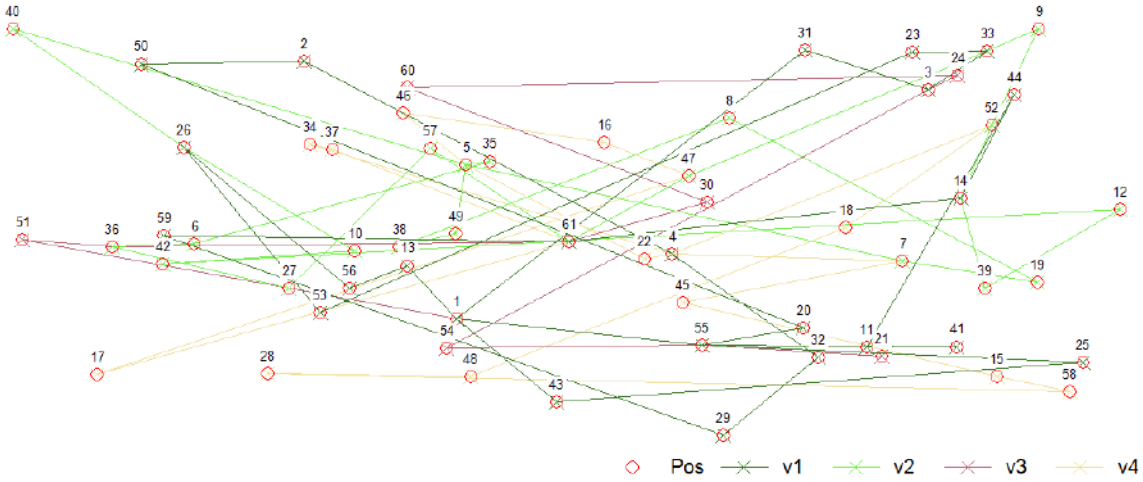


Fig. 1. Position of passengers' origin and destination and route of vehicles v_1-v_4 in data set AA30.

Based on the real world City of Tempe transportation network illustrated in Figure 2 with 1160 transportation nodes and 2493 links, we test our algorithm on randomly generated transportation request instances in order to demonstrate the computational efficiency of our model. In this examination, the vehicles' capacity is assumed to be 3, the vehicles' planning horizon is supposed to be $[0,700]$, and the load of each passenger is assumed to be 1. The ratio of $\frac{value\ of\ time\ (\$/min)}{value\ of\ distance\ (\$/mile)}$ is also assumed to be 1. Table 2 presents the results from City of Tempe.

Table 1. Results obtained from running our algorithm on Ropke and Cordeau (2009) instances.

Name	# of groups	# of Vehicles	# of Vehicles (Ropke et al. (2007))	Routing cost	UB of routing cost (Ropke et al. (2007))	Computation time (sec)	Computation time (sec) (Ropke et al. (2007))
AA30	5	4	5	41,316	51,317.40	25.6	27
AA35	5	5	5	51,506	51,343.53	44.2	33
AA40	6	6	6	61,759	61,609.44	44.67	41.4
AA45	7	6	6	62,029	61,693.01	52.57	49.8
AA50	8	6	7	62,213	71,932.03	54.25	58.8
AA55	8	8	8	82,405	82,185.31	95.63	64.2
AA60	9	8	9	82,629	92,366.70	94.44	76.8
AA65	10	7	8	72,783	82,331.12	98.3	87.6
AA70	10	8	11	82,950	112,458.28	143.7	98.4
AA75	11	8	9	83,044	92,529.42	141.82	112.8
BB30	5	4	5	41,267	51,193.62	28.4	28.2
BB35	5	5	6	51,580	61,400.07	38.2	32.4
BB40	6	5	5	51,785	51,421.35	60.83	44.4
BB45	7	6	6	61,950	61,787.28	71.86	49.2
BB50	8	7	7	72,164	71,889.75	89.5	58.8
BB55	8	8	8	82,483	82,080.73	122.75	64.2
BB60	9	11	10	112,988	102,323.77	122.22	73.8
BB65	10	10	8	103,211	82,623.98	122.9	85.2
BB70	10	11	9	113,534	92,647.75	160.4	100.8
BB75	11	11	9	113,558	92,476.30	154.18	112.8
CC30	5	5	5	51,358	51,145.18	44.2	28.2
CC35	5	5	5	51,578	51,235.64	51.8	34.2
CC40	6	5	6	51,695	61,473.91	49.5	43.2
CC45	7	5	8	51,955	81,408.89	53.57	49.8
CC50	8	7	6	72,154	61,936.27	51.38	63.6
CC55	8	10	6	102,460	61,930.55	90.88	71.4
CC60	9	8	7	82,546	72,104.00	84.89	82.8
CC65	10	9	8	92,803	82,326.62	102.9	90
CC70	10	9	9	92,963	92,613.68	149.3	102
CC75	11	9	9	93,220	92,711.74	149	112.8
DD30	5	4	6	41,426	61,040.10	41.4	27.6
DD35	5	5	7	51,614	71,308.04	68.2	33.6
DD40	6	5	6	51,851	61,531.68	68	43.2
DD45	7	5	8	51,960	81,601.63	72.86	48
DD50	8	6	7	62,131	71,761.23	70.25	60
DD55	8	6	7	62,358	72,051.95	121.38	69
DD60	9	7	8	72,521	82,308.08	113.78	78.6
DD65	10	7	8	72,825	82,200.77	120.6	90
DD70	10	8	8	83,034	82,631.56	173.6	102
DD75	11	9	9	93,255	92,970.84	165.45	109.8
XX30	5	4		41,093		101.4	
XX35	5	5		51,313		174.6	
XX40	6	5		51,540		166.33	
XX45	7	7		71,719		156.29	
XX50	8	6		61,707		126.88	
XX55	8	6		61,839		254.25	
XX60	9	6		62,033		299.44	
XX65	10	6		62,531		286.4	
XX70	10	7		72,775		400.1	
XX75	11	8		82,960		388.73	
YY30	5	4		41,195		76.2	
YY35	5	5		51,363		187.6	
YY40	6	6		61,608		161.33	
YY45	7	6		61,806		176.14	
YY50	8	6		61,966		203.88	
YY55	8	7		72,121		274.13	
YY60	9	6		62,321		276.56	
YY65	10	7		72,464		327.3	
YY70	10	7		72,586		419.6	
YY75	11	9		92,679		397.64	

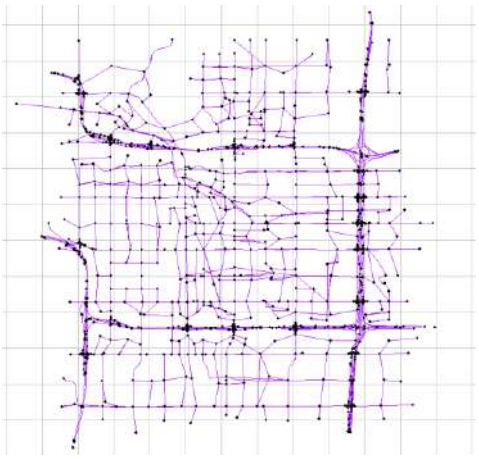


Figure 2 City of Tempe with 1160 nodes and 2493 links.

Table 2 Results from Tempe with 1160 nodes and 2493 links.

Test case	Number of passengers	Number of groups	Number of required vehicles	Routing cost	Computation time (s)
1	50	8	9	92,053	279
2	60	9	9	92,325	299
3	100	15	15	154,144	443
4	150	22	24	246,172	592
5	200	29	29	298,013	943
6	300	43	46	471,849	1233
7	400	58	53	547,230	2043

Conclusions

In this research, by extending the work pioneered by Bellman (1962), Held and Karp (1962) and Psaraftis (1980) on using the DP method to solve TSP and VRP, we embed many complex VRPPDTW constraints on a three-dimensional state-space-time network. In this hyper network construct, elements of time and load are explicitly added as new dimensions to the physical transportation network. In order to handle a real world large-scale transportation network with a large set of customers, we must split the large-sized primary VRPPDTW into a number of small-sized sub-problems in which passengers with the most compatibility are clustered together. We use a time-dependent forward DP algorithm to solve the time-dependent state-dependent least-cost assignment-path problem for the local clusters derived from the original VRPPDTW. In addition, in order to improve the vehicles' performance, we apply a number of rational rules to perform several tasks by a small set of vehicles. At the end, extensive computational results over the standard data sets and randomly generated data sets from the Phoenix subarea (City of Tempe) show the computational efficiency and solution optimality of our developed algorithm.

References

Bellman R (1962) Dynamic programming treatment of the traveling salesman problem. *Journal of the Association for Computing Machinery*, 9: 61-63.

Held M, Karp RM (1962) A dynamic-programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1): 196-210.

Psaraftis HN (1980) A dynamic programming approach to the single-vehicle, many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2): 130–154.

Raviv T, Tzur M, Forma IA (2013) Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and logistics*, 2: 187-229.

Ropke S, Cordeau JF, Laporte G (2007) Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49(4): 258–272.

Ropke S, Cordeau JF (2009) Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3): 267–286.

Order Acceptance Mechanisms for Same-Day Delivery

Mathias A. Klapp, Alan L. Erera, Alejandro Toriello

January 5, 2017

1 Introduction

Many retailers offer same-day delivery (SDD) for customers who make online purchases and who demand fast order fulfillment. To offer such a service, the retailer must operate a logistics system where delivery request acceptance, order processing at a fulfillment location (depot), and delivery to the customer location all occur within the same operating day. We formulate and study the Dynamic Dispatch Waves Problem with Immediate Acceptance (DDWP-IA) to integrate request management and order distribution decision making for SDD systems dispatching a single vehicle. Starting at time $T > 0$, customers place online order requests to be delivered to their geographic location $i \in I$ within the node set I until a cut-off time $t^{ct} \in (0, T)$, after which no more orders are accepted. Each order is not known until its disclosure time, but we assume that all probabilities of the arrival process are available and that arrivals are Markovian and independent between different locations. The DDWP-IA seeks to dynamically determine the set of requests to accept and a vehicle operation (dispatch plan) to serve these orders before time 0. A dispatch plan is defined as multiple delivery routes of a single vehicle dispatched from the depot at a subset of the finite set of feasible dispatch times (waves) $\{t_W = T, \ldots, t_1, t_0\}$; the index w represents the number of waves to go before $t_0 = 0$. The objective is to minimize total vehicle travel cost, plus penalties for rejected delivery requests.

The previous work on the Dynamic Dispatch Waves Problem (DDWP) [2, 3] models the setting where unattended requests are not formally rejected until the end of the day. This setting models a system where all realized unserved orders are covered using a secondary transportation mode.

In the DDWP-IA we study an alternative setting with reduced flexibility in which customers are offered SDD when placing an order, and if a customer selects the option then SDD is guaranteed. To do so, we use an *accept or reject* framework: a request is accepted (and thus delivered in the same day), or rejected (with

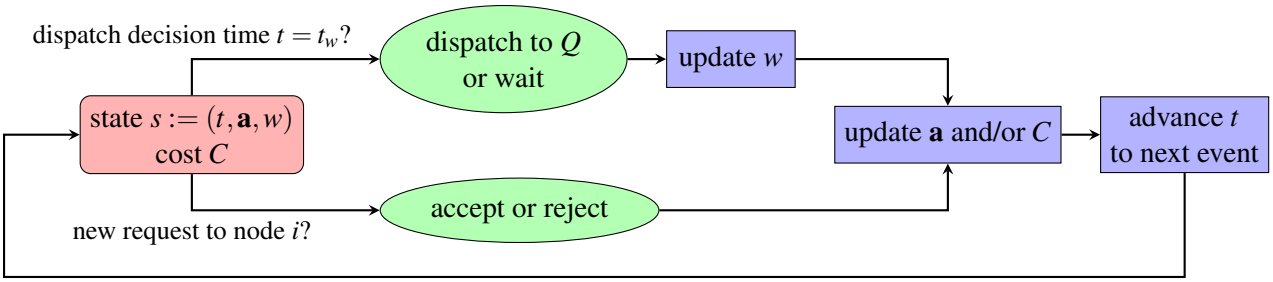


Figure 1: Flowchart of actions and transitions in state $(t, \mathbf{a}, w)$ for the DDWP-IA

a penalty) immediately when received. An order distribution system operates simultaneously with order acceptance and dynamically chooses at each wave whether or not to dispatch the vehicle with a subset of accepted orders ready for service; all accepted orders must be served by time 0. We also study the impact of an order processing time $p > 0$ at the stocking location, which models the fact that accepted requests may not be available for immediate dispatch, and instead must wait to be processed (picked and packed) before they can be loaded into the vehicle. The DDWP-IA models a setting such as Amazon's same-day delivery service.

2 Contribution

We formulate the DDWP-IA as a semi-Markov decision process [4]. The system state is $s = (t, \mathbf{a}, w)$ where $t \in [0, T]$ is the current time (counting down from T), $\mathbf{a}$ is the vector of commitments, and each component indicates the earliest possible wave $a_i \in \mathscr{W}$ in which location $i \in I$ can be visited to cover its accepted open orders.The value of $w < W : t_w \leq t$ represents the earliest upcoming wave when the vehicle is again available at the depot, and determines the next dispatch decision. Without loss of optimality, all commitments at one node can be covered in one visit and therefore the state s does not carry disaggregated order information. Figure 1 depicts a flowchart of all actions and transitions connected to a state s. An accept/reject decision is immediately made after a request is placed. Rejection costs $\beta_i > 0$, but keeps the system's state unaltered; acceptance is free of charge, but modifies commitments in $\mathbf{a}$. A vehicle dispatch decision is made at states where $t = t_w$ and represents a subset $Q \in I$ of node visits to be cleared from $\mathbf{a}$, paying the optimal TSP tour cost for those locations. The vehicle becomes available at the depot after the tour's duration; the special case $Q = \emptyset$ represents waiting for one wave at the depot at zero cost.

We solve a deterministic problem where the number of orders and arrival times are disclosed before the

operation starts. Under this setting the DDWP-IA model collapses to the deterministic DDWP solved in [2] via branch and cut approaches for routing problems. We develop a perfect information relaxation (PIR) that computes a different optimal solution for each scenario realization of the random parameters [1, 5].

We develop a framework to produce policies for the DDWP-IA, where a system state s is coupled with a state-feasible vehicle dispatch plan π (with potentially multiple planned trips) serving all commitments in s along with a set of potential future delivery requests that have not yet realized. This dispatch plan is also used to guide both vehicle dispatch and order acceptance decisions. Any policy P is fully determined by two features; an initial dispatch plan designed before the operation starts and a dynamic update of the plan when new information becomes available. One update of the plan might be performed immediately after a request's arrival and before its acceptance decision; the speed of this update is especially critical for immediate acceptance. Additionally, another update might be executed immediately before making vehicle dispatch decisions.

We initially provide a *myopic policy* (MP) that assumes no information regarding future arrivals. Thus, it only updates the after each request's arrival making optimal decisions with respect to the information disclosed so far. Ihe initial plan is a TSP tour visiting all locations considered in the initial commitments $\mathbf{a}^0$. MP tends to build a plan consisting of one single and long dispatch route, leaving few recourse possibilities (or none); we thus heuristically set a maximum route duration d^{max} to force returns to the depot.

We later build proactive policies that incorporate probabilistic information regarding potential future order requests. The first is an *a priori* policy (AP) in which a static dispatch plan π is determined before the operation starts, using all probabilistic information available at time $t = T$. We compute the optimal *a priori* policy in which no recourse actions are allowed and show its equivalence to solving a specific deterministic DDWP instance. If AP is used statically, then it does not provide good results. An alternative policy (MPF) uses AP's dispatch structure to plan vehicle returns to the depot in a myopic policy. We predetermine a subset of dispatch waves based on the optimal *a priori* solution, but dynamically assign orders to dispatches and routes based on myopic updates. We believe that such a policy emulates and improves a system that practitioners may use; it builds a reasonable dispatch structure based on probabilistic information and, once the daily operation starts, the dispatcher assigns requests to time slots. A better but more involved idea is to fully roll out the *a priori* policy (RP) and re-optimize the *a priori* problem and update the plan before each order acceptance decision and each vehicle dispatch decision.

Because RP may be computationally expensive, we propose the Heuristic Acceptance Rollout Policy

Table 1: Average results for each policy

metric \policy	**PIR** bound	**FLEX**	**MP** ($d^{max} = 252$)	**AP**	**MPF**	**RP**	**HARP**
cost per request	11.5	13.3	15.4	17.0	15.2	13.9	14.2
request fill rate	93.4%	88.6%	85.5%	80.1%	85.6%	87.8%	86.9%
distance per visit	9.1	8.8	8.9	9.4	9.0	8.9	8.9
gap^{LB}	N/A	15.9%	35.0%	48.0%	33.0%	21.0%	23.8%
$time_{disp}$ (sec.)	0.00	81.8	0.00	0.00	0.00	68.6	80.1
$time_{acc}$ (sec.)	0.00	0.00	2.4	0.00	1.2	18.1	1.1
number of routes	2.69	2.51	1.94	2.48	2.21	2.52	2.52
initial wait in waves	2.87	3.21	3.43	3.09	3.35	3.20	3.21

(HARP) that only re-optimizes the *a priori* problem before each dispatch decision and heuristically solves it upon order arrivals before acceptance decisions. We propose a fast solver-independent meta-heuristic to implement HARP that runs over any dispatch plan π. It is a local search procedure that exploits the wave structure of any feasible plan by running multiple neighborhood searches over it that solve instances of the prize-collecting TSP (PC-TSP). To avoid local optima points, we randomly destroy local solutions and randomize the evaluation of candidate solution. Moreover, each PC-TSP is solved with another metaheuristic that speeds up computation.

3 Computational Results

We designed a set of computational instances under different settings of geography, problem size up to $|I| = 50$ nodes, online request arrival rates, and accepted orders known before the operation starts. Table 1 presents average results for each policy. All experiments share a horizon of $T = 882$ time units, 7 homogeneously distributed waves, an order processing time $p = 20$, a cut-off time set at $2/7$ of the horizon, and penalties of the form $\beta_i = 2d_{0,i} + 1$, where $d_{0,i}$ is the distance cost from i to the depot. The cost of the best benchmark (MPF) is 9.3% higher than RP. The success of RP stems from optimization-guided decisions and increased recourse opportunities by executing more dispatches compared to myopic policies. Its marginal benefit is concentrated in improving fill rate rather than in routing efficiency; we found that RP especially increases fill rates of requests placed relatively later in the horizon and on locations relatively farther away from the depot. We also compared RP against an infeasible rollout of the optimal *a priori* policy that can postpone order acceptance decisions throughout the day (FLEX). The marginal cost per request added by

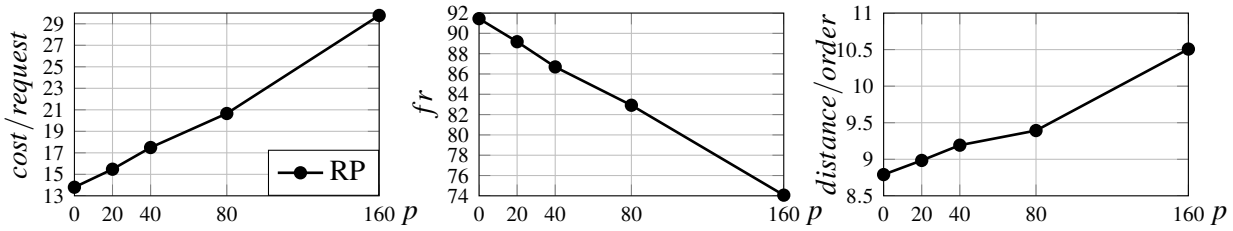


Figure 2: Average cost per request, fill rate (fr), and distance per order versus order processing time (p)

imposing immediate order acceptance is estimated to be 4.5%. All policies differ in computation per dispatch ($time_{disp}$) and per acceptance decision ($time_{acc}$). The HARP policy speeds up $time_{acc}$ 16.5 times on average over RP incurring relatively small additional cost (2.2%).

We also tested our policies with different values of order processing times p to evaluate the importance of implementing faster warehousing operations for SDD, and we conclude that an increase in p may directly be transferred to an increase in cost; see results in Figure 2. The value of p seems to affect the system in both aspects, routing efficiency and request fill rate. Moreover, we found that the average RP gap over PIR and the gap difference over the myopic policies decrease as p increases.

Our experiments also show that having more dynamism in the order arrival process (in the form of a later cut-off time) while keeping expected requests constant can significantly increase the system's cost per request; see Table 2. The design of the SDD service should set an appropriate value of t^{ct} that delivers the necessary amount of service flexibility to the customer while providing enough of a time buffer to the operation's planner.

Table 2: Average performance indicators of RP versus cutoff time (t^{ct})

t^{ct}	cost per order	fill rate (%)	distance per visit	accepted after t^{ct} (%)
126	27.4	76.2	10.1	21.0
252	17.1	86.8	8.9	65.0
378	9.9	95.7	7.5	94.8

References

[1] D. Brown, J. Smith, and P. Sun, *Information relaxations and duality in stochastic dynamic programs*, Operations research **58** (2010), 785–801.

[2] M. Klapp, A. Erera, and A Toriello, *The dynamic dispatch waves problem for same-day delivery*, under review (2016).

[3] ———, *The one-dimensional dynamic dispatch waves problem*, Transportation Science (2016), 1–14.

[4] M. Puterman, *Markov decision processes: discrete stochastic dynamic programming*, John Wiley & Sons, 2009.

[5] N. Secomandi and F. Margot, *Reoptimization approaches for the vehicle-routing problem with stochastic demands*, Operations Research **57** (2009), no. 1, 214–230.

Optimization algorithms for meal delivery operations

Damian Reyes, Alan Erera, and Martin Savelsbergh

January 5, 2017

1 Introduction

On-demand meal-ordering platforms – online marketplaces where diners order their favorite cravings from an array of restaurants – are growing at a fast pace [1], and the volume of meal delivery operations is rising quickly, opening up new economies of scope, scale, and density, which emerging providers are aiming to capitalize with the deployment of meal delivery networks. Such systems face complex capacity planning problems and increasingly large dynamic pickup and delivery problems that must be solved in (near) real-time [2, 5, 4, 3]. Due to the high dynamism and urgency of arriving orders [7], meal delivery is the ultimate last mile logistics challenge: a typical order is expected to be delivered within an hour (much less if possible), and within minutes of the food becoming ready, thus reduc- ing consolidation opportunities and imposing the need for more vehicles operating simultaneously and executing shorter routes (which can be costly).

A key to the success of meal delivery systems is their ability to respond to rapid swings in demand throughout the planning period (and the geography). Since employing a permanent fleet of vehicles can be prohibitively expensive, companies have resorted to "digital marketplace" business models – where the supply of drivers who are independent contractors [6] is controlled by economic incentives – an appealing strategy, first explored in the context of taxi and ride-sharing services, to externalize fixed costs while enhancing the ability to respond to sharp surges in demand. However, the independent contracting business model establishes a fundamentally different operating environment: in exchange for taking on some of the risks associated with demand uncertainty, drivers have a significant degree of autonomy, thereby adding yet another layer of complexity in the design of appropriate optimization technology.

In this paper we introduce optimization algorithms tailored to solve the driver assignment (vehicle routing) and capacity management (shift scheduling) problems in meal delivery. After a detailed exposition of the main features of our routing and scheduling algorithms we conduct extensive computer simulations to investigate the performance of our approach, and explore the characteristics of meal delivery systems, in particular those related to driver autonomy. While still preliminary, results suggest that our algorithmic ideas can be valuable in real-world implementations.

2 A rolling-horizon algorithm for driver assignment

2.1 A realistic simulation setup

To fully capture effects that propagate through time and geography, we focus on the study of performance metrics for a full day in a complete city. The instances used in our simulation study correspond to the activity logged by a major meal delivery company in Chicago, IL, during the

years 2015 and 2016. The number of individual orders arriving throughout a day in a single instance is typically above 2000 and can often exceed 3000.

The "real" time when orders are ready for pick-up, as retrieved from historical data, becomes known after the fact, as the "actual" pick-up availability of an order is checked before every optimization. For planning purposes, a constant preparation time is used (20 minutes from the placement time). If after this period the order is not available for pick-up, the ready time estimate is increased by f minutes, where f is the duration of the period between optimization runs.

Driver shifts start and end times, and their starting locations, are obtained from historical data. When drivers receive an assignment, they must immediately accept or reject it. Each decision is modeled as a binary random variable related to: the travel time to the assigned restaurant, and the proportion of assignments already rejected by the driver during their shift. After completing their last assignment, drivers are assumed to wait idle at their current location for a brief time. Afterwards, if still idle, drivers randomly select a nearby restaurant and move towards it. Closer restaurants are more likely to be chosen.

2.2 The algorithm

The proposed rolling-horizon matching-based algorithm solves an optimization problem every f minutes. At each optimization time t, it determines the available drivers best suited to deliver the orders in $U_t = \{o \in O_t : \ e_o \leq t + \Delta_u\}$, the set of *upcoming* orders with estimated ready times e_o at or before $t + \Delta_u$. Individual orders may be grouped to be picked-up as a batch and delivered together, in a specified route, by a single driver. Next, tentative driver - order assignments are defined by solving a sequence of matching problems. These assignments are examined and some, but not necessarily all, are communicated to drivers, following a specific "commitment strategy".

2.2.1 Batches and routes

To best utilize capacity, drivers may pick-up and deliver multiple orders, increasing the utilization of drivers at the expense of some freshness loss. To create routes, we decide how many orders should be batched together (target size of route), and then decide how orders are grouped and sequenced.

System intensity and a target batch size. A dynamic target size is defined in relation to the amount of work that must be completed with the available resources during a given period of time, *e.g* measured by means of a ratio of the form $^{(\#\text{orders ready})}/_{(\#\text{ drivers available})}$. It is intended to encourage quick deliveries when there are fewer orders than drivers, and favor larger batches when there are more orders than drivers and the system is under pressure. A parametric definition of the target order batch size at optimization time t is:

$$Z_t = \left\lceil \frac{|\{o \in O_t : \ e_o \leq t + \Delta_1\}|}{|\{d \in D_t : \ e_d \leq t + \Delta_2\}|} \right\rceil, \quad \Delta_1 > 0, \Delta_2 > 0 \tag{1}$$

where e_o is the estimated ready time of order o and e_d is the time when driver d becomes available for a new assignment. Note that it is possible that there are no drivers available before $t + \Delta_2$, in which case Z_t is set to some default value.

Creation of single-restaurant delivery routes. Once a system-wide target batch size Z_t has been determined at time t, the set of upcoming orders at each restaurant r, $U_{t,r}$, is partitioned into $m = \lceil {}^{|U_{t,r}|}/_{Z_t} \rceil$ batches, using a parallel insertion procedure:

Algorithm 1: Route generation at a single restaurant via parallel insertion

Input: Upcoming orders $U_{t,r}$, target batch size Z_t
Result: $m = \lceil |U_{t,r}|/Z_t \rceil$ batches to be assigned to drivers
Sort the orders in $U_{t,r}$ by non-decreasing estimated ready time;
Initialize empty routes (batches) $s_1, s_2, \ldots, s_m$;
for $o \in U_{t,r}$ **do**
  repeat
    Find the route $s \in \{s_1, \ldots, s_m\}$ and the insertion position i_s for order o into route s which results in the minimum increase in route cost, where the route cost of s is $\sum_{(p,q)\in s} TravelTime(p,q) + \beta \sum_{p\in s} ServiceDelay(p)$
  until $|s| < Z_t$ or insertion improves driver utilization;
  Insert o in route s at position i_s;
end

Note that once a batch reaches its target size, an order is only added if this increases driver utilization, i.e., the time per order delivered decreases. A multi-restaurant pick-up and delivery route generation variant has also been developed.

2.2.2 A sequence of matching problems

Before making assignments, upcoming orders are prioritized in three groups:

- **Group I**: orders whose target drop-off time is impossible to achieve.
- **Group II**: orders not in I which cannot be feasibly picked up at their estimated ready time.
- **Group III**: orders that do not fall into the prior categories.

A batch is assigned the highest priority of any of its elements. By creating assignments sequentially for these priority groups, urgent deliveries are likelier to find a better assignment than if all orders were part one matching. We use the following notation: N_s is the number of individual orders in route s; θ is a constant "penalty" for late pick-ups; $\pi_{s,d}$ is the pick-up time of batch s if assigned to driver d (by definition, $\pi_{s,d} \geq \max_{o\in s}\{e_o\}$); $\delta^s_{o,d}$ is the drop-off time of order o in batch s if assigned to driver d (depends on $\pi_{s,d}$); and $x_{s,d}$ is a binary variable for the assignment of batch s to driver d. The objective of the matching balances driver utilization, and loss of freshness:

$$\max \sum_{s\in U_t} \sum_{d\in D} \left(\frac{N_s}{\max_{o\in s}\{\delta^s_{o,d}\} - e_d} - \theta \left(\pi_{s,d} - \max_{o\in s}\{e_o\} \right) \right) x_{s,d}$$

2.2.3 Commitment strategies

Two-stage lazy commitment. An assignment can be decomposed into two travel segments: "inbound" travel to the restaurant, and an "outbound" delivery route. For each tentative assignment (s, d), of order batch s and driver d, in the solution of a matching problem, this strategy dictates:

1. If d can reach restaurant r_s before $t + f$ and all orders in s are estimated to be ready by $t + f$, make a *final commitment* of d to s: instruct d to travel to r_s, pick up and deliver orders in s.
2. If d cannot reach r_s by $t + f$, but completes their last assignment before $t + f$, make a *partial commitment* for d: instruct d to travel to r_s and wait there for a finalized order assignment.
3. If d cannot start a new assignment by $t + f$, ignore the assignment.

4. *Exception:* If any order in s has been ready for more than x minutes, override the rule and make a final commitment.

The motivation to send a driver to the restaurant without committing them to deliver a specific batch is that, while travel should begin, the driver can be matched again in the next optimization, while en-route,and the batch assigned may change. On the other hand, if the driver is busy before $t + f$, waiting for the next optimization will not delay the pick-up or delivery of any order.

Two-stage additive commitment. This strategy differs from the previous one only in case 2:

2. If d cannot reach r_s by $t + f$, but completes their last assignment before $t + f$, make a *partial commitment*: instruct d to travel to r_s and wait there for a finalized order assignment, which is guaranteed to include orders in s, and possibly more.

We call this variation "additive" because if s is partially committed to d at time t, then at optimization time $t+f$, we force the batch assigned to d to include s. Hence, orders in a batch partially assigned to d are guaranteed to be in the batch finally assigned to d. Such consistency is desirable if drivers are paid in relation to some property of the orders they deliver.

3 Driver shift scheduling

Our approach to driver capacity scheduling relies on the solution of a "shift-cover" problem, which, given a set of allowed driver shift structures, allocates enough drivers to cover an "activity profile" – an estimate of the number of drivers required in the system at any time of a day. The shift-cover problem is to minimize the total shift length required to cover the activity profile, i.e., to minimize the area under a feasible resource profile, given the shift start times and lengths allowed. The shift-cover problem can be formulated as a linear program.

4 Preliminary results

After tuning the algorithm parameters on a small sample of instances, we have conducted a series of simulations in a set of day-long instances from Chicago, IL.We focus on two key performance measures: (1) **click-to-door**, the time since an order is placed until the order is delivered, and (2) **ready-to-door**, the time since an order is ready for pick-up until the order is delivered. The table below summarizes our main preliminary results, with the caveat that not all the five experiments shown have been conducted using the final implementation of the algorithm: all experiments must be run again on a better controlled setting for a final draft. To protect the confidentiality of our industry partner, we report all values relative to historical performance, *e.g.* a simulation with click-to-door of 1.08 indicates a value 8% larger than in the historical record.

Average performance	**Click-to-Door**	**Ready-to-Door**
Baseline settings (historical shifts)	1.257	1.339
Commitment Strategy		
Two-Stage Addditive	1.253	1.334
Two-Stage Lazy	1.261	1.345
Routes allowed		
Single-restaurant	1.249	1.326
Multi-restaurant	1.266	1.352
Assignment rejection assumption		
Always accept	1.142	1.167
Exponential rejection rule	1.146	1.173
Optimized shift-schedules (baseline settings)	0.92	0.86

References

[1] Evan Bakker. *The on-demand meal delivery report: Sizing the market, outlining the business models, and determining the future market leaders*. `http://read.bi/2bU7EuD`, 2016. Accessed: 2016-12-01.

[2] G. Berbeglia, J.F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010.

[3] Francesco Ferrucci and Stefan Bock. Real-time control of express pickup and delivery processes in a dynamic environment. *Transportation Research Part B: Methodological*, 63:1 – 14, 2014.

[4] G. Ghiani, E. Manni, and B.W. Thomas. A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem. *Transportation Science*, 46(3):374–387, 2012.

[5] H.N. Psaraftis, M. Wen, and C.A. Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016.

[6] Reuters. *How Amazon is making package delivery even cheaper*. `http://fortune.com/2016/02/18/amazon-flex-deliveries/`, 2016.

[7] Rinde RS van Lon, Eliseo Ferrante, Ali E Turgut, Tom Wenseleers, Greet Vanden Berghe, and Tom Holvoet. Measures of dynamism and urgency in logistics. *European Journal of Operational Research*, 253(3):614–624, 2016.

VEHICLE ROUTING MODELS & APPLICATIONS

FB3: STOCHASTIC ROUTING

Friday 1:00 – 2:30 PM

Session Chair: Stefan Minner

1:00 **Vehicle Routing with Space- and Time-Dependent Stochastic Travel Times**

[1]*Stein W. Wallace**, [2]*Zhaoxia Guo*, [3]*Michal Kaut*

[1]*Norwegian School of Economic*, [2]*Sichuan University*, [3]*Norwegian University of Science and Technology*

1:30 **A Two-Phase Safe Vehicle Routing and Scheduling Problem: Formulations and Solution Algorithms**

[1]*Aschkan Omidvar**, [2]*Eren Ozguven*, [2]*Arda Vanli*, [3]*Reza Tavakkoli-Moghaddam*

[1]*Department of Civil and Coastal Engineering-University of Florida*, [2]*Florida State University*, [3]*University of Tehran*

2:00 **Optimal A-Priori Tour and Restocking Policies for the Vehicle Routing Problem with Stochastic Demands**

[1]*Alexandre Florio*, [1]*Richard F. Hartl*, [2]*Stefan Minner**

[1]*University of Vienna*, [2]*Technische Universität München*

Vehicle routing with space- and time-dependent stochastic travel times

Stein W. Wallace* Zhaoxia Guo† Michal Kaut‡

Vehicle routing, in all its variants, is one of the most studied problems in logistics. But for historical as well as numerical reasons, the vast majority of papers are on deterministic problems. Stochastic versions have started to occur, see for example the reviews (Pillac et al. 2013, Oyola et al. 2016, Ritzinger et al. 2016). The most studied stochastic phenomenon is demand, followed by travel time (speed) , service time, and finally, probably, random occurrence of customers.

Most vehicle routing problems (VRPs) are solved using heuristics, and a natural part of doing so is to evaluate the objective function for a given solution. If that was easy, most existing heuristics for vehicle routing could fairly easily be adopted to the case of dependent stochastic travel times (or speeds). After all, most heuristics have two parts, the search part and the evaluation part. This paper is only about the evaluation part. In order to demonstrate our approach we shall need both a test case within the family of VRPs and a heuristic, but neither of these represent contributions.

Our contributions are closely related to the many challenges set out in Gendreau et al. (2016) on how to represent the high-dimensional dependent random vector of travel times, how to generate scenarios for the underlying stochastic program, and how to perform function evaluations in the case where travel times are stochastic and dependent in time and space. By "time dependence" we mean that the travel time on a link in one period is correlated with the travel time on the same link in other nearby time periods. By "space dependence" we mean that travel times on close by links are correlated, so that if there is a traffic jam on one link, most likely, but not for sure, there is a long travel time also on neighboring links. This must be distinguished from literature (stochastic as well as deterministic) covering "time dependent travel times", but where the time dependence only means that travel times (or expected travel times in a few cases) are different in different time periods. To the best of our knowledge, no paper allows the stochastics in travel times themselves to be time dependent; even the stochastic approaches assume independence. In addition, space dependence has not been considered in the VRP literature, although real-world travel times are both time- and space-dependent.

*NHH Norwegian School of Economics, Bergen, Norway
†Business School of Sichuan University
‡SINTEF, Trondheim, Norway

Hence, our starting point is very general. We are concerned with the evaluation of the objective function for a feasible solution (a set of vehicle routes) to any VRP, when travel times are stochastic and dependent in time and space. The major question will be how we can represent the stochastics, including how scenarios can be generated (as discrete random variables will be needed), and how many scenarios are needed. The number of random variables will easily be in the thousands (the number of road links times the number of time periods), and care has to be taken in order to handle such big dimensions.

Modeling the routes

In VRP formulations, the nodes in the underlying network are normally the customers, described by the set $\mathcal{N} = \{1, \ldots, N\}$, and travel times (or distances) are modeled for pairs of customers. But if we have data for travel times in an area, they are most likely "on the map" – on the real road network – and not on pairs of customers. We shall describe this map by the directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V} = \{0, 1, \ldots, V\}$ are the vertices (nodes) and $\mathcal{L} = \{1, \ldots, L\}$ are the (road) links. We have that $\mathcal{N} \subset \mathcal{V}$, and $\mathcal{V}$ contains node 0, which is used to represent the depot. In deterministic models, moving from distances on the real road network to pairs of customers is not much of a problem. We can start from $\mathcal{G}$ and calculate the shortest travel time between each pair of customers, and proceed using customer-pair travel times. But if travel times are random, this is not quite as easy.

The shortest travel time path for a pair of customers will vary depending on the realizations of road link travel times. In principle, one could calculate the distribution of travel times for a given pair, but it would be rather challenging, as dependencies between travel times between different pairs of customers would be extremely hard to describe. Whenever two pairs of nodes shared a road link, the stochastic customer-pair travel times would be positively correlated. Note that even if all road link travel times were independent, the pairwise customer travel times would not. So assuming independence for customer-to-customer travel times would imply that no pairs ever shared a road link on the map, which to us seems extremely unrealistic.

If we accept that the stochastic data for the VRP lives on $\mathcal{G}$, another modeling question arises. Will a route in the VRP be given as a sequence of nodes from $\mathcal{V}$ in which two consecutive nodes are directly connected by a road link or a sequence of customers from $\mathcal{N}$? Both may make sense, and the two may of course occasionally coincide. In this work we focus on the case when a route is a sequence of customer nodes, but our general approach covers many variations. Our main focus is to handle travel time dependencies consistently.

The above discussion is done as if there was only one time period. But we shall in fact assume there are P time periods, and that there are travel time dependencies not only in space but also in time. We therefore assume that a route in the solution to a VRP is not just a sequence of nodes but also a starting time from the depot.

Multi-dimensional distributions and scenario generation

Stochastic programs need discrete random variables. Creating these discrete random variables from the background data (or from some modeling activity) is called scenario generation. For VRPs with time- and space-dependent stochastic travel times, this is a major undertaking, in fact, most likely the main bottleneck for solving such problems. And if the sole goal of the scenario generation is to come up with examples in order to test new algorithms, the issue is as straightforward as it is challenging; to come up with test cases in very high dimensions *that are meaningful for such investigations*.

Most of the literature on scenario generation is about how to generate scenarios from a distribution of some sort. That is an issue here as well, but there is another more fundamental question when we have dependence in very high dimensions: To find a meaningful distribution that can be the starting point for scenario generation. This is particularly true if we wish to sample scenarios, as sampling requires something to sample from. We shall spend some time here on analyzing the difficulty of coming up with a reasonable probability distribution, as that is a problem likely to be an issue for anybody facing dependent travel times. In our test cases we operate with up to 25,080 dependnt random variables. In such a case we are facing over 300 million distinct correlations. Obviously, some shortcuts are needed.

We use the scenario-generation method from Kaut (2014) that allows for specifying marginal distributions plus a subset of correlations, so we can concentrate on the most important pairs of variables: speeds on links sharing a node (neighbors) in one period and speeds on a given link in two consecutive periods. Notice that this is consistent with the structure of solutions to VRPs – routes. This way, the number of specified correlations (dependencies) become linear in the number of random variables. The method has the property that marginal distributions are retained, so, in particular, the expected length of any route is correct in any scenario set. But the method will produce some strange correlations among pairs of random variables that are not part of the specification. We shall see that this can be handled.

Stability

Gendreau et al. (2016) point out that the number of scenarios will *likely be quite large*. But how large is that? The classical problem here is a trade-off between a solvable problem but where the results are just noise due to a bad representation, and a numerically unsolvable problem (due to size) but where the representation is good.

This is a central question for any attempt to solve stochastic VRPs, and the answer will depend on what method is used to generate scenarios. Since scenarios will be expensive in the VRPs, meaning that the overall size of the optimization model very much depends on it, and just storing the scenarios might be an issue, we want to have as few as possible. But at the same time we need to know the quality of the solutions obtained using the scenarios. In the talk we illustrate one way to test this question for a *given* optimization problem, and a *given* scenario generation method. We shall show how a stability test

Table 1: Out-of-sample values for SP and DP solutions for some triplets (N, K, P)

(18,3,3)		(32,5,3)		(48,7,3)		(48,7,30)		(48,7,60)	
DP	SP	DP	SP	DP	SP	DP	SP	DP	SP
0.87	0.85	2.26	1.98	6.07	5.46	12.68	11.38	12.68	11.65

can be developed to determine the necessary number of scenarios.

Test case

As a general observation, stochastic speeds are only of major importance when there is an asymmetry between the effects of being early and being late. In particular, just minimizing expected travel times with no special constraints such as time windows (hard or soft) will not make it worthwhile to study stochastics. Hence, time windows (where the effects of being early or late are very different) or penalties for particularly long working days, are cases where stochastics is likely to matter.

As an example, we consider a two-stage stochastic VRP where there are penalties for late arrivals of vehicles back to the depot, but no gains for early arrivals. The instances are defined on a map of Beijing with 142 nodes and 418 road links. An instance is defines by N (the number of customer nodes), K (the number of vehicles) and P (the number of periods). N, K, vehicle capacity and customer demands are based mainly on numbers from http://neo.lcc.uma.es/vrp/known-best-results/ so that the numbers for each test instance make sense. So the cases we study have $418 \times P$ random variables. The objective is to minimize expected overtime pay.

So the first stage decisions is to pick K routes, in terms of sequences of customer nodes, and the second stage decision is to travel these routes, picking paths between them (as N is much smaller than V) to minimize expected overtime.

Let us report a few numerical results, using the Beijing map, at this point. The sole purpose is to show that this VRP is indeed one where stochastics matter. Table 1 shows the out-of-sample results of five problem instances. As expected, the SP generates much lower out-of-sample values than the DP. Taking problem (48,7,3) as an example, the out-of-sample value (5.46 hours) generated by the SP is around 10% less than the value (6.07 hours) generated by the DP. This shows that stochastics has a large effect on the solutions in the test case. So, the main point of Table 1 is that our test case in this paper is indeed one where stochastics matter. Notice that the largest case has $418*60 = 25,080$ dependent random variables.

Experimental Results

The basis for our tests is still the map of Beijing with 142 nodes and 418 road links. Consider Table 2. We have there studied stability for different values of N, K and P.

Table 2: Number of scenarios needed and corresponding relative errors

(N,K)	$P=3$		$P=5$		$P=15$		$P=30$		$P=60$	
	S	RD	S	RD	S	RD	S	RD	S	RD
(18,3)	50	2.3%	60	1.0%	30	1.0%	30	0.9%	20	0.9%
(32,5)	50	1.0%	15	1.0%	25	0.8%	15	1.0%	15	0.7%
(48,7)	15	0.9%	10	0.9%	10	0.7%	10	0.6%	10	0.7%
(64,9)	10	0.9%	10	0.9%	10	0.6%	10	0.6%	10	0.3%

Ideally, stability should depend on just P and L, as they directly describe the random variables, but as expected, stability will depend on N and K as well, to some extent. The main result to observe here is that for the larger problems, only ten scenarios are needed to achieve solutions with a 1% error. We find that very encouraging. For the somewhat smaller problems more scenarios are needed. This is a result of the fact that our scenario generation method is a heuristic.

CPU times

The CPU time for solution evaluation involves two parts. The first part – $T1$ – is solution-independent, and is used to generate the set of scenarios according to the chosen scenario generation method, in our case Kaut (2014). The number of scenarios is taken from Table 2.

The second measure shows the CPU time (in seconds) needed to find the objective function value for one feasible solution. We do this by generating 5000 feasible solutions for each of 20 problem instances, and report the average as $T2$. These instances and the corresponding results are shown in Table 3. The tests were carried out on a laptop with Intel Core i7-5500U CPU @2.4GHz and 8 GB RAM using MATLAB version R2009a.

Let us consider $P = 30$. This could, for example, correspond to a 10-hour day split into 20 minutes intervals. In this case it too about 40 minutes to create the scenarios. It might seem excessive to use forty minutes on ten scenarios. But remember that these ten scenarios involve over 125,000 different numbers, and it is the carefulness of setting these up that makes it possible to have so few scenarios. There are certainly quicker ways to set up ten scenarios, but the question will then be if such stability can be achieved. And again, this needs to be done only once per map.

Ones a scenario set is defined and found stable, it is $T2$ that determines the speed of solving the stochastic VRP. In particular, it is interesting to see how much slower a stochastic VRP solves compared to a corresponding deterministic case. To get an estimate using our results, consider as an example the number 16 in the lower right corner of Table 3. It shows that for this large case, with over 25,000 random variables, it took 16 seconds on average to evaluate one feasible solution. This is a case with nine routes and 10 scenarios, so a deterministic case would take about one tenth of that to evaluate, since finding the deterministic travel time amounts to one scenario evaluation.

Table 3: CPU times in seconds needed for stochastic instances

(N,K)	$P=3$		$P=5$		$P=15$		$P=30$		$P=60$	
	$T1$	$T2$	$T1$	$T2$	$T1$	$T2$	$T1$	$T2$	$T1$	$T2$
(18,3)	38	1.3	102	3.8	569	4.7	3590	4.5	19509	11.7
(32,5)	38	2.6	18	1.6	558	5.6	2448	5.1	19493	8.1
(48,7)	9	1.6	15	1.5	323	3.7	2367	6.3	18811	13.8
(64,9)	7	1.3	15	1.9	323	4.4	2367	8.7	18811	16.0

In other words, the scenario count in Table 2 is a good measure of how much slower a stochastic approach would run. This is why it is so critical to have a good method for generating scenarios so that stable results (at a chosen accuracy) can be achieved with as few scenarios as possible. Our scenario generation method is particularly well suited to VRPs.

References

Gendreau, G., O. Jabali, and W. Rei (2016). Future research directions in stochastic vehicle routing. *Transportation Science 50*(4), 1163–1173.

Ichoua, S., M. Gendreau, and J.-Y. Potvin (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operations Research 144*, 379–396.

Kaut, M. (2014). A copula-based heuristic for scenario generation. *Computational Management Science 11*(4), 503–516.

Oyola, J., H. Arntzen, and D. L. Woodruf (2016). The stochastic vehicle routing problem: a literature review. Optimization online, e-print ID 2016-01-5299.

Pillac, V., M. Gendreau, C. Gueret, and A. L. Medaglia (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research 225*(1), 1–11.

Ritzinger, U., J. Puchinger, and R. F. Hartl (2016). A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research 54*(1), 215–231.

A Two-Phase Safe Vehicle Routing and Scheduling Problem: Formulations and Solution Algorithms

Aschkan Omidvar [a*], Eren Erman Ozguven [b], O. Arda Vanli [c], R. Tavakkoli-Moghaddam [d]

[a] *Department of Civil and Coastal Engineering, University of Florida, Gainesville, FL 32611, USA*
[b] *Department of Civil and Environmental Engineering, FAMU-FSU College of Engineering., Tallahassee, FL, 32310, USA*
[c] *Department of Industrial and Manufacturing Engineering, FAMU-FSU College of Engineering, Tallahassee, FL, 32310, USA*
[d] *School of Industrial Engineering, College of Engineering, University of Tehran, Iran*

1. Introduction

Traffic crashes and congestion are major costs to the collective and social well-being. According to the Federal Highway Administration, traffic crashes imposed an economic cost of $242.0 billion to the U.S. economy in 2010 [1]. Total estimated cost of congestion to Americans was also approximately $124 billion in 2013 [2]. These figures show the vital influence of congestion and crashes on our daily trips. For many years, vehicle routing and scheduling have been used to investigate the effects of congestion on the roadway networks. Example studies include dynamic and stochastic vehicle routing applications that focus on the service time and demand at the nodes [3], models considering speed variations and dynamic travel times [4], stochastic and time-varying network applications [5, 6], and queueing theory-based applications [7]. For further information on the static and dynamic vehicle routing problems (VRP), please refer to [3, 8]. To the best of authors' knowledge, traffic safety, in terms of crash risk on roadways, has not been introduced to the graph theory and transportation network optimization. This study is an important step towards filling this gap.

2. Mathematical Modelling

We propose a two phase dynamic vehicle routing and scheduling optimization model that identifies the safest routes, as a substitute for the classical objectives given in the literature such as shortest distance or travel time, through (1) avoiding recurring congestions, and (2) selecting routes that have a lower probability of crash occurrences and non-recurring congestion caused by those crashes. This modeling approach has two phases:

Phase I. In this phase, we formulate a mixed-integer linear programming which takes the dynamic speed variations into account on a graph of roadway networks, according to the time of day. The probability of crash as a function of the speed will be predicted using logistic regression models. However, in most locations, with an increase in the traffic density, and reduction in speed, the probability of having a crash increases [9]. Therefore, the first graph model identifies the routing of a fleet and sequence of nodes on the safest feasible paths. Several constraints, such as hard and soft time windows on each node, capacity, operation hours, and number of vehicles are introduced to ensure the fast and quality service. The speed variation with respect to the hour of the day is obtained via queuing models (e.g., *M*/*M*/1, *GI*/*G*/*m and M*/*G*/1) to capture the stochasticity of travel times more accurately. So, in our first model, travel times are dynamic and a function of speed. The objective function consists of two main components: (1) crash rates on each segment according to the time of the day and (2) modified Planning Time Index (PTI) [10], which is a function of travel time. In summary, the two components of the objective function allows one to (a) increase the trip safety by choosing segments with low crash probability, and (b) to avoid recurring and non-recurring congested segments. Figure 1a shows the initial network, and Figure 1b indicates the routes identified as a result of Phase 1.

Phase II. After the construction of the routes through the first model, we propose a second optimization model which considers each route as an independent transit path (fixed route with fixed node sequences). This model tries

* Corresponding author. Tel.: +1-850-405-6688
E-mail address: Aschkan@ufl.edu

to improve the service timing and avoid congestion by rescheduling the departure times of each vehicle from each node and finding the optimal speed on each arc. Figure 1c illustrates the scheduling decisions made in Phase 2. In the past two decades, heuristic and meta-heuristic algorithms such as tabu search, genetic algorithm, simulated annealing and ant colony optimization, have been successfully used to solve optimization models related to routing and scheduling [11, 12] and we will explore feasibility of these tools in our problem. In the following section, we discuss such solution methods to solve Phase 1 and Phase 2 problems.

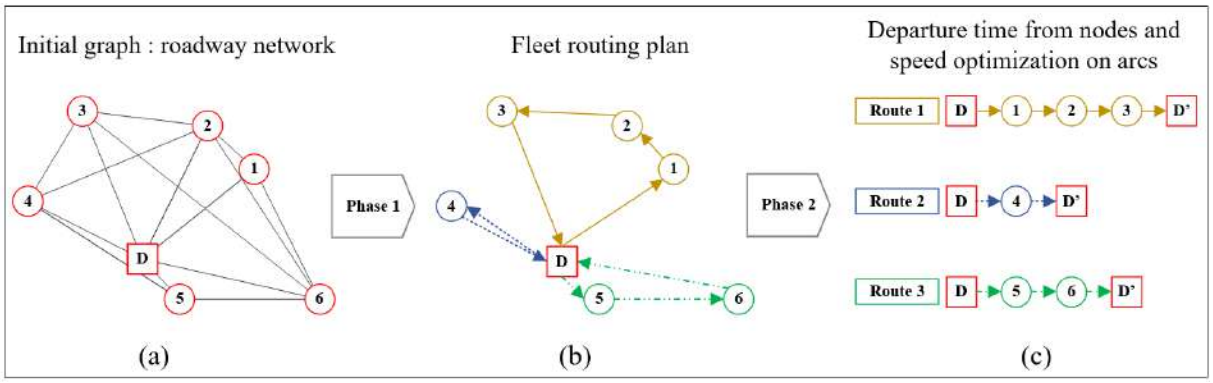


Figure 1 A schematic representation of modelling approach: (a) Roadway network (b) Result of Phase 1: routes identified (c) Result of Phase 2: route schedules determined, including departure time and the travel speed at each node

3. Solution Approach

Vehicle routing and scheduling are classified as NP-hard problems, hence, solving these models to optimality for medium to large scale networks is not computationally feasible. In this study, in order to achieve a fair tradeoff between the computation time and solution accuracy, we propose to solve this problem using a hybrid algorithm that combines a novel heuristic algorithm and an advanced meta-heuristic technique. A schematic approach of the solution approach is depicted in Figure 2. The solution of this high dimensional problem strongly depends on the initial solution. Therefore, in order to obtain a good initial feasible solution to the problem, we will use the heuristic algorithm developed by the authors in [13]. This approach uses a greedy local neighborhood search algorithm that works with polar coordinates of the nodes, where the depot is the initial pole, and expands the search range along the radius and azimuth, successively (See Figure 2a). The initial solution is then fed to the meta-heuristic algorithm, which develops itself to improve the solution quality at each iteration leading to suboptimal solutions. (Figure 2b).

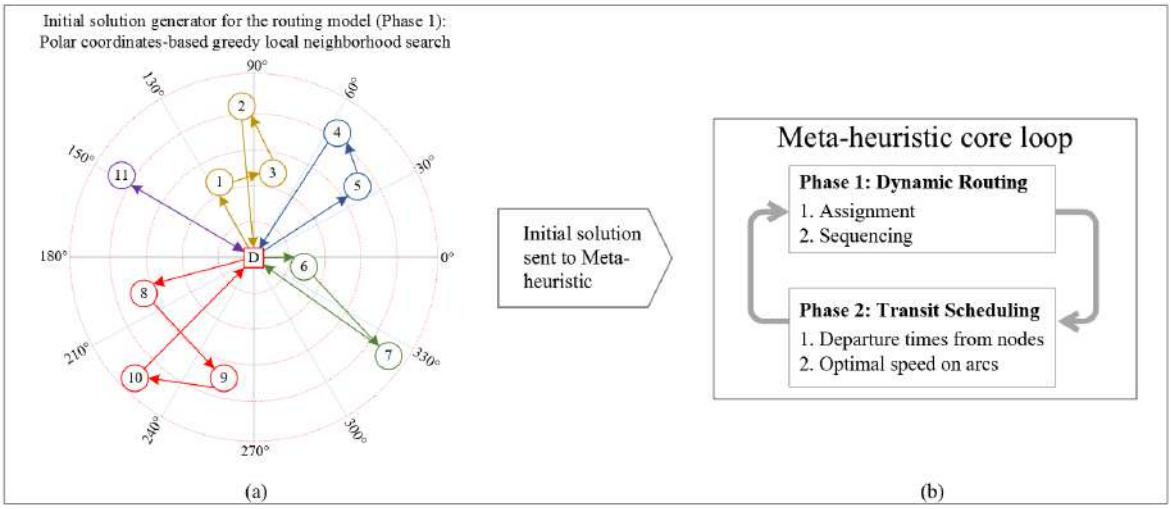


Figure 2 A schematic representation of solution approach (a) Polar coordinates-based heuristic to generate the initial solution (b) Meta-heuristic to solve Phase 1 and Phase 2 problems

The validity of both mathematical models is tested against several small scale test problems (See the next section). We also plan to evaluate the accuracy and responsiveness of the model with respect to several medium and large scale benchmark networks followed by case study applications in the State of Florida. During all these steps, we

plan to compare our model results to those from the classical models that only focus on the minimization of the travel time. This will let us identify those safe routes, which may have a higher travel time but less crash risk compared to those obtained from the classical models.

4. Preliminary Results

Figure 3 demonstrates a small test network in Miami, Florida, consisting of 4 nodes and a depot of interest (see Figure 3a). We studied arcs 3-4 and 2-4 for routing and scheduling decisions. For the two arcs, we modeled the relation between the observed crash frequencies (Figure 3b) and the speeds (Figure 3c) through GIS-based methods and logistic regression techniques (Figure 3d), and visually present this relation in terms of logit curves [14]. Please note that we focus on one iteration of the overall optimization process with two alternatives to travel from Node 4 to Node 3 (1) directly (Figure 3e), or (2) through node 2 (Figure 3d). The model in Phase 1 receives the crash probability and modified PTI as inputs, and determines the optimal route. In Phase 2, on the other hand, the model optimizes the scheduling of these current routes. Figure 3 shows how results may change according to the recurring and non-recurring traffic conditions for a safer and more reliable routing and scheduling. Figure 3e (travel directly) is optimal in the morning while Figure 3f (travel through node 2) is optimal in the afternoon.

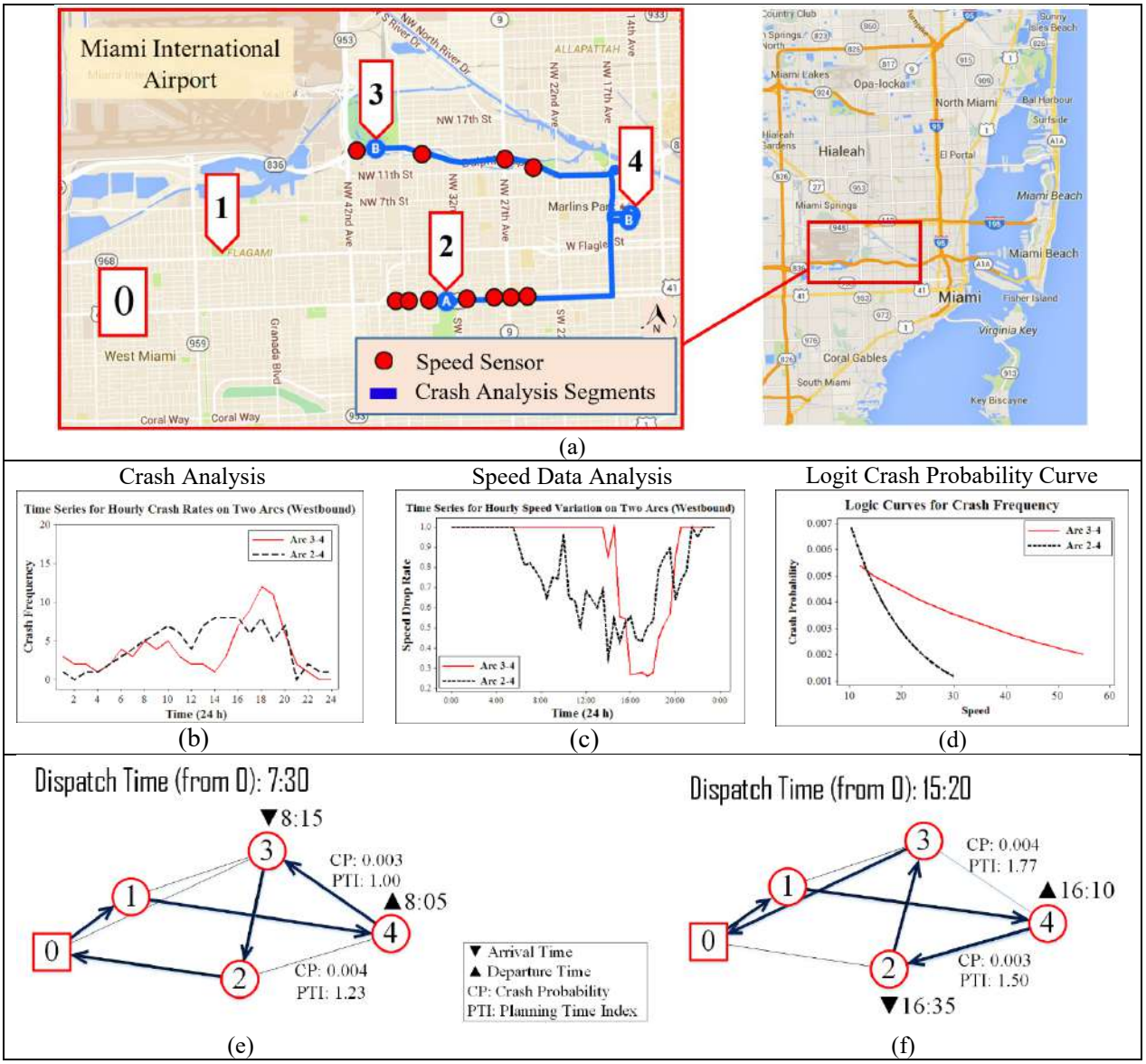


Figure 3 A case study in Miami, Florida

References

1. Federal Highway Administration website. (2015). http://safety.fhwa.dot.gov/facts_stats/.
2. ECMT (2007). Managing Urban Traffic Congestion.
3. Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. European Journal of Operational Research, 225 (1), 1-11.
4. Malandraki, C., & Daskin, M. S. (1992). Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. Transportation Science, 26 (3), 185-200.
5. Ziliaskopoulos, A. K., & Mahmassani, H. S. (1993). Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications. Transportation research record, 94-94.
6. Miller-Hooks, E. D. (1997). Optimal routing in time-varying, stochastic networks: algorithms and implementations. Ph.D. thesis, The University of Texas at Austin.
7. Van Woensel, T., & Vandaele, N. (2007). Modeling traffic flows with queueing models: a review. Asia-Pacific Journal of Operational Research, 24 (04), 435-461.
8. Toth, P., & Vigo, D. (Eds.). (2014). Vehicle routing: problems, methods, and applications (Vol. 18). Siam.
9. Blincoe, L. J., Miller, T. R., Zaloshnja, E., & Lawrence, B. A. (2015, May). The economic and societal impact of motor vehicle crashes, 2010. (Revised) (Report No. DOT HS 812 013). Washington, DC: National Highway Traffic Safety Administration.
10. Lomax, T., Schrank, D., Turner, S., & Margiotta, R. (2003). Selecting travel reliability measures. Texas Transportation Institute monograph.
11. Baldacci, R., Mingozzi, A., & Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. European Journal of Operational Research, 218(1), 1-6.
12. Golden, B. L., Raghavan, S., & Wasil, E. A. (Eds.). (2008). The vehicle routing problem: latest advances and new challenges (Vol. 43). Springer Science & Business Media.
13. Omidvar, A., & Tavakkoli-Moghaddam, R. (2012). Sustainable vehicle routing: Strategies for congestion management and refueling scheduling. In Energy Conference and Exhibition IEEE International, 1089-1094.
14. Omidvar, A., Ozguven, E.E., Vanli, O.A., Moses, R. (2016). Understanding the factors affecting the frequency and severity of aging population-involved crashes in Florida, Advances in Transportation Studies. (Article in Press)

Optimal A-Priori Tour and Restocking Policies for the Vehicle Routing Problem with Stochastic Demands

Alexandre Florio[1], Richard F. Hartl[1], Stefan Minner[2]

January 5, 2017

[1] Department of Business Administration, University of Vienna, Vienna, Austria, alexandre.de.macedo.florio@univie.ac.at, richard.hartl@univie.ac.at,

[2] Logistics and Supply Chain Management, TUM School of Management, Munich, Germany, stefan.minner@tum.de

1 Motivation and modeling approach

The Vehicle Routing Problem (VRP) is one of the classic optimization problems in logistics. In the VRP, a set of vehicles is available for delivering (or picking-up) goods to (from) customers. Each customer has some demand, and the vehicles have capacities. A valid route begins at some designated depot, visits a sequence of customers, and finishes again at the depot. Moreover, in a valid route the sum of the demands of the visited customers does not exceed the capacity of the vehicle. The objective in the VRP is to design a set of valid routes of minimum total cost visiting all the customers. The cost of a route is usually measured in terms of its total length or duration.

In many practical scenarios some of the problem data might not be known at route planning time. For example, traffic conditions might vary causing uncertainty in the travel times. In other cases, customer demands are unknown and only disclosed upon arrival of the vehicle at their locations. When randomness is present in the input data, the general class of derived problems is called stochastic VRP ([5]). Many subclasses arise depending on the element (or combination of elements) considered stochastic. We consider

in this paper the case of VRP with stochastic demands (VRPSD), which undoubtedly is the most studied by the scientific community.

Most of the approaches to solve the VRPSD are based on two-stage stochastic programming. The first-stage decisions determine the *a priori tour* (single-vehicle case) or *a priori routes* (multi-vehicle case). The second-stage decisions determine the *recourse action* with associated *recourse cost*. The objective in these models is the minimization of the total expected cost, composed of the first-stage (or a priori) cost and the expected recourse cost. The recourse costs are determined by the *restocking policy*, which is a set of rules that govern when the vehicle should perform a replenishment trip. Virtually all methods developed for the VRPSD assume the *detour-to-depot* restocking policy, first stated in [3]. This policy prescribes a replenishment trip if and only if the vehicle does not have sufficient capacity to serve the current customer. This is clearly non-optimal, especially when the current customer is located far away from the depot. However, incorporating more involved policies in the (already complicated) two-stage models may quickly lead to intractability. On the other hand, the problem of finding the optimal restocking policy given a fixed a-priori route has been solved by a simple stochastic dynamic programming algorithm ([7]). An optimal restocking policy potentially performs *preventive* replenishment trips to avoid failures further down the route where a return trip could be costly, and thus is significantly more sophisticated than the detour-to-depot policy. Optimal restocking policies have been incorporated in intelligent heuristics for the VRPSD ([6]). However, approaches to exactly solve the VRPSD considering the use of the optimal restocking policy are not available. Our main contribution is to close this gap in the literature, addressing concerns that methods for the VRPSD considering new recourse actions need be developed ([4]).

The distinctive characteristic of our model is the simultaneous optimization of the a-priori tour and restocking policy. We do so by first defining a general Stochastic Dynamic Program allowing both sequencing and restocking decisions, and then restricting such process by applying *policy constraining*. These constraints define a class of control policies for the general SDP which must comply with some a-priori tour. By finding the optimal policy in this class, we also arrive at the optimal restocking policy for some a-priori tour. Therefore, we do not rely on the dynamic programming algorithm of [7] for route

evaluation, but have the optimal restocking policy naturally as an output of the model. In some sense, we unify the a-priori problem (finding an optimal tour given a fixed policy) and the restocking problem (finding an optimal policy given a fixed tour) in the same model, and hence we call it a *unified model*. This is a different approach than the one proposed by [1] for the generalized VRP with stochastic demands, where route cost evaluation relies on the dynamic programming algorithm.

The techniques we employ to build such model, in particular the policy constraining ideas, provide a general framework for developing other models for the VRPSD, which we also consider a significant contribution. Unfortunately, the price to pay for having a unified model is a mixed-integer linear program with a number of decision variables and constraints polynomial in the capacity of the vehicle. Nevertheless, by using policy constraining and decomposition in different ways, one might derive models which compromise on the optimal restocking policy in favor of simplicity, still doing better than the traditional detour-to-depot recourse models.

2 Computational results

The model was implemented and solved with CPLEX version 12.6.1 branch-and-bound MILP solver. The subtour elimination constraints were added dynamically to the model, whenever at some node of the search tree they were violated. All experiments were performed on a Intel i7-4790 3.6GHz 4-core processor with available memory of 16 gigabytes. Parallel processing was used whenever available. Our tests were conducted on a set of simplified instances derived from literature instances. In these instances, the demand of some customer i is zero with probability p_0 and u_i with probability $1 - p_0$, where u_i is an integer ranging from 1 to 4. In all instances the capacity of the vehicle is set to 10. The *load* of an instance is defined as the total expected demand divided by the capacity of the vehicle. Each instance was tested under six different load scenarios, obtained by varying p_0. We compare the results with the solution obtained by applying the optimal restocking policy to the TSP-optimal a priori tour (both orientations considered). We also compare with the optimal solution when the detour-to-depot policy is used.

The standard implementation was able to solve instances of up to 50 nodes with a low

vehicle load (0.75) within the time limit of two hours. Considering all load scenarios, the optimal solution obtained with the unified model is on average 2.8% better than the optimal detour-to-depot solution. This difference can be up to 8.7% in high load scenarios. The results also indicate that the optimal detour-to-depot solution is inferior to the TSP-optimal a-priori tour coupled with the optimal restocking policy. The difference is relatively small for low load (0.75 to 1.00) scenarios (average of 0.8%, maximum of 2.1%), but becomes significant in medium load (1.25 to 1.50) scenarios (average of 2.9%, maximum of 4.9%) and high in high load (2.00 to 2.50) scenarios (average of 4.1%, maximum of 7.1%). Considering the much smaller computational effort required to obtain the TSP-solution and optimal restocking policy, we conclude that using the detour-to-depot policy in the single-vehicle case is unjustified. With the unified model, we are able to further improve the solution quality. In low load scenarios, the difference is rather small (average of 0.1%, maximum of 0.4%) and increases in higher load scenarios (average of 0.8% and maximum of 1.3% for medium load, and average of 1.7% and maximum of 2.0% for high load).

References

[1] Benjamin Biesinger, Bin Hu, and Günther Raidl. An integer l-shaped method for the generalized vehicle routing problem with stochastic demands. *Electronic Notes in Discrete Mathematics*, 52:245–252, Jun 2016.

[2] Moshe Dror, Gilbert Laporte, and Pierre Trudeau. Vehicle routing with stochastic demands: Properties and solution frameworks. *Transportation Science*, 23(3):166–176, Aug 1989.

[3] Michel Gendreau, Ola Jabali, and Walter Rei. 50th anniversary invited article—future research directions in stochastic vehicle routing. *Transportation Science*, 50(4):1163–1173, 2016.

[4] Michel Gendreau, Gilbert Laporte, and René Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, Jan 1996.

[5] Wen-Huei Yang, Kamlesh Mathur, and Ronald H. Ballou. Stochastic vehicle routing problem with restocking. *Transportation Science*, 34(1):99–112, Feb 2000.

[6] James R. Yee and Bruce L. Golden. A note on determining operating strategies for probabilistic vehicle routing. *Naval Research Logistics Quarterly*, 27(1):159–163, Mar 1980.

VEHICLE ROUTING MODELS & APPLICATIONS

FC3: STATISTICAL DATA ANALYSIS FOR ROUTING AND LOCATION

Friday 2:45 – 4:15 PM

Session Chair: Bruce Golden

2:45 **A Novel Statistical Algorithm for Very Large-scale Vehicle Routing Problems with Time Windows**
Mayank Baranwal, Lavanya Marla, Srinivasa Salapaka, Carolyn Beck*
University of Illinois at Urbana-Champaign

3:15 **Facility Location and Design Decisions from Public Data**
*[1]Kalyan Talluri, [2]Muge Tekin**
[1]Imperial College Business School, [2]Universitat Pompeu Fabra

3:45 **Addressing Uncertainty in Meter Reading for Utility Companies Using RFID Technology**
[1]Debdatta Sinha Roy, [1]Bruce Golden, [2]Edward Wasil*
[1]Robert H. Smith School of Business-University of Maryland, [2]Kogod School of Business-American University

A Novel Statistical Algorithm for Very Large-scale Vehicle Routing Problems with Time Windows

Mayank Baranwal, Lavanya Marla, Carolyn Beck, Srinivasa Salapaka
University of Illinois at Urbana-Champaign
Urbana, IL, USA

The Vehicle Routing Problem with Time-Windows (VRPTW) is a core problem in the class of network-based resource allocation problems. These problems arise in city logistics, telecommunications, military applications, last mile delivery problems, liner shipping and inter-city logistics. Significant, and sometimes majority, of operating costs incurred by large carriers are transportation costs, and so, even small percentage improvements in cost are often huge gains in absolute costs. Improvements of the order to 5-20% [1] have been achieved using computerized models for transportation planning, which lead to significant gains for the industry as a whole.

While the VRPTW is a well-studied problem, the ever-advancing nature of the logistics industry, advancement of global positioning systems and radio frequency identification, and parallel computing has motivated further advances in both exact and heuristic methods. The continuing importance of the CVRP and VRPTW is highlighted by the fact that several variants of the VRP exist, with stochasticity, backhauls, with heterogeneous vehicle fleet, split deliveries, periodic routing and dynamic VRPs. More recent variations on the VRP include vehicle routing with electric vehicles and VRP with drones.

The purpose of this paper is to introduce our *novel modeling approach inspired from the field of information theory*, to the VRPTW, and to demonstrate its significant advantage in achieving *cost effective solutions* with *highly competitive solution times*. Our approach, termed *Deterministic Annealing* (not to be confused with the deterministic variant of simulated annealing) is based on a statistical technique used commonly in data compression and model aggregation. Our approach is a unified approach that is applicable to vehicle routing problems of varying complexity. In this paper, we address the basic VRP, the CVRP, and the VRPTW using our approach. In future work we will present variations of our method to more sophisticated variants of the VRP. We present computational results in which we show that our approach provides competitive results in high efficient solution times, rendering large-scale solutions of the VRPTW possible in a few minutes.

Our Approach

Deterministic Annealing (DA) is an approach based on clustering methods developed in information theory. DA is well-suited to combinatorial clustering and resource allocation problems that require obtaining an optimal partition of an underlying domain, and optimally assigning resources to each cell of the partition. DA-based methods have been reported in a number of applications such as minimum-distortion problems in data compression [2], model aggregation [3], routing in multi-agent networks [4], locational optimization problems [5], and coverage control[6].

At this juncture, it is important to distinguish our Deterministic Annealing (DA) approach from the deterministic variants of simulated annealing, which are essentially large-scale neighborhood

search-based approaches. As described in [7], the deterministic annealing approach used thus far in the literature is one in which the rule for accepting a new solution in the neighborhood during the annealing process is deterministic [8, 9, 10, 11].

In contrast to this, our modeling framework is not a large-scale neighborhood approach in the classical sense. DA is based upon formulating the system as that of maximizing the Shannon entropy (a global measure of the system), while minimizing the distortion (a local measure). This is achieved by minimizing the free energy, which is written as a Lagrangean function. In the DA algorithm, the free energy is deterministically minimized at successively reduced temperatures over repeated iterations. To ensure clarity, we will refer to this approach as the *DA* approach while we will refer to the deterministic variant of simulated annealing as the deterministic annealing approach.

Most of the problems that DA has been applied to are highly non-convex, computationally complex and contain several poor local optima that riddle the objective function surface. Similar characteristics exist for the VRP and its variants, including the phenomenon of clustering that is a natural aspect of grouping customer points. In this paper, we present an enhanced version of DA, which we call the *Routing-enhanced DA*, which simultaneously incorporates routing, scheduling and capacity considerations, which have not been considered in earlier DA literature.

In our Routing-enhanced DA approach, scheduling and routing are not viewed as different problems, but are viewed as forms of clustering problems in a larger dimensional space. The free energy function is written as a Lagrangean, therefore constraints in the problem are not treated as `hard' constraints but violations are penalized. The approach is modeled as a continuous problem in terms of the free energy function, which is minimized using an annealing approach at successively reduced temperatures. During the annealing solution process, our solution methodology follows a cluster-first-route-second approach, where we find the set of customers to be served in the same cluster and route following that. While the variables in the formulation are integers, these constraints are relaxed to begin with. The annealing approach decreases the temperature with multiple iterations at each temperature value. However, once the annealing temperature reaches zero, we get a solution that solves the unrelaxed problem, that is, a solution that satisfies the integrality constraints.

This property of working with the relaxed continuous problem before reaching the solution provides a significant benefit in terms of searching for global optima while reducing computational time. Our approach also incorporates a tunable set of `velocity' parameters that controls the relative importance of scheduling and routing; allowing for small violation in time-windows to reduce total travel time or decreasing scheduling violation to increase travel distance.

Contributions

We present a flexible and generalizable heuristic modeling approach for the VRPTW that can solve large-scale instances significantly more effectively than models currently existing in the literature.

(1) Our first contribution is to enhance the information-theoretic DA method from its basic clustering framework to a general framework in which capacity, routing and scheduling constraints are incorporated simultaneously. We formulate our DA-based VRP approach

to the basic VRP, the CVRP, and the VRPTW. Our Routing-enhanced DA formulation flexibly incorporates constraints that enforce specific customers to be served by the same vehicle, fuel burn constraints, constraints on package volumes or weights and the number of shipments in each vehicle.

(2) Analytically, we demonstrate that the CVRP and the VRPTW formulations form a second-order non-linear system, and show that the algorithm convergences, and is equivalent to a gradient descent. We also show that the fast rate of convergence of this approach is due to the relative insensitivity of the objective function to changes at temperatures other than `phase transition' temperatures.

(3) Quantitatively, we show how the *velocity parameters* of this approach can be used to trade off between the routing and scheduling constraints. We discuss the changes in the route design with varying values of these parameters, and how this provides flexibility to operators in designing such solutions.

(4) We show that our Routing-enhanced DA approach is practical and particularly effective for very large-scale instances. We demonstrate quick and good-quality results on proprietary instances drawn from the real-world operations of a last-mile delivery carrier. In addition, we test our approach for the Gehring and Homberger benchmark instances [12], which have 200, 400, 600, 800 or 1000 customers. On these instances, we show that our approach can result in solutions with tour lengths within 5% of the best-known solutions, with small time-window infeasibilities.

The Basic DA algorithm

The basic DA algorithm addresses the question of clustering or facility location – given N number of demand points i=1,2,…N with priorities $w(x_i)$, find the locations y_j of j=1,2,,..,K facilities that minimize the distance traveled by the customers to the facilities. The objective therefore is:

$$\min_{\substack{\{v_{ij}\},\, v_{ij}\in\{0,1\} \\ \sum_{j=1}^{K} v_{ij}=1}} \underbrace{\sum_{i=1}^{N} w(x_i) \sum_{j=1}^{K} v_{ij}\, d(x_i, y_j)}_{D}$$

Previously existing heuristics such as Lloyd's algorithm are sensitive to initialization and converge to local minima. Deterministic Annealing (DA) addresses the facility location problem by smoothing the objective function and tracking the minimum as smoothing vanishes.

DA addresses smoothing of the objective function by ascribing *partial* associations of each x_i to every facility y_j through association probabilities $p(y_j|x_i)$.

Let $\overline{D}$ be the modified **distortion term** given by $\overline{D} = \sum_{i=1}^{N} p(x_i) \sum_{j=1}^{K} p(y_j|x_i) d(x_i, y_j)$

DA chooses a probability distribution $p(y_j|x_i)$ that maximizes **Shannon's entropy**

$$H = -\sum_{i=1}^{N} p(x_i) \sum_{j=1}^{K} p(y_j|x_i) \log p(y_j|x_i)$$

under the constraint of minimizing distortion. Thus we consider *minimization* of the following Lagrangian (Free Energy) function: $\overline{D} - \frac{1}{\beta} H := F$

F is shown to be convex at low β, and is non-convex as β increases. β is also defined as the inverse of the annealing temperature T, that is decreased in successive iterations.

DA algorithm:

Step 1: $\forall \beta > 0$, iterate between the following two steps until convergence:

(a) Compute the Gibbs distribution that minimizes free-energy F, i.e.

$$p(y_j|x_i) = \frac{e^{-\beta d(x_i,y_j)}}{\sum_{j=1}^{K} e^{-\beta d(x_i,y_j)}}$$

The Gibbs distribution captures the association of vehicle j to demand point i.

(b) Update the resource locations $\{y_j\}$ as

$$\sum_{i=1}^{N} p(x_i)p(y_j|x_i)\frac{\partial d(x_i,y_j)}{\partial y_j} = 0$$

Note: For $d(x_i, y_j) = ||x_i - y_j||_2^{\ 2}$, the update equations simplify to

$$y_j = \frac{\sum_{i=1}^{N} p(x_i)p(y_j|x_i)x_i}{\sum_{i=1}^{N} p(x_i)p(y_j|x_i)}$$

Step 2: Increment β till β_{max} and repeat Step 1.

Our approach: Routing-enhanced DA enhanced for the VRPTW

The Routing-enhanced DA algorithm enhances the basic DA from a clustering formulation to include scheduling, routing and capacity constraints. We allocate a **vehicle** j to a **demand point** i in a pre-ascribed **time window** $[t_{i,start}, t_{i,end}]$ such that the capacity constraints of each vehicle are not violated and a feasible tour for each vehicle across the customers it serves is found.

Modeling scheduling constraints: For simplicity, we choose property x_i of demand point i to be the **mid-point** of the associated time-window, i.e., $x_i = \frac{t_{i,start}+t_{i,end}}{2}$ The scheduling component of the distortion function $d(x_i, y_j)$, captures the penalty of not servicing shipment, i, by vehicle, j in the middle of the time window. Expected distortion, $\overline{D} = \sum_{i,j} p(x_i)p(y_j|x_i)d(x_i, y_j)$, where, $d(x_i, y_j) = ||x_i - y_j||_2^2$

Modeling Routing and tour-length constraints: We let the property $x_i \in \mathbb{R}^2$ of demand point i represents its locational coordinates, while adding the minimum tour-length constraint into the free-energy term as:

$$F' = F + \eta\left(\sum_{j=1}^{K} d(y_j, y_{j+1}) - L\right)$$

We adopt an *Elastic Net* approach to find optimal tour-length, L. The tour-length is controlled by appropriately varying η and β.

Modeling capacity constraints: Let the total allocated capacity of **vehicle** (facility) j be λ_j, we address capacity constraints through a modified Gibbs distribution

$$p(y_j|x_i) = \frac{\lambda_j e^{-\beta d(x_i,y_j)}}{\sum_{k=1}^{K} \lambda_k e^{-\beta d(x_i,y_k)}}$$

Because the marginal distribution, $p(y_j) = \sum_i p(x_i)p(y_j|x_i)$, therefore, we have $p(y_1):p(y_2):\cdots:p(y_K) \approx \lambda_1:\lambda_2:\cdots:\lambda_K$, and capacity is allocated in a proportional fashion.

Modeling routing and scheduling constraints: For each customer location i, we define property

$x_i = \begin{bmatrix} x_i^{(1)} \\ x_i^{(2)} \\ \frac{v}{2}(t_{i,start} + t_{i,end}) \end{bmatrix}$, where $\left(x_i^{(1)}, x_i^{(2)}\right)$ are the locational coordinates of the demand point i, and v is a **velocity parameter** that allows the tradeoff of routing and scheduling constraints. We provide details in our paper on the approach to solve the modified free energy function and guidelines on how the velocity parameters can be tuned during the algorithm.

We also provide results on convergence properties of the algorithm and other theoretical properties that demonstrate the efficiency of our approach.

Preliminary Computational Results

Our preliminary computational results on the Gehring-Homberger instances [12] of size 1000 customers (consisting of clustered customers C2, random customer locations R2 and mixed locations RC2) demonstrate that improved tour distances and highly improved computational times can be achieved through our approach. These improved travel distances are a the cost of violation of about 5% of time windows to small extents, which can be further traded off by varying the velocity parameter v, possibly resulting in increased travel distances. However, our approach also provides a practical and controllable way for the system operator for trading off distances and times, and computing the value of on-time service.

	Existing best (distance, computation time)	Our DA approach (distance, computational time)
C2	16680.16 (≈120 min)	15415.31 (18 min)
R2	28874.02 (≈80 min)	18488.60 (12 min)
RC2	24005.78 (≈100 min)	17302.47 (12 min)

References

[1] P. Toth and D. Vigo, Eds., The Vehicle Routing Problem, SIAM, Philadelphia, 2002.
[2] K. Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," Proceedings of the IEEE, vol. 86, no. 11, pp. 2210–2239, 1998.

[3] Y. Xu, S. M. Salapaka, and C. L. Beck, “Aggregation of graph models and markov chains by deterministic annealing,” Automatic Control, IEEE Transactions on, vol. 59, no. 10, pp. 2807–2812, 2014.
[4] N. V. Kale and S. M. Salapaka, “Maximum entropy principle-based algorithm for simultaneous resource location and multihop routing in multiagent networks,” Mobile Computing, IEEE Transactions on, vol. 11, no. 4, pp. 591–602, 2012.
[5] S. Salapaka, A. Khalak, and M. Dahleh, “Constraints on locational optimization problems,” in Decision and Control, 2003. Proceedings of the 42nd IEEE Conference on, vol. 2. IEEE, 2003, pp. 1741–1746.
[6] Y. Xu, S. M. Salapaka, and C. L. Beck, “Clustering and coverage control for systems with acceleration-driven dynamics,” Automatic Control, IEEE Transactions on, vol. 59, no. 5, pp. 1342–1347, 2014.
[7] "G. Laporte and S. Ropke and T. Vidal, “Heuristics for the Vehicle Routing Problem”, in “Vehicle Routing - Problems, Methods and Applications, SIAM Monographs on Discrete Mathematics and Applications", P. Toth and D. Vigo, eds., SIAM and the Mathematical Optimization Society, Amsterdam, 2014, pp. 87–116.
[8] O. Braysy, W. Dullaert, G. Hasle, D. Mester and M. Gendreau, “An effective multi-restart deterministic annealing metaheuristic for the fleet size and mix vehicle routing problem with time windows”, Transportation Science, Vol 42, no. 3, 2008, pp 371-386.
[9] B.L. Golden, E. A. Wasil, J. P. Kelly and I-M. Chao, "Metaheuristics in vehicle routing", in "Fleet Management and Logistics", T. G. Crainic and G. Laporte, eds., Kluwer, Boston, 1998, pp 33-56.
[10] F. Li, B. Golden and E. Wasil, “Very Large-scale Vehicle Routing: New Test Problems, Algorithms, and Results”, in Computers and Operations Research, Vol 32 (5) 2005, pp 1165—1179.
[11] C. Groer, B.L. Golden and E. A. Wasil, “A library of local search heuristics for the vehicle routing problem”, in Mathematical Programming Computation, Vol 2, 2010, pp 79-101.
[12] H. Gehring and J. Homberger, “A parallel two-phase metaheuristic for routing problems with time windows”, Asia-Pacific Journal of Operational Research, Vol 18, 2001, pp 35—47.

Facility Location and Design Decisions From Public Data

Kalyan Talluri*and Müge Tekin†

December 9, 2016

1 Abstract

There has been a tremendous increase in the size, scope and availability of public data, yet it is not clear how a firm might effectively use it. Analytical studies rarely seem to go beyond summary statistics and attractive visualizations. In this paper, we present an application based only on publicly available data in which a restaurant chain makes location and design decisions (e.g. cuisine type, price point, size, capability to serve groups) for a new restaurant so as to maximize its profit. We assume customers patronize restaurants based on review ratings and proximity to the restaurant. We combine Yelp review data sets with demographic and geographic data to build a model of demand and use it to formulate an optimization problem that recommends the top k alternatives.
Keywords: facility location, multinomial logit, mixed integer programming, online customer reviews.

2 Introduction

Traditional location theory used to assume that customers patronize the closest facility. The seminal work in this line is by [1] that studies the conditions of equilibrium in case of two facilities on a line. Later, [3] generalizes it to a network. However, customers are affected by other factors, too. For instance, in the case of choosing a restaurant they may consider price, service quality, parking option etc.. Thus, facility attractiveness levels need to be incorporated into the model.

Customer choice models generally assume that customers are utility maximizers and multinomial logit (MNL) framework is commonly used as a utility model. MNL can also be used in facility location models and alternative solution methodologies are proposed to optimally locate k facilities. For example, [4], [5] and [8] study non-linear model formulations. [9] present a comparison of linear reformulations. Considering only locational decisions is a limitation since it is necessary to simultaneously determine the facility's features. Later on, [6] and [7] extend it including both location and design as decision variables. Yet, they assume customer choice parameter estimates are known a priori. Conversely, [2] study to estimate the demand at a single new facility but there is no optimization aspect.

In this paper, we assume customers patronize restaurants based on review ratings and proximity to the restaurant. We essentially build up a facility location model and validate it based on how

*Imperial College Business School, South Kensington, London SW7 2AZ, e-mail: kalyan.talluri@imperial.ac.uk
†Universitat Pompeu Fabra, Ramon Trias Fargas 25-27, 08005 Barcelona, Spain

customer reviews affect this decision. Our study fills an important gap by combining estimation of customer choice model and optimization of facility location and design in a competitive environment based on customer tastes and demographics.

2.1 Data and Model

This study consists of two steps. The first step is about demand modeling and the second one is about location and product design decision once demand information is gathered. Next, we give a brief description on the available data.

We collect restaurant data from the Yelp Dataset Challenge [13] that contains business (e.g. cuisine type, price range−number of dollars, 1$ to 4$, latitude/longitude coordinates, zip code) and review (e.g. number of reviews, average review rating, distribution of ratings) data. It also provides true/false information on restaurant attributes such as delivery, credit card option, parking service.

We limit our study to restaurants from Las Vegas, NV, which includes 4428 restaurants and 498923 customer reviews. We gather spatial consumer distribution from [12] at grid level, that is given approximately at 1 km^2 distance. Moreover, we obtain transient tourist population via hotels. Local population of Las Vegas is 2027828, after accounting for transient population, it increases to 2131185. We assume demand is aggregated at these grids and in total 1810 grids exist. Next, we study customer choice model using MNL to obtain estimates of customer choice behavior parameters.

2.1.1 Customer Choice Model

We gather geographic data at grid level to account for variation of population size. Grids represent the neighborhoods of population zones, denoted by G and restaurants are located in nodes I. Let n_i denote the number of reviews for a specific restaurant. Then, the total number of reviews can be obtained as: $n = \sum_i n_i$. In our model, we assume customers choose facilities with certain probabilities and the customer choice is influenced by MNL function of their utility. We consider utility of a restaurant as a linear combination of review rating r_i and proximity to the restaurant d_{ig}. The utility of a restaurant i from grid point g is

$$U_{ig} = V_{ig} + \epsilon_{ig}$$

where

$$V_{ig} = \alpha r_i + \beta d_{ig}$$

and the probability of restaurant i to be chosen by a population located at grid g is defined by the expression:

$$Pr_{ig} = \frac{e^{V_{ig}}}{\sum_i e^{V_{ig}}}$$

Total number of customers residing at grid g is denoted by $M_g,\ g \in G$. We take number of reviews as a proxy on demand at each restaurant. As a supporting evidence, we collect a sample of booking data from [10] and verify that the correlation between number of reviews and demand is quite high, for this sample it is 80.16%. Therefore, based on the assumption that a fixed fraction f of the visitors leave reviews, demand at any restaurant is given by

$$N_i = f(\sum_g M_g Pr_{ig}) \tag{1}$$

Summing this overall restaurants in the city, we obtain total number of reviews

$$N = \sum_i N_i = f(\sum_g M_g \sum_i Pr_{ig}) \tag{2}$$

Dividing (1) to (2), we obtain

$$\frac{N_i}{N} = \frac{\sum_g M_g Pr_{ig}}{\sum_g M_g \sum_i Pr_{ig}} = \frac{\sum_g M_g Pr_{ig}}{\sum_g M_g}$$

where

$$\sum_i Pr_{ig} = 1$$

	Estimate	Std. Error	t value	Pr(>|t|)
α	0.46	0.05	9.43	0.00
β	-0.07	0.04	-2.02	0.04

Table 1: Demand as a function of ratings and distance

We assume demand D for a given review rating r is a function of grid g, cuisine c and features vector $\boldsymbol{a}$; $D(g, c, \boldsymbol{a}|r)$. We consider the probability of getting a review rating r depending on location, cuisine and features of a restaurant; $Pr(r|g, c, \boldsymbol{a})$. Estimation of this probability creates a challenge since there are many grids without a certain cuisine-features combination of a restaurant. Therefore, review rating distribution is unknown for them. We apply non-negative matrix factorization on Yelp reviews data to develop a spatial notion of attractiveness of a location.

2.1.2 Non-negative Matrix Factorization with Side Information

Each restaurant i of cuisine c has some restaurant-specific information (e.g: price-point, take-out option etc.) that are captured in $\boldsymbol{a}_i$. We would like an estimation model that given some planned restaurant features will make a prediction of its rating distribution. Our model is that rating distribution of restaurant i is given by

$$r_i \sim Poisson(\gamma \boldsymbol{a}_i + \boldsymbol{p}_g^T \boldsymbol{q}_c)$$

The parameters are γ's and the latent factors are $\boldsymbol{p}$ and $\boldsymbol{q}$'s. Note that γ is common to all restaurant types. $\boldsymbol{p}$ is grid-specific and $\boldsymbol{q}$ is cuisine-specific. The aim is to find the optimal parameters to minimize the loss function:

$$\min_{\boldsymbol{p}*, \boldsymbol{q}*, \gamma*} \sum_i (r_i - \gamma \boldsymbol{a}_i - \boldsymbol{p}_g^T \boldsymbol{q}_c)^2 + \lambda(||\boldsymbol{p}_g^T||^2 + ||\boldsymbol{q}_c||^2 + ||\gamma||^2)$$

where λ is the regularization term to avoid overfitting.

We apply stochastic gradient descent method. We take regularization weight, $\lambda = 0.1$, dimensionality of latent feature space, $k = 3$, number of iterations 30 and learning rate $\gamma = 0.01$. We use Root Mean Squared Error (RMSE) to evaluate accuracy of predicted ratings:

$$\text{RMSE} = \sqrt{\frac{\sum (r_i - \hat{r}_i)^2}{T}}$$

where T is the number of observation, r_i is the actual rating for restaurant i and $\hat{r}_i$ is the predicted rating. We find that addition of price range, p, good for groups attribute, g, take-out option, t, credit-card option, cc, as side information improves test data set errors by around 13% (Table 2). We also notice that training data set errors without side information is lower, this can be due to the restricted model structure with features and not allowing as free training as no features case. Overall, the model performs quite well, we observe low RMSE. Also, we compare actual and mean of predicted ratings given by the rate of Poisson distribution on a sample of cuisines (Table 3).

	No features	**Good for Groups**	**Take-out Possibility**	**Accepts Credit Cards**	**Price Range $1,$2,$3,$4**
Estimation	$\gamma = 0$	$\gamma_g = 0.641$	$\gamma_t = 0.506$	$\gamma_c = 0.742$	$\gamma_p = 0.575$
Test data	0.855			0.742	
Train data	0.591			0.656	

Table 2: Multiple Features k=3, 30 iterations: RMSE, covariate weight on rating, $\boldsymbol{\gamma}$

Cuisine	**Actual Rating**	**Predicted Rating**
Barbeque	3	2.800
Burger	2.5	2.602
Chicken	3	2.636
Fast Food	2.75	2.724
Indian	3	3.208
Peruvian	2.75	2.831
Sea Food	3.75	3.188

Table 3: Cuisine rating predictions at grid centered at (-115.3792,36.4958). Total number of grids are 639, cuisines are 61. The sparsity level is 92.2%

2.1.3 Optimization

Once we obtain customer choice behavior parameters, we can use them as an input for the optimization problem. We assume that all the grids in the network are candidates to location of the facility. Let us denote the locational decision variable x_g, taking value 1 if a new facility is located at grid g and 0 otherwise. We obtain commercial rent prices at zip code level from [11] and we use ($/square feet) as a fixed cost, C, of opening a facility. Our aim is to recommend the top k alternatives of location-design combination. The problem can be modeled as a mixed integer non-linear program:

$$
\begin{aligned}
\max_{g,c,\boldsymbol{a}} \quad & p_i E[D_i(g,c,\boldsymbol{a})] - C_{ig}x_g \\
\text{s.t.} \quad & D_i(g,c,\boldsymbol{a}) = \sum_g M_g Pr_{ig} \quad \forall i \in G, \\
& E[D_i(g,c,\boldsymbol{a})] = \sum_r E[D_i(g,c,\boldsymbol{a})|r] Pr_i(r|g,c,\boldsymbol{a}) \quad \forall i \in G, \\
& Pr_{ig} = \frac{x_g e^{V_{ig}}}{\sum_{j\in G} x_j e^{V_{jg}} + \sum_{j\in I} e^{V_{jg}}} \quad \forall i,g \in G, \\
& V_{ig} = \alpha r_i + \beta d_{ig} \quad \forall i,g \in G, \\
& \sum_{g\in G} x_g = 1,
\end{aligned}
\tag{3}
$$

$$
\begin{aligned}
&Pr_{ig} \in [0,1] \quad \forall i, g \in G, \\
&D_i(g, c, \boldsymbol{a}) \geq 0 \quad \forall i \in G, \\
&x_g \in \{0,1\} \quad \forall g \in G.
\end{aligned} \tag{4}
$$

3 Contribution and Future Extensions

We present an application based only on publicly available and free data. Specifically, we focus on restaurants but our idea can be beneficial to a lot of businesses such as hotels, retail stores for which customer reviews play an important role in decision making. Our study offers a twist to facility location problems by estimating customer choice model parameters and optimizing facility location and product design. We aim to extend it by integrating demographics information and customer choice factors such as variety-seeking, saturation effects.

References

[1] Hotelling H. 1929. Stability in competition. *Economic Journal* **39** 41–57.

[2] Nakanishi M. and Cooper L. G. 1974. Parameter estimates for multiplicative competitive interaction models-least square approach. *Journal of Marketing Research* **11** 303–311.

[3] Hakimi S.L. 1983. On locating new facilities in a competitive environment. *European Journal of Operations Research* **12** 29–35.

[4] de Palma, A. and Ginsburgh V. and Labbe H. M., and Thisse J.F. 1989. Competitive location with random utilities. *Transportation Science* **23** 244–252.

[5] Benati S. 1999. The maximum capture problem with heterogeneous customers. *Computers and Operations Research* **26** 1351–1367.

[6] Aboolian R. and Berman O. and Krass D. 2007. Competitive facility location and design problem. *European Journal of Operations Research* **182** 40–62.

[7] Fernandez J. and Pelegrin B. and Plastria F. and Toth B. 2007. Solving a huff-like competitive location and design model for profit maximization in the plane. *European Journal of Operations Research* **179** 1274–1287.

[8] Marianov V. and Rios M. and Icaza M.J. 2008. Facility location for market capture when users rank facilities by shorter travel and waiting times. *European Journal of Operations Research* **191** 32–44.

[9] Haase K. and Müller S. 2014. A comparison of linear reformulations for multinomial logit choice probabilities in facility location models. *European Journal of Operations Research* **232** 689–691.

[10] 2016. Booking information. `http://www.opentable.com`. Accessed on 2016-11-30.

[11] 2016. Craigslist commercial rent prices. `https://craigslist.org/`. Accessed on 2016-11-30.

[12] 2016. NASA Socioeconomic data and applications center (Sedac). `http://sedac.ciesin.columbia.edu/data/collection/gpw-v4`. Accessed on 2016-11-30.

[13] 2016. Yelp dataset challenge. `https://www.yelp.com/dataset_challenge`. Accessed on 2016-11-30.

Extended Abstract for TSL Conference 2017

Title: Addressing uncertainty in meter reading for utility companies using RFID technology

Co-authors: Debdatta Sinha Roy [a], Bruce Golden [a], Edward Wasil [b]

[a] Robert H. Smith School of Business, University of Maryland, College Park, MD

[b] Kogod School of Business, American University, Washington, DC

Introduction

Utility companies read the electric, gas, and water meters of their residential and commercial customers on a regular basis. Typically, for a residential customer, a meter reader visits and manually reads a meter at the site. Utility companies are interested in generating short and balanced routes where all streets with meters that have to be read are traversed.

In the late 2000s, the use of radio-frequency identification (RFID) technology increased and automatic meter reading (AMR) using RFID technology was adopted for commercial use by utility companies. An AMR system has two parts: RFID tags and a truck-mounted reading device. An RFID tag is connected to a physical meter. The tag encodes the identification number of the meter and its current reading into a digital signal. The truck-mounted reading device collects the data automatically when it approaches an RFID tag within a certain distance. Utility companies would like to design the routes of the trucks (vehicles) to cover all customers (meters) in the service area and minimize the total length of the routes or the total cost of the routes. The use of RFID technology in meter reading changes the routing problem from a standard vehicle routing problem (VRP) over a street network to a close enough VRP (CEVRP) over a street network. Substantial savings over traditional solutions are possible by developing routes that exploit this close enough feature, i.e., the meter readers have to be within a certain distance from the meters to read them and not manually visit each one.

There are issues with RFID technology that gives the meter reading problem a great amount of inherent uncertainty. The signal transmitted by an RFID tag occurs at regular time intervals, and is not continuous for energy conservation purposes. This leads to the possibility of a missed capture of a signal if the truck with the receiver is within the range of the meter only for a short time. Also, the signal range of a meter can vary from the range specified by the manufacturer. The signal range can be affected by weather conditions and surrounding obstacles. These unplanned missed reads can lead to increased

costs for a utility company when a truck has to be sent at some later time to read the missed meters.

Research Goal

Published papers have not considered the issues with RFID technology and thereby have not taken into account the inherent uncertainty in the meter reading problem. The main contribution of our research is to address the issues of RFID technology by generating robust routes that minimize the number of missed reads. Our goal is to bring together data analytics and optimization techniques from vehicle routing to generate robust routes. We want to design routes that are both short in length and better at capturing the uncertain signals from meters.

Description of the Data Set

Data are provided by RouteSmart Technologies from ITRON (a technology and services company) which manufactures the RFID transmitters and receivers that are used by utility companies for meter reading. Data are in geographic information system (GIS) format and have three layers.

1. Street Level Data. Information about the shape, length, and type of street segments.
2. Service Location Data. Geographic locations of all meters that are to be read. Each meter is indexed by an unique account identifier (account id).
3. Reading Events Data. Records of all read events by the utility vehicle in the form of the time of the read (with a resolution of 1 second), the account id of the meter that is read, and the geographic location of the vehicle during the read event.

Integer Programming Formulation

The meter reading problem with RFID technology is formulated as a two-stage IP. The Stage 1 IP finds the street segments that are to be traversed and the number of times they are to be traversed for reading each meter with a pre-specified amount of likelihood from the full route. The Stage 2 IP solves a mixed rural postman problem (MRPP) that adds deadhead segments to the solution of the Stage 1 IP to obtain the full route. One

interesting thing to note here is that the deadhead segments added in the Stage 2 IP further increase the likelihood of reading the meters.

Let E be the set of the edges (undirected links) and A be the set of the arcs (directed links) in a street network, denoted by $G = (V, E \cup A)$, where V is the set of nodes. Let $c_j \geq 0$ be the cost (length) of street segment j. Let I be the set of the meters. Let p_{ij} be the probability that meter i is read at least once from street segment j. Let $L_i \in [0, 1]$ be the pre-specified amount of likelihood of reading meter i from the full route. Let x_j be the decision variable denoting the number of times street segment j should be traversed in the full route. The two-stage IP formulations generate Pareto optimal solutions of L_i and the length of the shortest feasible route, i.e., the length of the shortest feasible route will be greater for larger values of L_i fed to the Stage 1 IP. The Stage 1 IP formulation is given by

$$\min \quad \sum_{j \in E \cup A} c_j x_j \tag{1}$$

$$\text{s.t.} \quad \prod_{j \in E \cup A} (1 - p_{ij})^{x_j} \leq (1 - L_i) \quad \forall i \in I \tag{2}$$

$$x_j \geq 0 \text{ and integer} \quad \forall j \in E \cup A \tag{3}$$

In (2), the x_j's are selected accordingly so that the probability of reading meter i from the full route is at least L_i. In (3), only integer values of the x_j's are allowed. The objective function (1) minimizes the total length of the route. Note that (2) can be linearized in the decision variables $\sum_{j \in E \cup A} x_j \times \log(1 - p_{ij}) \leq \log(1 - L_i)$ producing a linear integer program. If we have more than one meter reading vehicle, then we can think of min-max objectives to balance out the routes for each vehicle.

Regression and Bayesian Updating

In order to solve the Stage 1 IP, we need to estimate the p_{ij}'s. We use a regression model. The data are in the form of 1 and 0, where 1 indicates that meter i is read from street segment j and 0 indicates that meter i is not read from street segment j. The predicted values of the dependent variable in the regression model have to be between 0 and 1, which will denote the probability p_{ij}. Based on the type of the data we have and our requirements on the dependent variable, logit and probit models serve our purpose. The independent variables in the models are: Shortest_Distance$_{ij}$ (shortest distance between meter i and street segment j), No_of_Pulses$_j$ (number of pulses the meter reading vehi-

cle can receive from meters while traveling on street segment j), and No_of_Customers$_i$ (number of meters within 500 feet surrounding meter i). Shortest_Distance$_{ij}$ should have a negative coefficient because the larger the shortest distance between meter i and street segment j is, the lower the p_{ij}. No_of_Pulses$_j$ is obtained from the amount of time the meter reading vehicle spent on street segment j divided by the time interval between the RFID signal transmissions. If the meter reading vehicle travels at a higher speed through street segment j, then the time spent by the vehicle on street segment j is smaller and, therefore, the No_of_Pulses$_j$ is lower. No_of_Pulses$_j$ should have a positive coefficient because the greater the number of pulses the meter reading vehicle can receive from meters while traveling on street segment j, the higher the p_{ij}. No_of_Customers$_i$ is a measure of density of meters in a region. It is important because with a large number of meters in a region, the interference of the RFID signals are greater, so the signals die out quickly. No_of_Customers$_i$ should have a negative coefficient because the greater the number of meters surrounding meter i, the lower the p_{ij}.

Every time the meter reading vehicle collects readings, it adds more data to the previous readings. The more data we have, the better will be the estimates of the p_{ij}'s and, therefore, the routes generated by the two-stage IP will be of higher quality, i.e., the routes will be even better at capturing the uncertain signals thereby further reducing the number of missed reads. There are some serious issues if we use regression to update the estimates of the p_{ij}'s at every stage with the new data. Suppose in time period 1 we have data y_1. We run the regression on y_1. In time period 2, there is new data y_2, so we run the regression on y_1 and y_2 together as a single data set, and so on. We are regressing on the older data sets repeatedly which makes this process of updating inefficient. Data sets from different time periods are given equal weights in the regression which should not be the case in practice. If we update the estimates of the p_{ij}'s at every stage when new data comes in using concepts from Bayesian statistics then we can avoid the two drawbacks faced while updating using regression. Bayesian updating works as follows

$$
\begin{aligned}
&P(\theta|y_1) \propto P(\theta)L(y_1|\theta) \\
&P(\theta|y_1, y_2) \propto P(\theta|y_1)L(y_2|\theta) \\
&\vdots \\
&P(\theta|y_1, \ldots, y_T) \propto P(\theta|y_1, \ldots, y_{T-1})L(y_T|\theta) = P(\theta)L(y_1, \ldots, y_T|\theta)
\end{aligned}
$$

θ is the vector of coefficients of the independent variables from the regression models. $P(\theta)$ is the prior probability distribution on θ. $L(y_1|\theta)$ is the likelihood function based

on data y_1 in time period 1. $P(\theta|y_1)$ is the posterior probability distribution on θ at the end of time period 1. All the information in y_1 is captured in $P(\theta|y_1)$. When new data y_2 comes in time period 2, then $P(\theta|y_1)$ becomes the prior, $L(y_2|\theta)$ is the likelihood function based on data y_2, and $P(\theta|y_1, y_2)$ is the posterior at the end of time period 2. All the information in y_1 and y_2 is captured in $P(\theta|y_1, y_2)$, which becomes the prior for the next time period. Given data $y_1, \ldots, y_T$, the posterior at the end of time period T will be $P(\theta|y_1, \ldots, y_T)$ and it does not depend on the sequence in which data arrive. Bayesian updating is much faster than regression since analysis is done only on the new incoming data at each stage. New data can be given more weight than old data in Bayesian updating.

Bayesian updating to estimate the p_{ij}'s can be done for both logit and probit models. Both models have their pros and cons. Error terms in logit models have a logistic distribution, whereas error terms in probit models have a normal distribution. The logistic distribution has fatter tails compared to the normal distribution, so logit models are more robust than probit models. Logit models have a better fit to data that are more spread out in the tails. However, Bayesian updating is quite easy in probit models as compared to logit models. An exponential family distribution is not the conjugate prior of the likelihood function in logit models, so the posterior distribution is difficult to calculate. In cases where both the logit and probit models fit the data equally well, it is more convenient to do the Bayesian updating for the probit models. A more complex method for estimating the p_{ij}'s performs Bayesian updating for hierarchical probit models. The estimates of the p_{ij}'s from hierarchical probit models are more accurate but difficult to implement as compared to the logit and probit models. Hierarchical probit models account for the uncertain behavior of each meter separately while also accounting for the similarity between meters.

Conclusion

Bayesian updating helps us to solve the two-stage IP at every time period when new data comes in, thereby potentially helping to produce more robust routes by updating the estimates of the p_{ij}'s. The main contribution of our research is combining vehicle routing with Bayesian statistics and data analytics to address the uncertainty in the meter reading problem. As mentioned, we will demonstrate these ideas using real-world data.

VEHICLE ROUTING MODELS & APPLICATIONS

FD3: ROUTING APPLICATIONS

Friday 4:30 – 6:00 PM

Session Chair: Burcu Keskin

4:30 **Heuristics and Lower Bounds for Robust Heterogeneous Vehicle Routing Problems Under Demand Uncertainty**
[1]Anirudh Subramanyam, [2]Panagiotis Repoussis, [1]Chrysanthos Gounaris*
[1]Carnegie Mellon University, [2]Stevens Institute of Technology

5:00 **Optimal Snow Plow Routing with Route Continuity Constraint**
Luning Zhang, Jing Dong*
Iowa State University

5:30 **Patrol Routing with Hybrid Electric Vehicles**
Mesut Yavuz, Burcu Keskin, Huseyin Ergin*
University of Alabama

Heuristics and Lower Bounds for Robust Heterogeneous Vehicle Routing Problems Under Demand Uncertainty

Anirudh Subramanyam[1], Panagiotis P. Repoussis[2], and Chrysanthos E. Gounaris*[1]

[1]Carnegie Mellon University, Pittsburgh, PA, USA

[2]Athens University of Economics and Business, Athens, Greece and Stevens Institute of Technology, Hoboken, NJ, USA

Motivation and Literature Review

The Capacitated Vehicle Routing Problem (CVRP) aims to determine a cost-optimal transportation plan for a fixed fleet of homogeneous capacitated vehicles so as to serve a set of customers with known demand. The key decisions are the assignment of customers to vehicles and the sequencing of customer locations that are visited by each vehicle. The objective is to minimize the total transportation cost, which is often proportional to the total distance traveled. The Heterogeneous Vehicle Routing Problem (HVRP) is a generalization of the CVRP wherein the fleet of vehicles is not homogeneous. In the HVRP, one must additionally decide the fleet composition from a number of available vehicle types, which may differ in capacity, maintenance costs, speed etc. Moreover, the transportation cost is often composed of a fixed/one-time component (for e.g., reflecting rental or capital amortization costs), in addition to a variable/recurring component proportional to the traveled distance that is modeled in the CVRP. We refer the reader to the recent survey by Koç et al. [2016] and the references therein for an up-to-date review of existing studies of the HVRP, including applications, models and solution methods.

In all (but one) of the existing studies on the HVRP, it is assumed that customer demands are deterministically known at the time of decision-making. However, in several real-world applications, this information is often not available, and the actual demand is observed only during the execution of the transportation plan. A popular approach is to solve the HVRP using some nominal values

*gounaris@cmu.edu

of customer demand and implement the resulting solution. The consequence of adopting this approach is that the resulting routing plan may become infeasible during actual execution, leading to potentially severe contractual, reputational and other kinds of penalties. These penalties are particularly significant in the case of HVRPs since they are also associated with long-term fleet dimensioning decisions. Therefore, anticipating and incorporating this uncertainty at the decision-making stage is crucial.

In this regard, the work by Teodorovic et al. [1995] considers a stochastic HVRP where customer demands are modeled as uniform random variables. The authors propose a continuous approximation heuristic to construct a number of potential routes for each vehicle type such that the probability of failure/infeasibility along the routes is small. In contrast, our approach is based on *robust optimization* and aims to determine a cost-optimal fleet composition and transportation plan that is *robust feasible*, that is, feasible for all anticipated demand realizations. We assume that only the support or *uncertainty set* of the uncertain customer demands is known, and that it has a known polyhedral description. In the context of the CVRP under demand uncertainty, such approaches based on robust optimization have been proposed by Sungur et al. [2008], Erera et al. [2010], Gounaris et al. [2013, 2016]. However, it is not straightforward how these approaches can be generalized to the HVRP, particularly in the context of efficient exact and lower bounding methods. This paper attempts to achieve this objective and extend it by generalizing known metaheuristic approaches for the deterministic HVRP to the robust setting.

Contribution

Our contribution is two-fold. First, we extend an Adaptive Memory Programming (AMP) metaheuristic algorithm developed for the deterministic HVRP [Repoussis and Tarantilis, 2010] to generate high quality solutions for the robust HVRP. Second, we develop a new integer programming (IP) formulation and a branch-and-cut solution framework that produces lower bounds on the optimal robust HVRP solution, thus allowing us to quantify the quality of the AMP heuristic solutions. A novel feature of this formulation is the generalization of the rounded capacity inequalities (RCI) from the CVRP to the HVRP setting, which is of interest in its own right because it results in a new formulation for the deterministic HVRP.

In both upper and lower bounding approaches, we use known results [Gounaris et al., 2013] from the robust CVRP to expedite the associated solution algorithms for two specially structured but broad classes of customer demand supports. In the metaheuristic approach, we exploit these results

to expedite the verification of robust feasibility of candidate solutions, while in the branch-and-cut approach, we use them to improve the efficiency of the separation routine for the generalized RCI inequalities.

Computational experiments conducted on a number of medium and large scale literature benchmark instances indicate that the AMP metaheuristic produces high quality solutions within short computational times. We remark that our techniques are readily generalizable to a number of routing models that are subsumed by the generic HVRP model, including the Fleet Size and Mix VRP, the Fixed Fleet HVRP (both with vehicle dependent routing costs), the Site Dependent VRP and the Multi-Depot CVRP. In the remainder of this abstract, we highlight the main features of the branch-and-cut and the AMP approaches.

Branch-and-Cut Approach

The HVRP is defined on a complete undirected graph $G = (V, E)$ with nodes $V = \{0, 1, \ldots, n\}$ and edges E. Node $0 \in V$ represents the unique depot, and each node $i \in V_C \coloneqq V \setminus \{0\}$ corresponds to a customer with demand $q_i \in \mathbb{R}_+$. We model the customer demands as random variables and assume only that their support is a known, non-empty polyhedron, $\mathcal{Q} \subseteq \mathbb{R}^n_+$. The depot is equipped with a set $K = \{1, \ldots, m\}$ of m different vehicle types and each vehicle of type $k \in K$ has a capacity equal to Q_k. We use the index set $K_i = \{k \in K : \max\{q_i : q \in \mathcal{Q}\} \leq Q_k\}$ to denote the set of vehicle types that can feasibly visit customer $i \in V_C$ under any demand realization.

Our IP formulation uses three-index integer variables x_{ijk} to count the number of times edge $(i, j) \in E$ is traversed by a vehicle of type $k \in K$. In addition, binary variables y_{ik} indicate if customer $i \in V_C$ is visited by a vehicle of type $k \in K$. In the interest of space, we present only the *robust generalized RCI* constraints from our formulation. In the following, $\delta(S)$ denotes the subset of edges in E with exactly one end-point in $S \subseteq V_C$.

$$\sum_{(i,j)\in\delta(S)} x_{ijk} + 2\sum_{i\in S}(1 - y_{ik}) \geq 2\left\lceil \frac{1}{Q_k} \max_{q\in\mathcal{Q}} \sum_{i\in S} q_i \right\rceil \quad \forall\, S \subseteq \{i \in V_C : K_i \ni k\},\ \forall\, k \in K \qquad (1)$$

These constraints enforce conditional lower bounds on the number of vehicles of type $k \in K$ that must serve a customer set $S \subseteq V_C$. For any support $\mathcal{Q}$, it can be shown that the above robust generalized RCI constraints are both necessary and sufficient to induce a set of robust feasible routes for the HVRP. Observe that since the maximization on the right-hand side does not depend on any decision variables, we can replace this quantity with the optimal objective value of the corresponding optimization problem.

Our IP formulation contains an exponential number of inequalities (1), and hence, we use a branch-and-cut solution algorithm. We remove the inequalities (1) and dynamically re-introduce them whenever the current node solution is found to violate them by solving the associated separation problem. We use a Tabu Search separation heuristic that iteratively constructs candidate subsets $S \subseteq V_C$ through a number of greedy perturbations. Since our heuristic typically constructs a large number of such subsets, it is crucial that the corresponding right-hand side of (1) can be evaluated efficiently. In the case of general polyhedral supports $\mathcal{Q}$, this would require the solution of a linear program. However, in the case of specially structured *disjoint budget* or *factor model* supports, this quantity can be computed analytically (see [Gounaris et al., 2013]), enabling fast separation of the robust generalized RCI constraints.

AMP Approach

The AMP framework is based on the idea that good-quality locally optimal solutions encountered during the search process are likely to share common features and characteristics (for e.g., sequences of customers visited by the same vehicle type); therefore, it utilizes a set of long-term memories for the iterative construction of so-called *provisional solutions*. These solutions are used as references for restarting and intensifying the search procedure, while adaptive learning mechanisms are used to update and modify the memory structures. In our approach, a reference set M maintains a pool of diversified "elite" HVRP solutions encountered during the search process. This central memory structure M is divided into subsets M_k, each corresponding to a particular vehicle type $k \in K$, which are used to maintain vehicle routes belonging to solutions in M.

The proposed AMP approach consists of an initialization and an exploitation phase. In the initialization phase, the initial reference set M is populated. For this purpose, a number of solutions are generated via a greedy randomized semi-parallel construction heuristic algorithm, and they are further improved via a Tabu Search algorithm. In the exploitation phase, a provisional solution is constructed at each iteration using promising vehicle routes extracted from the memory structures M_k. The score given to each stored route is based on the cost of the corresponding parent solution and the appearance frequency of edge $(i, j) \in E$ amongst the vehicle routes which are stored in M_k. Note that only those routes are considered which constitute a structurally feasible solution; that is, no two routes overlap and each customer is visited exactly once. Lastly, local search is applied to the provisional solution using a Tabu Search algorithm and if a set of admission criteria are satisfied, then the best encountered solution is used to update the reference set M.

Both the initialization and exploitation phases consist of a number of node- and edge-exchange local moves, and we only accept moves which guarantee that the resulting solution is robust feasible. That is, if $R_k = (0, R_{k,1}, \ldots, R_{k,n_k}, 0)$ denotes a candidate route traversed by a vehicle of type $k \in K$, we must verify that the total demand on the route is less than the vehicle capacity $\sum_{l=1}^{n_k} q_{R_{k,l}} \leq Q_k$ for all realizations $q \in \mathcal{Q}$. Observe that this is equivalent to checking if $\max_{q \in \mathcal{Q}} \sum_{l=1}^{n_k} q_{R_{k,l}} \leq Q_k$ and this latter quantity is exactly the same as the one which appears in the right-hand side of the robust generalized RCI constraint (1) associated with $S = R_k$. Thus, verification of robust feasibility in the AMP framework requires computation of the maximum of a linear function over a polytope. For general supports $\mathcal{Q}$, this entails the solution of a linear program. However, for disjoint budget or factor model supports, this quantity can be computed analytically, enabling fast verification of robust feasibility of candidate vehicle routes.

References

A. L. Erera, J. C. Morales, and M. Savelsbergh. The Vehicle Routing Problem with Stochastic Demand and Duration Constraints. *Transportation Science*, 44(4):474–492, 2010.

C. E. Gounaris, W. Wiesemann, and C. A. Floudas. The Robust Capacitated Vehicle Routing Problem Under Demand Uncertainty. *Operations Research*, 61(3):677–693, 2013.

C. E. Gounaris, P. P. Repoussis, C. D. Tarantilis, W. Wiesemann, and C. A. Floudas. An Adaptive Memory Programming Framework for the Robust Capacitated Vehicle Routing Problem. *Transportation Science*, 50(4):1239–1260, 2016.

Ç. Koç, T. Bekta, O. Jabali, and G. Laporte. Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research*, 249(1):1–21, 2016.

P. Repoussis and C. Tarantilis. Solving the Fleet Size and Mix Vehicle Routing Problem with Time Windows via Adaptive Memory Programming. *Transportation Research Part C: Emerging Technologies*, 18(5):695–712, 2010.

I. Sungur, F. Ordóñez, and M. Dessouky. A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. *IIE Transactions*, 40(5):509–523, 2008.

D. Teodorovic, E. Krcmar-Nozic, and G. Pavkovic. The mixed fleet stochastic vehicle routing problem. *Transportation Planning and Technology*, 19(1):31–43, 1995.

Optimal Snow Plow Routing with Route Continuity Constraint

Luning Zhang
Graduate Student
Iowa State University
Institute for Transportation, 2711 South Loop Drive, Suite 4700, Ames, Iowa, United State of America, 50010-8664
Tel: (+1) 515-708-0239
Fax: 515-294-0467
Email: lzhang@iastate.edu

Jing Dong (corresponding author)
Assistant Professor
Iowa State University
Institute for Transportation, 2711 South Loop Drive, Suite 4700, Ames, Iowa, United State of America, 50010-8664
Tel: (+1) 515-294-3957
Fax: 515-294-0467
Email: jingdong@iastate.edu

Winter road maintenance includes the removal of snow and ice on roadways and spreading materials for anti-icing, de-icing or increasing friction. Snow removal and material spreading are often accomplished by a fleet of snow plow trucks. Winter road maintenance is essential for providing safe and efficient service for road users (Haghani and Qiao 2001). It is also very expensive due to the high cost of equipment, crew, and materials. According to a recent survey by the American Association of State Highway and Transportation Officials (AASHTO), 23 reporting states spent in total approximately $1.1 billion from October 2014 to mid-April 2015 to pre-treat, plow, and spread chemicals and other materials on roadways. Optimizing winter road maintenance operations could result in significant cost savings, improved safety and mobility, and reduced environmental and social impacts (Haghani and Qiao 2001).

This paper formulates and solves an arc routing problem (ARP) to optimize snow plow routing considering route continuity constraint. The transportation network is described as a connected direct graph. Given a preset depot location and sector region, the ARP is to design a set of routes such that all the road segments of a transportation network are serviced by a fleet of trucks based at that depot, subject to a set of constraints with the objective of minimizing the operation cost. There are two problem included in the routing design: 1) how often a road segment should be serviced, and 2) how to minimize the operational cost.

How often a road segment needs to be plowed depends on the targeted Level of service (LOS). In the context of winter maintenance, LOS of a roadway is defined by the number of plows per day during a full-day storm. As shown in Table 1, a level A road requires 12 plows a day, a level B road requires 11 plows, and less busy roads need fewer services. Since service cycles are different for different roads, a service schedule table is generated to combine road segments of different service levels. Then, route optimization is applied to each service levels' networks.

One way to minimize operation cost is to minimize the total deadhead distance, that is, the distance traversed without servicing the road segment. In snow plow operation, there are three possible situations when a truck is traveling on a road segment—plowing the segment, traversing

the segment when it has already been plowed, and traversing the segment when it has not been plowed. The first case usually involves spreading materials as well, and the last two cases are considered as deadhead. In practice, the third case should be avoided because it increases deadhead and the traverse speed would be lower as the truck had to drive on the snow covered road. A good measure in this situation is the route connectivity. A connected route may consist of alternated service road segments and deadhead road segment, but it will not allow trucks traversing the segment when it has not been plowed.

Level of Service

LOS in the context of winter maintenance operation is a set of operational instruction and routing process that determines when, which and how often a road network should be serviced (Blackburn et al. 2004). Highway agencies of different states provide their own way to characterize LOS. (Haghani and Qiao 2001) and (Perrier et al. 2008) use priority treatment, road segment with high priority must be service before road segment with low priority. (Jang et al. 2010) use frequency treatment, road segments of different LOS have different cycle time, but their service route are created exclusively for each LOS class.

Iowa department of transportation (DOT) defines the quality of winter maintenance service by service frequency. Roadways with higher traffic demand require more frequent service. For example, roadways classified as service level "A" need to be serviced 12 or 13 times during a full-day storm. That is, such roadways must be serviced at least once every 2 hours during a continuous storm. As shown in Table 1, service frequency requirements are defined for roadways based on the number of vehicles per lane per day. The number of vehicles is the sum of number of passenger cars and 1.8 times number of trucks.

TABLE 1. Maintenance Service Levels

Service Level	Number of service per day	Vehicles per lane per day
A	12 or 13	> 8001
B	10 or 11	5001 - 8000
C	9	2501 - 5000
D	7	1501 - 2500
E	5	801 - 1500
F	3 or 4	0 - 800

A service network schedule that determines whether or not a road segment needs to be serviced in a timeslot is generated for the study network.

Formulation

A capacitated arc routing problem (CARP) is formulated to optimize route design. The objective is to minimize the deadhead distance to improve the operational efficiency and reduce cost. Let $G = (V, A)$ be a directed graph where $V = \{v_0, v_1, \dots, v_n\}$ is a set of nodes, and $A = \{(v_i, v_j): v_i, v_j \in V \; and \; i \neq j\}$ is a set of arcs. The depot is represented by node v_0. Define $R \subseteq$ A as the set arcs that requires service. Each arc $(v_i, v_j) \in R$ is associated with a demand q_{ij},

expressed as the total amount of material needed to service the arc, a distance d_{ij} corresponding to the length of the road segment, and a time t_{ij} corresponding to the servicing time. Every arc $(v_i, v_j) \in A$ is associated with a deadhead time t'_{ij}. Define n_{ij} as the number of times arc $(v_i, v_j) \in R$ should be serviced in a service cycle. Let K be the set of vehicles. For every arc $(v_i, v_j) \in A$ let x_{ijk} and y_{ijk} be binary variables, which equal to 1 if and only if arc (v_i, v_j) is serviced or traversed as deadhead from i to j in route k, respectively. Let T and W be the maximum route duration and the capacity of all vehicles. The CARP is formulated as follows.

Minimize $\sum_{k\in K}\sum_{(v_i,v_j)\in A} d_{ij}\, y_{ijk}$ (1)

Subject to

$\sum_{\{v_j:(v_j,v_i)\in A\}}(y_{jik} + x_{jik}) - \sum_{\{v_j:(v_j,v_i)\in A\}}(y_{ijk} + x_{ijk}) = 0 \quad (v_i \in V, k \in K)$ (2)

$\sum_{k\in K} x_{ijk} = n_{ij} \quad ((v_i, v_j) \in R)$ (3)

$\sum_{(v_i,v_j)\in R} q_{ij} x_{ijk} \le W \quad (k \in K)$ (4)

$\sum_{k\in K} y_{ijk} = 0 \quad ((v_i, v_j) \in R, \sum_{k\in K} x_{ijk} < n_{ij})$ (5)

$\sum_{\{v_i:(v_i,v_0)\in A\}}(x_{i0k} + y_{i0k}) = 1 \quad (k \in K)$ (6)

$\sum_{\{v_j:(v_0,v_j)\in A\}}(x_{0jk} + y_{0jk}) = 1 \quad (k \in K)$ (7)

$\sum_{(v_i,v_j)\in S}(x_{ijk} + y_{ijk}) \le |S| - 1 + |V|^2 u_k^S \quad (S \subseteq \mathrm{V}\backslash\{v_0\}, \mathrm{S} \neq \emptyset, k \in K)$ (8)

$\sum_{v_i\in S}\sum_{v_j\in S}(x_{ijk} + y_{ijk}) \ge 1 - w_k^S \quad (S \subseteq \mathrm{V}\backslash\{v_0\}, \mathrm{S} \neq \emptyset, k \in K)$ (9)

$u_k^S + w_k^S \le 1 \quad (S \subseteq \mathrm{V}\backslash\{v_0\}, \mathrm{S} \neq \emptyset, k \in K)$ (10)

$u_k^S, w_k^S \in \{0,1\} \quad (S \subseteq \mathrm{V}\backslash\{v_0\}, \mathrm{S} \neq \emptyset, k \in K)$ (11)

$x_{ijk}, y_{ijk} \in \{0,1\} \quad ((v_i, v_j) \in A, k \in K)$ (12)

The objective function (1) minimizes the total deadheading distance. Constraints (2) are flow conservation equations for each vehicle. Constraints (3) state that each arc is serviced as required number of times in that service cycle. Constraints (4) guarantee that the capacity of each vehicle is never exceeded. Constraints (5) state that trucks cannot traverse on an unplowed road segment as deadhead. Constraints (6) and (7) require all routes to start and end at the depot. Constraints (8) - (11) prohibit the formation of disconnected sub tours and were explained in detail by Golden and Wong (1981). Finally, constraint (12) restricts x_{ijk}, y_{ijk} to be binary.

Route Connectivity Constraints

Route continuity is a practical concern for operational convenience. Service route continuity constraint was first introduced by (Haghani and Qiao 2002). In their definition, service arcs in a route must be connected to each other, which is referred to as *strong continuity* in this paper. As shown in Figure 1, the top route satisfies strong continuity.

As mentioned earlier, unplowed segments should not be traversed as deadhead. However, truck can traverse on an already plowed arcs as deadhead in middle of a service route. This is referred

to as *loose continuity*. In Figure 1, both the top route and the middle route satisfy loose continuity, although the middle route has deadhead segment between service arcs.

Because loose connectivity allows routes including deadhead-connected service arcs, it tends to make the full usage of truck capacity and thus hele to reduce fleet size. In the situation where the number of lanes changes along a road, if trucks can service only one lane per run, it is highly probable that road segments with less lanes will be completed earlier than road segments that have more lanes. These serviced deadhead road segments will become a barrier to strong connected routes. Since the neighboring segments of these segments still need to be serviced. Loose connected routes are not affected by the heterogeneity in number of lanes along a road.

The third route in Figure 1 is allowed by neither strong nor loose continuity.

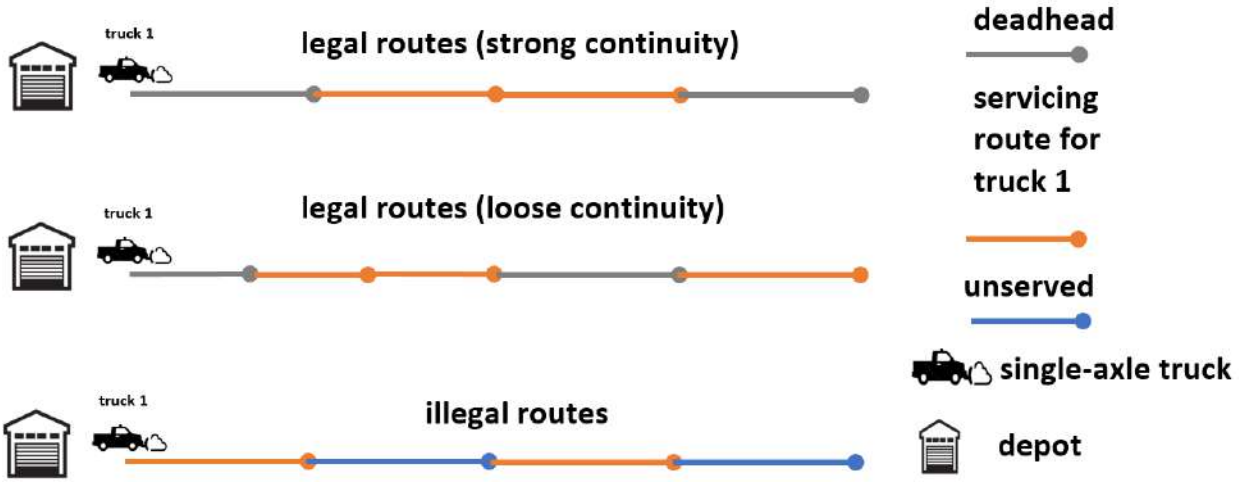


FIGURE 1 Service Continuity

Solution Algorithm

Since CARP is NP-hard (Golden and Wong 1981), exact method can only solve small size instances. This paper uses memetic algorithms (MA) (Lacomme et al. 2002) to solve the proposed model for snow plow routing. Single insertion, double insertion and swap (Tang et al. 2009) are used as local search move operators. To account for the route continuity constraint, after initialization and after each local search movement, the solution sequence must be adjusted to guarantee legal routes.

A case study is conducted for the state highway road network in Dubuque County, Iowa. The transportation network includes 18 nodes and 36 directed arcs. Solutions can be found by the MA. However, MA generally took a long time to reach its optimal (e.g. 2 hours to solve one instance on the case study network). This is because loose continuity has largely limited feasible solutions. Future research will focus on developing algorithms that can generate feasible solutions in a more reasonable computational time.

References

American Association of State Highway and Transportation Officials. Survey of 23 States Shows $1 billion Spent, 8 Million Work Hours Logged and Six Million Tons of Salt Used to Battle Winter Weather. Retrieved July 25, 2016, from http://www.aashtojournal.org/pages/NewsReleaseDetail.aspx?NewsReleaseID=1443.

Blackburn, Robert, KARIN M. BAUER, SR. DUANE E. AMSLER, S. EDWARD BOSELLY, and A. DEAN MCELROY. 2004. *Snow and Ice Control: Guidelines for Materials and Methods*. *Transportation Research*. Vol. 10.

Golden B.L., and Wong R. Capacitated arc routing problems. Networks, 1981, 11:305-315.

Haghani, Ali, and Haiying Qiao. 2001. "Decision Support System for Snow Emergency Vehicle Routing: Algorithms and Application." *Transportation Research Record* 1771 (1): 172–78. doi:10.3141/1771-22.

———. 2002. "Snow Emergency Vehicle Routing with Route Continuity Constraints." *Transportation Research Record* 1783 (1): 119–24. doi:10.3141/1783-15.

Jang, Wooseung, James S Noble, and Thomas Hutsel. 2010. "An Integrated Model to Solve the Winter Asset and Road Maintenance Problem." *IIE Transactions* 42 (9): 675. doi:10.1080/07408171003705375.

Lacomme, Philippe, CHRISTIAN PRINS, and WAHIBA RAMDANE-CHERIF. 2002. "Competitive Memetic Algorithms for Arc Routing Problems." *Annals of Operations Research*, 159–85.

Perrier, Nathalie, André Langevin, and Ciro-Alberto Amaya. 2008. "Vehicle Routing for Urban Snow Plowing Operations." *Transportation Science* 42 (1): 44–56. doi:10.1287/trsc.1070.0195.

Tang, Ke, Yi Mei, and Xin Yao. 2009. "Memetic Algorithm with Extended Neighborhood Search for Capacitated Arc Routing Problems." *IEEE Transactions on Evolutionary Computation* 13 (5): 1151–66. doi:10.1109/TEVC.2009.2023449.

Patrol Routing with Hybrid Electric Vehicles

Mesut Yavuz, Burcu B. Keskin
Information Systems, Statistics, and Management Science
University of Alabama
myavuz@cba.ua.edu, bkeskin@cba.ua.edu

Hüseyin Ergin
Computer Science
University of Alabama
hergin@crimson.ua.edu

January 5, 2017

Extended Abstract

Due to both economic and humanitarian importance, maintaining roadway travel safety has aroused widespread interest from all levels of the society, including citizens, government officials, industry practitioners, and academicians. In the United States, the National Highway Traffic Safety Administration estimates the cost of improving safety by various ways of law enforcement as high as $230.6 billion, a year-nearly 2.3 percent of the nation's gross domestic product (Blincoe et al. 2002). In addition, environmental sustainability concerns make reduction of oil consumption an important goal for the transportation sector, which is the biggest consumer of petroleum in the U.S. (69% in 2011) (see Yavuz and Capar Forthcoming, and the references therein). An important way of reducing transportation oil consumption is to improve fuel efficiency of vehicles in use nationwide. Toward this objective, all major automotive manufacturers today offer not only more fuel-efficient traditional (running solely on petroleum) vehicles, but also various alternative-fuel vehicles.

Electric vehicles are particularly interesting due to (i) the ubiquity of electricity, (ii) lower per-mile cost than petroleum with today's technology, and (iii) lower per-mile greenhouse gas emission than petroleum with today's electricity generation mix. Greenhouse gas emission can be further lowered with increased renewable electricity generation. However, (pure) electric vehicle adoption is limited primarily due to the so-called *range anxiety*.

In the field of law enforcement, many police departments have already adopted electric or other alternative-fuel vehicles. In this work, we focus on highway patrol operations performed by state troopers, which differs from police operations in regards to (i) geographical area coverage, (ii) number of miles traveled by a vehicle, and (iii) the higher probability of involvement in high-speed, long-distance chases. Therefore, pure electric vehicles do not fit the problem addressed in this work because of its finite driving range and limited recharging infrastructure. Hybrid electric vehicles (HEVs) that run on electricity and gasoline fit well and can lead to both cost and greenhouse gas emission savings by state troopers.

Scholarly works (see Steil and Parrish 2009, Keskin et al. 2012, Lou et al. 2011, Willemse and Joubert 2012, for example) are also rigorously pushing forward more effective law enforcement plans, including effective patrolling plans. One way of improving patrolling efficiency is focusing on patrolling critical locations with high crash frequencies.

Speeding, driving under the influence, and other aggressive driving behaviors are among the leading causes of highway crashes and fatalities. State officials continuously work to discourage such behaviors in several ways, including: (i) increased data-driven enforcement; (ii) technological advances, such as automated enforcement; and (iii) public information and education programs (GHSA 2013). Data-driven enforcement involves developing strategic countermeasures and operational plans using locally collected data and hot spot information.

Hot spots are defined as certain combinations of highway stretches and time of day with high frequencies of crashes. The visibility of law enforcement officers at hot spots is known to be one of the key deterrents to aggressive driving. Two streams of research have emerged in data-driven law enforcement. The former is related to defining hot spots via clustering analysis of historical crash and citation data (Anderson 2006, Chen and Quddus 2003, Cheng and Washington 2005, Gatrell et al. 1996, McCullagh 2006, Miranda-Moreno et al. 2007, Steil and Parrish 2009, Steil 2010). The latter, to which this paper contributes, is concerned with the use of predetermined hot spot

information in setting effective operational plans that enhance public safety (Keskin et al. 2012, Li and Keskin 2014, Çapar et al. 2015).

In this work, for a given mixed patrol fleet consisting of traditional (gasoline or diesel) vehicles and HEVs and, we investigate optimal patrol routes to visit time-critical hot spots. Our overall goal is to maximize the visibility of the state troopers while minimizing the costs associated with utilization of troopers. Therefore, we tackle a bi-criteria optimization problem.

Let the fleet consist of two types of vehicles, namely hybrid electric vehicles (HEVs), and traditional gasoline or diesel vehicles (GDVs). Sets V_1 and V_2 denote the sets of HEVs and GDVs, respectively, and set $V = V_1 \cup V_2$ is the set of all vehicles. Fleet composition is known, i.e., $|V_1|$ and $|V_2|$ are given. The HEVs and the GDVs are homogeneous within their respective groups.

The HEVs have two modes of operation, electric and gas, and they operate exclusively in the electric mode when the state-of-charge (SOC) is positive. When the battery is depleted (i.e., SOC is zero) they switch to operating in the gasoline mode and stay in that mode until SOC is increased through a recharge. The SOC is defined in the closed interval of $[0, 1]$ and changes linearly with time. That is, it increases by γ per minute of recharging and decreases by β per minute of traveling. We would like to note that γ depends on vehicle as well as charging station characteristics, in that it is the minimum of the vehicle's and charging station's limit charging rate per minute. In the problem studied here, homogeneous HEVs and homogeneous charging stations are assumed, thereby resulting in a constant γ. The HEV cost of traveling in the electric and gas modes are c^{he} and c^{hg}, respectively, and the GDV cost of traveling is c^g per minute of travel.

The problem is defined on a network $G = (N, A)$. $N = S' \cup H \cup D \cup R \cup S''$ consists of five disjoint sets of nodes. S' is the set containing only node 0, which is the source node where each vehicle starts the workday. H, D, and R represent the hot spots, administrative task stations and recharging stations, respectively. Finally, S'' is the set containing only node $|H| + |D| + |R| + 1$, which is the sink node where each vehicle ends the workday.

The sets H, D, and R are populated as follows. H is obtained by splitting an original set of hot spots into time sections described in Dewil et al. (2015). D is obtained by replicating each original administrative task station $|V|$ times. Similarly, R is obtained by replicating each original recharging station k times, where k is the maximum number of times a recharging station can be visited in a workday. If there is a recharging station at the depot, it is

included in the original set of recharging stations, thus R does not overlap with S' or S''. If a hot spot or an administrative task location coincides with a recharging station, it is reflected in the maximum allowed SOC gain at that node, thus R does not overlap with H or D. Essentially R represents dedicated recharging station visits, and the time spent on them cannot be used for covering a hot spot or performing administrative tasks. Finally, troopers are not allowed to perform administrative tasks while they cover hot spots, due to the nature of the two activities, both of which require a trooper's full attention.

In the remainder of this work, we (i) formulate the emerging bi-criteria optimization problem, (ii) investigate mathematical properties and special cases of the problem that lend themselves to quick analytic solutions or transformations to polynomially-solvable problems such as the minimum cost network flow problem, and (iii) present a computational study using appropriate algorithms on real-life data obtained from the Alabama Law Enforcement Agency.

References

Anderson, T. 2006. Comparison of spatial methods for measuring road accident "hotspots": a case study of London. *Journal of Maps* 55–63.

Blincoe, L., A. Seay, E. Zaloshnja, T. Miller, E. Romano, S. Luchter, R. Spicer. 2002. The economic impact of motor vehicle crashes, 2000. Crash Statistics DOT HS 809 446, NHTSA.

Çapar, İbrahim, Burcu B Keskin, Paul A Rubin. 2015. An improved formulation for the maximum coverage patrol routing problem. *Computers & Operations Research* **59** 1–10.

Chen, H. C., M. A. Quddus. 2003. Applying the random effect negative binomial model to examine traffic accident occurrence at signalized intersections. *Accident Analysis and Prevention* **35** 253–259.

Cheng, W., S. P. Washington. 2005. Experimental evaluation of hotspot identification methods. *Accident Analysis and Prevention* **37**(5) 870–881.

Dewil, Reginald, Pieter Vansteenwegen, Dirk Cattrysse, Dirk Van Oudheusden. 2015. A minimum cost network flow model for the maximum covering and patrol routing problem. *European Journal of Operational Research* **247**(1) 27–36.

Gatrell, A. C., T. C. Bailey, P. J. Diggle, B. S. Rowlingson. 1996. Spatial point pat-

tern analysis and its application in geographical epidemiology. *Transactions of the Institute of British Geographers* **21**(1) 256–274.

GHSA. 2013. http://www.ghsa.org/html/issues/speeding.html. Governer's Highway Safety Association.

Keskin, B. B., S. R. Li, D. Steil, S. Spiller. 2012. Analysis of an integrated maximum covering and patrol routing problem. *Transportation Research Part E: Logistics and Transportation Review* **48**(1) 215–232.

Li, S. R., B. B. Keskin. 2014. Bi-criteria dynamic location-routing problem for patrol coverage. *Journal of the Operational Research Society* **65** 1711–1725.

Lou, Y., Y. Yin, S. Lawphongpanich. 2011. Freeway service patrol deployment planning for incident management and congestion mitigation. *Transportation Research Part C: Emerging Technologies* **19**(2) 283–295.

McCullagh, M. J. 2006. Detecting hotspots in time and space. *Proceedings of International Symposium and Exhibition on Geoinformation*. 1–18.

Miranda-Moreno, L. F., A. Labbe, L. Fu. 2007. Bayesian multiple testing procedures for hotspot identification. *Accident Analysis and prevention* **39**(6) 1192–1201.

Steil, D., A. Parrish. 2009. Hit: A gis-based hotspot identification taxonomy. *International Journal of Computers and their Applications* **16**(2) 1–10.

Steil, D. A. 2010. Creation of crash-countermeasure police patrol routes targeting hotspot road segments. Ph.D. thesis, The University of Alabama.

Willemse, Elias J, Johan W Joubert. 2012. Applying min–max k postmen problems to the routing of security guards. *Journal of the Operational Research Society* **63**(2) 245–260.

Yavuz, M., I. Capar. Forthcoming. Alternative-fuel vehicle adoption in service fleets: Impact evaluation through optimization modeling. *Transportation Science* URL `http://dx.doi.org/10.1287/trsc.2016.0697`. Published online in Articles in Advance 26 Jul 2016.

VEHICLE ROUTING MODELS & APPLICATIONS

SA3: VRP IN SERVICES

Saturday 9:00 – 11:00 AM

Session Chair: Lei Zhao

9:00 **Self-Sustained Car-And-Ride Sharing Design and Optimization for Improving the Mobility of Underserved Communities**

[1]*Miao Yu**, [2]*Siqian Shen*
[1]*University of Michigan*, [2]*Industrial and Operations Engineering-University of Michigan*

9:30 **The Multi-Period Service Planning and Routing Problem**

*Albert H. Schrotenboer**, *Evrim Ursavas*, *Iris F.A. Vis*
University of Groningen

10:00 **Covering Tour Problem with an Application to School Bus Routing: Analysis of Single Vehicle Tours on a Grid**

*Liwei Zeng**, *Sunil Chopra*, *Karen Smilowitz*
Northwestern University

10:30 **Coordinated Delivery to Nanostores in Megacities**

[1]*Ruidian Song*, [1]*Lei Zhao**, [2]*Jan C. Fransoo*, [2]*Tom Van Woensel*
[1]*Tsinghua University*, [2]*Technische Universiteit Eindhoven*

Self-Sustained Car-and-Ride Sharing Design and Optimization for Improving the Mobility of Underserved Communities

Miao Yu, Siqian Shen
Emails: miaoyu, siqian@umich.edu
Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA

Abstract

We focus on the design and optimization of an innovative self-sustained car-and-ride sharing system. We study one of its special applications in improving the mobility of underserved communities. In this system, we assume two types of demands: Type 1 demand arises from customers who rent shared cars for private use and Type 2 demand arises from customers who cannot drive but require ridesharing services that could be fulfilled by Type 1 drivers. We propose a two-phase approach for optimizing resource pooling, supply-demand matching, and service scheduling, to maximize the fulfillment of car/ride-sharing demand while maintaining the cost self-sustainability of running the system. In the first phase, we allocate shared vehicle fleet and maximize the total car/ride sharing requests to satisfy, while ensuring that enough Type 1 drivers are accepted to serve Type 2 non-driver users. In the second phase, we design routing and scheduling for both types of users. We provide preliminary computational results of our two-phase approach by testing diverse instances.

1 Introduction

Sharing services are undergoing a fast rise in popularity and industrial growth in recent years. Car-sharing companies such as Zipcar, Car2Go, and Maven have shown their success with fast growing users. Via efficiently pooling idle resources, sharing services provide effective solutions to people's daily life needs. In this paper, we aim to design and optimize an innovative car-and-ride sharing (CRS) system in which both carsharing and ridesharing services are provided to heterogeneous users. In addition, the system is self-sustained such that customers with ridesharing demands are served by volunteer drivers with carsharing demands. We focus on its application in improving the mobility of underserved communities, where residents are experiencing transportation scarcity due to poverty or geographically dispersed from resources.

1.1 Problem Description and Solution Approaches

We consider a CRS system that provides both carsharing and ridesharing services in a service region. We target at underserved communities where transportation is a scarce resource and vast wealth gap across racial groups exists. We partition the service region into zones, and each zone represents a community. Shared vehicles are located *a priori* in some designated parking spaces in each zone. We classify two types of customers: Type 1 customers who want shared cars for private use and Type 2 customers who have ridesharing demand but cannot drive cars themselves. We aim to build a self-sustained CRS system that encourages Type 1 customers "serve" Type 2 customers at their capability to gain discount for renting cars. To implement the system, we develop an online reservation system that enables Type 1 customers to reserve cars by providing their car rental information along with available time windows for serving others, and Type 2 customers to

input their ridesharing requests with origin/destination of their trips and time windows for pick-up. Our goal is to maximize the fulfillment of two types of demands, while self-covering the operational cost by daily revenue.

We only provide round-trip carsharing services to Type I customers since a one-way trip demand can be considered as a ridesharing service. We assume that demands arise at discrete time periods over a finite horizon. In each period, the unfulfilled demand is immediately lost. We propose a two-phase approach: in Phase I, we maximize the demand fulfillment by accepting enough Type 1 drivers to serve Type 2 users while maintaining the cost self-sustainability of the system; in Phase II, we optimize routing and scheduling for both types of customers. We formulate two integer programs for solving the problems in the two phases and will investigate decomposition-based algorithms for improving the computational efficiency.

1.2 Contributions and Main Results

This paper focuses on developing a new CRS system targeting at underserved communities. We aim to optimize resource pooling, supply-demand matching and shared service scheduling within and across multiple communities. The contributions of this paper are three-fold. Firstly, we propose a new design of sharing service system that integrate carsharing and ridesharing services. Secondly, the classification of the two types of customers/services leads to self-sustained operations of the system via supply-demand matching within the service region. Thirdly, the system will provide prominent solutions to underserved communities, which are currently being left behind by sharing service technologies despite their potential large demand.

We plan to test our approach on instances generated on surveys from underserved populations in Detroit area. The preliminary results on randomly generated instances show the computational efficiency of the two-phase approach. The computational time depends on the size of the fleet and the number of service demands. The system maintains high quality of service as well as high vehicle utilization rate.

2 Problem Formulation

We are given a fleet of vehicles $K = \{1, 2, \ldots, |K|\}$ for serving a set I of zones. We aim to maximize the total number of car/ridesharing requests fulfilled while maintaining the net revenue above a certain threshold value, S, to ensure self-sustainable operations. Let $L = \{1, 2, \ldots, |L|\}$ be the set of reservations received from Type 1 users. Each $l \in L$ corresponds to a tuple, $(i_l, s_l, t_l, g_l^L, h_l^L)$, which indicates the zone i_l for picking up and returning a rental car, the time window $[s_l, t_l]$ during which a car is needed, and the time window $[g_l^L, h_l^L]$ during which the corresponding Type 1 user can provide ridesharing service. Let J be the set of ridesharing demands from Type 2 (non-driver) users. Each $j \in J$ corresponds to a tuple $(o_j, d_j, e_j, g_j^J, h_j^J)$, where o_j and d_j represent the trip's origin and destination, respectively, e_j is the total time needed (including driving time from o_j to d_j, time of loading passengers or goods, etc.), and $[g_j^J, h_j^J]$ is the time window for picking up the corresponding Type 2 user at o_j. We describe the cost parameter settings for fulfilling demands in L and J later when we describe the modeling details for Phase I and Phase II problems.

2.1 Phase I: Carsharing Planning and Operations

For a CRS system, we determine initial locations of $|K|$ cars and construct a spatial-temporal network [1], $G = (N, A)$, to model car movement over T periods to operate the CRS system. Each node $n_{it} \in N$ represents a zone $i \in I$ at period $t \in \{0, 1, \ldots, T\}$. Let the arc set $A = A^R \cup A^{\text{idle}}$, where $A^R = \{(n_{i_l s_l}, n_{i_l t_l}) : l \in L\}$ and $A^{\text{idle}} = \{(n_{it}, n_{it+1}) : i \in I, t \in \{0, 1, \ldots, T-1\}\}$, conveying

arc flows that represent decisions of letting a Type 1 user $l \in L$ rent a car from zone i_l during the time $[s_l, t_l]$ and letting car(s) sit idle at zone $i \in I$ from period t to period $t+1$, for all $t = 1, \ldots, T-1$, respectively.

For each Type 1 demand $l \in L$, we charge r_l^L per period of car use (which depend on the pick-up location and time window to use the car). We also pay the user p_l for each time period he/she serves others. For each Type 2 request $j \in J$, we charge r_j^J per period dependent on the origin, destination, and time window needing the service. A car in use will incur a service cost c^{ser} (including maintainace, insurance, and other costs) per period to the system oprator and will incur an idle cost c_i^{idle} (including necessary costs in c^{ser} and parking cost) if it sits idle at zone $i \in I$. Let f_a and u_a represent the net revenue of generating one unit of flow on arc $a \in A$ and the arc capacity, respectively. For each $a_l \in A^R$, we have $f_{a_l} = (t_l - s_l)(r_l^L - c^{\text{ser}}) - p_l(h_l^L - g_l^L)$ and $u_{a_l} = 1$. For each idle arc $a = (n_{it}, n_{it+1}) \in A^{\text{idle}}$, $f_a = -c_i^{\text{idle}}$ and $u_a = |K|$.

We define integer decision vector $x = (x_i,\ i \in I)^T$ where x_i is the number of cars intially located at zone $i \in I$, binary decision vector $y = (y_a^k,\ a \in A,\ k \in K)^T$ where $y_a^k = 1$ indicates that vehicle k flows on arc a, and 0 otherwise, and binary decision vector $z = (z_{jl}, j \in J,\ l \in L)^T$ where $z_{jl} = 1$ indicates that Type 2 demand $j \in J$ is served by Type 1 demand $l \in L$, and 0 otherwise. We define a binary parameter vector $w = (w_{jl}, j \in J,\ l \in L)^T$, where $w_{jl} = 1$ if ridesharing $j \in J$ can be served by carsharing trip $l \in L$ and 0 otherwise. We have $w_{jl} = 1$ if $|[g_j^J, h_j^J] \cap [g_l^L, h_l^L]| \geq e_j$, meaning that Type 2 user's request $j \in J$ can be performed within the time window specified by Type 1 user $l \in L$ to serve others. We let $\delta^+(n_{it})$, $\delta^-(n_{it})$ be the set of arcs to which node n_{it} is their tail and head, respectively, and formulate an integer program P1 as follows.

$$\textbf{(P1)}\ \max \quad \sum_{a_l \in A^R} \sum_{k \in K} y_{a_l}^k + \sum_{j \in J} \sum_{l \in L} z_{jl} \tag{1}$$

$$\text{s.t.} \quad \sum_{i \in I} x_i \leq |K| \tag{2}$$

$$\sum_{k \in K} \sum_{a \in A} f_a y_a^k + \sum_{j \in J} r_j^J \sum_{l \in L} e_j z_{jl} \geq S \tag{3}$$

$$\sum_{k \in K} \sum_{a \in \delta^+(n_{i0})} y_a^k \leq x_i \qquad \forall i \in I \tag{4}$$

$$\sum_{i \in I} \sum_{a \in \delta^+(n_{it})} y_a^k - \sum_{i \in I} \sum_{a \in \delta^-(n_{it})} y_a^k = \begin{cases} 1 & \text{if } t = 0 \\ 0 & \text{if } t \in \{1, \ldots, T-1\} \\ -1 & \text{if } t = T \end{cases} \qquad k \in K, \tag{5}$$

$$\sum_{k \in K} y_a^k \leq u_a \qquad \forall a \in A \tag{6}$$

$$\sum_{j \in J} z_{jl} e_j \leq \sum_{k \in K} y_{a_l}^k (h_l^L - g_l^L) \qquad \forall l \in L \tag{7}$$

$$z_{jl} \leq w_{jl} \qquad \forall j \in J,\ \forall l \in L \tag{8}$$

$$\sum_{l \in L} z_{jl} \leq 1 \qquad \forall j \in J \tag{9}$$

$$x \in \mathbb{Z}_+^{|I|},\ y \in \{0,1\}^{|A| \times |K|},\ z \in \{0,1\}^{|J| \times |L|}, \tag{10}$$

where the objective (1) maximizes the total number of sharing service fulfillment; constraint (2) limits the number of available cars for the system; constraint (3) ensures that the system achieve the minimum daily revenue requirement; constraints (4) enforce that the sum of outgoing flows from node n_{i0} is bounded by x_i for all $i \in I$; constraints (5) are flow balance constraints; constraints (6) are the capacity constraints for each arc; constraints (7) ensure that the accepted carsharing requests can potentiallly provide sufficient total time to serve accepted ridesharing requests; constraints (8)

ensure that $z_{jl} = 1$ only if carsharing trip l can serve ridesharing demand j; constraint (9) ensures that each ridesharing request will be served at most once.

2.2 Phase II: Ridesharing Routing and Scheduling

The solution to P1 gives the information on shared cars that are available to provide ridesharing. Now we construct Phase II model to find the routing and scheduling decisions for both customers.

We define a network $G'(V, E)$ where V is the node set and E is the edge set. Let $V = V_0 \cup V_1 \cup V_2$ where $V_0 = V_1 = \{i_l : l \in L\}$ is the location set of the depots and $V_2 = \{o_j, d_j : j \in J\}$ is the location set of Type 2 users. Let $E = E_0 \cup E_1 \cup E_2 \cup E_3$ where $E_0 = \{(i_l, o_j) : i_l \in V_0, o_j \in V_2\}$ contains edges connecting depots to ridesharing locations, $E_1 = \{(o_j, d_j), j \in J\}$ contains edge connecting the origin and destination of ridesharing service, $E_2 = \{(d_j, o_v) : j \neq v \in V_2\}$ contains edges connecting different ridesharing locations, and $E_3 = \{(d_j, i_l) : d_j \in V_2, i_l \in V_1\}$ contains edges connecting ridesharing locations back to depots. We use c_{uv} as the traveling time between two nodes such that $(u, v) \in E$.

Define binary decision vector $\alpha = (\alpha^l_{uv},\ (u, v) \in E,\ l \in L)^T$ where $\alpha^l_{uv} = 1$ indicates that carsharing service l travels along arc (u, v), and 0 otherwise, binary vector $\beta = (\beta_j,\ j \in J)^T$ where $\beta_j = 1$ indicates that the ridesharing service $j \in J$ gets served, and 0 otherwise, continuous decision vector $\gamma = (\gamma_v,\ v \in V)^T$ where γ_v is the time that location v gets visited. We formulate ridesharing scheduling problem as a vehicle routing problem with time windows and multiple depots [2]:

$$
\begin{aligned}
(\mathbf{P2})\ \text{maximize} \quad & \sum_{j \in J} \beta_j && && (11)\\
\text{subject to} \quad & \sum_{j \in J} r^J_j \beta_j \geq \sum_{l \in L} p_l (h^L_l - g^L_l) && && (12)\\
& \sum_{l \in L} \sum_{u:(u,o_j) \in E_1 \cup E_2} \alpha^l_{uo_j} \leq \beta_j && \forall o_j \in V_2, && (13)\\
& \sum_{v:(i_l,v) \in E_0} \alpha^l_{i_l v} \leq 1 && \forall l \in L && (14)\\
& \sum_{u:(u,i_l) \in E_3} \alpha^l_{u i_l} \leq 1 && \forall l \in L, && (15)\\
& \sum_{v:(u,v) \in E} \alpha^l_{uv} - \sum_{v:(v,u) \in E} \alpha^l_{vu} = 0 && \forall u \in V_2,\ \forall l \in L, && (16)\\
& \gamma_{i_l} + c_{i_l v} - T(1 - \alpha^l_{i_l v}) \leq \gamma_v && \forall (i_l, v) \in E_0, && (17)\\
& \gamma_{o_j} + e_j + c_{d_j v} - T(1 - \sum_{l \in L} \alpha^l_{d_j v}) \leq \gamma_v && \forall (d_j, v) \in E_2 \cup E_3, && (18)\\
& g^L_l \leq \gamma_{i_l} \leq h^L_l && \forall i_l \in V_0 \cup V_1, && (19)\\
& g^J_j \leq \gamma_{o_j} \leq h^J_j && \forall o_j \in V_2, && (20)\\
& \alpha^l_{uv} \in \{0, 1\} && \forall (u, v) \in E,\ \forall l \in L, && (21)\\
& \beta_j \in \{0, 1\} && \forall j \in J, && (22)
\end{aligned}
$$

where the objective (11) maximizes the total number of ridesharing services we fulfill; constraint (12) ensures that the revenue we receive from fulfilling Type 2 demand is sufficient to cover the payment to Type 1 drivers; constraints (13) ensure that $\beta_j = 1$ if the location o_j is visited by a driver; (14)–(15) specify the origin and destination of the flow for each driver with demand $l \in L$; constraints (16) are the flow balance constraints; constraints (17)–(20) ensure that the arrival time at each location satisfying corresponding time window. Moreover, constraints (17)-(18) also serve

as subtour elimination constraints by enforcing the arrival time at location v to be greater than the arrival time at location u plus traveling and service time if a driver travels from u to v directly.

3 Numerical Results

We test our model on randomly generated instances to demonstrate the preliminary computational results for our approach. We generate 6 instances with various sizes of fleet and different amount of demands in a system with $|I| = 5$ and $T = 8$. The carsharing demands are uniformly distributed in each zone and the rental time windows are randomly generated with length at most 4 periods. We generate the location of carsharing stations and ridesharing demands on a $[0, 1]^2$ square and set the traveling time between two locations as their Euclidean distance. The time windows and service time for ridesharing demands are also randomly generated. The price/cost parameters are set according to real practice from Zipcar operations in Detroit area[1]. The numerical experiments are conducted on a desktop with i7-6700K CPU at 4.00GHz and 32GB memory. We use Gurobi 7.0.1 as the integer programming solver for P1 and P2. We report the computational time and the quality of service in Table 1 with different fleet sizes ($|K|$), various amount of carsharing demands ($|L|$) and ridesharing demands ($|J|$).

Table 1: Numerical Results for Proposed Model

	$\|K\|$	$\|L\|$	$\|J\|$	Phase I Time(s)	Phase II Time(s)	Fulfilled Carsharing	Fulfilled Ridesharing
Instance 1	20	20	40	0.03	10.39	19	40
Instance 2	20	40	40	0.01	68.62	40	40
Instance 3	20	40	80	24.81	832.03	40	80
Instance 4	30	30	60	0.25	115.84	30	60
Instance 5	30	60	60	0.04	2015.63	57	60
Instance 6	30	60	120	276.46	1592.48	59	120

The results show that we can use general commercial optimization tool to solve the models efficiently for small instances. However, with an increasing number of demands, the solution time for Phase II model grows fast. Results also show that the proposed CRS system has high quality of service for providing sharing services. Our preliminary results indicate that the size of fleet is the key factor that causes demand loss for carsharing in Phase I. According to the solutions, for the carsharing demands that have been rejected, they in general offer narrower time windows for providing ridesharing. In addition, we notice that the ridesharing demands can be effectively fulfilled in the test instances.

For the next step, we plan to demonstrate the computational results of our model by testing instances generated on surveys from underserved populations in Detroit area and investigate decomposition-based algorithms to improve the computational efficiency.

References

[1] Lu, M., Shen, S., and Chen, Z. (2016). Optimizing the profitability and quality of service in carshare systems under demand uncertainty. *Working paper*.

[2] Toth, P. and Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications*, volume 18. Siam.

[1] http://www.zipcar.com/detroit?zipfleet_id=1072097971

The Multi-Period Service Planning and Routing Problem

A.H. Schrotenboer, E. Ursavas, and I.F.A. Vis
Department of Operations, Faculty of Economics and Business, University of Groningen,
a.h.schrotenboer@rug.nl, http://www.rug.nl/staff/a.h.schrotenboer

We study a multi-period multi-commodity pickup and delivery problem with a heterogeneous fleet inspired by the short-term planning of maintenance activities for offshore operations. In this Multi-Period Service Planning and Routing Problem, each maintenance service requires spare parts as well as different types of servicemen. This leads to a mix of one-to-one and many-to-many pickup and delivery structures for the construction of the daily vehicle routes. Travel times and costs are period dependent to model a wide variety of application dependent characteristics in a unified way. We propose a branch-and-price-and-cut algorithm to solve this problem. It relies on efficiently solving a new variant of the Resource Constrained Elementary Shortest Path Problem. Preliminary experiments show that the algorithm easily finds solutions to instances with 25 maintenance services, demanding three types of servicemen, over a five-period planning horizon.

Key words: Pickup and Delivery Problem, Multi-commodity, Multi-Period, Branch-and-Price-and-Cut, Maintenance

1. Introduction

The short term planning of maintenance services that take place at geographically scattered locations is a frequently encountered decision problem in maintenance service logistics. In the Multi-Period Service Planning and Routing Problem (MSPRP), we consider a maintenance service provider for offshore operations that needs to perform such a set of geographically scattered services $\{1,\ldots,n\}$, which all need to be planned over a time horizon $\{1,\ldots,T\}$. A service is started if the right amount of spare parts and the right number of differently skilled servicemen are delivered at the service location. After completion of the service, the servicemen need to be picked up again to be delivered at their next-scheduled services. These delivery and pickup tasks are performed by K heterogeneous vehicles, capacitated in the total weight of the spare parts and the number of serviceman. Every service can be started and completed within a single day, and we assume that they are planned in such a way. Vehicles are allowed to continue with the remaining pickup/delivery tasks following the delivery of the servicemen at a service location. However, each vehicle is responsible for the pickup of servicemen if it has done the corresponding delivery. They should thereby respect service times and the designated maximum daily working hours.

We model the MSPRP as a pickup and delivery problem: We aim to develop cost-minimizing routes such that spare parts and servicemen are picked up and delivered between service locations, for each vehicle in each period, assuring the performance of all maintenance services. To encompass restrictions of offshore operations, we let travel times and costs be vehicle and period dependent, allowing the

modelling of a wide variety of application dependent characteristics in a unified way. For example, the different cost structures of corrective and preventive maintenance services, as well as the influence of weather conditions on the travel times, can be characterized in this way.

We present an efficient branch-and-price-and-cut algorithm for solving the MSPRP. It relies on efficiently solving pricing problems that are obtained by decomposing the problem for each vehicle and period. The pricing problems are a new variant of the Elementary Resource Constrained Shortest Path Problem (ERCSPP). The ERCSPP is solved by an algorithm that first provides lower bounds of partial paths, and second, uses that information in a depth first search to prune partial paths, in the line with the approach of Lozano, Duque, and Medaglia (2015). We show its applicability by applying it to a practically inspired case constructed for the maintenance service logistics at offshore wind farms (see, e.g., Irawan et al. (2016)).

The MSPRP has distinguishing characteristics that makes it a novel problem to study. First, it mixes a one-to-one pickup and delivery structure with a many-to-many pickup and delivery structure, see Berbeglia et al. (2007) for an overview of these structures. A one-to-one delivery and pickup structure (see, e.g., Ropke and Cordeau (2009)) exists between nodes that represent the start and completion of a service, respectively. The many-to-many pickup and delivery structure (see, e.g., Hernández-Pérez and Salazar-González (2014)) exists between multiple services; servicemen of a completed service can be used to start other services. Second, respecting the service times yield so-called delayed precedence constraints between the delivery and pickup of servicemen, i.e., time windows at the pickup nodes depend on the arrival time at the corresponding delivery nodes. Third, the MSPRP is essentially a multi-commodity pickup and delivery problem in a multiple vehicle setting, for which no sophisticated branch and bound algorithms have been developed so far.

The remainder of this abstract is as follows. The next section introduces the mathematical model used to solve the MSPRP. We conclude with some promising preliminary experiments in Section 3.

2. Model description

Let $G = (N, A)$ be a directed graph with a set of nodes N and a set of arcs A. The node set N consists of delivery nodes $N_d = \{1, \dots, n\}$, pickup nodes $N_p = \{n+1, \dots 2n\}$, and the origin and destination depot $\{0, 2n+1\}$. Every delivery node i has a corresponding pickup node $i+n$, who represent the start and the completion of service i, respectively. Each service demands known amounts of L different types of servicemen. The number of available servicemen of type ℓ in period T equals $\tilde{Q}_{\ell t}$. A vehicle $k \in \mathcal{K}$ has spare parts capacity $\bar{Q}^1$ and servicemen capacity $\bar{Q}^2$.

We propose a set covering formulation to solve this problem. First, let $\mathcal{R}$ be the set of all feasible routes that can be constructed in the MSPRP. Due to the heterogeneity of vehicles and discrepancies between periods, a route's costs and feasibility may differ between vehicles and periods. We therefore let $\mathcal{R} = \cup_{k \in \mathcal{K}, t \in \mathcal{T}} \mathcal{R}_{kt}$, where $\mathcal{R}_{kt}$ are the sets of feasible routes for vehicle k in period t. For some route $r \in \mathcal{R}_{kt}$, let y_r^{kt} be a binary decision variable that equals 1 if route r is chosen and 0 otherwise. In addition, let c_r^{kt} be the corresponding costs , β_i^{ktr} be the number of times node $i \in N_d \cup N_p$ is visited,

and γ_ℓ^{ktr} be the number of commodities (i.e, the number of servicemen types) of type $\ell \in \mathcal{L}$ needed in route $r \in \mathcal{R}_{kt}$

A set covering formulation is then given by:

$$\min \quad \sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_{kt}} c_r^{kt} y_r^{kt} \tag{1}$$

$$\text{S.t.} \sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_{kt}} y_r^{kt}\beta_i^{ktr} \geq 1 \qquad \forall\ i \in N_d \tag{2}$$

$$\sum_{r\in\mathcal{R}_{kt}} y_r^{kt} \leq 1 \qquad \forall\ k \in \mathcal{K},\ t \in \mathcal{T} \tag{3}$$

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_{kt}} y_r^{kt}\gamma_\ell^{ktr} \leq \tilde{Q}_{lt} \qquad \forall\ t \in \mathcal{T},\ \ell \in \mathcal{L} \tag{4}$$

$$y_r^{kt} \in \{0,1\} \qquad \forall\ k \in \mathcal{K},\ t \in \mathcal{T},\ r \in \mathcal{R}_{kt} \tag{5}$$

The objective (1) calculates the costs for using the selected routes from each subset $\mathcal{R}_{kt}$. Equation (2) ensures that every node is visited at least once. Equation (3) ensures that every vehicle in every period is used at most once, which is necessary due to heterogeneity of vehicles and discrepancies between periods. Equation (4) ensures that there are enough commodities available at the depot. We will refer to the model described by equations (1) - (5) as the Integer Programming Master (IPM) problem.

To find the LP relaxation of (IPM), we consider restricted route sets $\overline{\mathcal{R}}_{kt} \subset \mathcal{R}_{kt}$. We iteratively solve the LP relaxation of (IPM) subject to $\overline{\mathcal{R}}_{kt}$ and generate paths for each $\overline{\mathcal{R}}_{kt}$ by solving the (k,t)-pricing problem. The LP relaxation of (IPM) is found if no path of negative reduced cost is found for each (k,t)-pricing problem.

To formulate the (k,t)-pricing problems, let μ_i, λ_{kt}, and $\pi_{\ell t}$ be dual variables corresponding to constraints (2) - (4), respectively. Let d_{ij}^{kt} be the costs of traversing arc (i,j) in some (k,t)-pricing problem. We define

$$d_{ij}^{kt} = \begin{cases} c_{ij}^{kt} - \mu_i & \text{if } j \in N_d, \\ c_{ij}^{kt} & \text{otherwise.} \end{cases}$$

Similarly, let $\tilde{d}_{\ell t} = \tilde{c}_{\ell t} - \pi_{\ell t}$. Then the (k,t)-pricing problems are given by

$$\min_{r\in\mathcal{R}_{kt}} \left\{ \sum_{(i,j)\in A} d_{ij}^{kt} r_{ij}^{kt} + \sum_{\ell\in\mathcal{L}} \tilde{d}_{\ell t} r_{\ell t} - \lambda_{kt} \right\}, \forall\ k \in \mathcal{K},\ t \in \mathcal{T},$$

where $r_{ij}^{kt} = 1$ if arc (i,j) is used by path $r \in \mathcal{R}_{kt}$, and $r_{0\ell}$ are the number of demanded servicemen of type $\ell \in \mathcal{L}$ in path $r \in \mathcal{R}_{kt}$. The (k,t)-pricing problems are new variants of the Resource Constrained Elementary Shortest Path Problem, namely, it comprises a multi commodity mixed pickup and delivery structure combined with time based precedence constraints. We developed an efficient algorithm to solve the pricing problems. It first provides lower bounds to partial paths, and afterwards, this information is used in a depth first search to efficiently prune partial paths. It is a rigorously adapted variant of the algorithm by Lozano, Duque, and Medaglia (2015), i.e., a similar approach for solving the pricing problems arising in a vehicle routing problem with time windows.

Table 1 Characteristics of the vehicles used in the instance sets

vehicle	$\bar{Q}^1$	$\bar{Q}^2$	relative fuel cost	relative travel speed
1	600	12	1	1
2	1000	16	1.2	0.8
3	1000	12	1.1	0.9

To strengthen the LP relaxations, valid inequalities are added after solving the pricing problems at a branch and bound node. We adapted 2-path inequalities, fork inequalities and capacity inequalities (see, e.g., Ropke, Cordeau, and Laporte (2007)) such that they are valid for the MSPRP.

Branching is done in a three-stage approach. First, we branch on the number of vehicles (Ropke and Cordeau 2009), i.e., we consider the number of outgoing edges from the depot in the LP relaxation. If we cannot branch by this rule, i.e., if the number of vehicles is integer, we continue with the second branching rule. It sums the use of an edge over the vehicle and period dimension. If no such summations are fractional, we consider branching on individual edges of the underlying graph G. This three-stage approach outperforms branching on the set partition variable y_r^{kt}, since that leads to unbalanced branch and bound trees (Irnich and Desaulniers 2005).

3. Preliminary Results

In this section, we study a practically inspired case of maintenance service logistics for offshore wind farms, first introduced by Dai, Stålhane, and Utne (2015). Recently, Irawan et al. (2016) propose an exact solution method based on a set partitioning formulation and a total enumeration of all possible routes for a multi-depot setting. Their approach inherently causes memory problems for larger problem sizes, although the inclusion of multiple depots causes a more restricted solution space, i.e., services are not allowed to be served from all depots and only services within a single wind farm (out of three) could be visited by a single route. The largest instances they proposed solutions for (without guaranteed optimality) consists of routes with at most 12 services, i.e., 12 turbines per wind farm.

We created two sets of new instances to test the efficiency of the branch-and-price-and-cut algorithm. We implemented it using the framework for constraint programming SCIP 3.2.1 (Gamrath et al. 2016) in combination with CPLEX 12.6.1 as an LP-solver. The overall program is coded in C++. Daily working hours are randomly drawn between 8 and 12 hours, spare parts weights are randomly drawn between 200 and 400 kg. Three different types of servicemen are included, and the demands for these types of servicemen are randomly between 1 and 2. The number of available servicemen are per period per type randomly drawn between 4 and 6. The vehicles characteristics are given in Table 1. The first two vehicles are used in instance set 1, and all vehicles are included in instance set 2. The results are presented in Table 2. As can be seen, even with a time limit of 120 seconds on the computation time, instances up to 25 maintenance services can easily be solved.

4. Conclusion

In this paper, we researched a pickup and delivery problem arising in offshore maintenance operations. We proposed a branch-and-price-and-cut algorithm based on a set partition formulation decomposed for every vehicle and period. Its efficiency is shown by solving two new sets of instances.

Table 2 Results of preliminary experiments

Set	Instance	n	K	T	LB	UB	Gap (%)	time (s)
1	1	20	2	5	40797.32	40797.32	**0.00**	3.36
	2	20	2	5	38700.80	38717.07	0.04	120.00
	3	20	2	5	34253.41	34253.41	**0.00**	26.57
	4	20	2	5	41249.51	41249.51	**0.00**	36.51
	5	20	2	5	36819.32	36819.32	**0.00**	8.10
	6	25	2	5	50394.84	50730.17	0.67	120.00
	7	25	2	5	47364.08	47950.53	1.24	120.00
	8	25	2	5	47554.99	47554.99	**0.00**	55.04
	9	25	2	5	54591.77	54834.14	0.44	120.00
	10	25	2	5	44350.86	44350.86	**0.00**	29.30
2	1	20	3	5	37545.88	37591.17	0.12	120.00
	2	20	3	5	32211.08	32211.08	**0.00**	8.02
	3	20	3	5	30694.82	30694.82	**0.00**	24.63
	4	20	3	5	32907.11	32907.11	**0.00**	11.00
	5	20	3	5	39882.72	39882.72	**0.00**	70.08
	6	25	3	5	45135.46	45264.21	0.29	120.00
	7	25	3	5	43730.12	44404.96	1.54	120.00
	8	25	3	5	42960.33	42960.33	**0.00**	47.91
	9	25	3	5	40772.26	40772.26	**0.00**	37.26
	10	25	3	5	38274.49	38529.93	0.67	120.00

References

Berbeglia G, Cordeau JF, Gribkovskaia I, Laporte G, 2007 *Rejoinder on: Static pickup and delivery problems: a classification scheme and survey. TOP* 15(1):45–47, URL `http://dx.doi.org/10.1007/s11750-007-0015-2`.

Dai L, Stålhane M, Utne IB, 2015 *Routing and scheduling of maintenance fleet for offshore wind farms. Wind Engineering* 39(1):15–30.

Gamrath G, Fischer T, Gally T, Gleixner AM, Hendel G, Koch T, Maher SJ, Miltenberger M, Müller B, Pfetsch ME, Puchert C, Rehfeldt D, Schenker S, Schwarz R, Serrano F, Shinano Y, Vigerske S, Weninger D, Winkler M, Witt JT, Witzig J, 2016 *The scip optimization suite 3.2.* Technical Report 15-60, ZIB, Takustr.7, 14195 Berlin.

Hernández-Pérez H, Salazar-González JJ, 2014 *The multi-commodity pickup-and-delivery traveling salesman problem. Networks* 63(1):46–59.

Irawan CA, Ouelhadj D, Jones D, Stålhane M, Sperstad IB, 2016 *Optimisation of maintenance routing and scheduling for offshore wind farms. European Journal of Operational Research* .

Irnich S, Desaulniers G, 2005 *Shortest path problems with resource constraints. Column generation*, 33–65 (Springer).

Lozano L, Duque D, Medaglia AL, 2015 *An exact algorithm for the elementary shortest path problem with resource constraints. Transportation Science* 50(1):348–357.

Ropke S, Cordeau JF, 2009 *Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science* 43(3):267–286.

Ropke S, Cordeau JF, Laporte G, 2007 *Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks* 49(4):258–272.

Covering Tour Problem with an Application to School Bus Routing: Analysis of Single Vehicle Tours on a Grid

January 5, 2017

1 Introduction

In the face of critical budget cuts, many school districts are looking to reduce transportation expenditures. This paper presents initial work on the School Bus Routing Problem (SBRP), motivated by a partnership between Evanston/Skokie District 65 (D65), a pre-K - 8 public school district north of Chicago, and Northwestern University focused on bus transportation. The SBRP has been studied by the operations research community for fifty years, identifying creative routing and scheduling approaches for school districts. The SBRP itself is a composite of five decision subproblems: data preparation, bus stop selection, bus route generation, school bell time adjustment, and route scheduling. In our work, we take a new approach to the joint problem of bus stop selection and bus route generation, exploiting the underlying grid-like structure of the road network present in the D65 service region to obtain robust, easy-to-implement solutions.

An efficient and sustainable mechanism for route design is crucial in the SBRP; such a mechanism would provide benefit in many aspects, including saving money for schools/organizations and reducing walking time for students. In many cases of SBRP (especially in urban areas), students are not picked up at their homes, but rather are assigned to a bus stop such that the walking distance for each student is no more than a predetermined constant. It is natural to link bus stop selection and bus route generation. We model the joint problem as a covering tour problem (CTP). The CTP is a variant of the traveling salesman problem (TSP). The major difference between these two problems is that we are not required to visit every node directly in CTP. Instead, we identify the least cost tour of a subset of the nodes such that every node not on the tour is within some predetermined distance of a node that is on the tour.

In developing solution approaches for the CTP, we exploit characteristics of urban bus routing, most notably the underlying grid structure. A grid graph is a graph whose nodes correspond to integer points in the plane, x-coordinate being in the range 1 to m, y-coordinate in 1 to n. Two nodes are connected if they are at distance 1. We show in our analysis that the grid structure greatly reduces the complexity of the problem. We begin our analysis in this paper with a stylized setting of one bus serving students located uniformly on a complete grid. In the presentation, we will introduce our approach to the CTP and its applications to SBRP. In this abstract, we summarize key findings for the simple setting that will inform solution approaches for more general settings. The presentation will discuss both the simple settings and generalizations for more complex settings.

2 Literature review

The CTP, first introduced by Current (1981), is NP-hard as it reduces to TSP when the covering distance goes to 0. The problem was first formulated as an integer linear program in Current and Schilling (1989). Gendreau et al. (1997) provide polyhedral study for this problem as well as a branch-and-cut algorithm. The CTP can also be formulated as a generalized traveling salesman problem (GTSP) (see Fischetti et al. (1997)): given several sets of nodes, the GTSP seeks to determine a shortest tour passing at least once through each set. Recent CTP work continues the design of heuristics for both CTP and multi-vehicle CTP, see Jozefowiez (2014), Murakami (2014) and Leticia Vargas et al. (2015). Different from these papers, our approach is to impose structure on the underlying graph and develop solution methods with provable bounds based on the structure.

Some of the most fundamental combinatorial optimization problems are well studied on grid graphs. Itai et al. (1982) proved that the Hamiltonian problem is NP-hard on general grid graphs, but can be solved in polynomial time on graph without holes (Umans and Lenhart (1997)). Recent advances in the TSP also indicate potential benefits of working on grid. Arkin et al. (2000) provide a polynomial algorithm for TSP on simple grid graph with approximation ratio $\frac{6}{5}$. Gharan et al. (2011) provide a $1.5 - \varepsilon$ polynomial approximation algorithm for graph TSP, which is the first result that beat Christofides heuristic with 1.5 approximation ratio. This ratio was later improved to 1.4 by Sebo and Vygen (2012). Our work is a continuation of this stream of work to use grid characteristics for complex problems.

3 Problem setting

Consider a m by n unit grid with $N = mn$ nodes. Given $k \in \mathbb{N}$, we say node A covers node B if the distance between A and B is no more than k. The covering tour problem on grid can be stated as follows: for given parameters m, n, k, find a minimum cost tour such that each node in the grid graph is covered by at least one node in the tour. Here we use l_1 norm to measure distance, which is the shortest path length between two nodes if we are only allowed to travel along edges in the grid graph. We consider two costs motivated by our school bus routing problem: tour cost, which is a function of tour length; and fixed cost, which is a function of the number open bus stops, since each stop requires a fixed stopping time. In what follows, we first minimize each cost individually and then use these results to analyze the trade-off when both costs are considered.

3.0.1 Main results: 1 school, 1 bus, complete grid

In the stylized setting of a single vehicle serving a single school on a complete grid, we have established the following fundamental results that will be used in more general settings. We solve the following decision making problem in different settings: given a m by n grid and covering radius k, for parameters L, T, determine if there exist a covering tour such that the tour length is L and the number of stops is T. We present the following main results of this initial study.

Theorem 1 (**Minimum stop count**)**.** The minimum number of stops in a covering tour is $T^* = \frac{N}{2k^2+2k+1} + O(m)$.

To achieve the minimum fixed cost (minimum stop count), open stops are arranged on the grid such that each open stop covers a diamond region of $(2k^2 + 2k + 1)$ nodes. In this manner, this diamond region forms a tessellation on $\mathbb{Z}^2$ so that $\frac{N}{2k^2+2k+1} + O(m)$ stops are needed to cover the grid. The $O(m)$ stops are used to cover the boundary of the grid.

Theorem 2 (**Minimum tour length**)**.** The minimum length of a covering tour is $L^* = \frac{N}{2k+1} + O(m)$.

Notably, these two minima can often not be achieved simultaneously.

Theorem 3 (**Multiple objective trade-off**). In the multi-objective setting considering both tour length and fixed stop cost, one can not achieve L^* and T^* simultaneously. Moreover, the two costs satisfy

$$N \leq T\hat{f}(\frac{L}{T})$$

where $\hat{f}$ is the piecewise-linear function associated with the following discrete function f:
$f(1) = 2k+1$, $f(2k+1) = 2k^2+2k+1$;
$f(d) = d(2k+1-\frac{d}{2})$ for $d = 2, 4, \cdots, 2k$;
$f(d) = d(2k+1-\frac{d}{2}) - \frac{1}{2}$ for $d = 3, 5, \cdots, 2k-1$.

For minimum tour cost and the multi-objective case, we point out that the average distance between consecutive stops $\frac{L}{T}$ quantifies the trade-off between tour cost and fixed cost. In general, small $\frac{L}{T}$ leads to lower tour cost and large $\frac{L}{T}$ helps minimize fixed cost. Below is the sketch of proof for above theorems.

Let $F_1 - F_2 - \cdots - F_T - F_1$ be a covering tour and d_i is the distance between F_i and F_{i+1} ($F_{T+1} = F_1$), then the total tour length is

$$L = \sum_{i=1}^{T} d_i.$$

Let S_i denote the set of nodes covered by F_i, since all nodes are covered, then

$$N \leq |\cup_{i=1}^{T} S_i| \leq \sum_{i=1}^{T} |S_{i+1} - S_i| \leq \sum_{i=1}^{T} f(d_i).$$

From the concavity of f we have

$$N \leq \sum_{i=1}^{T} |S_{i+1} - S_i| \leq T\hat{f}(\frac{L}{T}).$$

Note that $\hat{f}$ is a concave piecewise-linear function and can be reformulated as the minimum of several linear functions; i.e. $\hat{f}(d) = \min_i\{a_i d + b_i\}$. Hence, $N \leq T\hat{f}(\frac{L}{T})$ is equivalent to $N \leq \min_i\{a_i L + b_i T\}$, the boundary of which is a piecewise-linear convex function.

The turning points of the boundary are:

$$T(d) = \frac{N}{f(d)}, L(d) = \frac{Nd}{f(d)}$$

where $d = 1, 2, 4, 6, \cdots, 2k-2, 2k, 2k+1$.

Theorem 4 (**Tightness of trade-off inequality**). The inequality in Theorem 3 characterizes the boundary of all feasible pairs of (L, T) almost exactly.

Remark: Notice that Theorem 3 implies Theorem 2 if we set covering radius $k = 0$. We also note that for general grid graphs with N nodes, Theorems 1 and 2 provide lower bounds for tour cost and fixed cost and the inequality in Theorem 3 still holds.

To show the tightness result, we first construct zigzag tours with $T = T(d) + O(m)$ and $L = L(d) + O(m)$, where $d = 1, 2, 4, \cdots, 2k-2, 2k, 2k+1$. Then we reach all other pairs of (L, T) by combining two of these covers. Figure 1 shows the tightness result when $m = n = 2000, k = 10$. The red polyline represents the trade-off constraint while the blue points are data from constructions. The small gap between theoretical bound and real data is of order $O(m)$, which confirms that the theoretical bound is indeed tight.

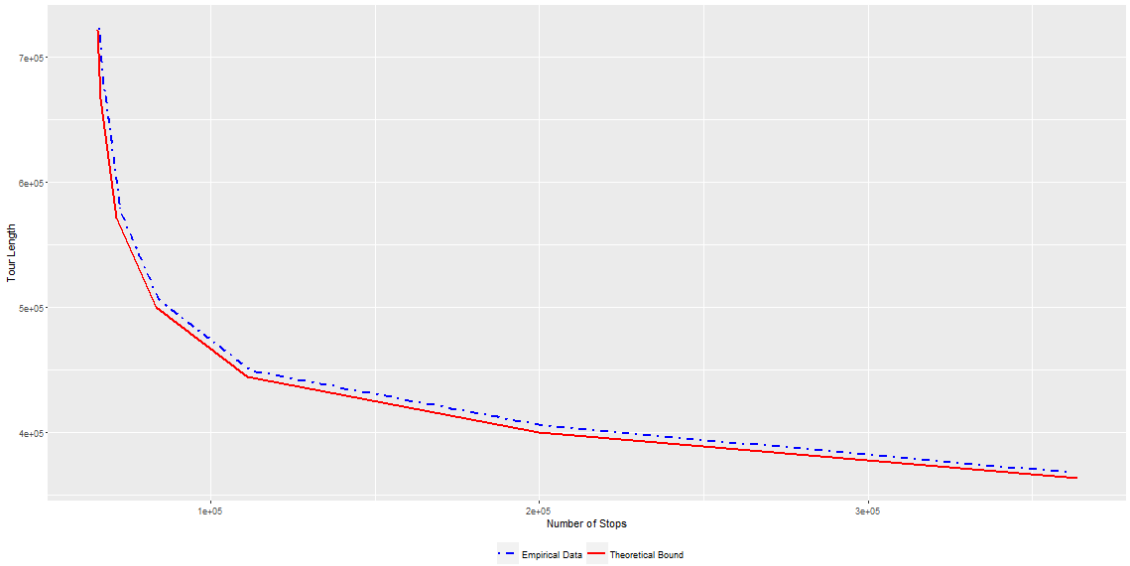


Figure 1: Tightness results

4 Conclusion and future work

These results for a stylized setting provide insights in the trade-off between tour cost and fixed cost, and lay the foundation for our analysis of the SBRP for D65. Based on simplicity of these preliminary results and the structure of the true D65 road network, our future work will continue to generalize the problem setting. This work joins a growing body of work that looks to more efficiently solve real life routing problems by exploiting the underlying well-structured graph.

References

E. M. Arkin, S. P. Fekete, and J. S. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1):25–50, 2000.

J. R. Current. Multiobjective design of transportation networks. Technical report, 1981.

J. R. Current and D. A. Schilling. The covering salesman problem. *Transportation Science*, 23(3): 208–213, 1989.

M. Fischetti, J. J. Salazar González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997.

M. Gendreau, G. Laporte, and F. Semet. The covering tour problem. *Operations Research*, 45(4): 568–576, 1997.

S. O. Gharan, A. Saberi, and M. Singh. A randomized rounding approach to the traveling salesman problem. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 550–559. IEEE, 2011.

A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

N. Jozefowiez. A branch-and-price algorithm for the multivehicle covering tour problem. *Networks*, 64(3):160–168, 2014.

B. Leticia Vargas, N. Jozefowiez, and S. U. Ngueveu. A selector operator-based adaptive large neighborhood search for the covering tour problem. In *Learning and Intelligent Optimization: 9th International Conference, LION 9, Lille, France, January 12-15, 2015. Revised Selected Papers*, volume 8994, page 170. Springer, 2015.

K. Murakami. A column generation approach for the multi-vehicle covering tour problem. In *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1063–1068. IEEE, 2014.

A. Sebo and J. Vygen. Shorter tours by nicer ears. *arXiv preprint arXiv:1201.1870*, 2012.

C. Umans and W. Lenhart. Hamiltonian cycles in solid grid graphs. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 496–505. IEEE, 1997.

Coordinated Delivery to Nanostores in Megacities

Ruidian Song[a*], Lei Zhao[a†], Jan C. Fransoo[b‡], Tom Van Woensel[b§]
[a]Department of Industrial Engineering, Tsinghua University, Beijing, 100084, China
[b]School of Industrial Engineering, Eindhoven University of Technology, Eindhoven, Netherlands

In developing economies, especially in Asia and Latin America, the major consumer goods retail channel consists of thousands of small traditional "mom-and-pop" retail stores. We refer to these small, family-owned, and independently operated retail stores as *nanostores*. It is estimated that there are more than 30,000 nanostores in the core area of Beijing, China. In Mexico, Coca-Cola supplies 1.2 million points of sale (Blanco and Fransoo, 2013).

To better understand these nanostores, we interviewed 50 store owners in Beijing from 2012 to 2014. Unlike modern channel retailers such as chain hyper/supermarkets and convenience stores, nanostores are normally not equipped with retail management information systems and are operated mostly by the store owners with experiences (or "heuristics"). Most nanostores are located in and serve a residential neighborhood. They customize their assortments to adapt to the neighborhood and the demands are relatively stable.

They may (visually) review their inventory routinely (e.g., daily) or when a customer shops for a particular item. When the inventory drops below a certain threshold ("safety stock"), they place an order to the wholesaler (or the manufacturer with direct distribution channel). The order quantity has to be multiple of certain batch size and constrained by the available shelf space.

The deliveries vary from half an hour to two days. Normally, store owners are not very strict on delivery time, unless a late delivery causes out-of-stock or lost-sale in the store. Sometimes, the wholesaler (or manufacturer) send pre-sales representatives to visit the nanostores, checking the presentation of their products, introducing new products, and convincing/helping the store owners to place a replenishment order.

Constrained by the shelf space and available cash or credit, the store owners replenish their inventories with small order sizes and high order frequencies. Consequently, the replenishment to the nanostores results in a large number of uncoordinated deliveries to these stores, which poses serious challenges in the corresponding logistics management.

In this paper, we study, from the wholesaler's (or manufacturer's) perspective, how to coordinate the delivery to the nanostores to reduce the logistics cost, while maintaining service quality to these nanostores. We refer to this as the Coordinated Delivery Problem (CDP). We study two versions of the CDP. In the *Passive* CDP, the wholesaler can delivery to a nanostore only after the store owner places an order, while in the *Proactive* CDP, the wholesaler is allowed to deliver before the store owner place an order, e.g., via pre-sales effort.

*srd13@mails.tsinghua.edu.cn
†lzhao@tsinghua.edu.cn
‡J.C.Fransoo@tue.nl
§T.v.Woensel@tue.nl

Specifically in our paper, a wholesaler serves a set of N nanostores over a finite (cyclic) planning horizon T (e.g., a week). Each nanostore faces deterministic demand of a single product, which can be store specific and time varying within T. The inventory policy of nanostore i can be approximated as an (s_i, S_i, nB) policy, where s_i represents the reorder point, S_i the upper limit of replenishment quantity, B the batch size, and n the number of batches in the order. Note that while s_i and S_i are store specific, the batch size B is common to all nanostores. At a reorder point, nanostore i orders as many batches as possible under the S_i.

The wholesaler has a single vehicle with capacity Q. The vehicle can travel multiple trips to serve the nanostores within the duration of each period (e.g., a day). The wholesaler can schedule the delivery to the nanostores to coordinate and consolidate the orders to save delivery cost. However, unmet demand at a nanostore will result in unsatisfactory of the store owner to the wholesaler's delivery service, represented as a penalty to the wholesaler. Further, postponed delivery (in the passive and proactive CDP) and advance delivery (in the proactive CDP) are also subject to penalty cost. The penalty for advance delivery in the proactive CDP represents the pre-sales effort to convince the store owner to place an order earlier than the reorder point. Besides, to ensure the cyclic pattern of the planning horizon, we also penalize the deviation from the target inventory level at each nanostore. We extend the two-commodity flow formulation (Baldacci et al., 2004) to multiple periods to model the CDP as a mixed integer program (MIP).

Our work falls into the stream of literature on Inventory Routing Problem (IRP), first introduced in gas industrial distribution by Bell et al. (1983). We refer interested readers to Bertazzi and Speranza (2013) and Coelho et al. (2014) for comprehensive reviews. In our paper, we study the (s, S, nB) policy at nanostores, with batch size consideration of replenishment orders. Besides, stockouts or lost sales at nanostores are allowed but penalized to the wholesaler, and both postponed and advance deliveries to nanostores are also penalized.

We plan for the coordinated delivery to nanostores within a finite time horizon T. Therefore, CDP is also closely related to the Periodic Vehicle Routing Problem (PVRP), first introduced by Beltrami and Bodin (1974). In PVRP, customers (e.g., grocery stores, garbage stations) require one or multiple visits within a (cyclic) planning horizon (e.g., a week) and there are a set of feasible visit options (delivery *scheme*s, or day-combinations, e.g., {Monday, Wednesday, Friday} and {Tuesday, Thursday, Saturday}) for each customer. The typical objective is to assign each customer a feasible delivery scheme to minimize the total travel distance in the planning horizon. A recent review on PVRP is by Campbell and Wilson (2014). Conventionally in PVRP, each customer has a specific set of delivery schemes to select, whereas in CDP, the delivery pattern to each nanostore is influenced by both the nanostore's inventory policy and the wholesaler's delivery decisions. Furthermore, in PVRP, the delivery quantity of each customer is assumed to be constant, and stockout is not allowed, while in CDP, the reorder (thus delivery) quantity may vary and stockout is allowed and penalized.

As for the solution method, while a few researchers study exact methods (Archetti et al., 2007; Baldacci et al., 2011; Coelho and Laporte, 2013), most resort to (meta-)heuristic approaches, such as tabu search (Cordeau et al., 1997; Alonso et al., 2008), a clustering heuristic for the first phase and insertion heuristic for the second phase (Campbell and Savelsbergh, 2004; Laganà et al., 2015), genetic algorithm (Vidal et al., 2012; Park et al., 2016), a hybrid heuristic combining tabu search ingredients and mixed integer programming models (Archetti et al., 2012), an integer-programming based heuristic for the first phase and a record-to-record travel algorithm for the second phase (Gulczynski et al., 2011). The solution approach developed in this paper includes two phases and a mixed integer programming is solved in the first phase, which is inspired by Campbell and Savelsbergh (2004), Gulczynski et al. (2011), and Laganà et al. (2015).

The contributions of this paper are three-fold. First, we propose the coordinated delivery problem (CDP) to nanostores and study two versions (passive and proactive) of coordination. CDP enriches the IRP and PVRP literature with several new features. Second, we develop an efficient integer programming based heuristic method, inspired by Gulczynski et al. (2011), to solve the CDP instances with realistic sizes. Last, we design and perform extensive numerical experiments to study the efficiency of the solution algorithm as well as the impact of different coordination strategies in delivery to nanostores with different inventory policies.

Keywords: Nanostores; Coordinated delivery; Inventory routing problem; Periodic vehicle routing problem; Integer programming based heuristic

References

Alonso, F., Alvarez, M. J., Beasley, J. E., 2008. A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. Journal of the Operational Research Society 59 (7), 963–976.

Archetti, C., Bertazzi, L., Hertz, A., Speranza, M. G., 2012. A hybrid heuristic for an inventory routing problem. INFORMS Journal on Computing 24 (1), 101–116.

Archetti, C., Bertazzi, L., Laporte, G., Speranza, M. G., 2007. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. Transportation Science 41 (3), 382–391.

Baldacci, R., Bartolini, E., Mingozzi, A., Valletta, A., 2011. An exact algorithm for the period routing problem. Operations Research 59 (1), 228–241.

Baldacci, R., Hadjiconstantinou, E., Mingozzi, A., 2004. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. Operations Research 52 (5), 723–738.

Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., Mack, R. G., Prutzman, P. J., 1983. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. Interfaces 13 (6), 4–23.

Beltrami, E. J., Bodin, L. D., 1974. Networks and vehicle routing for municipal waste collection. Networks 4 (1), 65–94.

Bertazzi, L., Speranza, M. G., 2013. Inventory routing problems with multiple customers. EURO Journal on Transportation and Logistics 2 (3), 255–275.

Blanco, E. E., Fransoo, J. C., 2013. Reaching 50 million nanostores: retail distribution in emerging megacities. Working Paper WP 404, Eindhoven University of Technology.

Campbell, A. M., Savelsbergh, M. W. P., 2004. A decomposition approach for the inventory-routing problem. Transportation Science 38 (4), 488–502.

Campbell, A. M., Wilson, J. H., 2014. Forty years of periodic vehicle routing. Networks 63 (1), 2–15.

Coelho, L. C., Cordeau, J., Laporte, G., 2014. Thirty years of inventory routing. Transportation Science 48 (1), 1–19.

Coelho, L. C., Laporte, G., 2013. A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. International Journal of Production Research 51 (23–24), 7156–7169.

Cordeau, J., Gendreau, M., Laporte, G., 1997. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30 (2), 105–119.

Gulczynski, D., Golden, B., Wasil, E., 2011. The period vehicle routing problem: new heuristics and real-world variants. Transportation Research Part E: Logistics and Transportation Review 47 (5), 648–668.

Laganà, D., Longo, F., Santoro, F., 2015. Multi-product inventory-routing problem in the supermarket distribution industry. International Journal of Food Engineering 11 (6), 747–766.

Park, Y., Yoo, J., Park, H., 2016. A genetic algorithm for the vendor-managed inventory routing problem with lost sales. Expert Systems With Applications 53, 149–159.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., Rei, W., 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research 60 (3).

VEHICLE ROUTING MODELS & APPLICATIONS

SB3: VRP IN HEALTH AND FOOD DELIVERY

Saturday 11:15 – 12:45 PM

Session Chair: Ann Campbell

11:15 **An ALNS For a Rich Home Health Care Routing and Scheduling Problem**
Florian Grenouilleau, Antoine Legrain, Nadia Lahrichi, Louis-Martin Rousseau*
CIRRELT-Montréal, Canada

11:45 **Team Orienteering with Uncertain Rewards and Service Times with an Application to Phlebotomist Intra-Hospital Routing**
[1]Huan Jin, [2]Barrett Thomas*
[1]Ningbo Supply Chain Innovation Institute Center, [2]University of Iowa

12:15 **The Restaurant Delivery Problem**
[1]Marlin Ulmer, [2]Barrett Thomas, [2]Ann Campbell, [2]Nicholas Woyak*
[1]Technische Universität Braunschweig, [2]Department of Management Sciences-University of Iowa

An ALNS For A Rich Home Health Care Routing And Scheduling Problem

Florian Grenouilleau[1], Antoine Legrain[1], Nadia Lahrichi[1], and Louis-Martin Rousseau[1]

[1]Ecole Polytechnique de Montréal, Montréal, Canada

1 Introduction

In Canada, as in many other developed countries, home health care services are expanding. Such care provides support or medical services to people in their own homes. It ranges from basic hygiene support to more complex tasks such as insulin injections or wound care. Home health care increases patients' comfort by allowing them to stay with their families, and it reduces general health-care costs by decreasing the length of hospital stays and the number of hospital beds needed. To ensure that their services are cost-effective, home health care organizations need to utilize their critical resources as efficiently as possible; this topic has been widely studied in recent years. Software companies such as Alayacare, our partner in Montreal, develop platforms to help these organizations plan their operations.

We investigate the home health care routing and scheduling problem (HHCRSP), which determines the assignment and routing of a set of home visits over a week. It can be described as a multi-depot periodic vehicle routing problem with time windows, consistency, and time-dependent travel times. Moreover, the home care context adds constraints focusing on the caregivers' skills and the patients' requirements (both mandatory and optional) as well as the caregivers' contracts. We use real business constraints provided by Alayacare and real data (for North American clients) to develop our algorithm.

Recently, [Fikar and Hirsch, 2016] presented a review of the HHCRSP. Metaheuristics have been developed for large problems (Bertels and Fahle [2005], Mankowska and Meisel [2014], Braekers et al. [2015]), but these studies solve the problem for a single day and do not integrate the periodic and consistency dimensions that stem from the patient–caregiver relationship. We propose a more comprehensive adaptive large neighborhood search (ALNS) for the HHCRSP. The contribution of this work is twofold: we tackle a rich planning problem that encompasses most of the real-world requirements of home health care, and we develop new ALNS operators, fitted for the problem.

2 Problem Definition

The HHCRSP can be described as a rich vehicle routing problem with skill requirements and workload balancing. The objective is to schedule a set of home health care visits in a defined period (a week in our context). For each visit we must determine on which day the patient will be visited, by which caregiver, and at what time while respecting the constraints and maximizing the satisfaction of the patients' and caregivers' preferences.

Home health care agencies must carefully assign their caregivers while taking into account the mandatory and optional expertise required for each patient, the skills of the caregivers, and continuity of care. Continuity of care is a measure of the strength of the relationship in each patient–caregiver pair; it is modeled by a dynamic score. A high score indicates that this (patient, caregiver) pair occurs frequently. Simultaneously with the caregiver assignments, agencies must build routes that take into account patient and caregiver availability (indicated by time windows; the travel times consider the time-dependent aspect) and the skill

match. In addition, they must consider the duration of each caregiver's day (and week). Caregivers with too little or too much work may not have job satisfaction.

3 Solution Method

The ALNS [Pisinger and Ropke, 2004] is an extension of the LNS [Shaw and Ilog, 1998] that uses the *ruin and recreate* principle of Schrimpf et al. [2000]. This metaheuristic iteratively destroys part of the current solution and then repairs it in a different way to improve its quality. A full description can be found in [Gendreau and Potvin, 2011].

In our algorithm, the initial solution is found using a constructive greedy heuristic that tries to assign all the visits over the week using a lowest-cost-insertion approach. Then, at each iteration, we choose destroy and repair operators by wheel selection. The destroy operator removes q visits from the current solution, where q is randomly chosen in an interval. We apply five destroy operators. We use the well-known $ShawRemoval$,$WorstRemoval$ and $RandomRemoval$ defined by Pisinger and Ropke [2004]. In addition, we introduce the $ServiceRemoval$ operator that randomly removes all the scheduled visits of a subset of the patients and the $FlexibleAvailRemoval$ operator that deletes from the current schedule the patients with the most availability (i.e., highest value of $\frac{NumberOfAvailableDays}{NumberOfVisitsToSchedule}$).

We use six repair operators including the classical $Greedy\ Heuristic$, $regret$-2 and $regret$-3 from Pisinger and Ropke [2004]. In addition, we introduce the $RandomService$ operator that randomly chooses a patient and schedules all the visits for that person. The remaining operators sort the unscheduled visits and then call a greedy allocation method. In the $PercentUnscheduled$ operator the sorting criterion is the percentage of visits currently scheduled for each patient and in the $PossibleRoutesWorkTime$ operator, the criterion is the level of congestion of the possible routes.

We then analyze the new schedule to determine if it improves the current or best solution. We also use a simulated annealing mechanism [Pisinger and Ropke, 2004] as an acceptance criterion. We use an adaptive mechanism to update the score of the operators chosen for this iteration according to the quality of the new solution. These scores have an impact on the probabilities of the wheel selection. Finally, we apply two termination criteria: a maximum number of iterations and a maximum computational time.

The ALNS is decomposed into segments of N iterations (where $N = 2000$ in our context). At the end of each segment, we reset the scores of the operators using a uniformly distributed wheel selection to restart the search. Moreover, we use a local search procedure for an intensification phase on the best solution found. The local search sequentially searches for improving relocations and then focuses on part of the problem by launching the ALNS on a reduced number of days and routes.

We compute the time-dependent travel time via the algorithm described by Ichoua et al. [2003]. This procedure respects the FIFO logic and computes the travel time according to the start and end locations and departure time. Moreover, we use the $forward\ time\ slack$ principle introduced by Savelsbergh [1992] to speed up the insertion tests made by the repair operators.

4 Computational Results

The algorithm is implemented in C++, and the tests were carried out on a Macbook Pro with a 2.5-GHz CPU and 16 GB of memory. The process was terminated after 600 seconds or 80000 ALNS iterations. We compared the classical routing operators and the new ones introduced for the home care context on a set of generated instances. We solved real-world instances to validate the algorithm. According to Alayacare, the priorities are travel time and continuity of care, so our results focus on these indicators.

4.1 Generated Instances

The goal here is to observe the impact of the created operators on the solutions. To do so, we define 4 operators classes :

- CL : The classical operators
- SE : ServiceRemoval + RandomService operators
- FL : FlexibleAvailRemoval + PercentUnscheduled operators
- WT : PossibleRoutesWorkTime operator

We test the combinations of these operators on 2 sets $P\_V\_C\_R$ of 5 instances where P is the number of patients, V the number of visits, C the number of caregivers, and R the number of routes (number of work days).

	40_100_6_20		100_250_12_50		
Scenarios	*Mean cost*	*Gap*	*Mean cost*	*Gap*	*Mean gap*
CL	229923.08	0%	308738.43	0%	0%
CL + SE	210530.51	-8.43%	295868.13	-4.17%	-6.30%
CL + FL	211552.13	-7.99%	282117.42	-8.62%	-8.31%
CL + WT	220996.31	-3.88 %	283214.77	-8.27 %	-6.07 %
CL + SE + FL	201685.98	-12.28 %	285484.38	-7.53 %	-9.91 %
CL + SE + WT	211846.03	-7.86 %	284709.74	-7.78 %	-7.82 %
CL + FL + WT	201957.96	-12.16 %	285357.29	-7.57 %	-9.87 %
CL + SE + FL + WT	192496.89	-16.28 %	284722.76	-7.78 %	**-12.03 %**
SE + FL + WT	192428.43	-16.31 %	290022.43	-6.06 %	-11.18 %

Table 1: Comparison of the operators on generated instances

A comparison of the operators is presented in Table 1. For each scenario and each set of instances, *Mean cost* represents the mean of the best solutions' costs, *Gap* represents the gap between the scenario and the 'CL' one according to the *Mean cost* value. *Mean gap* gives the mean of the gaps.

According to the Table 1, it appears that each category of created operators, associated with the classical ones, permits to improve the best solutions. Despite the fact that the scenario only using the created operators (FL + RS + WT) permits to reduce by 11.18% the cost of the best solutions, the best combination seems to be the one using all the operators, which reduces by 12.03% the mean of the costs.

4.2 Real-World Instances

	Current solution		**ALNS solution**		**Δ**	
Instance	*TT*	*CC*	*TT*	*CC*	*TT*	*CC*
149_325_11_40	3516.16	60%	**2220.62**	**60.90%**	-36.68%	+0.90%
137_340_11_40	3580.28	62.33%	**2489.19**	**66.22%**	-30.47%	+3.89%
145_311_11_35	3051.96	71.69%	**2103.80**	**81.06%**	-31.07%	+9.37%
146_324_11_40	3034.67	64.45%	**2173.90**	**76.26%**	-28.36%	+11.81%
Mean	3295.77	64.62%	2151.6	71,11%	**-31.69%**	**+6.49%**

Table 2: Comparison of the current and ALNS solutions on real instances

A comparison of Alayacare's current client solutions and ALNS's solutions on 4 real instances is presented in Table 2. According to Alayacare's advices, we focus here on two major indicators, the total travel time (TT) and the continuity of care (CC, i.e., the percentage of visits for which the assigned caregiver visited the patient in previous weeks).

In each case, ALNS improves the solution, in terms of both total travel time and continuity of care. On average, the ALNS reduces the total travel time by 31.69% and improves the continuity of care by 6.49%.

5 Conclusion

This project considers the HHCRSP in practical settings, taking into account all the industrial constraints. We have developed a specialized version of the ALNS for this problem and introduced 5 efficient operators (2 destroy and 3 repair). This approach improves Alayacare's current client solutions as measured by two major indicators: total travel time and continuity of care. The ALNS reduces the total travel time by more than 31% and increases the continuity of care by more than 6%. The algorithm will be included in our partner's software in the near future.

References

S. Bertels and T. Fahle. A hybrid setup for a hybrid scenario : combining heuristics for the home health care problem . *Computers & Operations Research*, 33:2866–2890, 2005.

K. Braekers, R. F. Hartl, S. N. Parragh, and F. Tricoire. A bi-objective home care scheduling problem : Analyzing the trade-off between costs and client inconvenience. *European Journal of Operational Research*, 248(2):428–443, 2015.

C. Fikar and P. Hirsch. Computers & Operations Research Home health care routing and scheduling : A review. *Computers and Operation Research*, 77:86–95, 2016.

M. Gendreau and J.-Y. Potvin. *International Series in Operations Research & Management Science Series*, volume 157. 2011.

S. Ichoua, M. Gendreau, and J. Y. Potvin. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379–396, 2003.

D. S. Mankowska and F. Meisel. The home health care routing and scheduling problem with interdependent services. *Home Care Management Science*, 17:15–30, 2014.

D. Pisinger and S. Ropke. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 40:455–472, 2004.

M. W. P. Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA J. Comput.*, 4:146–154, 1992.

G. Schrimpf, J. Schneider, H. Stamm-wilbrandt, and G. Dueck. Record Breaking Optimization Results Using the Ruin and Recreate Principle. *Journal of Computational Physics*, 159:139–171, 2000.

P. Shaw and S. A. Ilog. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. *Principles and Practice of Constraint Programming CP'98*, pages 417–431, 1998.

Team Orienteering with Uncertain Rewards and Service Times with an Application to Phlebotomist Intra-Hospital Routing

Huan Jin

Ningbo Supply China Innovation Institute Center

Zhejiang, China, 315000

Barrett Thomas

Department of Management Sciences

Tippie College of Business, University of Iowa,

Iowa City, IA, USA, 52242

January 2017

1 Abstract

Research shows that laboratory performance affects approximately 60% to 70% of the most critical medical decisions related to admission, discharge, and the medication of inpatients (Da Rin, 2009). Phlebotomists are a key part of these laboratory services. Phlebotomists primarily draw blood, urine,

and other samples from patients and are in fact often the patients' only contact with medical laboratories. With an increasing demand for healthcare services, the demand for phlebotomists is expected to grow 25% between 2014 and 2024 (Bureau of Labor Statistics, U.S. Department of Labor, 2016). With the demand for healthcare workers at an all-time high (Mattice, 2016) and healthcare dominating the American Staffing Associations Skills Gap Index, a measure of the hardest to fill jobs (American Staffing Association, 2016), it seems unlikely that the demand for phlebotomists can be met. Thus, increasing the efficiency of phlebotomists will be crucial to avoid delays in the admissions, discharge, and medication of patients.

With this need in mind, this study focuses on the intra-hospital routing of phlebotomists. This study is specifically motivated by the Department of Pathology at the University of Iowa Hospitals and Clinics (UIHC). At UIHC, there is a team of phlebotomists that works in the morning and another team that works later in the day. We focus on the morning shift as demand for these phlebotomists is often greater than capacity. Typically, the morning shift of phlebotomists works from 05:30am to 09:30am and serves 27 patient units located in different buildings across UIHC. This team of phlebotomists performs approximately 40% of the daily draws at UIHC.

When the phlebotomists arrive at work at 05:30, they can see the current set of outstanding orders. These orders are certain and called "pre-orders." Additional orders that arrive randomly between 05:30 and 09:30 am are called "add-ons." As a result, the service time and the number of orders to be served at a unit is random. The add-ons associated with a unit arrive before a phlebotomist reaches a unit and while a until is being serviced. The latter

case can be modeled as a queueing process because add-ons continue to arrive even when a phlebotomist is in a unit serving patients. A phlebotomist cannot leave a unit until all pre-orders and all add-ons, even those arriving once service has begun, are served.

In this work, we seek to route the team of phlebotomists for a given day. The objective is to maximize the expected number of orders served by the morning shift. Because it may not be possible to serve all the demand by the 9:30am end of the morning shift, we view the problem as an orienteering problem. Because the rewards and service times are random, the intra-hospital routing problem studied can be considered a variant of the team orienteering problem with stochastic rewards and service times (TOPSRST). While motivated by phlebotomist intra-hospital routing, the problem described in this study also has applications in airport screening and ticket inspection for some local rail operations (Thorlacius et al., 2010; Yan et al., 2016).

To solve the problem, we propose an a priori routing approach or an a priori policy. A priori policies are characterized by a priori routes or predetermined sequences of locations. In this research, an a priori policy requires phlebotomists to visit patient units in the order specified by a set of predefined routes. A priori policies handle uncertainty using what are called recourse rules. In this problem, we require a recourse to handle the possibility that we cannot serve all units by the end of the shift. In that case, our recourse policy simply sends the phlebotomist back to the origin, earning reward for just those patients served up to the end of the shift.

We also consider the case in which multiple phlebotomists are allowed to serve a unit at the same time. A unit is said to be "swarmed" when a group of

phlebotomists is scheduled to serve this unit. The advantage of swarming is that it allows the phlebotomists to serve orders at a unit as fast as possible, avoiding receiving and serving too many add-ons when phlebotomists are present. UIHC currently uses this practice to ensure that some larger units are served without sacrificing visits to smaller units.

To generate our a priori tours, we propose a variable neighborhood search (VNS). Because of the queueing aspects of the problem, we cannot exactly evaluate the value of a solution. However, we derive and demonstrate an iterative monte-carlo sampling approach to provide estimates of the value. We embed this sampling in our VNS.

This research makes the following contributions to the literature. First, this paper is the first that formulate and solve a team orienteering problem with stochastic rewards and service times, particularly one in which the rewards and service times are governed by a queueing process. Second, we propose an a priori solution approach and derive an expression of the objective that leads to an iterative a priori sampling scheme that can be used to estimate the value of a complicated objective function. Finally, using data from UIHC, we demonstrate that the proposed approach can be more effective than the approach currently used in practice.

References

American Staffing Association. Asa skills gap index, 2016. URL `https://americanstaffing.net/staffing-research-data/asa-staffing-industry-data/asa-skills-gap-index/`. [Online;

accessed 24-June-2016].

Bureau of Labor Statistics, U.S. Department of Labor. Phlebotomists. In *Occupational Outlook Handbook, 2016-2017*. Available from `http://www.bls.gov/ooh/healthcare/phlebotomists.htm`, accessed on June 23, 2016, 2016.

Giorgio Da Rin. Pre-analytical workstations: a tool for reducing laboratory errors. *Clinica Chimica Acta*, 404(1):68–74, June 2009. ISSN 1873-3492. doi: 10.1016/j.cca.2009.03.024. URL `http://www.ncbi.nlm.nih.gov/pubmed/19302988`.

Mattice. Healthcare job demand reaches all-time high, 2016. URL `http://www.beckershospitalreview.com/human-capital-and-risk/healthcare-job-demand-reaches-all-time-high.html`. [Online; accessed 24-June-2016].

Per Thorlacius, Jens Clausen, and Kalvebod Brygge. Scheduling of inspectors for ticket spot checking in urban rail transportation. *Proceedings of Trafikdage på Aalborg Universitet 2010*, available from `http://www.trafikdage.dk/td/papers/papers08/per_thorlacius_196.pdf`, accessed on October 1, 2016, August 2010.

Zhenzhen Yan, Sarah Yini Gao, and Chung Piaw Teo. On the design of sparse but efficient structures in operations. Submitted for publication, 2016.

The Restaurant Delivery Problem

Marlin Ulmer
Technische Universität Braunschweig, Braunschweig, Germany
Email: m.ulmer@tu-braunschweig.de

Barrett W. Thomas, Ann M. Campbell, Nicholas R. Woyak
University of Iowa, Iowa City, IA, USA

1 Motivation

People like to eat at home, but do not always like to prepare it. In a recent survey, 68% of survey respondents reported ordering takeout at least once a month, with 33% ordering at least once a week (Statista Survey, 2016). Seeking to satisfy this demand, recent years have a seen a surge in companies that deliver restaurant meals to customers. These companies include startups like Grubhub, OrderUp!, and UberEats. These companies offer customers the chance to place an order via a mobile application or internet website and choose from the full menu of dozens of restaurants. The delivery is typically promised between 30 to 40 minutes after the customer's call and costs between $4 and $7 per delivery (Macmillan, 2016). From May 2010 to May 2015, online orders in the US grew from approximately 400 million annually to over 900 million annually (NPD Group, 2015). Given this growth, it is estimated that application-based delivery services will represent a multi-billion-dollar market in the coming years (Isaac, 2016).

While ordering from a delivery startup app is simple and convenient for consumers, the companies providing these services face significant operational challenges. Customers want a reliable and fast service, cooperating restaurants want their product served fresh, and drivers want to serve enough orders to make a decent wage. Because late deliveries can make customers dissatisfied and not only less likely to use the delivery service again, delivery companies need to assign drivers to orders with the goal of avoiding delay (Maze, 2016).

With these operational challenges in mind, we introduce the Restaurant Delivery Problem (RDP). The RDP is characterized by a fleet of delivery vehicles that serve dynamic customer requests over the course of a day. The temporal distribution of customer requests is known, but no assumption is made on the knowledge of customer locations. Customers make their requests (e.g an order for food) through a mobile application or website choosing from a number of known providers (e.g. restaurants) throughout their area. Once a request is placed, the order is immediately relayed to the provider who assigns it to a driver who will pickup and deliver the order. The assignments of orders to drivers do not need to be immediate and it is possible for a driver to be assigned multiple outstanding orders at one time (bundling). Before delivery of an order, the driver must pickup the order from the appropriate restaurant. However,

the time to prepare a customer's food at each restaurant is random. While the delivery company knows a distribution on the time, and perhaps even a time-dependent distribution, the driver does not know exactly when the order will be ready. Thus, the driver may need to wait for the order's completion when arriving to a restaurant. Once the order has been retrieved, the driver delivers it to the customer who expects her/his order to be delivered by a certain deadline. The objective is to minimize the expected sum of positive differences between delivery times and delivery deadlines of orders over the service day.

To the best of our knowledge, this work is the first to address the RDP. In our talk, we will introduce a dynamic policy for making assignments of orders to drivers. The policy considers both the postponement of the assignment of an order to a driver and integrates a cost function approximation (CFA) to heuristically account for uncertainty in both ready times and customer orders. We will conclude the talk by presenting the results of a set of comprehensive computational tests that demonstrate the value of the proposed solution approach.

2 Solution Approach

The RDP can be modeled as a Markov decision process (MDP). For the sake of space, we omit the details of the model from this abstract. However, we note that at each decision point, we must determine what orders to assign to what vehicles and how to route them as well as what orders to postpone. This decision has both a subset selection component, as in an orienteering problem, as well as an assignment and routing component as in a vehicle routing problem.

A solution for the RDP is a policy $\pi \in \Pi$ assigning an action to each state. The optimal policy for the RDP minimizes the expected costs $c(X_k^\pi(S_k))$ of choosing an action X_k^π when in state S_k and can be expressed as

$$\pi^* = \arg\min_{\pi \in \Pi} \mathbb{E}\left[\sum_{k=0}^{K} c(X_k^\pi(S_k))|S_0\right]. \tag{1}$$

Because of the curse of dimensionality, we are not able to solve for the optimal policy. Instead, we solve Equation (1) approximately using a two-step procedure. First, we use heuristic solution methods to significantly restrict the set of actions and therefore operate on a set of restricted policies $\bar{\Pi} \subset \Pi$. Second, we use simulation of the information space to estimate Equation (1) for each policy. In the following, we describe how we design the subset of policies $\bar{\Pi}$ and how we conduct the simulation.

2.1 Restricted Policies

In the development of suitable policies for the RDP, we experience three major challenges. First, because decisions need to be made in real-time, the time available for calculation is limited. Second, because customer requests are random, myopic assignments may negatively impact the ability to serve later requests. Third, the RDP contains an additional uncertainty in the ready times. Because the ready-time distributions are usually long tailed, cost calculation based on mean values may lead to significant delay for some customers. In the following, we describe how we address these three challenges and how we use

simulation to determine an overall policy.

Real-Time Decision Making: Decision Space Reduction

We accommodate fast decision making by reducing the action space through the use of an insertion routing heuristic coupled with an assignment strategy based on a reduced subset selection. To implement this, we maintain and update a set of planned routes $\Theta = (\theta^1, \ldots, \theta^m)$ throughout the horizon. At each decision point, the routing heuristic inserts a set of open orders as follows. First, to insert a customer, the routing heuristic determines the vehicle into whose route the order (and the restaurant) can be integrated with the smallest increase in "myopic" cost. The cost $\mathcal{C}$ of inserting a customer in route θ is calculated myopically. The arrival time a is calculated based on mean ready times and assuming no future requests. Given the deadline d, the approximate increase in delay of a tour is

$$\mathcal{C}(\theta) = \sum_{i:\theta_i \text{ is Customer}} \max\{0, a(\theta_i) - d(\theta_i)\}. \tag{2}$$

To account for sets of open orders, the routing heuristic generates a set of all potential sequences of these open orders. For each ordered sequence, the heuristic subsequently applies the aforementioned insertion method for each order. The heuristic selects that sequence minimizing the (myopic) sum of delay for all vehicles.

Stochastic Requests: Postponements

In some cases, postponing assignments may lead to better decisions. Our postponement decisions are determined by three indicators. First, we postpone requests not impacted by the immediate action of our routing decision. This means that we do not postpone orders for which a vehicle is assigned to visit the corresponding restaurant as the vehicle's next stop. Second, to avoid multiple postponements of an order, leading to a potentially large increase in computation time associated with the routing and subset selection,, we limit the maximum amount of time an order is postponed. Third, because postponements increase the number of open orders, we limit the overall number of postponed orders. The values for both of these limits are set based on preliminary experiments.

Stochastic Ready Times: Temporal Buffers

The major challenge for the RDP is uncertainty in ready times. With the use of mean ready times, the described routing and assignment strategy can result in many delays. Even though Equation (2) may indicate no cost for a decision, the realization of ready times can lead to a delay. To address this, we implement a cost-function approximation (CFA, Powell 2014) . The general idea of a CFA is to estimate potential costs based on uncertainty by adding parameters in the cost calculation. To account for this, we add a time buffer λ to the myopically calculated delay. To implement the CFA, we replace Equation (2) with:

$$\mathcal{C}^{\lambda}(\theta) = \sum_{i:\theta_i \text{ is Customer}} \max\{0, a(\theta_i) - (d(\theta_i) + \lambda)\}.$$

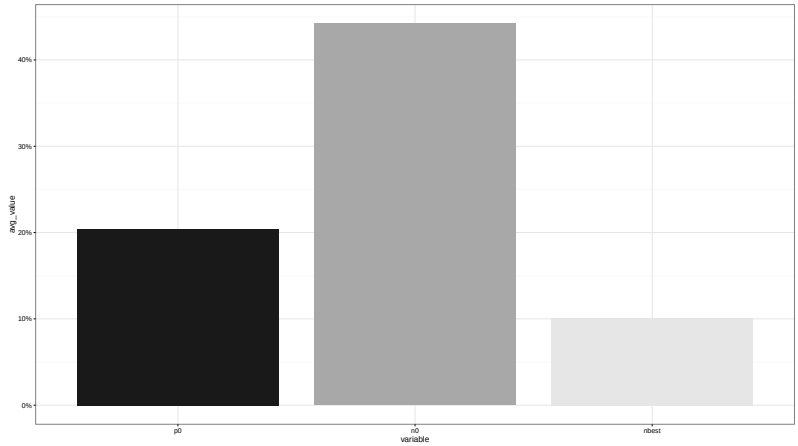


Figure 1: Comparison of Benchmarks to Proposed Approach

A buffer of zero results in Equation (2).

2.2 Policy Search via Simulation

The steps discussed in the previous section define a subset of policies $\bar{\Pi} = \{\pi^\lambda : 0 \leq \lambda \leq t_{\text{limit}}\} \subset \Pi$, where t_{limit} is the maximum amount of time between when an order is made and when it should be delivered. Because of the different number of expected orders and ready time distributions, λ may vary for different instance settings. Thus, we individually determine λ for each instance setting. We are not able to determine the exact expected value for each policy and thus use simulation. To this end, we simulate 1000 trajectories for each policy π^λ and select the policy minimizing the sum of delay over all simulation runs.

3 Results

We base our experiments on a delivery service based in Iowa City with 110 restaurants, 15 vehicles, and more than 30,000 possible customer locations. We consider 42 different instance settings derived from varying the coefficient of variation (COV) of the distributions of the ready times at restaurants, the arrival rate of customers requests, and whether or not the ready times at restaurants are homogeneous or heterogeneous. Once we have determined our parameters for an instance setting, we test the quality of our approach by running 1000 trials. We compare the results of our approach with those of several benchmarks.

Figure 1 presents the percentage difference between three benchmarks, $n0$, $p0$, and *nbest*, and the solution method described in the previous section, using both postponement and the approximate cost function. The $n0$ benchmark sets $\lambda = 0$ and does not allow postponement. By setting $\lambda = 0$, this benchmark policy does not approximate the cost function. The $p0$ benchmark sets $\lambda = 0$, but allows postponement. Again, this benchmark does not approximate the cost. Finally, *nbest* does not allow postponement, but sets λ to the best found value for each of the 42 instance settings.

The results show that both postponing and approximating the cost can have dramatic improvements

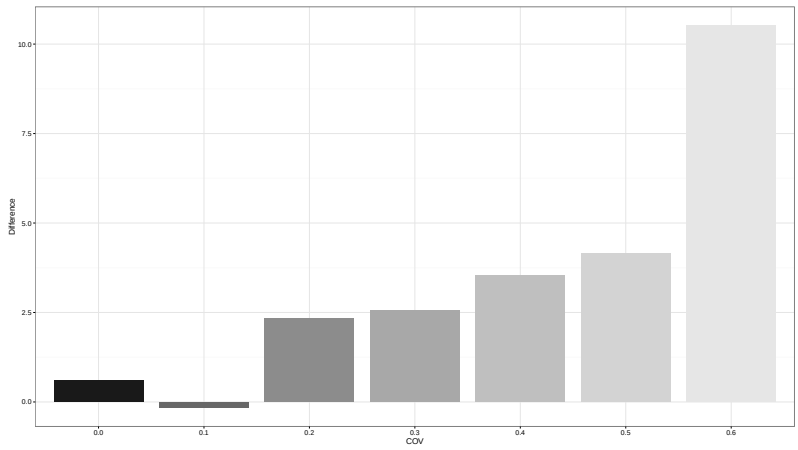


Figure 2: Difference in Sum of Delay between $p0$ and Proposed Approach by COV for 240 Customers

on solution quality. The results show that doing neither, $n0$, leads to solutions that are over 40% worse on average. Only allowing postponement, $p0$, leads to solutions that are over 20% worse on average. The best benchmark, *nbest*, is still 10% worse on average.

It is interesting to examine the impact of increasing variability on the solutions. Not surprisingly, the sum of the delays for an instance increases as the COV increases regardless of what policy we choose. However, the proposed CFA also has increasing value as the COV increases. Figure 2 presents the difference in sum delay between $p0$ and the proposed approach by COV for instances with 240 expected customers. The figure shows the increasing difference in the expected sum of delay. This increasing difference demonstrates the danger of planning with the mean, even if using postponement, in a highly variable and dynamic environment.

In our talk, we will also present results that the describe the proposed policy's performance with regard to the worst case delay, the delivery time, the freshness of the food upon delivery, and driver equity.

References

NPD Group. Food delivery orders made in the US, 2015. URL `\url{https://www.theatlas.com/charts/EJANqMTO}`. [Online; accessed 6-December-2016].

Mike Isaac. Delivery Start-Ups Face Road Bumps in Quest to Capture Untapped Market, February 11, 2016.

Douglas Macmillan. Uber Prepares Meal-Delivery Service, January 20, 2016. URL `http://www.wsj.com/articles/uber-to-ring-the-dinner-bell-in-10-u-s-cities-1453324399`.

Jonathon Maze. Not everything delivers: Saying no to delivery, April 28, 2016. URL `http://nrn.com/operations/not-everything-delivers-saying-no-delivery`. [Online; accessed 7-December-2016].

Warren B. Powell. *Clearing the Jungle of Stochastic Optimization*, chapter 5, pages 109–137. INFORMS, 2014. doi: 10.1287/educ.2014.0128. URL `http://pubsonline.informs.org/doi/abs/10.1287/educ.2014.0128`.

Statista Survey. How often do you order food for takeout?, November 2016. URL `\url{https://www.statista.com/statistics/319662/frequency-of-ordering-food-for-takeout-or-delivery-us/}`. [Online; accessed 6-December-2016].