Newsletter
of the INFORMS Computing Society
Volume 24, Number 1 Spring 2003

Contents

ICS Meeting News Flash!

Bruce Golden, S. Raghavan and Ed Wasil have agreed to chair the bi-annual ICS meeting, which will be held in Annapolis Maryland during the first week of January 2005. The dream-team co-chairs have tentatively scheduled the conference for January 5 through 7 (Wednesday through Friday) at Loew's Annapolis. January 2005 is sooner than you think! Put it on your calendar. It will be a great meeting.

Introduction to Extensible Markup Language (XML) with Operations Research Examples

Gordon Bradley
Naval Postgraduate School

Operations research applications are driven by data and increasingly that data is available in xml documents (and over the web). I like to characterize operations research as the field that uses models to turn data into information. Because xml has emerged as the de facto standard for sharing data among applications, we should look to using xml in our applications and research. The power tools to read, write, access, process, create, validate, and transform xml documents should be as much of a part of our computational toolbox as are spreadsheets and data analysis programs.

This short introduction will discuss the uses and benefits of xml, the construction of xml-based languages, the validation of xml documents, the use of software tools to construct, validate, process, and transform xml documents, and software API's to access, manipulate, and transform data from xml documents within computer programs. Included is a brief discussion of ongoing uses of xml in operations research.

Extensible Markup Language and Related Technologies

The Extensible Markup Language (xml) and the set of related technologies are platform and language independent open standards to represent, structure, validate, process, archive, transform, and present data. The driving motivation behind xml is

XML: continued on page 8

ICS Officers

Chair:

David L. Woodruff (dlwoodruff@ucdavis.edu)
Graduate School of Management
University of California-Davis
Davis, CA 95616
(530) 752-0515

Vice-Chair/Chair-Elect:

Ariela Sofer (asofer@gmu.edu)
Professor and Chair SEOR Dept., MS4A6
George Mason University
4400 University Drive
Fairfax, VA 22030
(703) 993-1692

Secretary/Treasurer:

John W. Chinneck (chinneck@sce.carleton.ca)
Systems and Computer Engineering
Carleton University
Ottawa, Ontario K1S5B6 CANADA
(613) 520-5733

Board of Directors

Karen Aardal (karen.aardal@isye.gatech.edu)
School of Industrial and Systems Engineering
Georgia Institute of Technology
765 Ferst Drive
Atlanta, GA 30332-0205
(404) 385-0770

Hamant Bhargava (bhargava@computer.org)
The Smeal College of Business
Penn State University, University Park, PA 16802
(814) 865-6253; term expires in 2003

Brian Borchers (borchers@nmt.edu)
Department of Mathematics
New Mexico Tech, Socorro, NM 87801
(505) 835-5813; term expires in 2004

Robert Fourer (4er@iems.nwu.edu)
Deptartment of Industrial Engineering
& Management Sciences
Northwestern University
2145 Sheridan Road, Room C234
Evanston, IL 60208-3119
(847) 491-3151; term expires in 2004

Robin Lougee-Heimer (robinlh@us.ibm.com)
IBM TJ Watson Research Center, Math Sciences Dept.
PO Box 218
Yorktown Heights, NY 10598

Sanjay Saigal (saigal@ilog.com)
ILOG Inc.
1005 Terminal Way, Suite 100
Reno, NV 89502
(775) 332-7600; term expires in 2003

Newsletter Editors

Acting Editor-in-Chief

David L. Woodruff (dlwoodruff@ucdavis.edu)
Graduate School of Management
University of California-Davis
Davis, CA 95616
(530) 752-0515

Managing Editor

Benjamin Y. Lou (bylou@ucdavis.edu)
c/o Professor Woodruff
Graduate School of Management
University of California-Davis
Davis, CA 95616

The ICS Newsletter is published semiannually by the INFORMS Computing Society (ICS). Manuscripts, news articles, advertisements and correspondence should be addressed to the Editors. Manuscripts submitted for publication will be reviewed and should be received by the Editor three months prior to the publication date. Requests for ICS membership information, orders for back issues of this Newsletter, and address changes should be addressed to:

**INFORMS Computing Society Business Office
901 Elkridge Landing Road, Suite 400
Linthicum, MD 21090-2909
phone: (410) 850-0300**

Letter From the Editor

By David L. Woodruff
ICS Chair and Acting Newsletter Editor
http://faculty.gsm.ucdavis.edu/~dlw/

Technically, spring lasts until June 21. I mention this because, technically, we are on time with the spring newsletter. The change from a printed to an electronic newsletter was pretty easy, really. Last time we held the presses in hopes of making an announcement concerning the bi-annual ICS meeting. In this issue we are delighted to announce that Bruce Golden, Ed Wasil and S. Raghavan are organizing the meeting in Annapolis in January of 2005. That seems like a long way away, but it is not! Mark your calendar and you'll see; the meeting is tentatively scheduled for Wednesday, January 5 through Friday the seventh.

We have a great feature piece in this newsletter on XML. Gordon Bradley has done a nice job of providing background information as well as citing emerging OR applications. The computing society remains at the forefront of the application of emerging computational technologies in Operations Research and Management Science. People like Karen Aardal have put us there. She is a director of the society and the subject of our member feature this month.

ICS track for the INFORMS meeting to be held in October in Atlanta is a full slate of diverse and interesting sessions. Kudos to Ariela Sofer for putting it together! Her term and vice-chair and chair-elect will end at the Atlanta meeting when she takes over as chair of the society.

MProbe Version 4.0 Released

MProbe is a general tool for analyzing the properties of mathematical programs, e.g. the convexity and effectiveness of constraints, finding near-feasible points etc. Version 4.0 is now available for download at http://www.sce.carleton.ca/faculty/chinneck/mprobe.html

New features include:

- a new Points Workshop which permits exchange of points with external programs such as solvers, analysis of points, etc.

- a method for finding approximately feasible points, even for nonconvex feasible regions

- the ability to read and manipulate MPS files directly

- a new method for shrinking variable bounds

- improved feasibility bootstrapping

Member Profile: Karen Aardal

Karen Aardal obtained her PhD from C.O.R.E, Universite' Catholique de Louvain, Belgium in 1992. Her main research interest is integer programming, and in particular algebraic approaches to integer programming. Additionally she is also interested in combinatorial optimization, in particular in facility location. Before joining Georgia Tech in October 2002 Karen held positions at the University of Essex, UK; and Erasmus University, Tilburg University and Utrecht University in the Netherlands.

In addition to serving on the Council of the Mathematical Programming Society, MPS,and currently the Publications Committee Chair of MPS, Karen also served on various other committees of MPS, such as the Symposium Advisory Committee and the Web Committee. She has been an INFORMS member since 1989, and is particularly interested in the activities of the Computing Society. In January 2003 she became a member of the Board of Directors of ICS.

She is an associate editor of Operations Research Letters, INFORMS Journal on Computing, and Mathematical Programming Series B, and was a previous editor of OPTIMA - The Newsletter of the Mathematical Programming Society.

What Karen feels is particularly attractive about ICS is that it provides a broad forum to exchange ideas on basically all aspects of computing related to OR. To have one such broad forum increases the understanding between areas such as computer science, OR, and mathematics and is clearly enhancing both the computational and theoretical sides of OR. Better theoretical insight leads to better algorithms and implementations, and good computational studies prompt new theoretical questions.

When Karen am not at work she loves spending time with her daughter Elisabeth who is 4 years old and she very recently had a second child, little Jacob. She used to drivea BMW motor bike, but right now sticks to more "responsible" hobbies such as photography and music. Karen is proud of her vintage Hasselblad camera and her three clarinets.

MathOptimizer Professional for Continuous Global and Convex Optimization

János D. Pintér , PCS Inc. and Dalhousie University
and
Frank J. Kampas, WAM Systems, Inc.

Global Optimization

The objective of global optimization (GO) is to find the best solution of nonlinear decision models that may have a multitude of global and local optima. GO has significant existing and potential applications in many fields of the sciences, engineering, econometrics, and finances. As of 2003, over a hundred books, many thousands of articles, and dozens of web sites are devoted to the subject: consult, e.g., Horst and Pardalos (1995) or Pardalos and Romeijn (2002).

LGO and MathOptimizer Professional

The LGO software serves to solve a broad range of nonlinear optimization models, making use of a robust and efficient suite of global and local scope solvers. The LGO software development is based on award-winning research (Pintér, 1996). LGO has been developed and maintained for over a decade: consult also e.g., Pintér (2001). The software has been peer-reviewed (Benson and Sun, 2000), and it is used by a growing clientele in education, research, and industry.

MathOptimizer Professional (Kampas and Pintér, 2003) combines the power of *Mathematica* (Wolfram, 1999) with the external LGO solver engine. This combination provides significantly enhanced modeling capabilities (supported by *Mathematica*), and a solver performance that is comparable to other compiler-based solver implementations.

Supported complier platforms for using MathOptimizer Professional currently include: Borland C/C++, Lahey Fortran 90/95, and Microsoft Visual C/C++. Various further personal computer and workstation platform implementations can be made available upon request. (*Mathematica* itself can be used across a range of hardware platforms and operation systems.)

The MathOptimizer Professional software product is developed and supported by Pintér Consulting Services, Inc., and by Dr. Frank J. Kampas. Please contact <jdpinter@hfx.eastlink.ca> and <fkampas@msn.com> for further information.

Acknowledgement

FJK and JDP wish to thank Dr. Mark Sofroniou (Wolfram Research, Inc.) for making availible his Format.m package used in the course of the MathOptimizer Professional development project.

References

Benson, H.P. and Sun, E. (2000) LGO - Versatile tool for global optimization. *ORMS Today* October 2001 Issue, pp. 52-55

Horst, R and Pardalos, P.M., Eds. (1995) *Handbook of Global Optimization, Vol. 1* Kluwer Academic Publishers, Dordrecth / Boston/ London.

Kampas, F.J. and Pintér, J.D. (2003) MathOptimizer Professional. Lecture presented at the *Mathematica Developer Conference*, Champaign, IL, April 10-12, 2003. (This presentation is available upon request, in *Mathematica* notebook format.)

Pardalos, P.M. and Romeijn, H.E., Eds. (2002) *Handbook of Global Optimization, Vol 2.* Kluwer Academic Publishers, Dordrecth / Boston / London.

Pintér, J.D. (1996) *Global Optimization in Action.* Kluwer Academic Publishers Dordrecth / Boston / London.

Pintér, J.D. (2001) *Computational Global Optimization in Nonlinear Systems.* Lionheart Publishing, Inc., Atlanta, GA, 2001

Wolfram, S. (1999) *The Mathematica Book.* (Fourth edition.) Wolfram Media, Champaign, IL, and Cambridge University Press, Cambridge.

"Handbook of Applied Optimization" named an AAP Outstanding Professional and Scholarly Title of 2002.

The Association of American Publishers' Professional/Scholarly Publishing (PSP) Division has singled out Handbook of Applied Optimization, co-edited by Panos M. Pardalos (Center for Applied Optimization, University of Florida, Gainesville) and Mauricio G. C. Resende (Internet and Network Systems Research Center, AT&T Labs Research, Florham Park) as an "Outstanding Professional and Scholarly Title of 2002" in the computer science category. The PSP Awards are given annually to acknowledge excellence in book, journal, and electronic publishing in all the disciplines represented by professional, scholarly, and reference publishing. Only two awards are presented in each category. Panos and Mauricio were invited to edit this handbook by Oxford University Press, Inc., which published the book.

The handbook provides a broad spectrum of advances in applied optimization with a focus on the algorithmic and computational aspect of the field. Leading experts in applied optimization contributed to the volume, which is geared towards engineers, scientists, operations researchers, and other applications specialists who are looking for the most appropriate and recent optimization tools to solve particular problems. The book is divided into three main parts: algorithms, applications, and software. In the algorithms section, the important algorithms in the major fields of optimization are described.

The section on applications is designed to provide the practitioner with a description of the relevant optimization issues in a number of specific application areas. The authors include overview articles discussing broad problem types, such as scheduling, vehicle routing, network design, bin packing, inventory management, traveling salesman, satisfiability, location and assignment problems, as well as some examples of applied optimization in specific areas. These applications cover a broad spectrum of fields, such as transportation, agriculture, manufacturing, aerospace, telecommunications, energy, biology, finance, and the environment.

The tools of applied optimization are described in the software section, with an emphasis on practical details: how to implement and test optimization algorithms, how to use existing optimization packages and the Internet, and how to use modeling languages to build optimization systems.

For more on Handbook of Applied Optimization,

visit: http://www.research.att.com/~mgcr/hao.html.

Special Issues of the INFORMS Journal on Computing coming up or in development:

— Special Issue on Mining Web-based Data for e-Business Applications, guest co-edited by Alex Tuzhilin and Louiqa Raschid, to appear as Vol. 15 No. 2 (Spring 2003); see http://joc.pubs.informs.org/WebDM.html and http://joc.pubs.informs.org/ForthcomingPapers.html for more information.

— Special Issue on Computational Molecular Biology/Bioinformatics, guest-edited by Harvey Greenberg; call for papers at http://joc.pubs.informs.org/CallSpecialIssueCompBio.html (submission deadline August 1, 2003).

— Special Issue on Operations Research in Electrical and Computer Engineering, guest-edited by John Chinneck; call for papers at http://joc.pubs.informs.org/CallSpecialIssueORinECE.html (submission deadline October 31, 2003).

Member Announcements

Donald Hill retired from Marathon Ashland Petroleum LLC in August, 2002. He and a partner have formed a consulting firm: GPSS Analysis LLC. He may

be contacted at: donhill3752@msn.com or
2620 Springmill Rd.
Findlay, Ohio 45840

CALL FOR PAPERS
Computers & Operations Research
Focused Issue on Operations Research and Data Mining

Guest Editor

Sigurdur Olafsson
Iowa State University
olafsson@iastate.edu

The field of data mining has seen an explosion of interest from both academia and industry and this interest continues to grow at a rapid rate. In this context, we use the term data mining to refer to all aspects of an automated or semi-automated process for extracting previously unknown and potentially useful knowledge and patterns from large databases. This process typically involves several steps ranging from data collection and integration to interpretation of the output, but we note in particular that the success of the process relies heavily on the availability of computer algorithms that can be effectively applied to large data sets to extract useful information.

The operations research community has recently made significant contributions in this area and in particular to the design and analysis of data mining algorithms. For example, mathematical programming formulations of support vector machines have been used for feature selection and data clustering. Metaheuristics and evolutionary methods have also been introduced to solve these and other data mining problems, and the opportunities for using both exact and heuristic optimization algorithms for data mining are extensive. This includes but is not limited to data mining problems such as feature and instance selection, classification, association rule discovery and data clustering, and OR methodologies such as mathematical programming, evolutionary methods, and metaheuristics.

However, the intersection of OR and data mining is not limited to algorithm design and data mining can play an important role in many OR applications. Applications in areas such as e-commerce, transportation and logistics, supply chain management, planning and scheduling, and inventory control often generate vast amounts of data and data mining can be used to extract structural information and insights from these datasets. An example of this that has been studied in the past is the use of data mining to learn when a particular dispatching rule is appropriate in production scheduling environments.

Finally, while the data mining process is usually effective for generating insights and patterns from large data sets, it is a model free approach and the insights are typically unstructured and require substantial interpretation. Thus, optimization methods can potentially be applied to the output of the data mining process to optimize the desired objective while accounting for relevant business constraints. Again, all of the methodologies or application areas mentioned above are relevant in this context as well.

This focused issue aims to broadly explore the synergy between the fields of operations research and data mining. We are therefore interested in receiving papers focusing on either applications or methodology and dealing with any aspect of the intersection between these two fields, including but not limited to the three perspectives described above.

The deadline for submission is December 31, 2003 but early submissions are encouraged. Prospective authors should send a postscript (.ps) or pdf formatted file containing their paper as an email attachment to the guest editor at olafsson@iastate.edu. Any questions regarding this focused issue should also be directed to the guest editor. All submitted papers will be peer reviewed according to the usual standards of a leading international journal.

xML: continued from page 1

to provide interoperability of different systems through the sharing of valid data. This has been a problem long before the introduction of computers, but the recent increase in the volume and variety of data, networks to quickly transfer information, the web to allow universal access to data across different computers and computer systems, and requirements for multiple views of data have made it a much more challenging issue.

It is difficult to grasp the enormity of the changes that have taken place since the introduction of the World Wide Web only ten years ago. The availability of large numbers of inexpensive, powerful computers connected in a world wide network was a more continuous, and thus more predictable, development over the past few decades. But the introduction of a small number of protocols (URI, HTTP, HTML) that enabled the web was a more sudden and disruptive development whose power and implications are yet to be fully seen and understood. One reason that it is difficult to grasp this impact is that the idea of easy interoperability of valid data was long active in the imagination of computer users. However, it is only in the last few years that that the technology to fully achieve this has been available. The initial standards for xml were only formally finished five years ago. While the web protocols provide the connectivity, it is the xml standards that have provided the open source standards for the universal interoperability of valid data.

Other approaches to representing computer data have often involved computer specific binary representations, propriety formats, and few, if any, open standards. This greatly impedes the interchange of information within an organization to say nothing of the movement of data among different organizations with different hardware and software systems. The cooperative development of license free, open standards has allowed the rapid development of data interoperability.

Relational databases and SQL are the de facto standards for data storage and data access; xml is the de facto standard for sharing data among applications. Thus relational databases and xml are complementary rather than competing data technologies. This distinction is not always clear because there are native xml databases, xml can be stored in relational databases, and database access can be used to exchange data. In spite of the overlay of the technologies, databases and xml technologies each have a distinct contribution. Xml technologies are often characterized as the glue between different systems and applications. While this is by no means the only use of xml, it is currently the most widespread use.

The general areas that have thus far seen most of the applications of xml are business to business (for example, supply chain), construction of books and reports, e-commerce, database access, scientific data transfer, and configuration files.

The language for web pages, HTML, is the markup language that most people are familiar with. For this reason, xml is often compared to HTML. This comparison is very misleading because xml is a meta language (that is, a set of rules) for creating xml-based languages each of which defines documents that are instances of the language. The confusion of comparing xml to HTML is made greater because HTML documents do not conform to xml rules and thus HTML is not an xml-based language and the technologies and tools for processing xml do not apply to HTML documents.

The term “language” is justified since each xml-based language has a specified syntax and grammar and we are able to employ a range of formal computer language tools. An xml-based language (also called a tag set, an xml vocabulary, a document type) defines a set of element and attribute names, a structure for including them in a

document, and constraints on the values of data in the document. There are many, many examples of these xml-based languages:

- MathML is a language for representing mathematical formulas (for searching, indexing, evaluation) and their presentation (for example, in web pages, books). MathML is supported in computer algebra systems such as Mathematica, Maple, and Mathcad as well as in mathematical typesetting systems such as Tex and Latex.
- Scalable Vector Graphics (SVG) is a language for describing two-dimensional graphics (for example, points, lines, and curves).
- Extensible Business Reporting Language (XBRL) is for describing company financial reports.
- Synchronized Multimedia Integration Language (SMIL) is for creating multimedia presentations.

Each xml-based language has a formal specification (called a schema) that defines the structure and content of documents in the language. An xml document is an instance of a particular xml-based language if it conforms to the schema that defines that language. Given a schema, there are validating parsers that can determine if a document adheres to the particular scheme (it may also add defaults and do other processing on the document).

The process that developed xml and the related technologies is very interesting and is perhaps the best demonstration of the power of license free software and open standards. In 1994 the World Wide Web Consortium (W3C) www.w3.org was formed to further the development and evolution of the Internet by improving HTTP and HTML and developing new standards to ensure interoperability. Any company or individual can join and participate in their ongoing construction of standards (called "recommendations"). The work that lead to the development of xml began with the effort to address the limitations of HTML. In particular it was seen that HTML focused on presentation of data rather than the content of the data. The basic goal of xml is to separate data content (held in xml) from presentation (held in HTML, XHTML, or other formats).

There has been a rush to define many xml-based languages to facilitate the easy interchange of data in specific domains. The characteristics of the domain often determine the type of organization that develops an xml-based language. Business oriented languages are developed by groups of software professionals from different companies working together through a non-profit organization such as W3C and The Organization for the Advancement of Structured Information Standards (OASIS) www.oasis-open.org while being paid by their company. Some business-oriented languages are developed by the employees of a single company that may then make it available license free, under license, or embedded in a commercial product. Languages for government use or developed by governments for widespread use are constructed by employees of the government entity. Volunteer software workers are typical for the standards developed for research domains and hobby domains. The goal of interoperability is achieved only if all the parties interested in a particular data domain use the same xml-based language. Thus the work of developing a successful xml-language is based on careful technical work, review and acceptance by potential users, and tradeoffs and compromises that assure that the language will be widely adopted and carefully adhered to. This has lead the interested parties to cooperate in the construction of languages they all then commit to. To locate xml work going on in a particular area, a web search is usually quite effective.

An xml-based language is defined by specifying and then publishing a schema for the language. With the schema anyone can construct xml documents based on the language and verify that they conform to the language (using a validating parser). Most of these languages are publicly available, open source, and license free. In addition most are controlled by an organization that has operating procedures for making changes that

assures users of the language that they have protection for the investment they made to put the data into the particular language.

The Basic Ideas

The process of adding meaning to symbols is a challenge that has been around as least as long as language. The recent development of xml and the set of related technologies is based on a small number of basic ideas:

1. *Data should be encoded using international standards that are independent of computer hardware, system software, and applications software.*

XML documents are text based (as opposed to binary) and conform to a number of 8-bit and 16-bit Unicode http://www.unicode.org/ standards. The UNICODE standards allow for easy and efficient encoding of English and other European language character sets and also cover the full range of encoding that embraces all the languages of the world including, for example, Japanese, Tibetan, and languages of the Indian subcontinent. The Unicode standards currently include over 95,000 individual characters.

Even though the vast majority of xml documents are constructed and processed exclusively by machine, because of their text based encoding, xml documents are usually referred to as being "human readable." Using established standards for encoding guarantees that all computational devices and software have unambiguous access to data with no concerns about binary formats, word lengths, proprietary formats, license fees, big-endian and little-endian, etc. The commercial power of the drive to open standards text based encoding of data is evidenced by the announcements that Microsoft Office documents and Adobe pdf documents will soon be represented in xml (and thus abandoning the special encoding of these documents that has impeded their access and construction).

2. *Metadata (that is data about data) should be embedded in the same document with the data.*

This is achieved by use of "markup" that provides both structure and partial description of the content of the document. The idea of markup of data has a long history in publishing. Xml is based on the Standard Generalized Markup Language (SGML) developed in the late 1970's and early 1980's that in turn is based on the Generalized Markup Language (GML) developed in the late 1960's. Data documents without markup need separate documentation to specify how to interpret the content of the data document and this description must be coded into any software that processes the data document.

The xml markup is also encoded to be human readable so it can include well-chosen names that can help document the data contents. This has lead to the claim that xml documents are "self describing". This is not completely true. For example, while the data: `House 7 2` is made more understandable by: `<Name feet = "7" inches = "2">House</Name>` it is still not clear whether this is a tall person or a short building. Only further context for the data yields a complete description.

These ideas can be seen in a simple xml-based language to specify network optimization problems. A common data format for network optimization problems is shown in Figure 1.

Figure 1

```
# node attribute time (int), arc attributes length(float), cost(int)
A1   2
A27  8
B2   5
B5   4
C13  7
A1   B2   5.2    8
B2   C13  0.0    5
A1   A27  6.3  234
A1   B5   5.3    3
A27  B5   4.333 12
B5   C13  3.33   21
```

Figure 2 shows the data content after it has been "marked up" in an xml-language that is defined in the schema contained in the file network.xsd.

Figure 2

```
<?xml version="1.0" encoding="UTF-8"?>
<Network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="network.xsd" arcs="directed"
  name="demo for INFORMS-Computing Society">
   <NodeSet>
      <Node name="A1" time="2"/>
      <Node name="A27" time="8"/>
      <Node name="B2" time="5"/>
      <Node name="B5" time="4"/>
      <Node name="C13" time="7"/>
   </NodeSet>
   <ArcSet>
      <Arc tail="A1" head="B2">
         <Length>5.2</Length>
         <Cost>8</Cost>
      </Arc>
      <Arc tail="B2" head="C13">
         <Length>0.0</Length>
         <Cost>5</Cost>
      </Arc>
      <Arc tail="A1" head="A27">
         <Length>6.3</Length>
```

Figure 2 cont.

```
        <Cost>234</Cost>
        </Arc>
        <Arc tail="A1" head="B5">
           <Length>5.3</Length>
           <Cost>3</Cost>
        </Arc>
        <Arc tail="A27" head="B5">
           <Length>4.333</Length>
           <Cost>12</Cost>
        </Arc>
        <Arc tail="B5" head="C13">
           <Length>3.33</Length>
           <Cost>21</Cost>
        </Arc>
      </ArcSet>
   </Network>
```

Figure 2 xml document (networkFigure2.xml)

The contents of an xml element is contained between the start tag and the matching end tag, for example: `<Length>5.2</Length>` or in the key-values attribute pairs, for example: `tail="B2" head="C13"`.

3. *All xml documents are structured as root trees.*

All elements in an xml document must have a start tag and a matching end tag and the elements must be correctly nested, that is, any start tag must be matched with its end tag inside any enclosing element. Each xml document must have a unique "root" element. The rooted tree structure that the markup imposes on the document makes it easy to construct software (for example, parsers and processors) that can work on any xml document. This means that once data is embedded in an xml document, there are a number of powerful computer language tools that can be used to process the data. The tree structure allows names of data items to be defined relative to their position in the tree; this local name scoping allows names to have their meaning fixed relative to their location in the tree structure. The tree structure also allows easy composition of xml documents. It is also possible to include some additional relationships between elements, giving the option to impose a graph structure within xml documents.

4. *Xml is not a programming language, but the formal structure of the documents make it easy to read, modify, process, create, and write xml documents inside a programming language.*

The W3C recommendations to support processing of xml documents have been specified as computer language Application Program Interfaces (APIs). This supports the design of software to parse and process xml documents in different programming languages. Extensive software has been developed in Java, C, Perl, Com, and other languages. The two primary API's are Document Object Model (DOM) and Simple API for XML (SAX). DOM reads an xml document and constructs in memory a tree structure that allows "random" access to any part of the document. This is effective as long as the document is small enough to represent in memory. SAX is an event driven API that processes the document from top to bottom. SAX efficiently processes large documents;

however, the organization of the document (that is the design laid out in the schema) must be such that it is effective to process it in a "single pass." The designs of the programming interfaces are very object-oriented and most of the license free, open source software has been developed in Java.

5. Schemas are used to give a formal definition of an xml-based language.

There are several different schemas for defining an xml-based language. The W3C supports two schemas: Document Type Definition (DTD) and the XML Schema. There are several other schemas developed by other groups. The schemas differ in their syntax and the kind and power of constraints they can impose on xml document structure and content. The more recently developed XML Schema allows considerably more structure and data constraints than DTD schemas. XML Schema has a well-developed notion of data types that can be applied to the contents of elements and to the values of attributes. There are data types for integer, double, Boolean, string, times, dates, URI's, etc. There are also further restrictions of these types, for example, positive, negative, and nonnegative integer. In all there are 44 data types with capabilities for a user to use the base data types to build others. For values with a string data type an enumerated list can be specified. Also for string values regular expressions can be specified. The regular expression capability is built on the Perl regular expressions. In addition to these "simple" data types there are "complex" data types that allow any subtree to be specified as a type. Complex data types have object-oriented language capabilities that allow derived data types. There are also ID/IDREF and KEY/KEYREF properties to allow easier compatibility with relational databases.

The XML Schema, network.xsd, for the network shown in Figure 2 specifies that the xml document must have one or more nodes, zero or more arcs, the two names that define the head and tail of each arc must be the name of a node, the node attribute "time" must be a nonnegative integer, the arc element "Length" must be a double, the arc element "Cost" must be a nonnegative integer, "Cost" and "time" are required, "Length" is optional, and the names of the nodes must begin with the letter A, B, or C followed by one or two digits. All these restrictions are enforced by a validating parser. Construction of network xml documents can be supported by an editor (based on the schema) that could, for example, when entering an arc provide pull down menus for the head and tail node that would only include nodes that have already been defined.

The designer of an xml-based language strives to include as many constraints as possible in the schema since this will enable many data errors to be detected automatically by a validating parser (this can greatly reduce but does not usually eliminate the data checking that must be done by other means). For example, in the schema that includes a "LastName" element, a schema could use a regular expression to specify that a name must begin with a letter and contain only letters and dashes, however, this would not flag "fly-by-night" as being an invalid last name. In all the different natural languages encoded in xml there is a well-defined character set with letters, digits, and other characters, thus it is possible to construct similar constraints in any natural language.

The designer who uses the XML Schema has extensive flexibility to specify an xml-based language that is tightly defined. Validation allows the creator of an xml document and any user of that document to be assured that the data values conform to the constraints defined in the schema. This is particularly important when documents are automatically produced and processed and when they move automatically across organizational boundaries. Many applications that have adopted xml for data input and output have seen a dramatic decrease in the amount of code necessary to do data error checking. Similar to the effect of using a database, xml can move nearly all the "data modeling" outside the application; this allows use of the powerful xml tools and directly supports interoperability of data among applications.

6. *For documents that are based on an xml-based language that is defined by a schema, there are power tools to automate many important tasks.*

For example, given a schema there is software to:

- Construct an xml document with the required structure and including optional data items.
- Construct documentation to describe the xml-based language.
- Construct an editor for creating an xml document that allows only valid data to be put into the document.
- Transform an xml document based on one xml language to an xml document based on a different xml language.
- Transform an xml document into an HTML or XHTML document for display in a web browser.
- Transform an xml document into a document of any type (that is, not necessarily HTML or another xml document)
- Convert the xml schema into a relational data base schema (and the reverse)

Other Related Technologies

There are related technologies to transform an xml document to another xml document, to an HTML document, or to a document that is neither (for example, a book or report). Extensible Stylesheet Language (XLS and XSL-FO), XPath, and Extensible Stylesheet Language Transformations (XSLT) are used for these transformations of xml documents. An XSLT transformation can be based on the schema for a particular xml-based language; thus the transformation applies to any xml document that is valid with respect to the schema. Transformation via XSLT is an alternative to using a programming language to read the document, process it, and write out the transformed document.

Figure 3 An XSLT document: letter.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/Letter">
  Dear <xsl:value-of select="Addressee/Title"/>
        <xsl:value-of select="Addressee/Last"/>:
       <p>We are writing to offer you a new credit card....
       </p>
        </xsl:template>
</xsl:stylesheet>
```

applied to an xml document: letter.xml

```
<?xml version="1.0"?>
<Letter>
 <Addressee>
  <First>John</First>
  <Last>Doe</Last>
  <Title>Mr.</Title>
 </Addressee>
</Letter>
```

The document shown in Figure 3 yields:

```
Dear  Mr.Doe:


We are writing to offer you a new credit card....
```

XSLT is an xml-based language that is also a programming language. It is a functional language as opposed to procedural languages such as Java and C. XSTL matches templates against the xml document. Like other functional languages (such as Lisp and Scheme), XSLT has no variables and no assignment statement. One of the benefits claimed for XSLT is that it is easy for non-programmers to master (I have found in my xml course that the initial learning curve is steep for programmers and non-programmers alike). While it is true that it is possible to do some simple transformations much more easily than in a procedural programming language, complex operations can be difficult (for example, to sum a column of numbers you need to use recursion). Since the transformation can be based on the schema for an xml-based language, there are very powerful tools that allow the construction of the XSLT using a drag and drop interface; these systems allow the construction of complex transformations with the details largely hidden from the user.

From the very beginning of the development of xml, there was a strong focus to separate the content of the data from how it should be presented (in a book, report, web page, PDA, cell phone). The notion is that there will be xml documents to hold the data and potentially many different ways to present it (some xml, other not). XSLT and programming language processing are the two ways to perform these transformations. The choice depends on the application and the skill set of the user, however, the constant improvements in the tools available for processing xml documents has kept this a difficult choice. New (and often free) tools can dramatically improve some portion of the task and thus suggest a more efficient approach.

Because there are many powerful tools to transform data content from one xml-based language to another, often the largest cost of adopting xml is initially getting data into an xml-based language. This often involves designing a schema for a new xml-based language and then constructing each xml document. The cost of this and the division of effort between design and construction depends on the form and volume of the data. Most data formats, even flat files, are already in some form of tree structure, so the initial move to an xml document is often straightforward. There are powerful tools to support these efforts, especially if the data is in a relational database. There are tools to automate the transformation from a database schema to an xml schema (and the reverse). Note that for large applications the design of an effective xml schema is as much work as constructing an effective database schema. Even with the same data the xml schema (tree structure) can be organized quite differently from the database schema (table structure).

Designing an xml-based language is equivalent to creating a computer language; the available xml software tools automatically and instantaneously create software that is equivalent to constructing a compiler for that language. The combination of the tree structure of all xml documents and the powerful computer language translation tools developed over the last three decades has made possible powerful xml tools. These tools also guarantee that the validity checks in the schema can be used to protect the data validity at each step of the process.

The tools for xml include capabilities to support object-oriented programs by generating objects directly from xml documents (this is called data binding). Because xml was developed the same time as Java

and because much of the open source software is written in Java, there is good support for transferring data between xml documents and object-oriented programs.

Operations Research Examples

The applications of xml technologies to operations research problems that I have been involved with demonstrate a few of the xml capabilities that were discussed above.

1. Xml technologies are effective for data input and output.

Xml is being widely used to format data that is shared among applications; today more and more of those applications are in different systems on different computers connected by a network. As discussed above, xml-based languages have been developed to support this interoperability of data. Most computer applications also have data input that is used to configure the application. This allows the application to be customized via data rather than using code that would have to be recompiled when it is modified. External configuration data also supports user operations to set and save options. Xml is being used widely for configuration software replacing flat files of data or files with lists of key-value pairs (often called property lists). There are a number of benefits that come from using xml:

- The data files can be formatted in a tree structure with human readable names to describe the data.
- There are API's for popular programming languages that can read in and, if needed, write out the data.
- A schema to define an xml-based language for the data can be constructed.
- Using the schema, documentation can be constructed automatically.
- Using the schema, an input editor can be constructed that facilitates the input of the data.
- Using the schema, anyone who constructs the data file can immediately validate it.
- Using the schema, the program that reads the data can validate the data (thus eliminating most, if not all, of the necessary data checking).
- The schema makes it easy to reuse the xml-based data language in whole or in part in other applications.

These advantages have lead many computer applications to convert from application specific data formats to xml.

Arnie Buss and I, together with a number of masters' students, have constructed an "Extensible Analyst Toolbox" to support the rapid construction of military planning systems from pre-existing components. The projected user of this system is an operations research analyst who is planning military operations for a particular locale. The user first identifies the maps, images, satellite photos, data sources, road networks, terrain data, drawings, etc. that are associated with the area of operation. This data is specified in xml files that are loaded into the planning system as layers. The user has the opportunity during the analysis to load one or more layers of various types (images, graphs, drawings, road networks, terrain, line-of-sight, etc.) Each planning situation is unique, so it cannot be anticipated what data and layers will be needed. The analyst constructs the required system by "loosely coupling" pre-existing components. One use of xml is to construct the configuration files that specify the user interface (menus, toolbars, tool tips, buttons, combo boxes, etc.) for the system. Each component layer has its own xml configuration file; the analyst can use a data editor constructed from the schema to specify the content and look of the user interface. The user-friendly data editors and the validation of the data files make it easy for the analyst to construct the layer interfaces.

MontereyMenu.xsd is the schema for the main window menu bar and MontereyMenu.xml is an instance document based on the schema. MontereyMenuSchemaDiagram.doc is a graphical view of the schema that is constructed by XMLSPY (a proprietary xml integrated development environment from Altova, Inc.) Note that the tree structure of the menus is easily captured in xml. When we switched to xml, the code to construct the interface for each layer was replaced by a single generic method to read them all. The validation of the xml eliminated the previous data checks and the generic code to construct all the interfaces is shorter than the previous code for each layer. This approach can be reused in any application. Our program is written in Java; we used the open source JDOM API and the Apache Software Foundation xerces parser. (This follows our software maxim to maximize the code written by others and minimizes the code written by us.)

2. Scenario specification for simulations

The most time consuming part of executing military simulations is the data preparation to specify the scenarios to be analyzed. High resolution military simulations require detailed and extensive information on terrain, weapons, and plans. These can involve tens or hundreds of thousands of data values. Some of the data must be generated by the analysts running the simulation. Other data is gathered from other organizations; this information is increasing available over the web and often in an xml document. Reuse of scenario data from previous analysis is often difficult especially if it is not possible to easily access and automatically process the data.

If the scenario is validated by hand (that is, "by eye") then a large fraction of the simulation code and a large fraction of the analysis time is devoted to identifying and correcting errors in the data. Often the simulation is used to debug data as in "these results can't be right, there's something wrong in the data".

The Combined Arms Analysis Tool for the 21st Century (COMBAT XXI) is a recent large-scale simulation project that was developed in Java with all the scenario data being specified in xml. It is a high-resolution, analytical combat simulation focused on tactical combat. It is being developed by the U.S. Army TRADOC Analysis Center-White Sands Missile Range (TRAC-WSMR) and the Marine Corps Combat Development Command (MCCDC). When the project was begun five years ago it was on the cutting edge as the xml technologies evolved. In the initial version of the project they had custom build code to generate system objects from data in xml files. When xml technology evolved to automatic, generic construction of objects (data binding) they were able to redesign their system and achieve an order of magnitude reduction in the number of lines of Java code. This demonstrates the impact of the power tools that are being developed (they are especially powerful for object-oriented designs). The use of the free, open source, general purpose, and thoroughly tested tools to replace application specific code is a very effective way to reduce development costs, time, and risk.

Several masters' thesis research projects at the Naval Postgraduate School have used xml for scenario specification. The initialization of a simulation written in an object-oriented language such as Java involves reading in data values from the scenario data files and then instantiating the many objects associated with the scenario.

The Multi-Agent Robert Swarm Simulation (MARSS) system developed by Alistair Dickie takes a innovative approach to using xml in an object-oriented simulation. MARSS is an agent based simulation of unmanned aerial vehicles; it includes a 2-dimensional and 3-dimensional viewer that presents the positions of the vehicles as the simulation unfolds. The thesis, a power point presentation, and the xml schemas can be downloaded from his web site. You can also download the simulation that automatically installs itself and any necessary Java

code needed to execute the code. http://diana.or.nps.navy.mil/~ajdickie/marss/. Currently the 3-dimensional viewer software is available only on Windows operating systems.

In MARSS in addition to the scenario data values, the xml files include the names of the Java objects to be constructed and the parameters that are necessary to instantiate them. Parameters to instantiate objects may be objects themselves that must first be instantiated with parameters that may be objects etc. etc.; thus it is necessary to write code to construct arbitrary Java objects that are specified in the xml document. The schema to specify the xml-based language and the Java code to read the xml is by necessity generic (works for any Java object) and recursive, however, it is dramatically less code than writing code to construct each needed Java object. The structure of the xml documents together with validation using schemas allows a general solution to what was previously longer and very problem specific.

The MARSS approach to initializing simulation objects from external files is a demonstration of the power of xml ideas and xml tools to directly address one of the most pressing problems of running large-scale simulations. As with other uses of xml, custom data editors and validation provide important functionality. The extensive data modeling and data validation using xml technologies allows the separation of data modeling and simulation logic that supports reuse in both.

3. Xml-based language for network optimization

I have been working to develop an xml language to represent and validate network and graph problems. Designing an xml language for a particular domain involves following certain design patterns and managing some difficult tradeoffs. For network optimization it is necessary to follow the xml principle of separating content and presentation. An important part of the design is to support capabilities to generate displays of the networks and graphs; however, this should be accomplished only by transformation of the xml documents that hold the network topology and node and arc properties. A difficult tradeoff is between validation and generality. If the standard puts restrictions on the naming of properties (as was done in Figure 2), then stronger validations are possible, but the standards can be applied to fewer networks. Another tradeoff is between readability (long descriptive names) and the size of the xml file (most network and graph models have large amounts of data). Another issue is the division of labor between the xml validation and data checking within a reader, that is, validations that are difficult or impossible to specify in the schema may be performed in a data reader that can be considered as part of the total data modeling. This is a work in progress, for the latest refer to: http://diana.or.nps.navy.mil/~ghbradle/xml/index.html.

4. Other use of xml in operations research

A. Leon Lopes and Bob Fourer have developed SNOML to provide an xml representation of stochastic programming problem data for problem instances. http://senna.iems.nwu.edu/xml

B. Bjarni Kristjansson has proposed the xml-based language Optimization Markup Language (OptML).

http://www.maximal-usa.com/slides/

C. The Computational Infrastructure for Operations Research (COIN-OR) has provided a forum for researchers interested in developing open-source xml-based languages for optimization to work together.

http://www-124.ibm.com/developerworks/opensource/coin/

D. Kipp Martin has shown how XSLT and XPath can be used to generate xml documents that contain an instance of a mixed integer linear programming problem. He shows how to go directly from the data for a particular problem to the xml document bypassing the use of traditional algebraic modeling languages. His work demonstrates the feasibility of developing an xml-based language for problem instances and using XSLT transformations to construct the xml documents that contain the problem instances. If this xml-based language was adopted as an input standard for optimization solvers, there would be a well-defined, open source, open standards path from the world of data in xml to the solution of optimization problems. He also discusses the access of problem instance data from xml data documents, relational databases, spreadsheets, and flat files. http://gsbkip.uchicago.edu/xslt/pdf/xmlmodeling.pdf

Final Comments

It is obvious that the operations research examples mentioned here touch on only a few of the many possible applications. In addition, they utilize only some of the xml technologies discussed earlier in the paper. For example, they don't show the use of xml editors that can be automatically generate from xml schemas, the constructing of XSLT transformations, or stylesheets to construct dynamic HTML or XHTML pages, the use of technology for client/server applications over the web, access to databases, automatic construction of xml and database schemas from other schemas.

I have been careful not to "oversell" xml as "self describing" data or as the ultimate answer for knowledge management. There are other technologies under development that are focused on taking the next step to more fully describe data and the context in which it is embedded. The W3C is working on the Resource Description Framework (RDF) and the "semantic web." Researchers in artificial intelligence and knowledge management are working on ontologies for particular domains. Xml is an important contributor to this work because these more powerful capabilities can be constructed on top of xml.

Acknowledgments

My research using xml in simulations and military mission planning systems is supported by the Air Force Office of Scientific Research. My work on designing an xml-based language for network and graph optimization is supported by the Office of Naval Research.

References

1. Bradley, Gordon, link to xml work: http://diana.or.nps.navy.mil/~ghbradle/xml/index.html

2. Bradley, Gordon, "Extensible Markup Language (XML) with Operations Research Examples," tutorial given at the Eighth INFORMS Computing Society Conference, January, 2003, Chandler, AZ, PowerPoint slides

   http://diana.or.nps.navy.mil/~ghbradle/xml/PaperMay2003/GBradleyXMLTutorialJan03.zip.

3. Common Optimization Interface for Operations Research (COIN-OR), http://www-124.ibm.com/developerworks/opensource/coin/

4. Cover, Robin, "The XML Cover Pages," http://xml.coverpages.org/xml.html good source of current xml information.

5. Dickie, Alistair, "Multi-Agent Robot Simulation System (MARSS)," http://diana.or.nps.navy.mil/~ajdickie/marss/

6. Hunter, David, et al, "*Beginning XML*," 2nd edition, WROX, 2002. This is the book I use in my beginning xml class; this is not a textbook, it is written by programmers for programmers. While there are many books (some quite good), I have not yet seen a good textbook, that is, a well-organized book with background material, clear examples, homeworks, projects, and references.

7. Lopes, Leo and Fourer, Robert, "SNOML," http://senna.iems.nwu.edu/xml/

8. JDOM, http://www.jdom.org, developed by Brett McLaughlin for his book "*Java and XML*" to be more natural for the Java programmer; it has grown into an major open source effort.

9. Kristjansson, Bjarni, "Optimization Modeling in Distributed Applications: How New Technologies such as XML and SOAP allow OR to provide Web-based Services," http://www.maximal-usa.com/slides/Svna01Max/index.htm and http://www.maximal-usa.com/slides/Montrl02/index.htm

10. Martin, Kipp, "A Modeling System for Mixed Integer Linear Programming Using XML Technologies," December 11, 2002, revised February 27, 2003, 34 pages. http://gsbkip.uchicago.edu/xslt/pdf/xmlmodeling.pdf

11. Microsoft's XML implementation, http://msdn.microsoft.com/xml

12. XML.com, http://www.xml.com/

13. The open source sites listed below contain a huge amount of information on the existing and evolving standards for xml and related technologies. There is a lot of good information, but it is very heavy going, not the place to start, and not for the faint of heart.

    World Wide Web Consortium (*W3C*) http://www.w3.org/
    Apache Foundation xerces http://xml.apache.org/
    JDOM http://www.jdom.org/
    Unicode http://www.unicode.org/
    OASIS http://www.oasis-open.org/
    XML.org http://www.xml.org/
    OMG http://www.omg.org/

901 Elkridge Landing Road, Suite 400
Linthicum, MD 21090-2909