Deadlock Avoidance Dispatching Algorithm for Generalized Track and Train Configurations

Geordie Roscoe

INFORMS 2020

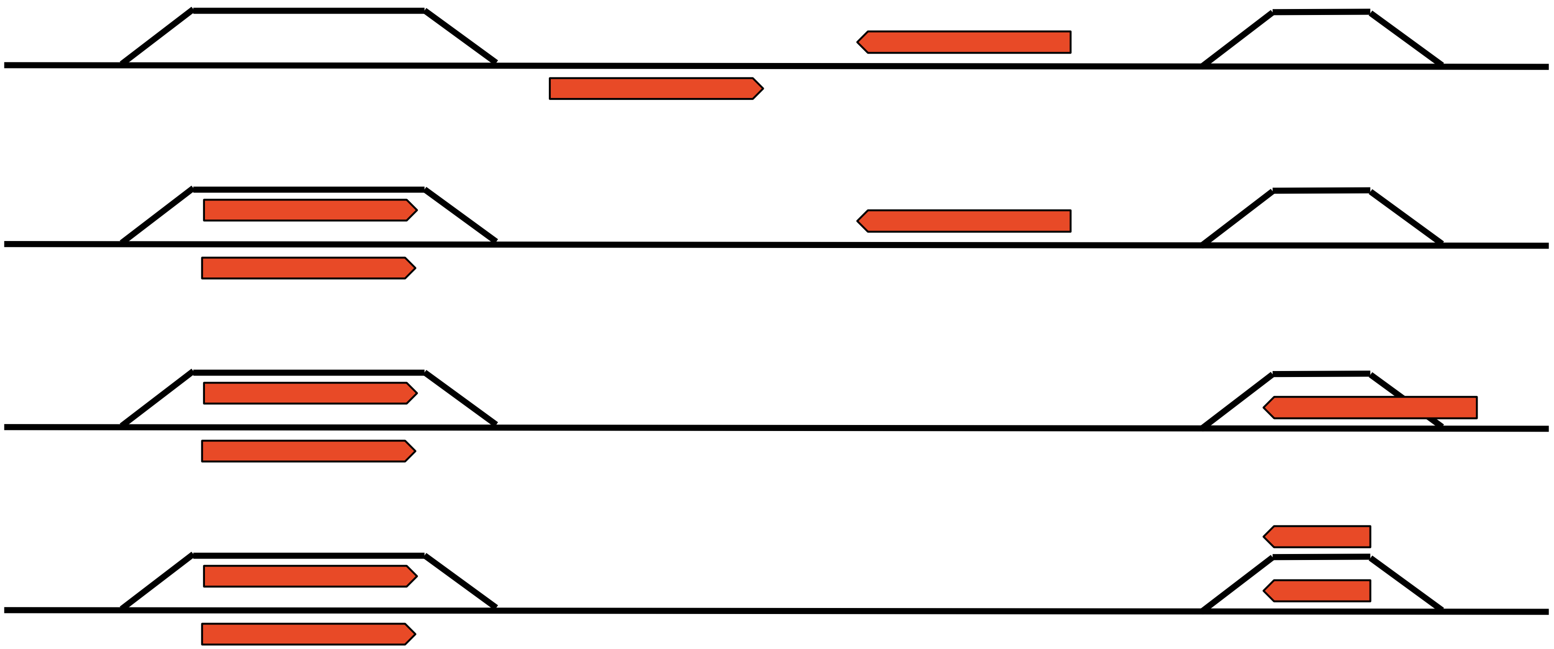
RAILTEC
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Dispatching Algorithm Overview

- **Objective:** develop a dispatching algorithm to resolve conflicts in simulations of new control systems on mainline railway corridors
- **Algorithm requirements**
  - **Track layout and train plan generality**
    - Handle any track configuration and combination of train origins, destinations, work events, and path restrictions
  - **Train length**
    - Consider when resolving conflicts and avoiding deadlock
  - **Solution quality**
    - Emulate a human dispatcher
  - **Solution speed**
    - Dispatch much quicker than real time
  - **Scalability**
    - Handle large geographic and temporal scope at high train density and capacity utilization
  - **Intermediate feasibility** (optional)
    - Allows partial re-simulation if the train plan changes

Deadlock Examples

- Deadlock occurs when one or more trains reach a point where one or more trains must reverse to resolve the conflicts



- How to prevent? Several options:
  - Schedule all moves in advance with mixed integer program
  - Advance and rewind if deadlock is encountered
  - Develop a dispatching algorithm that prevents deadlock by design

Dispatching Algorithm Approaches

Properties	Mixed Integer	Greedy Advancing with State Restore	Deadlock Prevention
Track and Train Generality	Slow when added	In base algorithm	Depends on algorithm
Train Length	Much slower when added	Minor effect on speed	Minor effect on speed
Solution Quality	Optimal	Good	Good
Solution Speed	Slow	Very quick to moderate for low to high train density	Quick for all train densities
Scalability	Low	Very good to moderate for low to high train density	Moderate for all train densities
Intermediate Feasibility	Some is possible	None	Every step
Example Implementation	Higgins model	RTC	GTMS, Railsys

Dispatching Algorithm Approaches

Properties	Mixed Integer	Greedy Advancing with State Restore	Deadlock Prevention
Track and Train Generality	Slow when added	In base algorithm	Depends on algorithm
Train Length	Much slower when added	Minor effect on speed	Minor effect on speed
Solution Quality	Optimal	Good	Good
Solution Speed	**Slow**	Very quick to moderate for low to high train density	Quick for all train densities
Scalability	**Low**	Very good to moderate for low to high train density	Moderate for all train densities
Intermediate Feasibility	Some is possible	**None**	Every step
Example Implementation	Higgins model	RTC	GTMS, Railsys

Dispatching Algorithm Approaches

Properties	Mixed Integer	Greedy Advancing with State Restore	Deadlock Prevention
Track and Train Generality	Slow when added	In base algorithm	Depends on algorithm
Train Length	Much slower when added	Minor effect on speed	Minor effect on speed
Solution Quality	Optimal	Good	Good
Solution Speed	**Slow**	Very quick to moderate for low to high train density	Quick for all train densities
Scalability	**Low**	Very good to moderate for low to high train density	Moderate for all train densities
Intermediate Feasibility	Some is possible	**None**	Every step
Example Implementation	Higgins model	RTC	GTMS, Railsys

Free Path Approach to Avoid Deadlocks

- Propose a **“free path”** approach to resolve conflicts and avoid deadlock

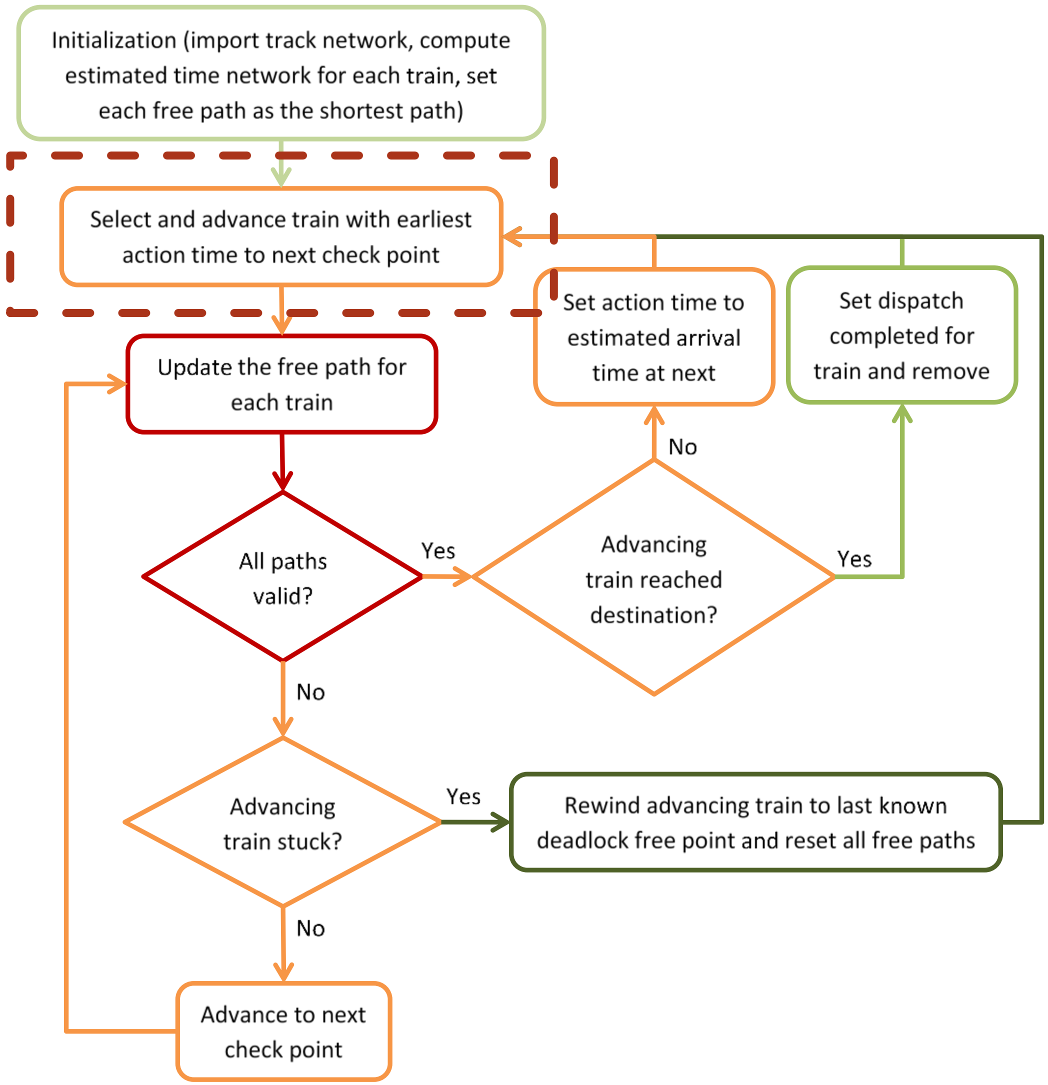
- Free path for a train:
  - Runs from current train position to its destination
  - Does not go through any opposing direction trains
- All trains must have a free path
  - Thus, free paths can go through same direction trains

- Example: for blue train to have a free path from its current position, opposing trains may occupy:
  - any combination of the green positions
  - one of the two orange positions

Deadlock Avoidance Dispatch Algorithm

- Every train has a free path from its current location to its destination **every time the boxed block is reached**

- Many deadlock-free train configurations are not allowed
  - Enables a faster and much more parallel algorithm



Updating Free Path

- Each train has a network listing the shortest path from each route section to its destination
  - Independently determined for each train
  - Shortest path becomes initial free path to destination

- Free path is checked for opposite direction train occupancy
  - If no conflicts are found, free path set to shortest path
  - If free path conflict is found
    - Move backwards from conflict point to first previous possible divergence point and diverge
    - Advance along the new shortest path until reaching a conflict (fail) or another part of the free path (success)
    - If the diverging path is successful, splice it into the free path

- If no valid free path was found, the dispatching state is unsafe (may lead to deadlock)

Base Time Complexity

- Variables
  - n is the total number of trains to be dispatched
- Shortest path network for each train
  - $O(n)$ because each train evaluates this independently
- Dispatch advancing trains
  - $O(n)$ because this step does not depend on any other trains
- Dispatch updating free path
  - $O(n^2)$ because each train advancement event may require updating all other trains
- Overall time complexity is $O(n^2)$

Time Complexity Improvements

- Assume trains beyond some conflict time horizon do not affect current operations
  - Reduces the number of trains requiring a free path update

- Variables
  - c is the maximum number of trains on the network concurrently
  - h is the maximum number of trains in any conflict time horizon

- Dispatch updating free path
  - $O(n^2)$ becomes $O\big(n \times \min\big(n, (c+h)\big)\big)$
  - Largest improvements seen for scenarios with long time horizons

- Overall time complexity of $O(n^2)$ becomes $O\big(n \times \min\big(n, (c+h)\big)\big)$

Time Complexity Parallelization

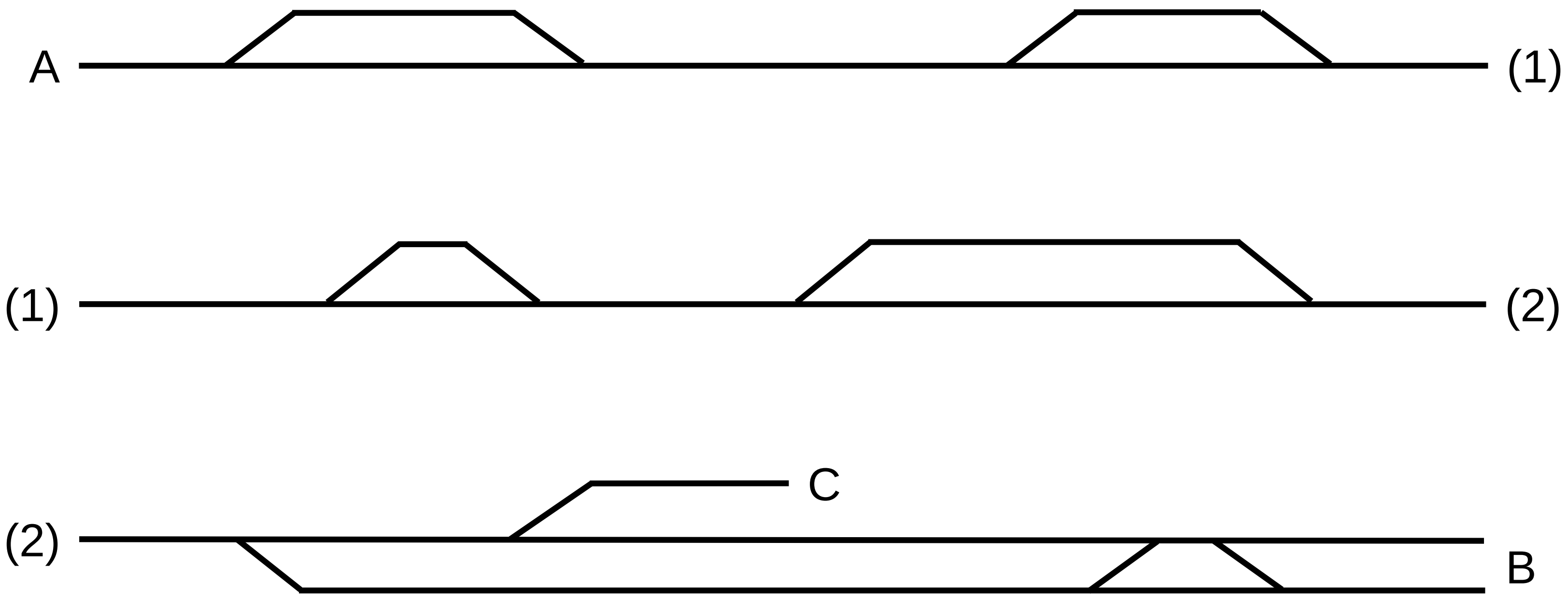
- Shortest path network for each train
  - Complete independence of trains allows linear speedup
- Dispatch advancing trains
  - Cannot be parallelized
- Dispatch updating free path
  - Conflict checking component, with time complexity $\min(n, (c + h))$, allows linear speedup

- Overall results
  - Time complexity is $O(n \times \min(n, (c + h)))$
  - Overall speedup is nearly linear
    - Testing on large scenario yielded ~7× speedup with 8 cores
  - On 1,500-mile network (~10× the size of a typical dispatcher territory), the algorithm runs at ~1000× real-time speed

Dispatching Example Train Plan

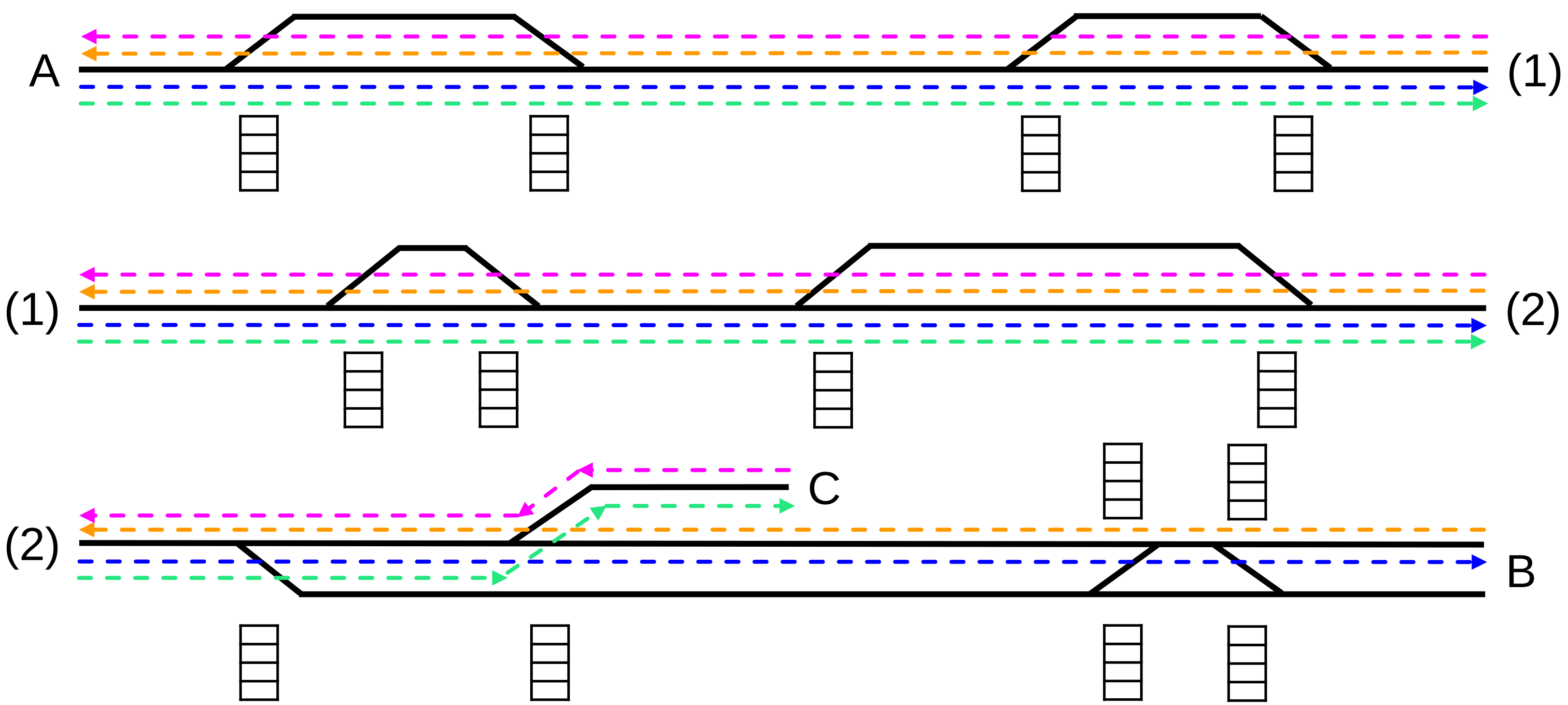
- The four trains scheduled over territory below listed in arrival time order on the right

Train Color	Order	Origin	Destination
Blue	1	A	B
Magenta	2	C	A
Yellow	3	B	A
Green	4	A	C



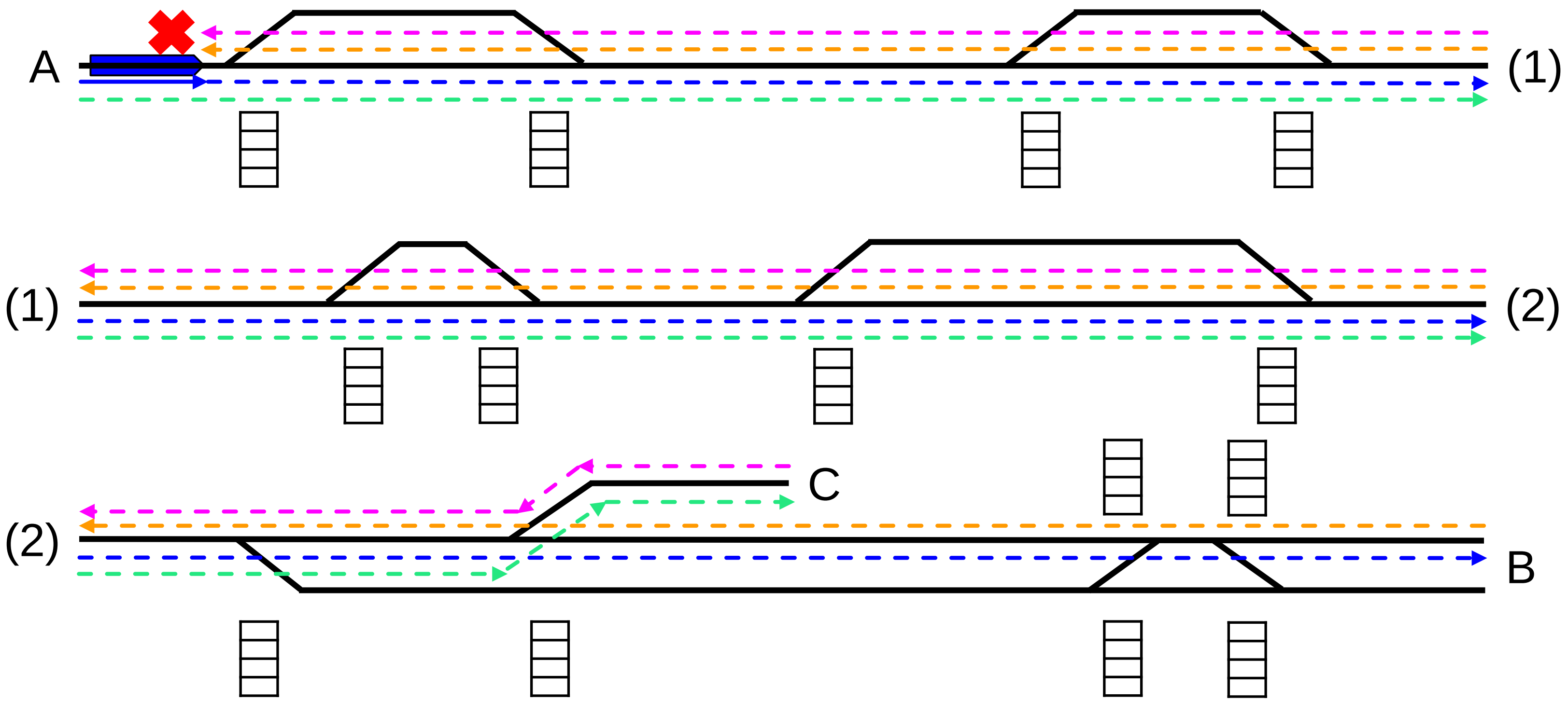
Dispatching Example

- All trains start off the territory and have their free path initialized as the shortest path from origin to destination
- Solid black lines are track, dashed colored lines indicate free path, and solid colored lines indicate traversed path



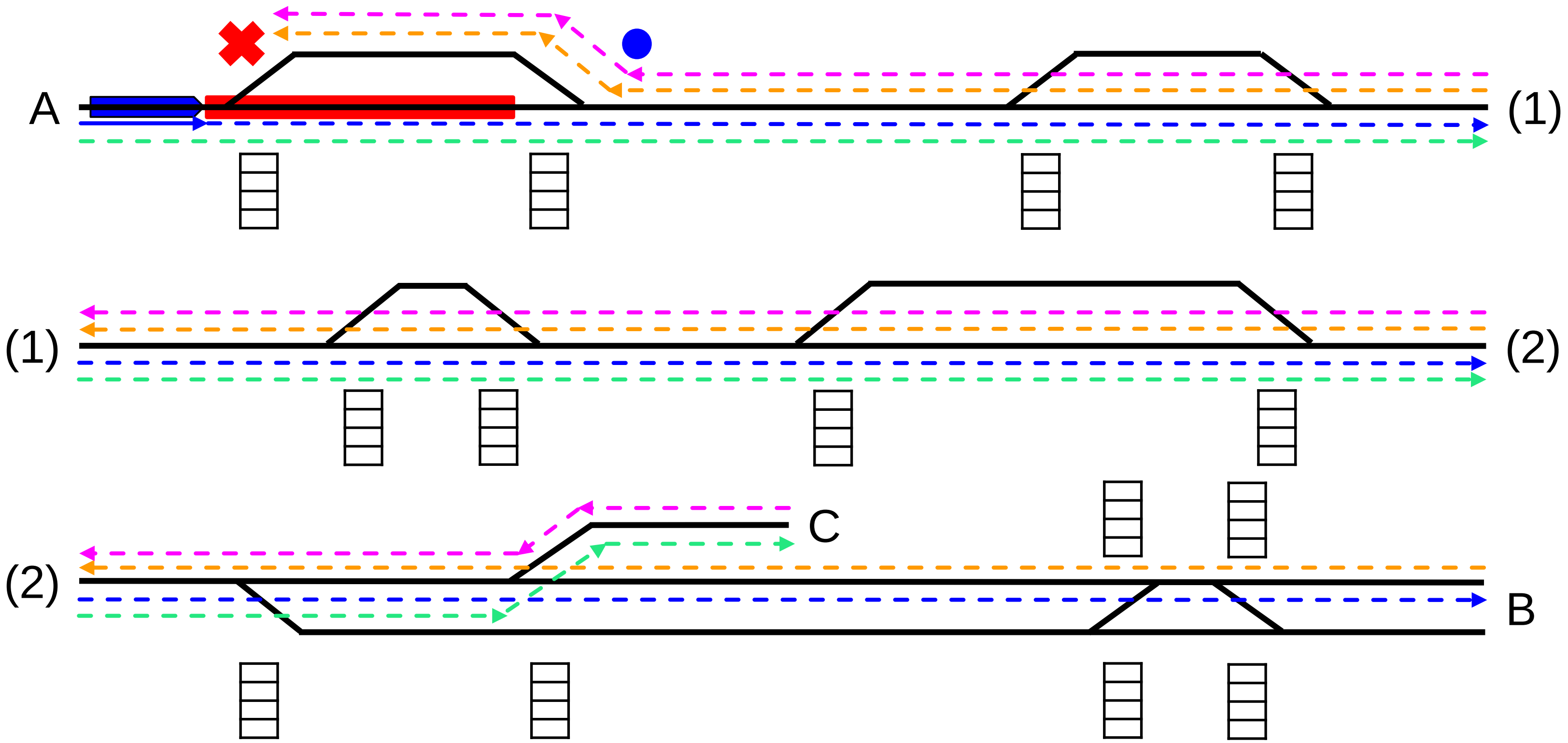
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



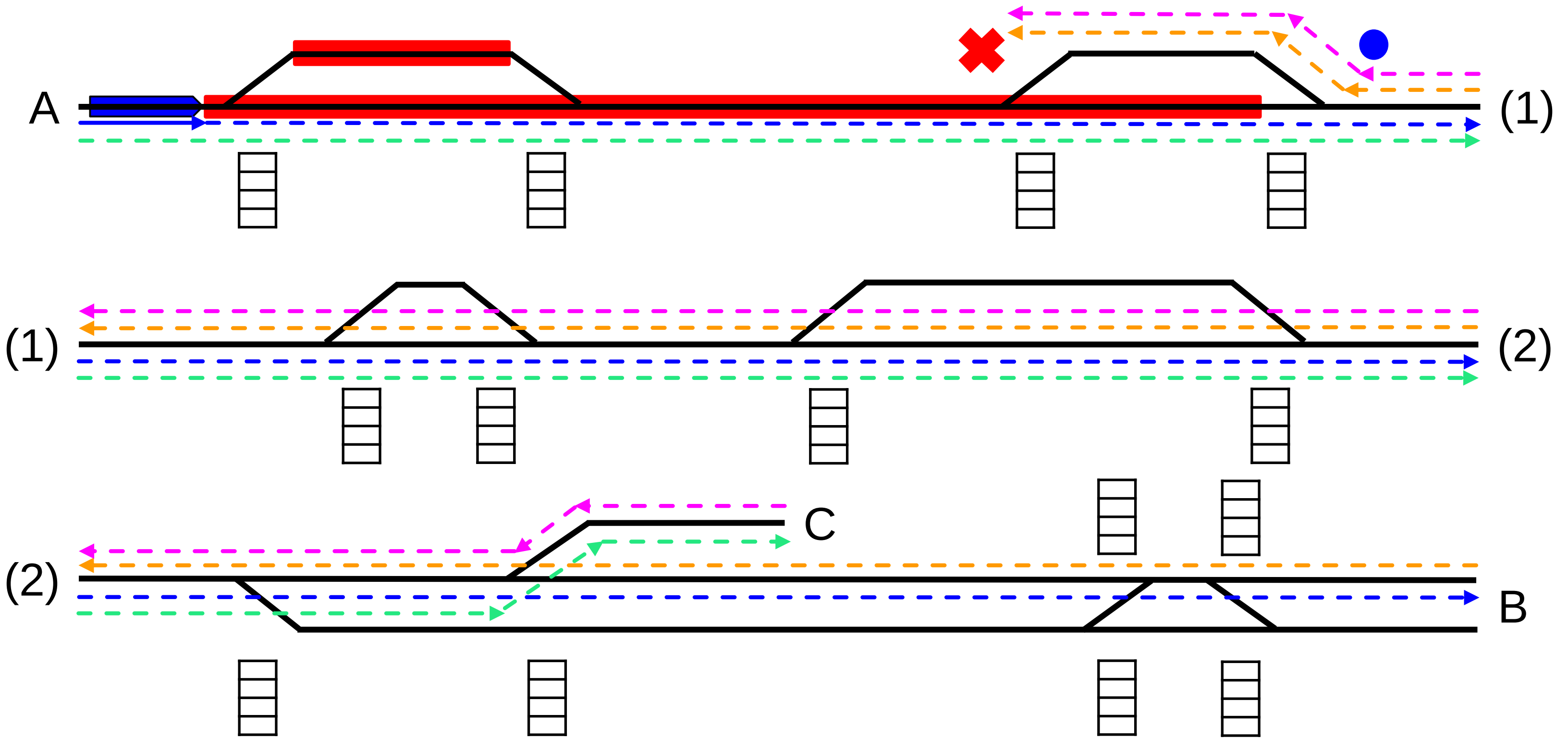
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



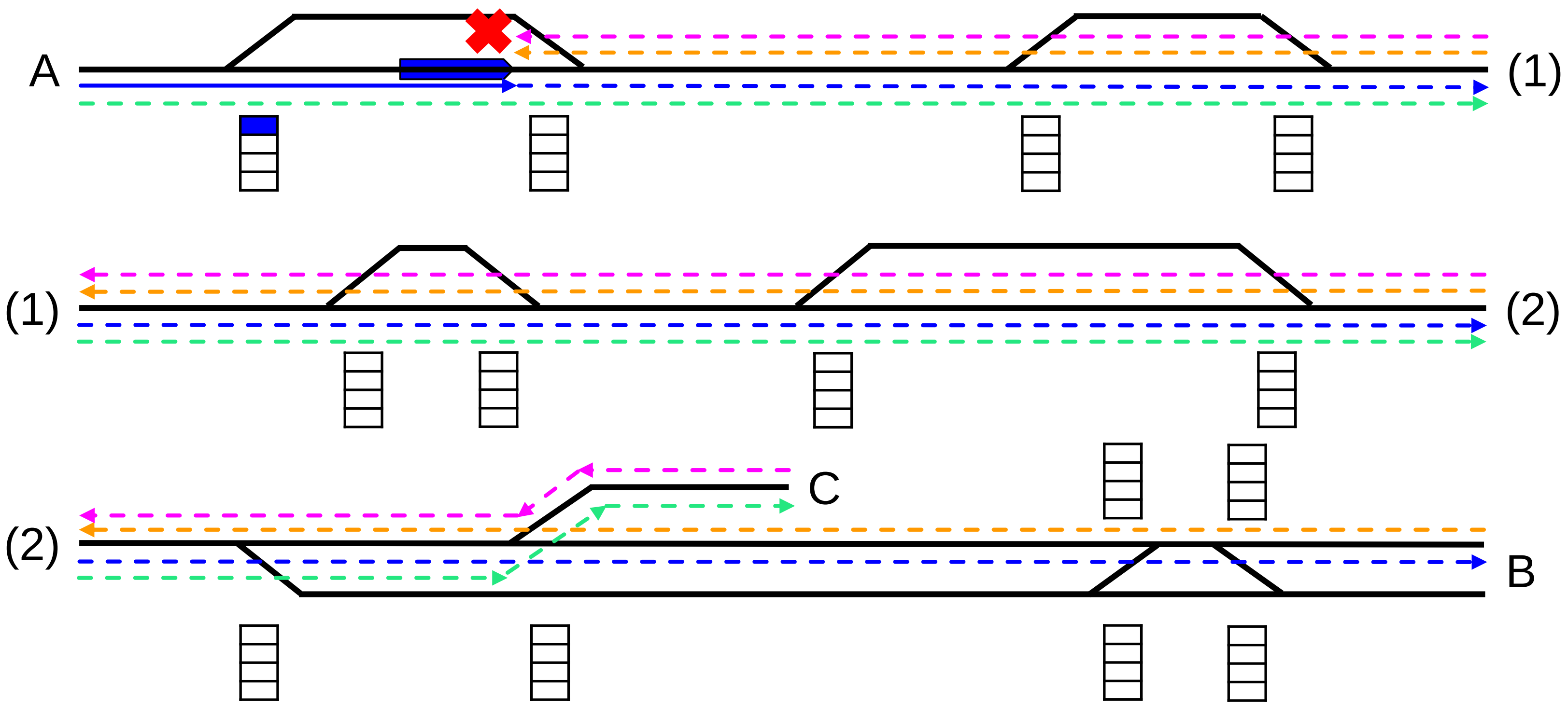
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



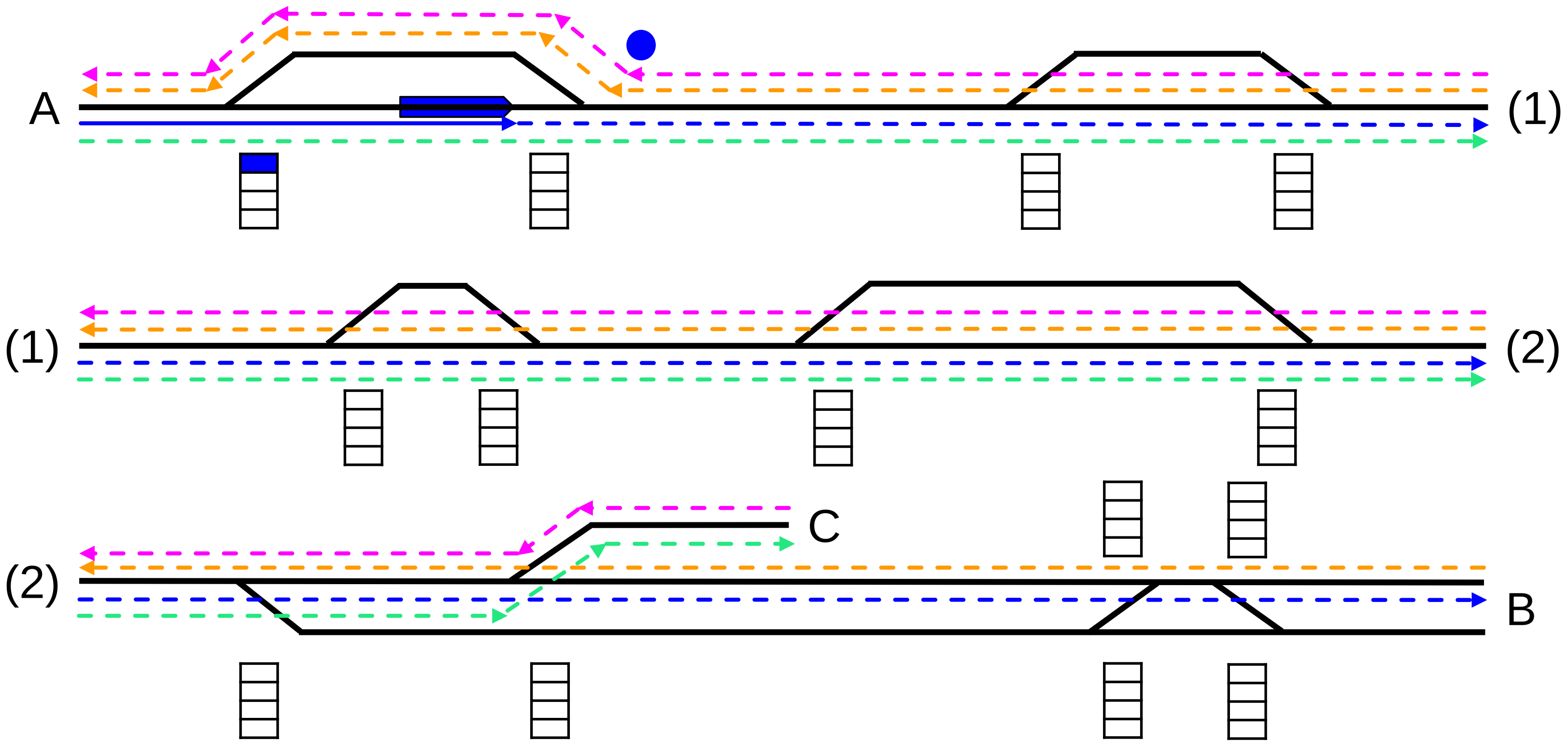
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



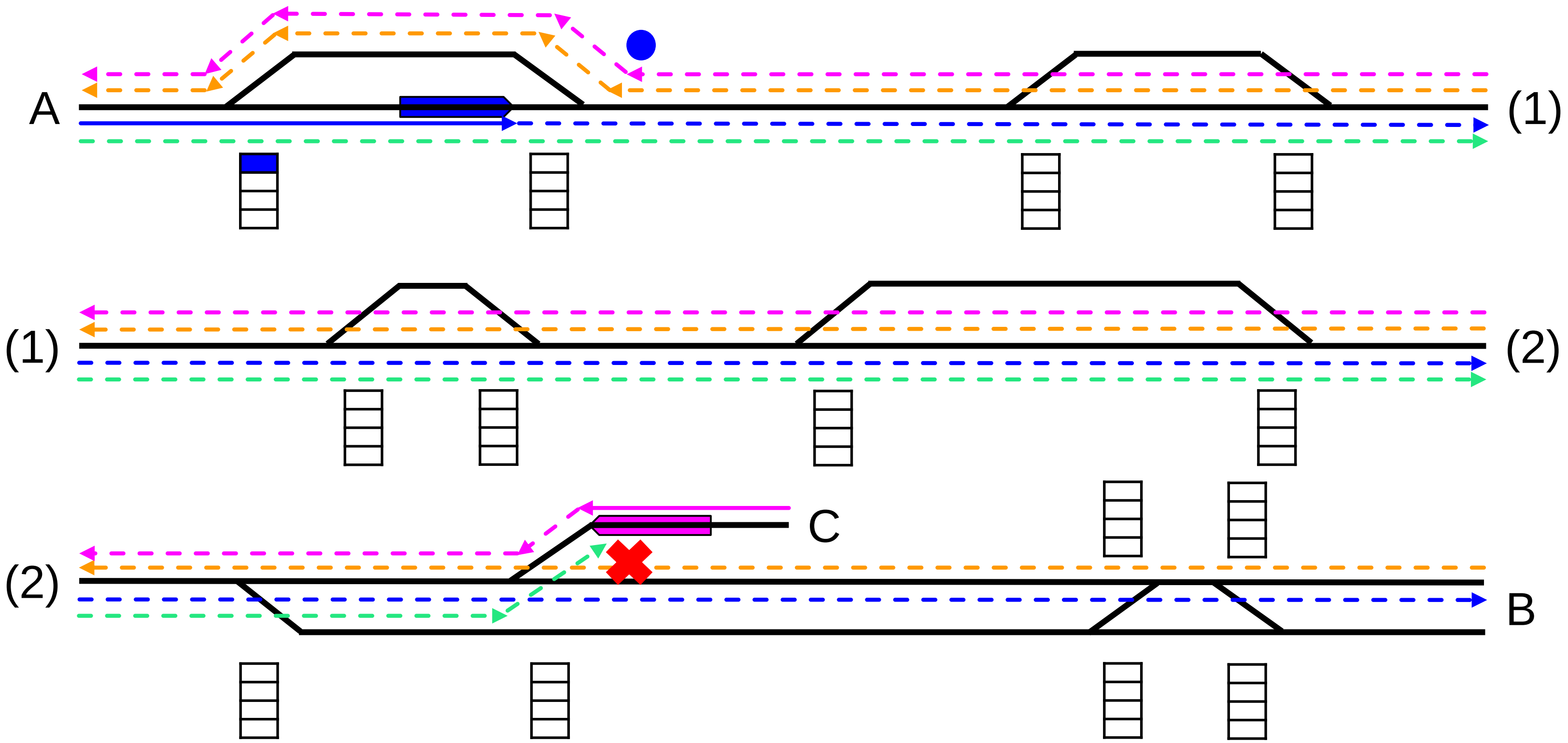
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



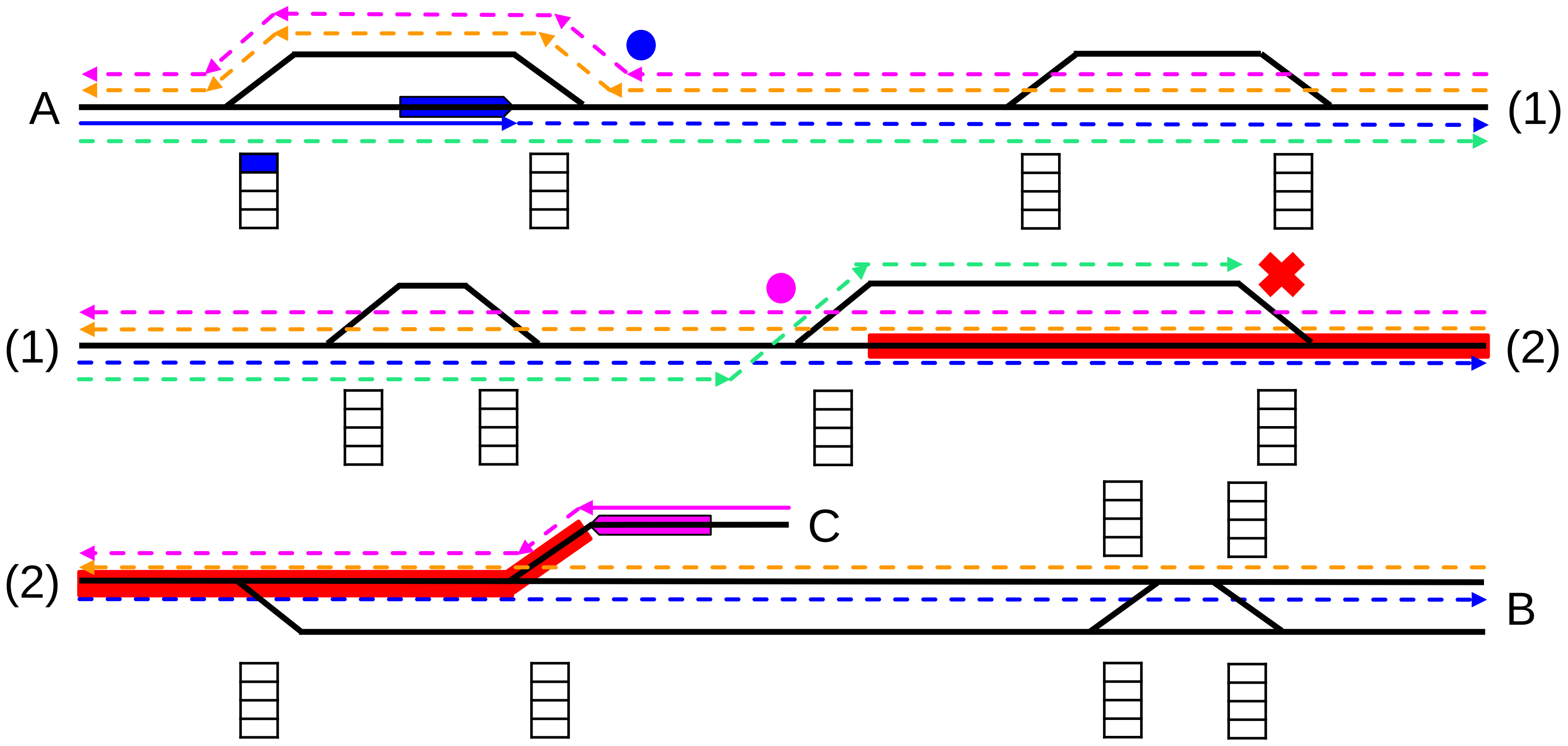
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



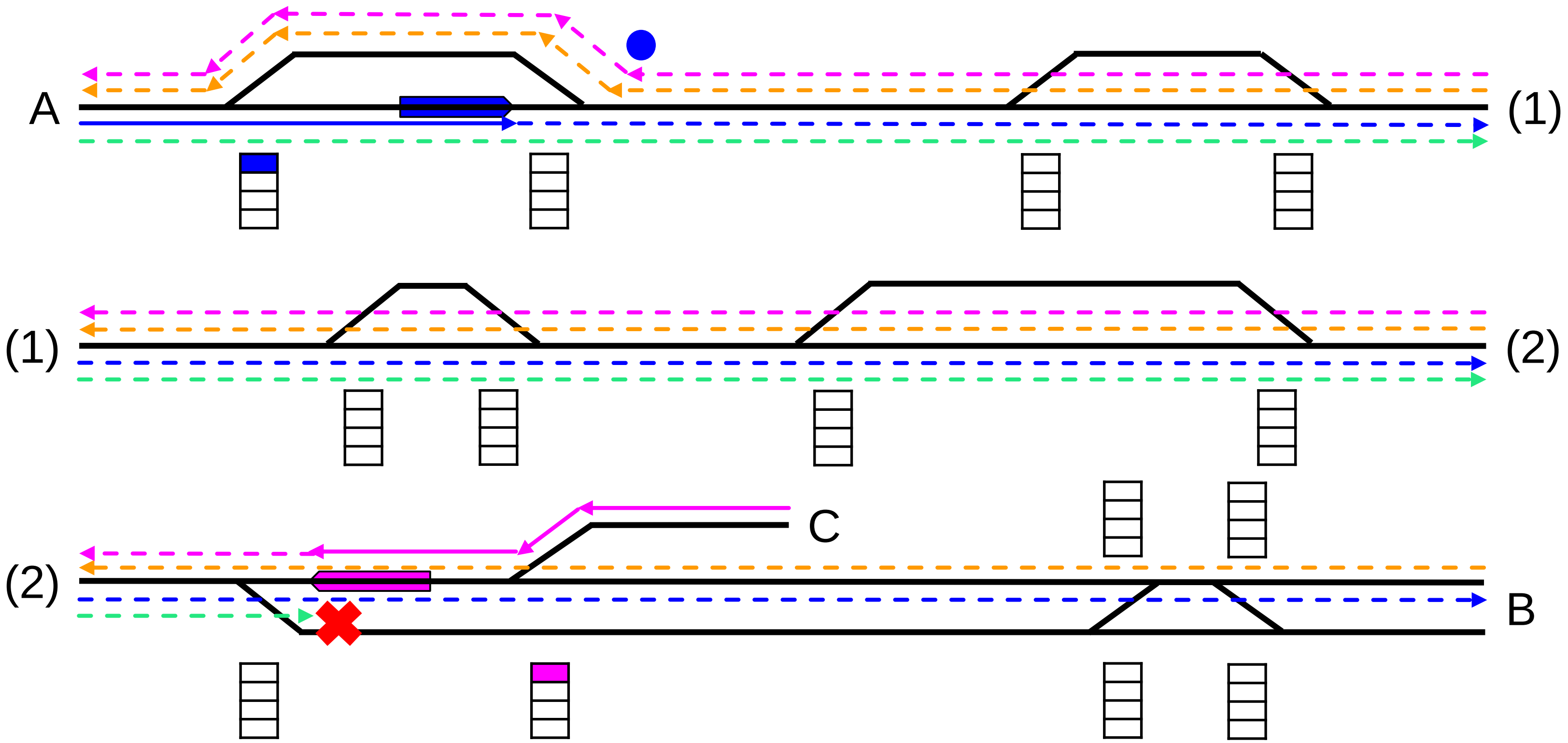
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



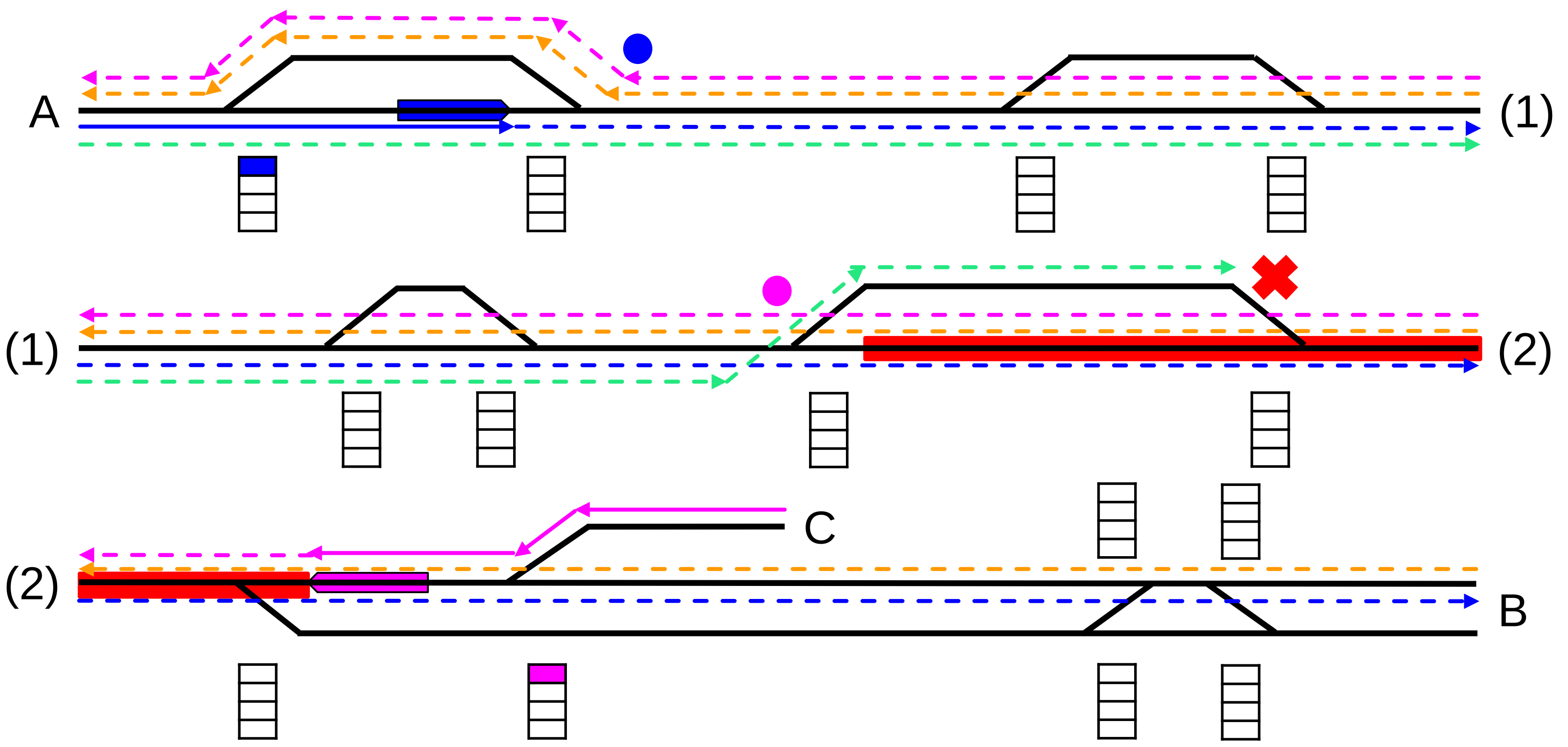
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



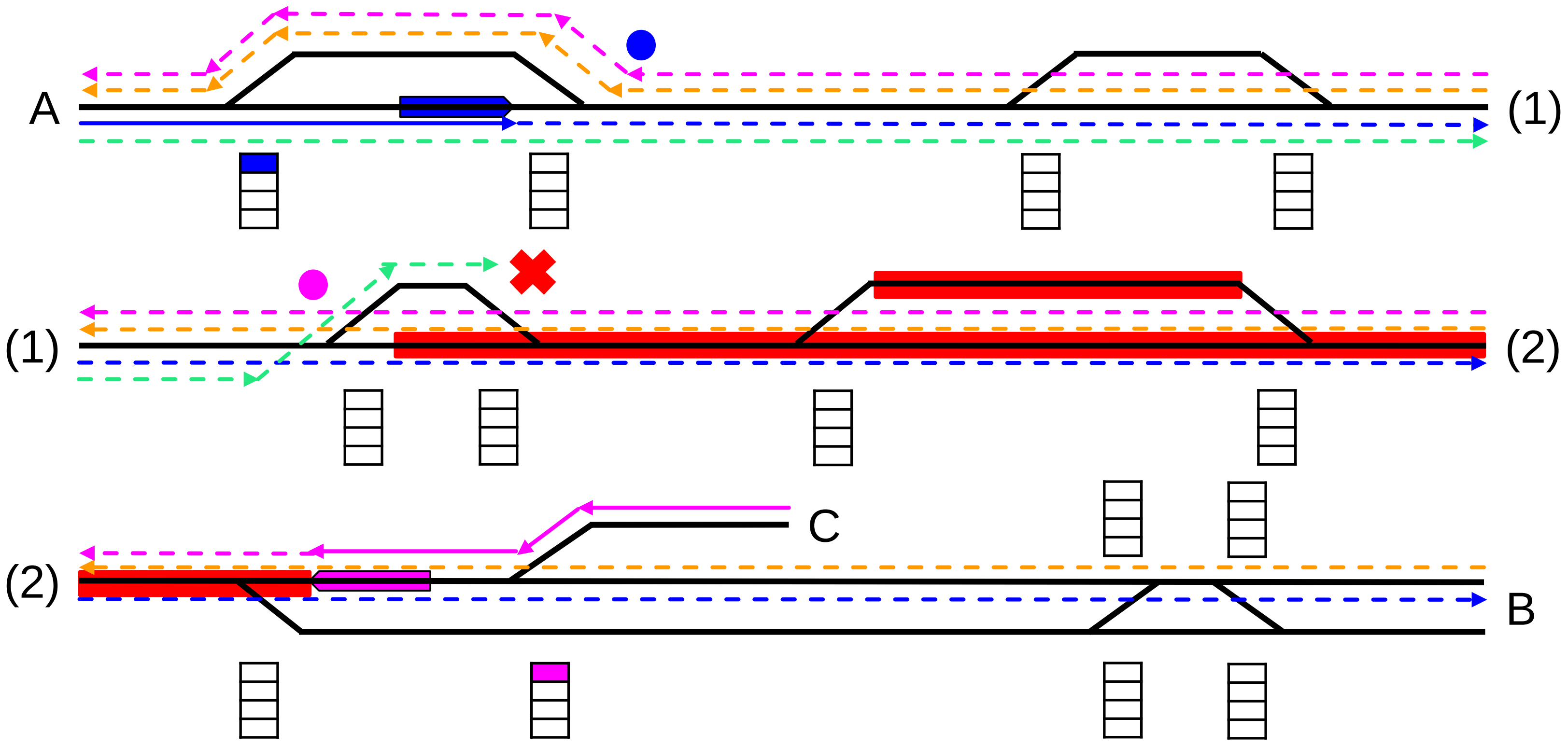
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



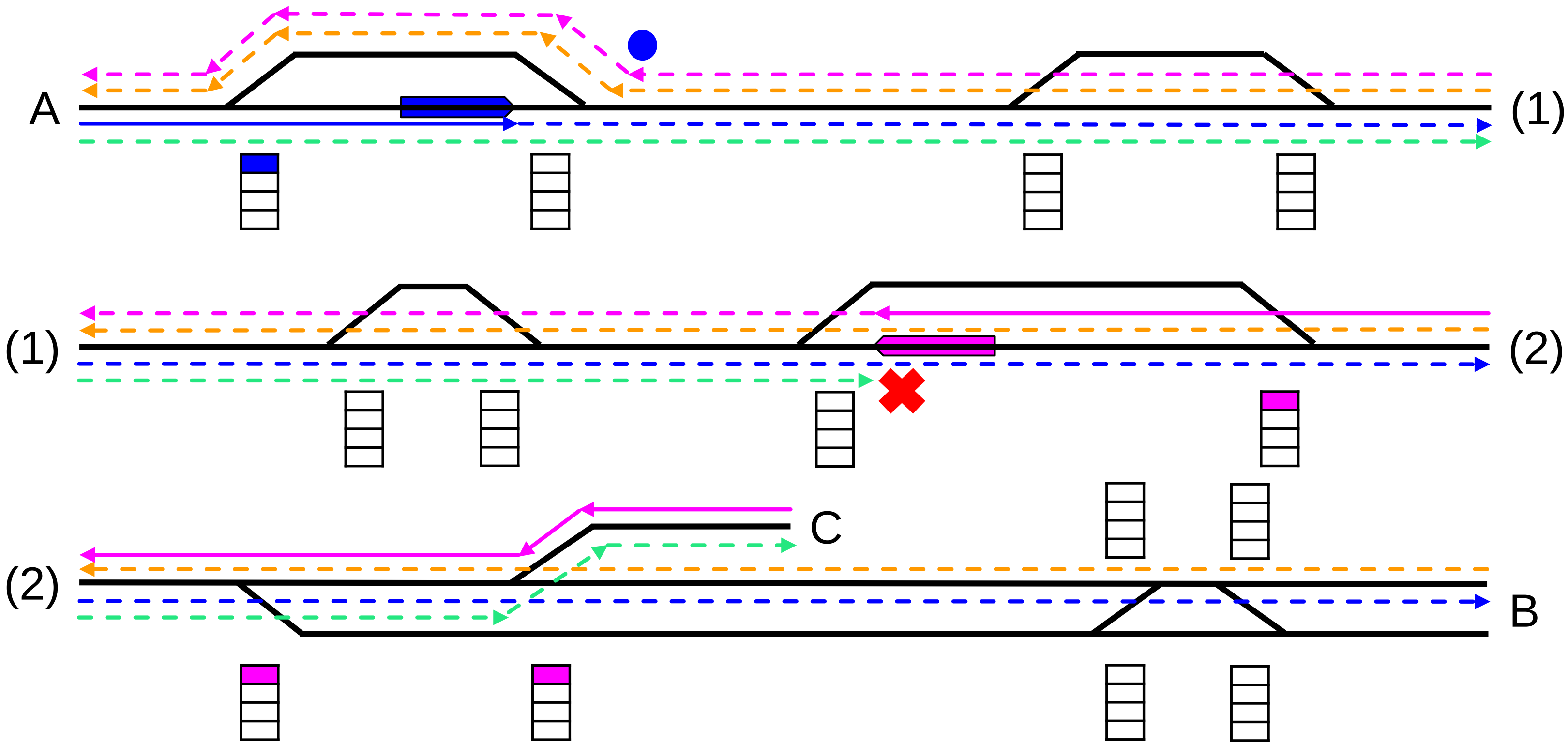
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



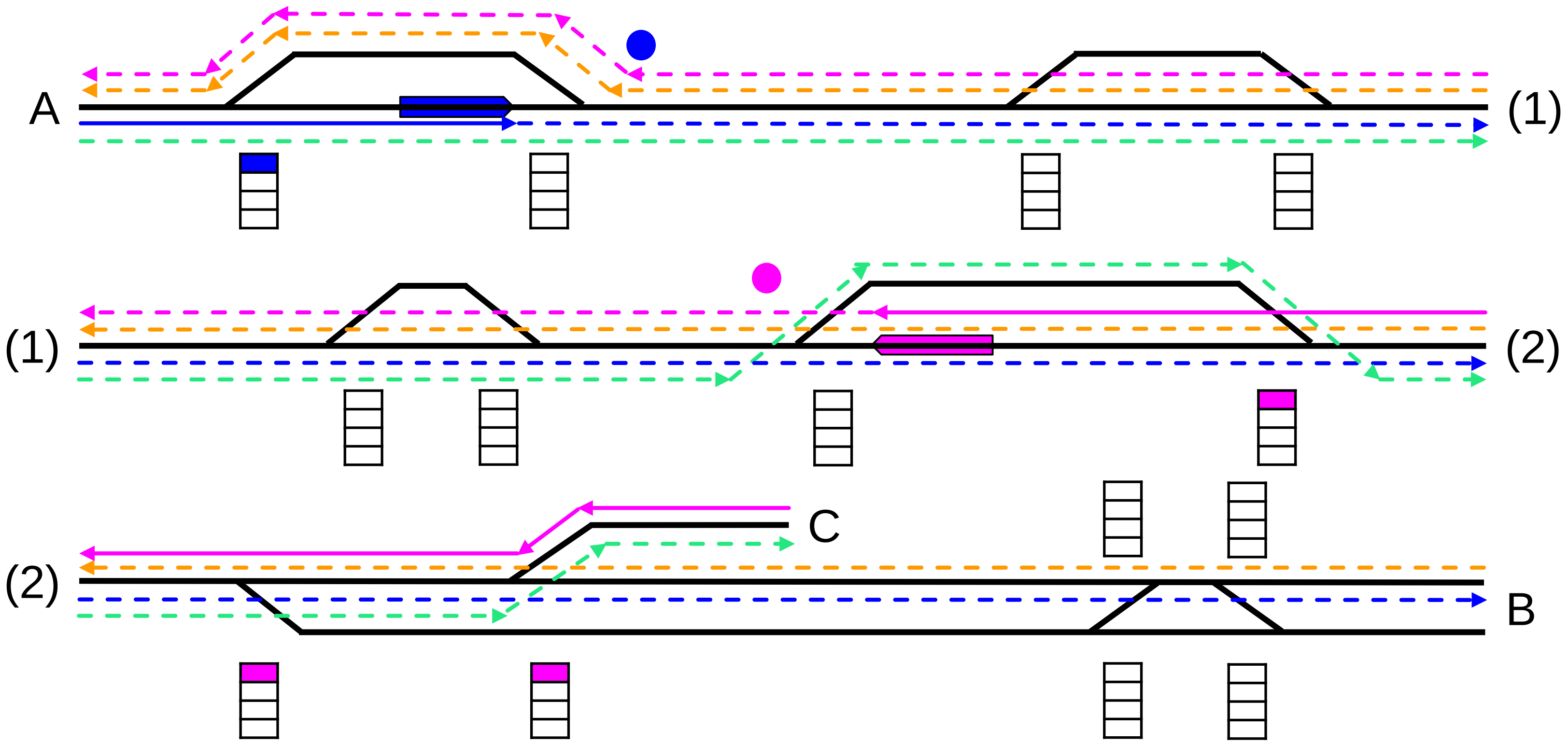
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



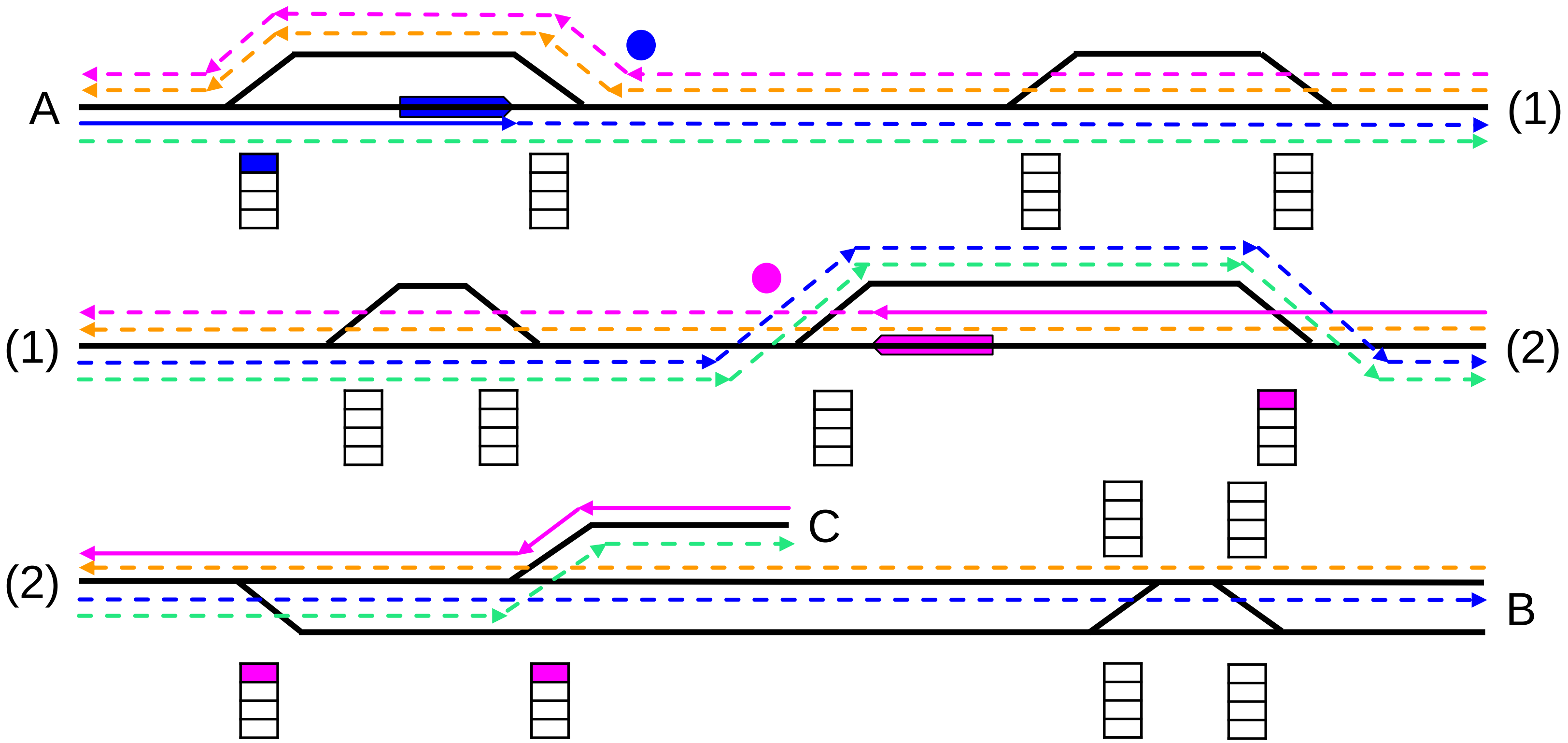
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



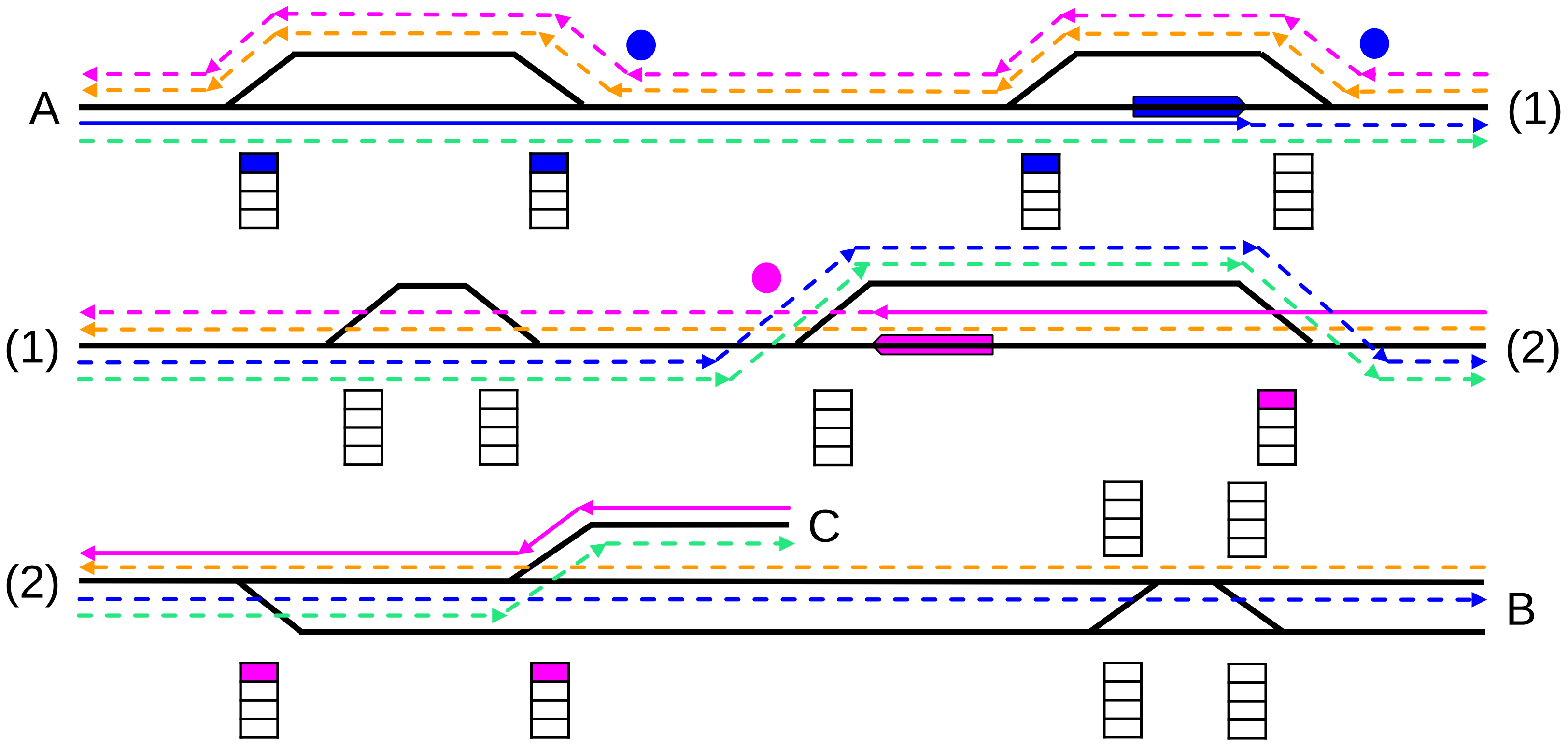
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



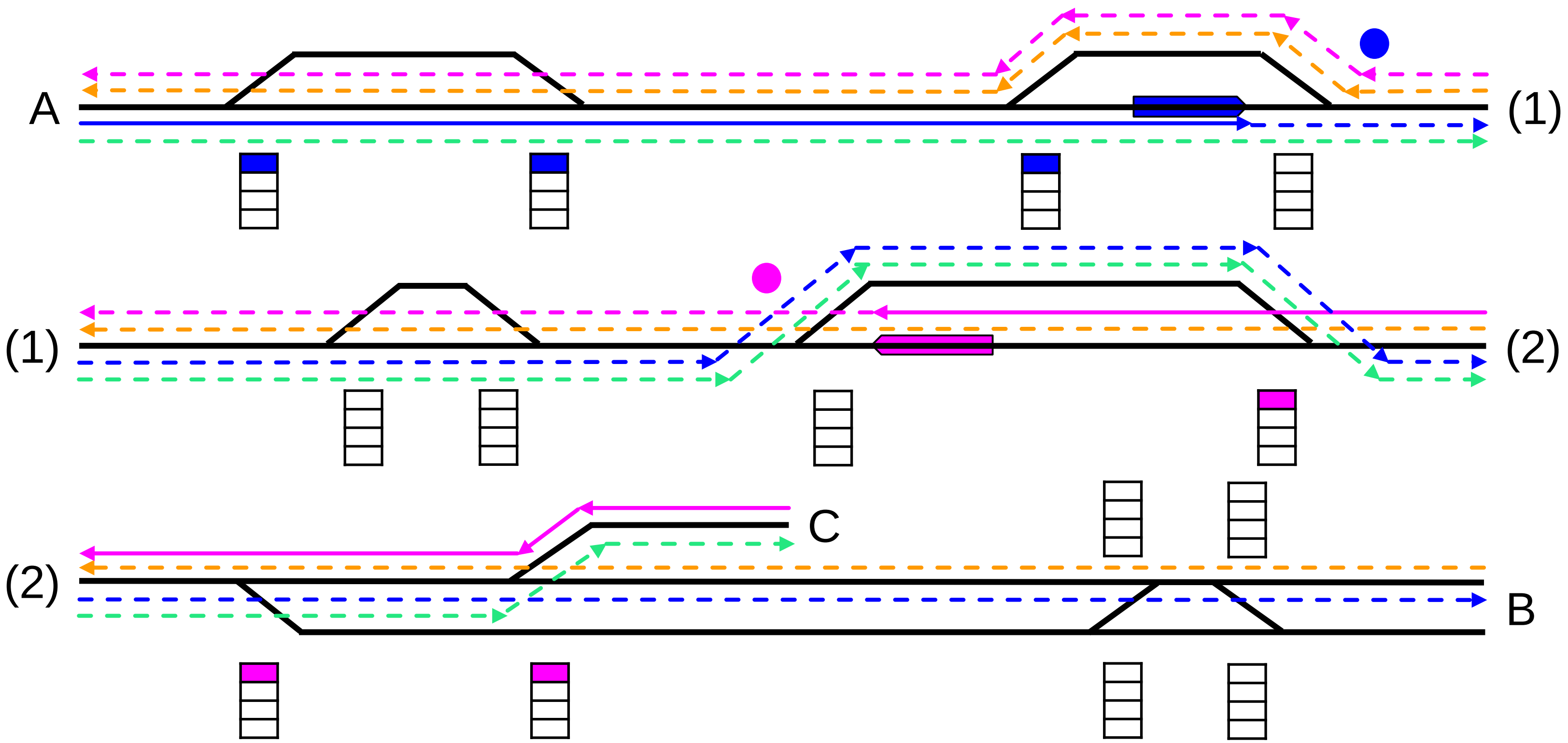
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



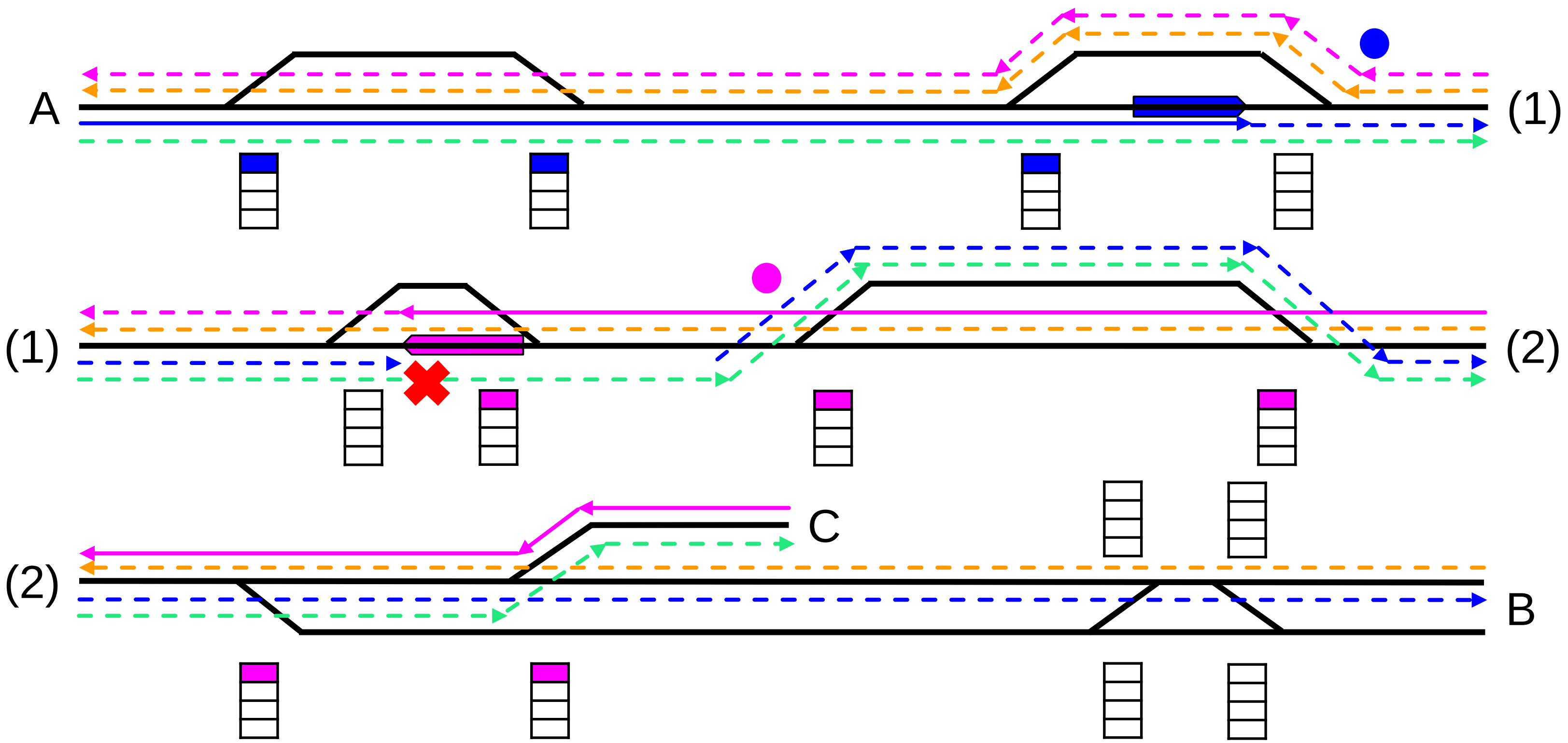
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



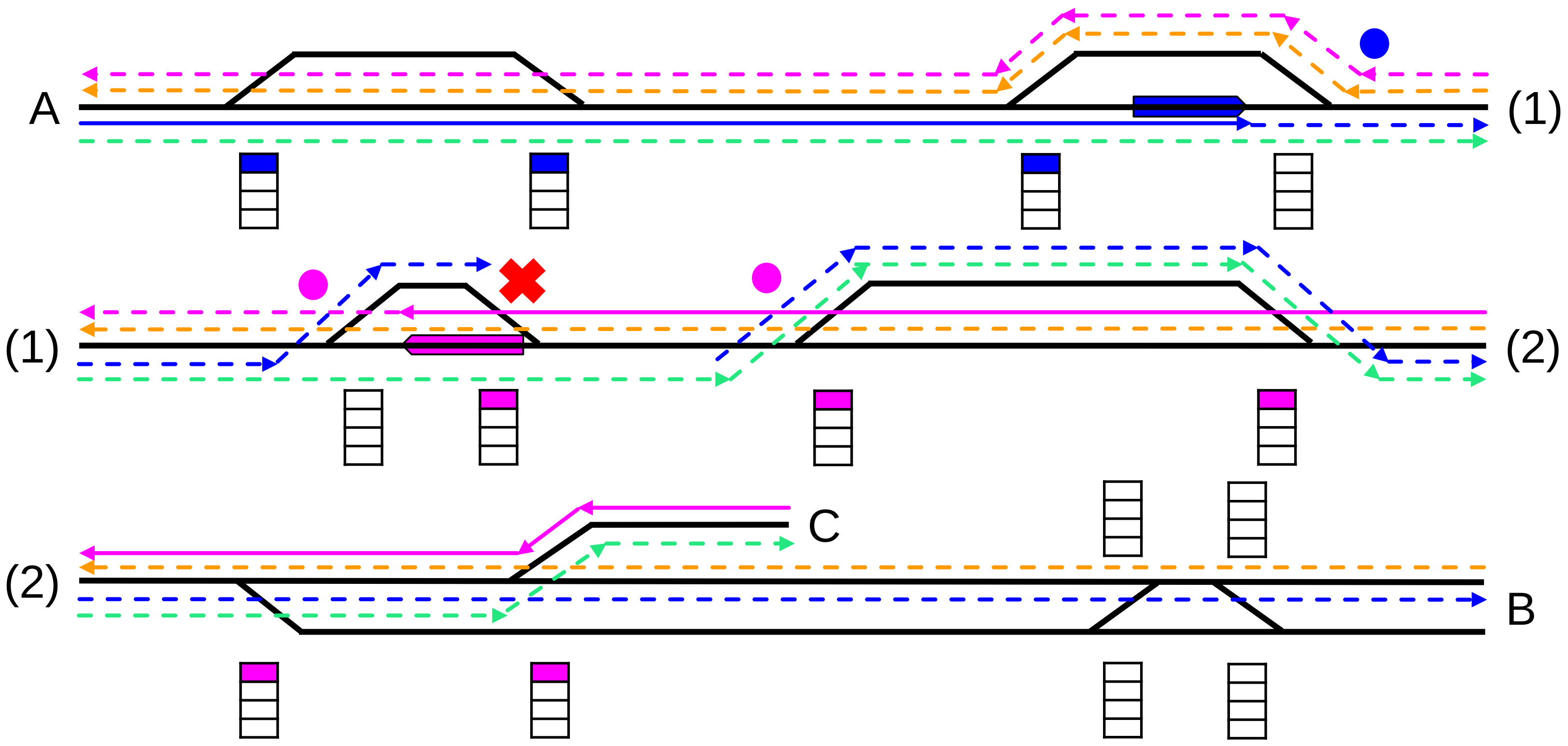
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



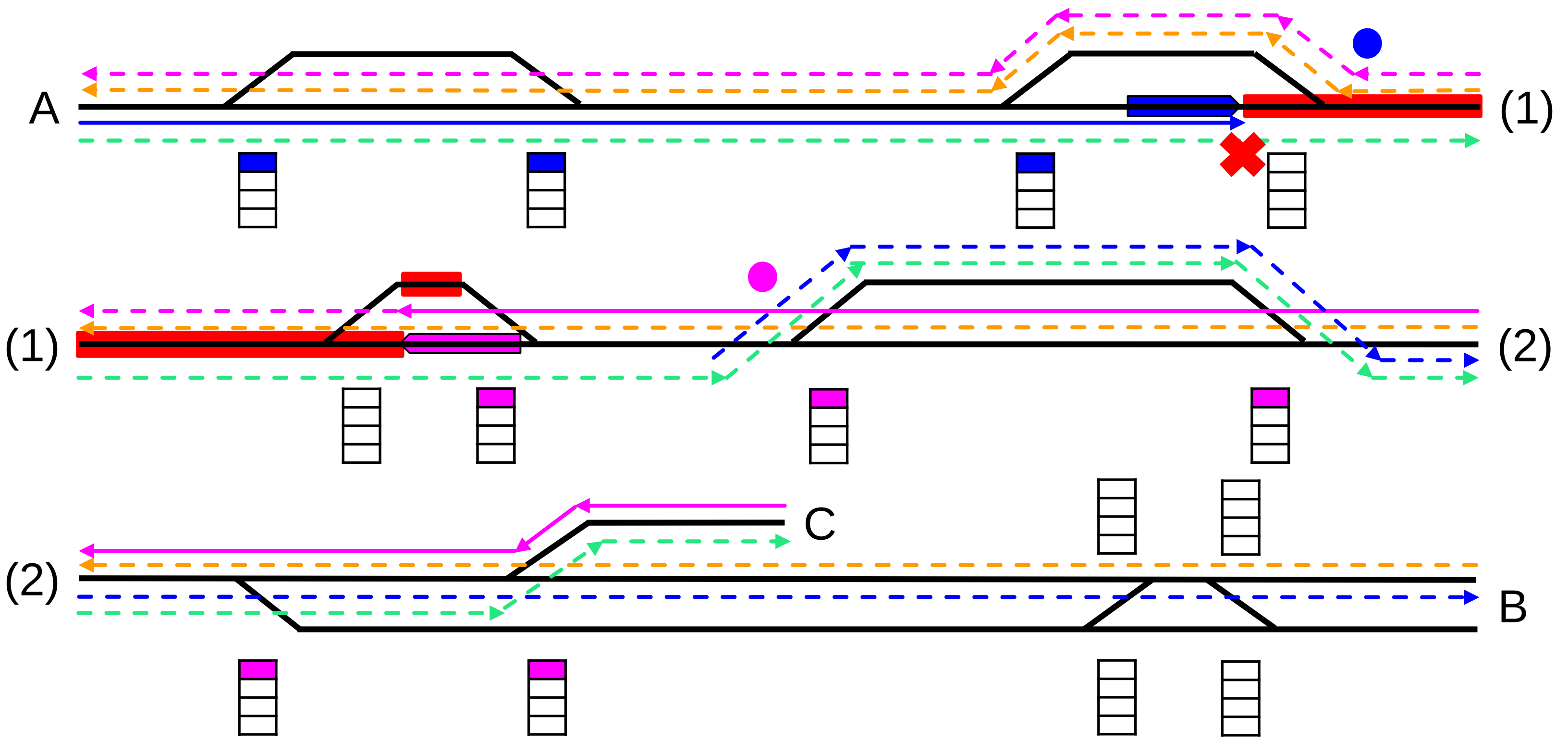
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



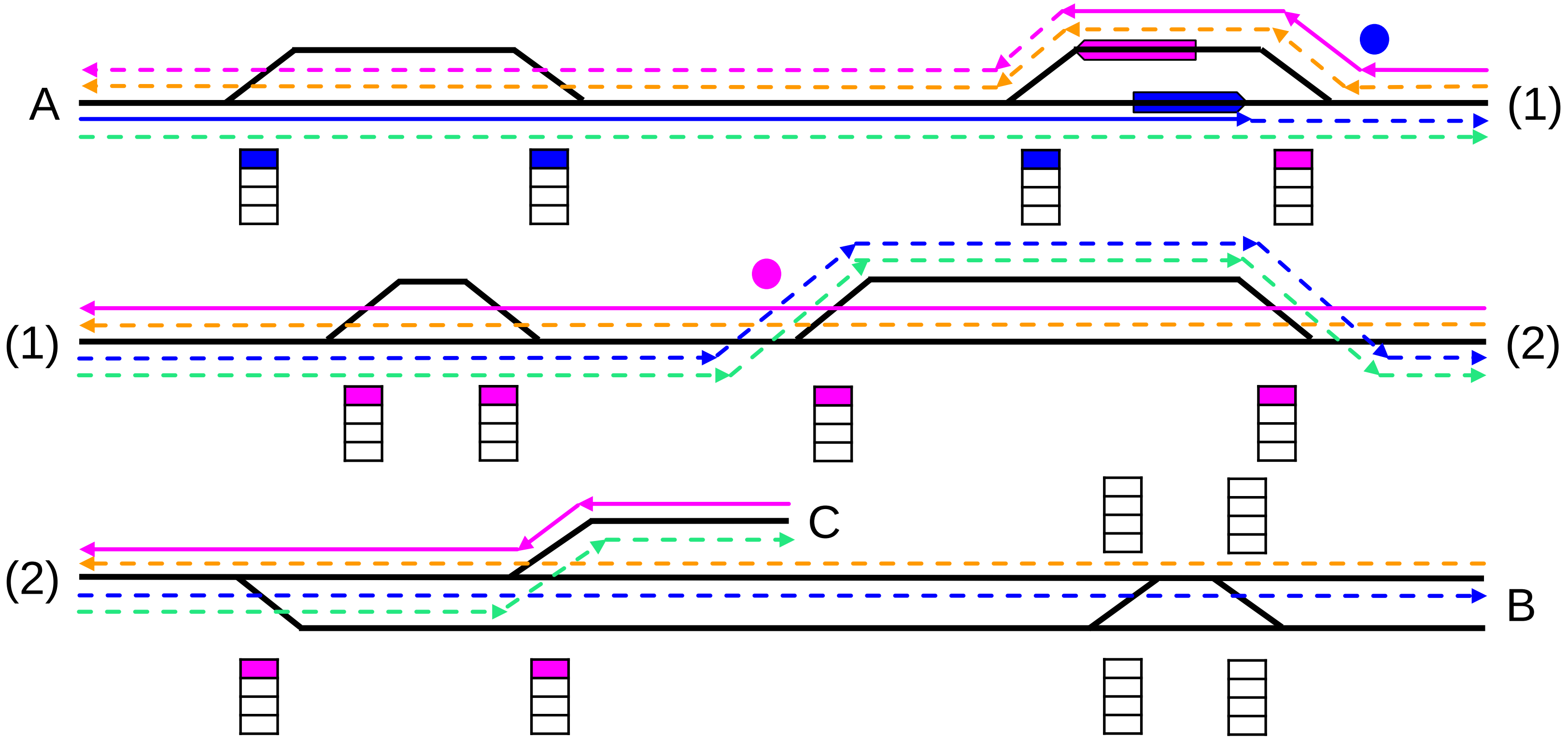
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



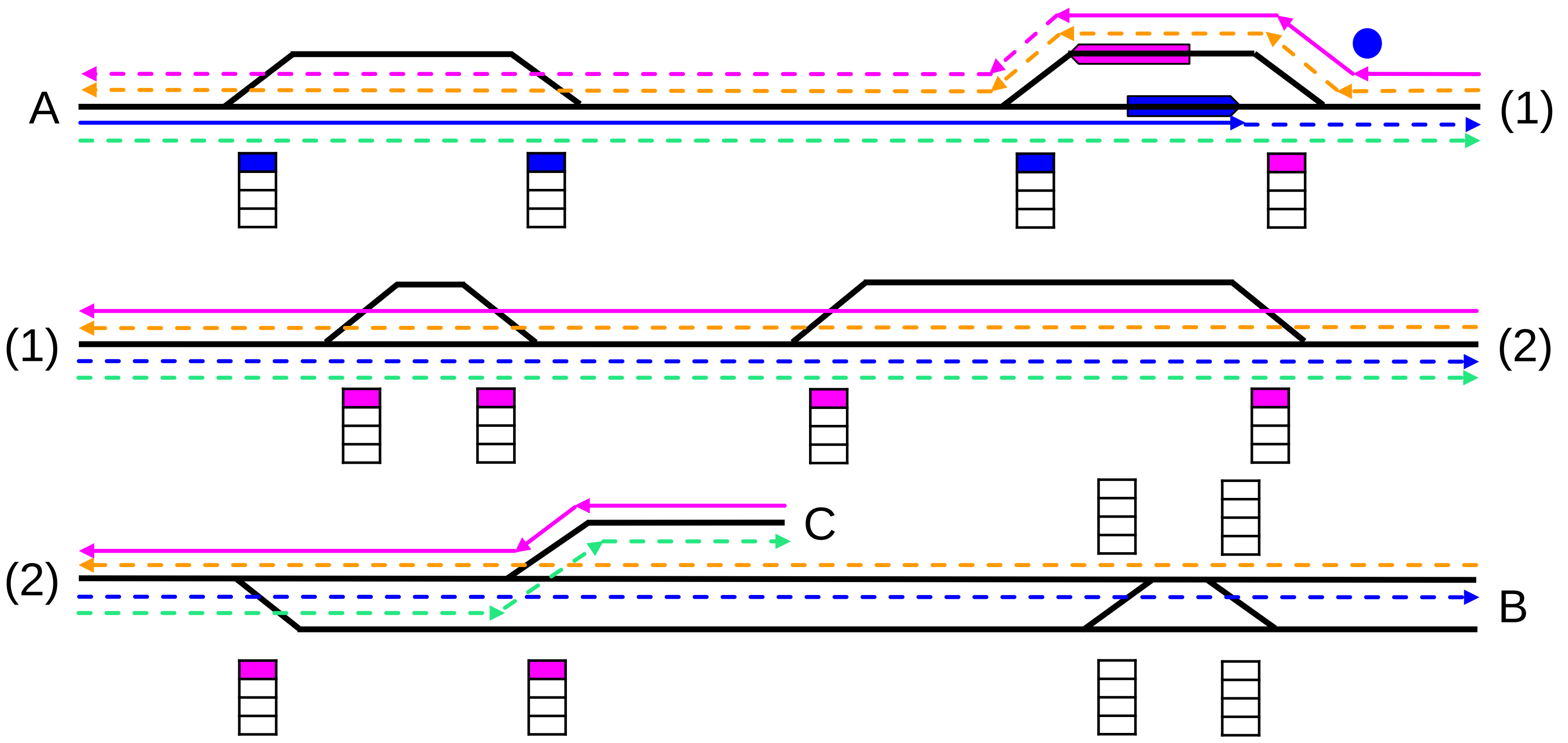
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



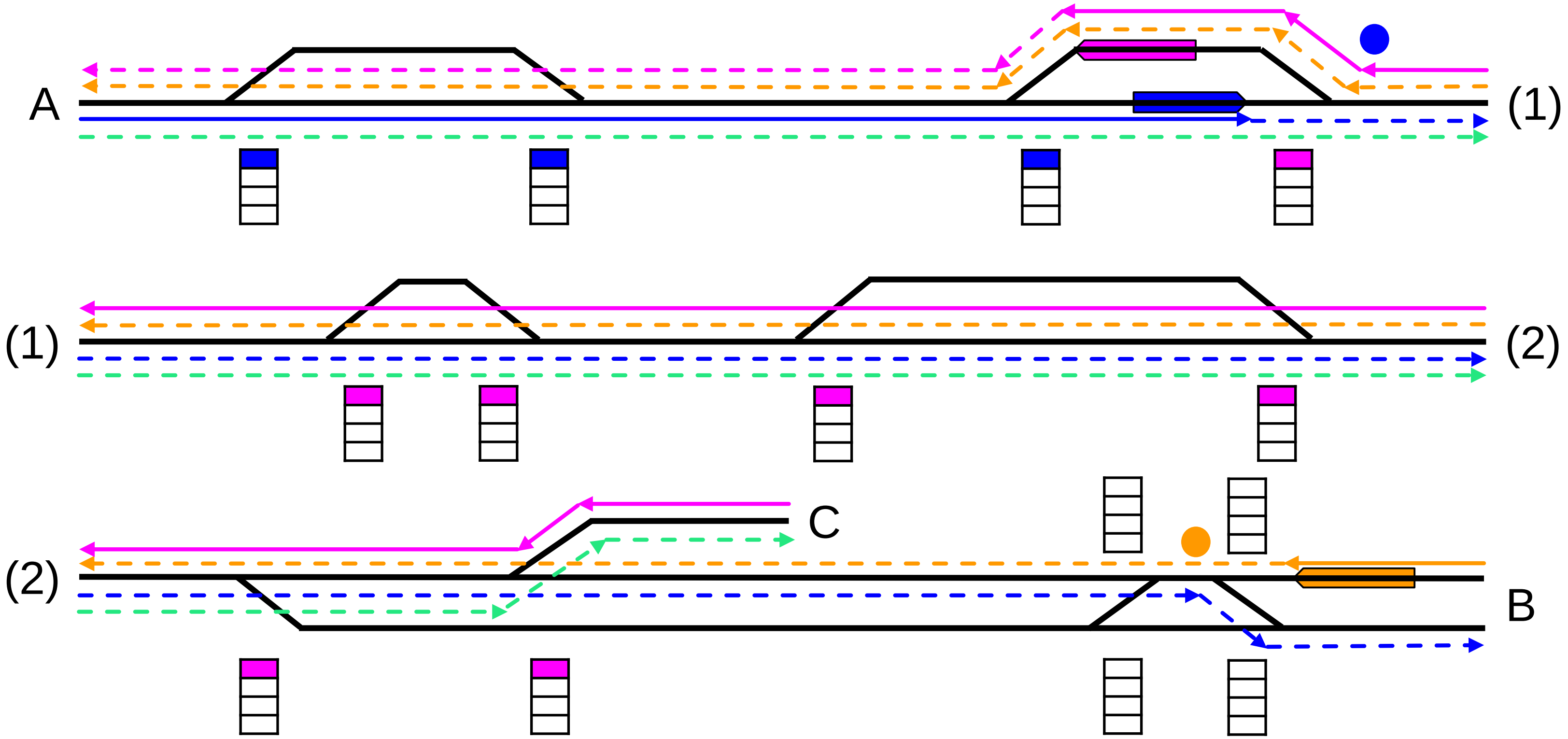
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



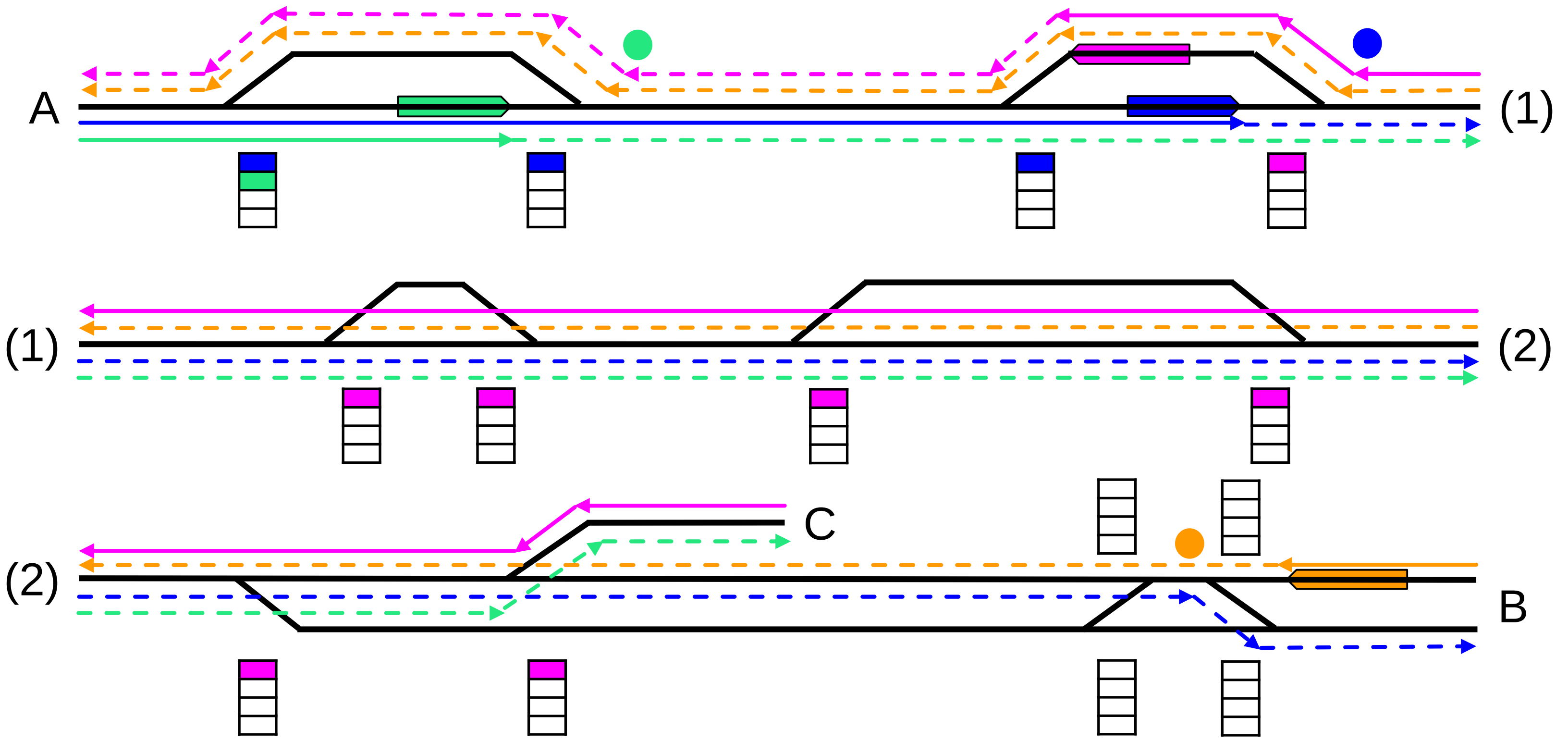
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



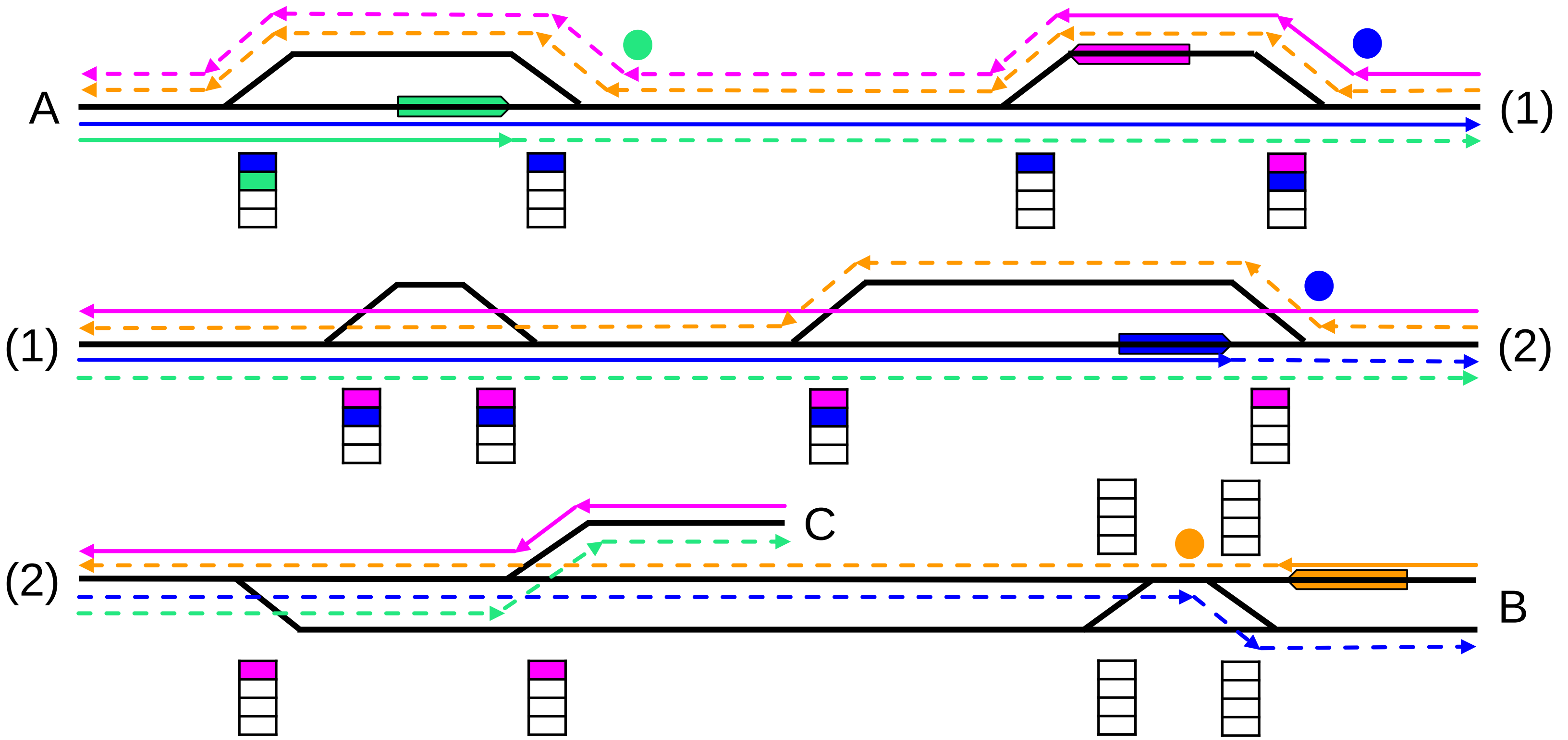
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



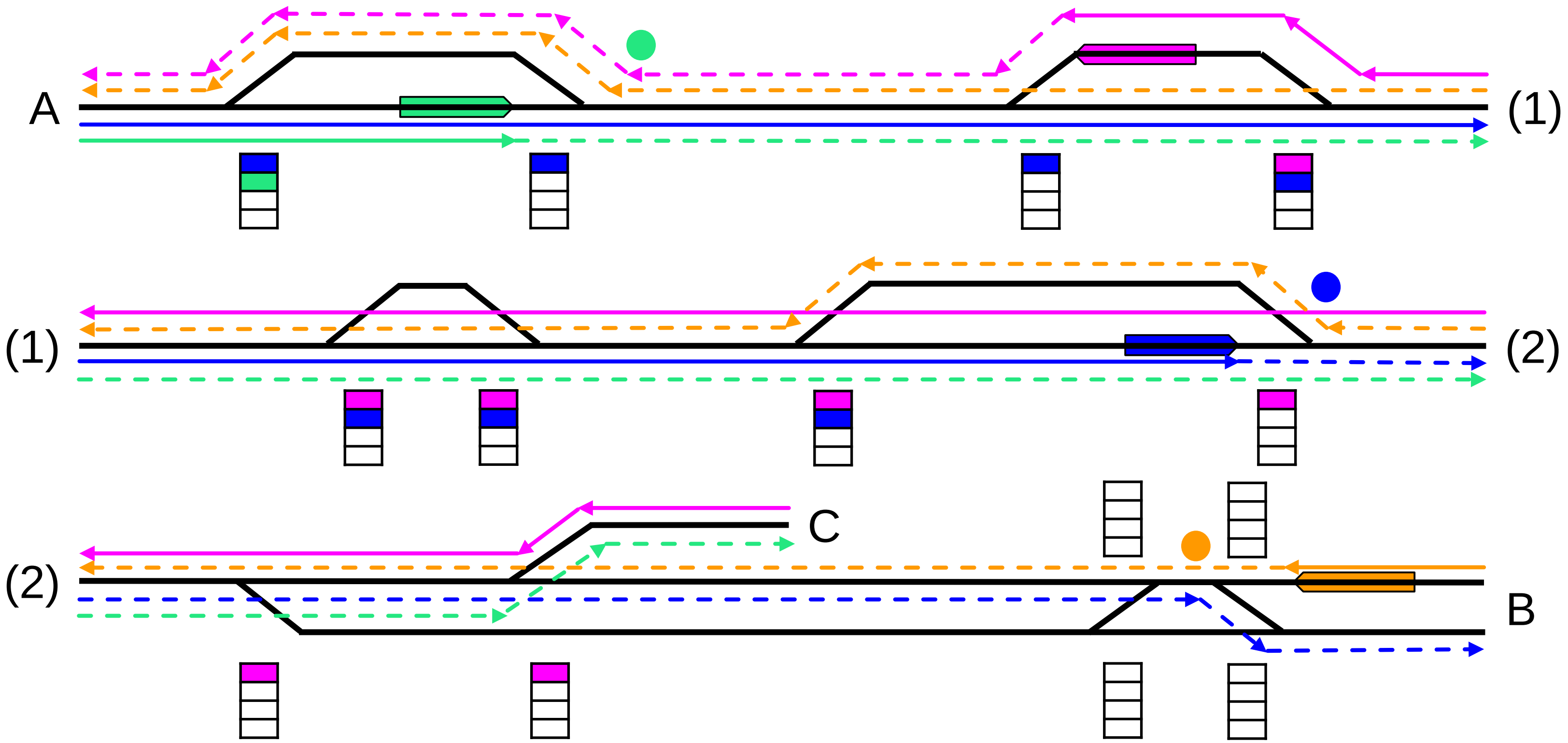
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



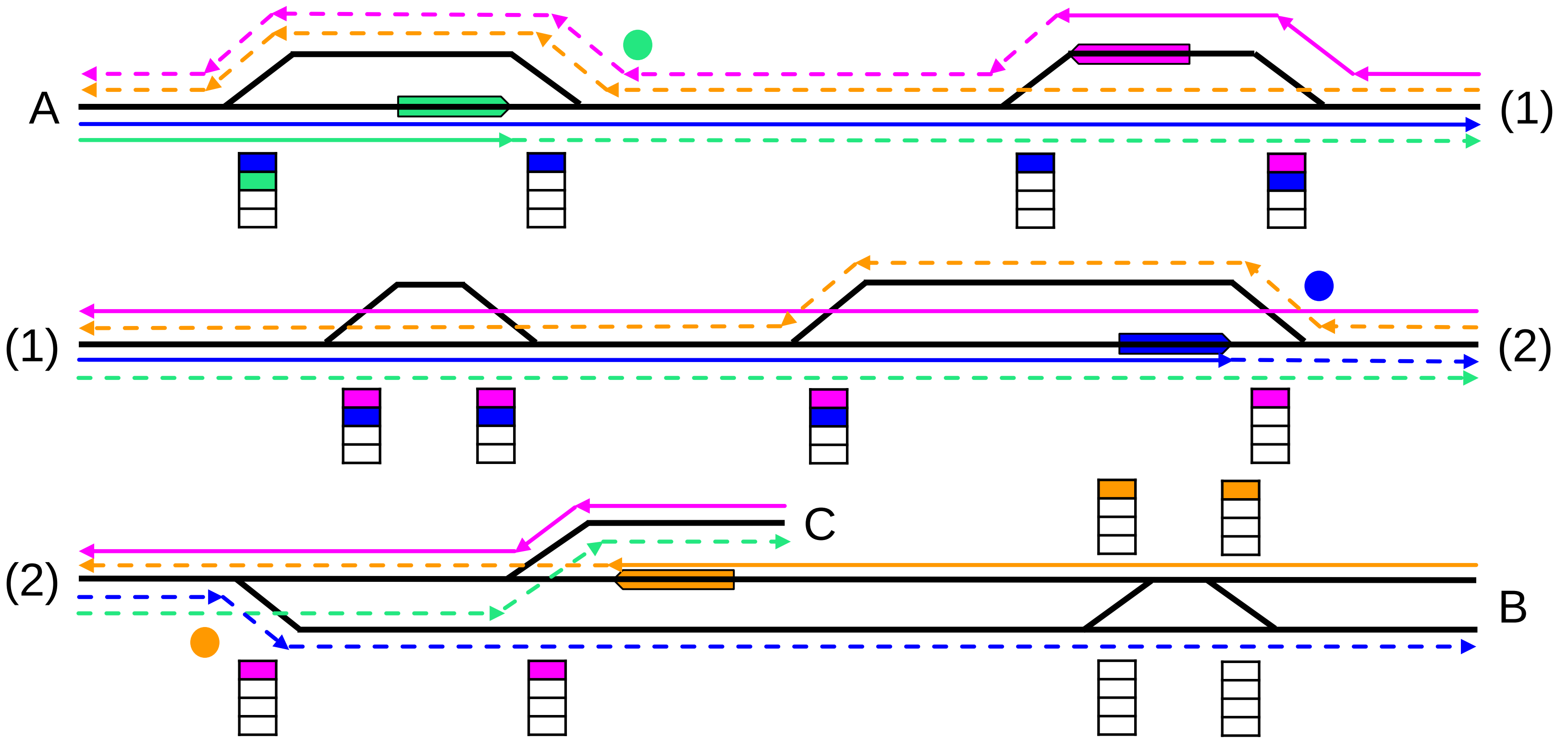
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



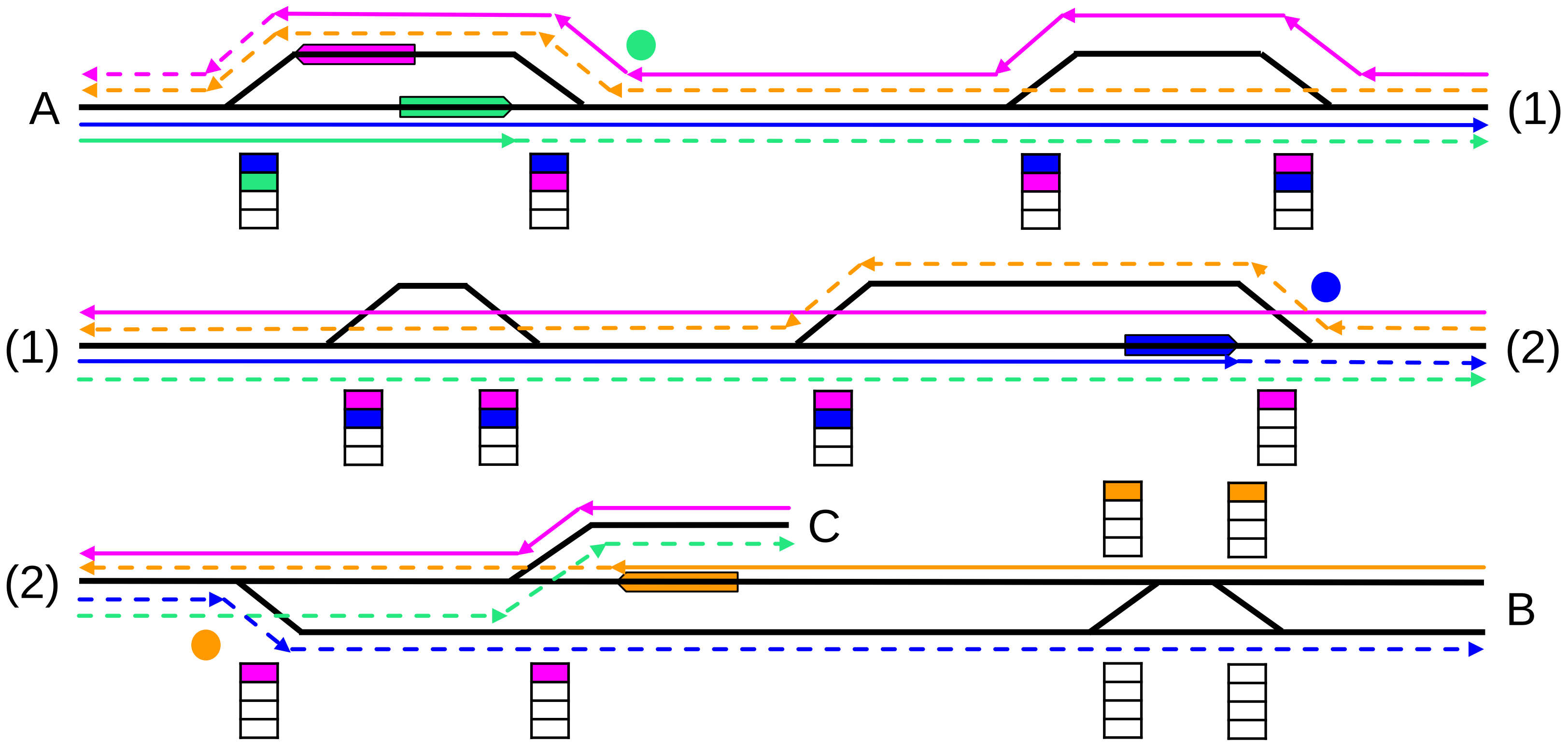
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



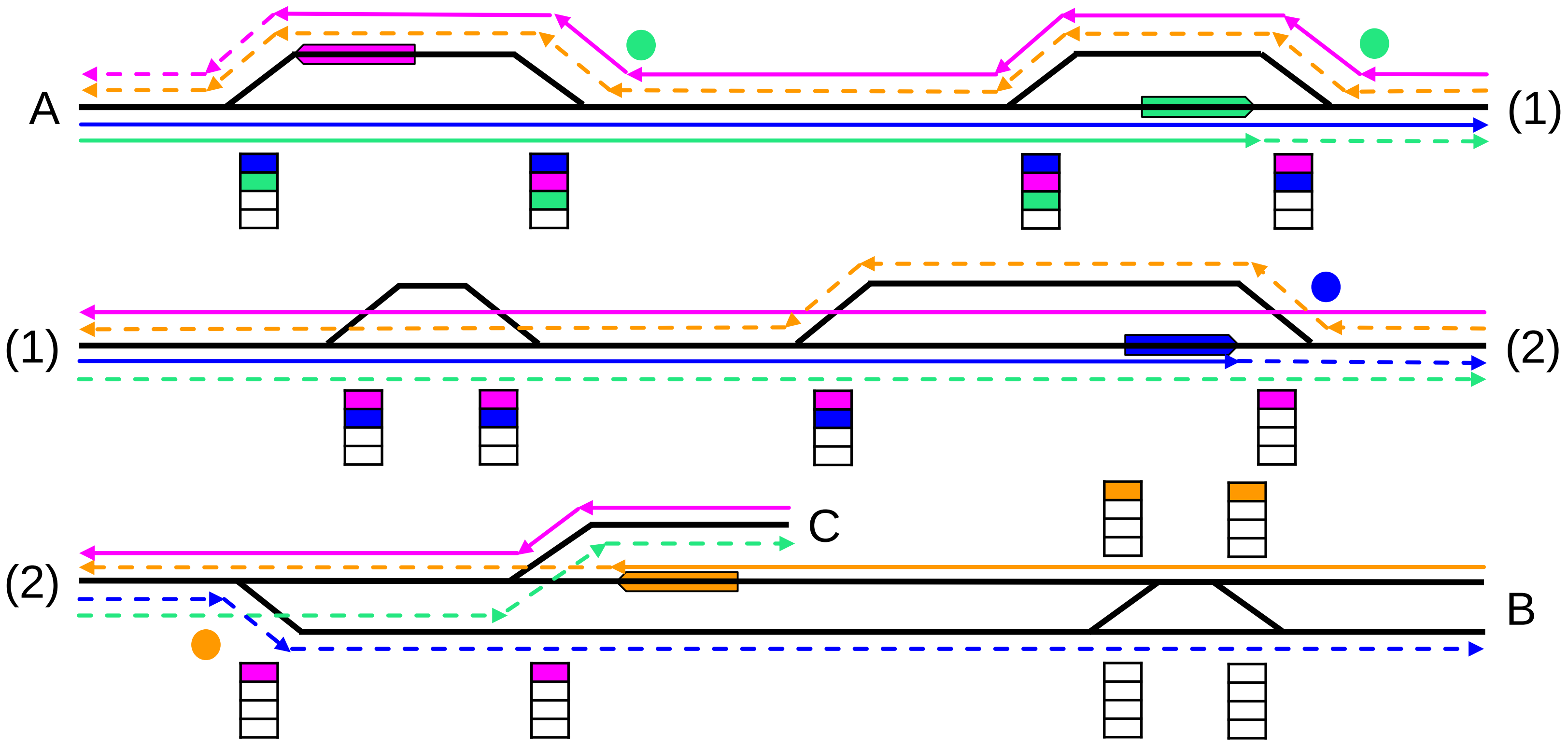
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



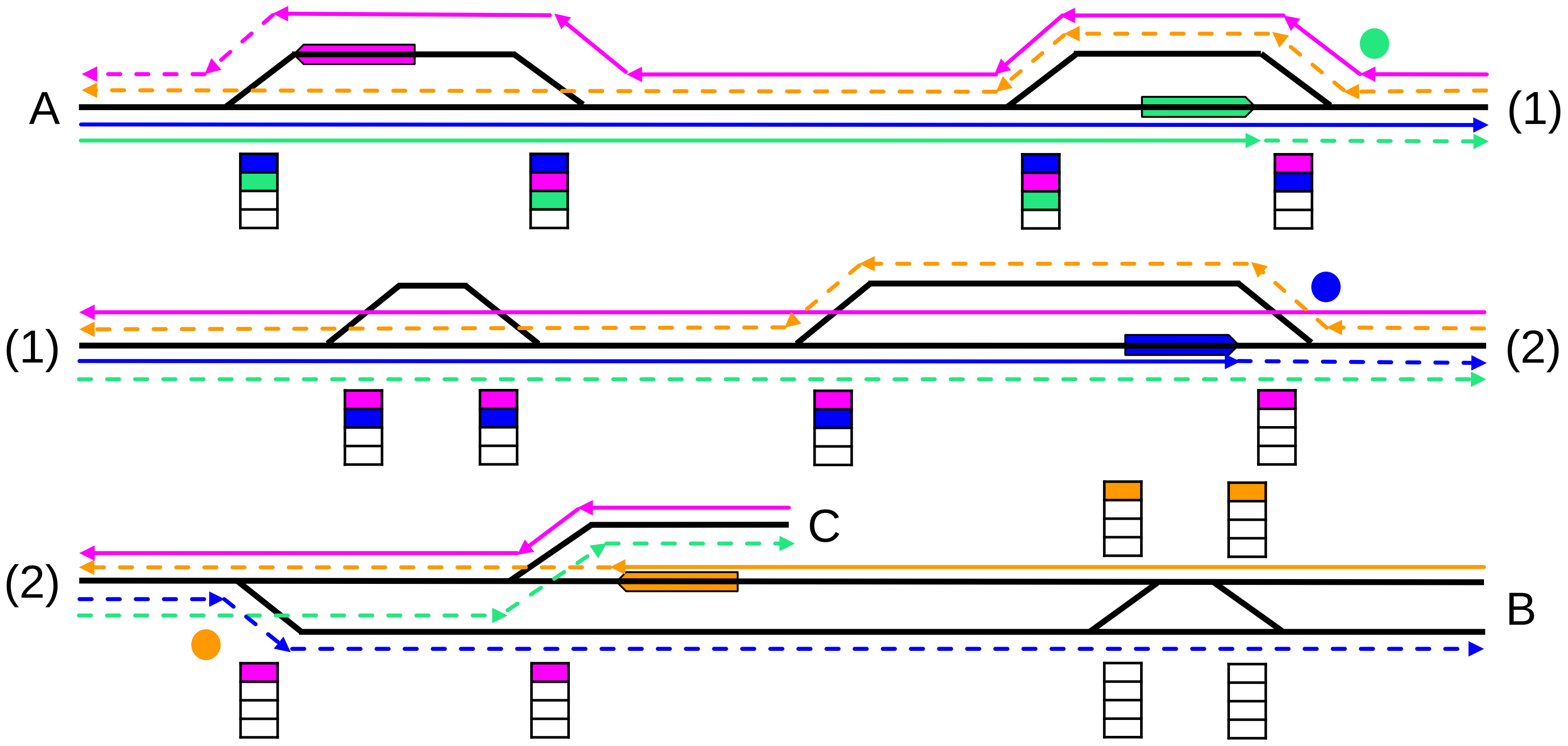
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



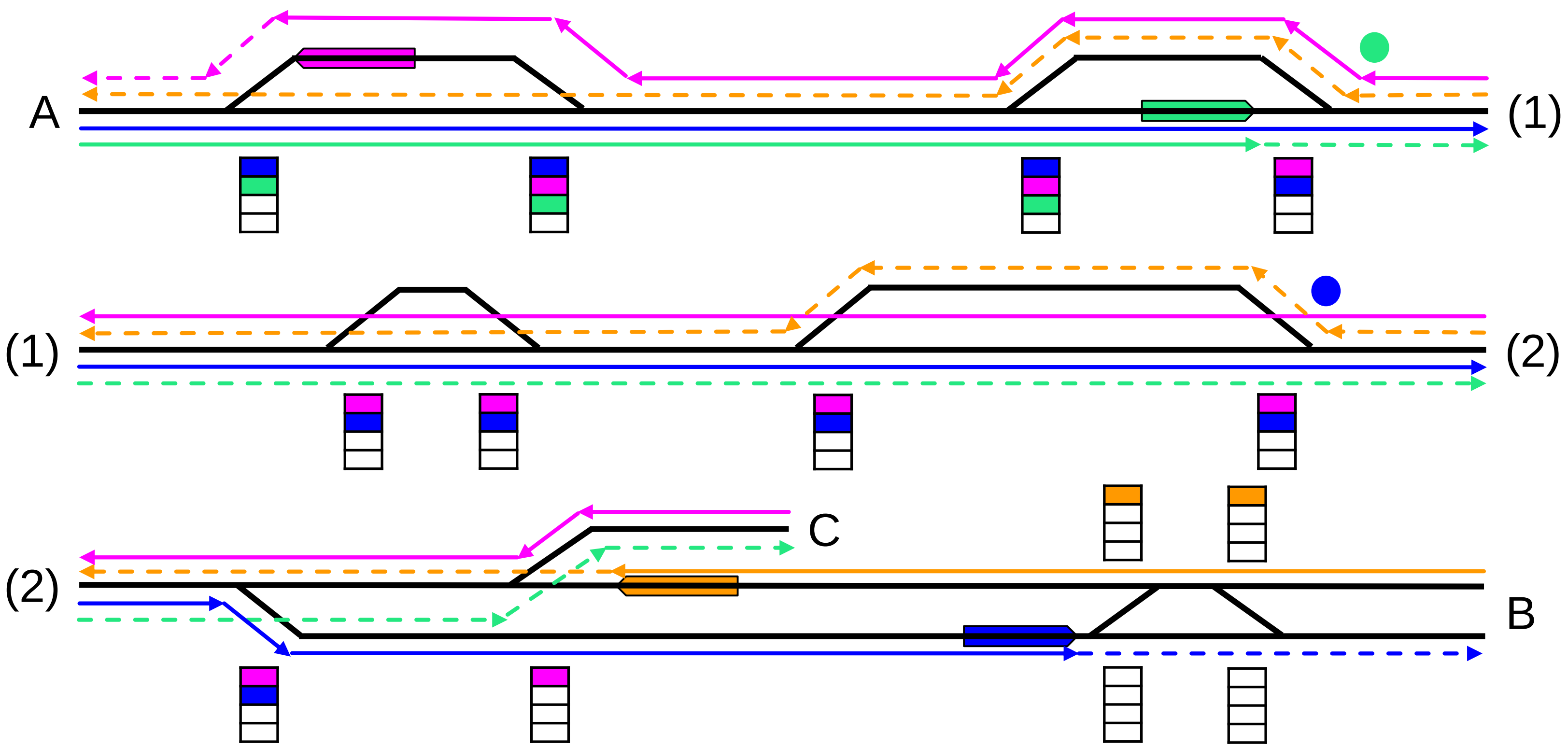
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



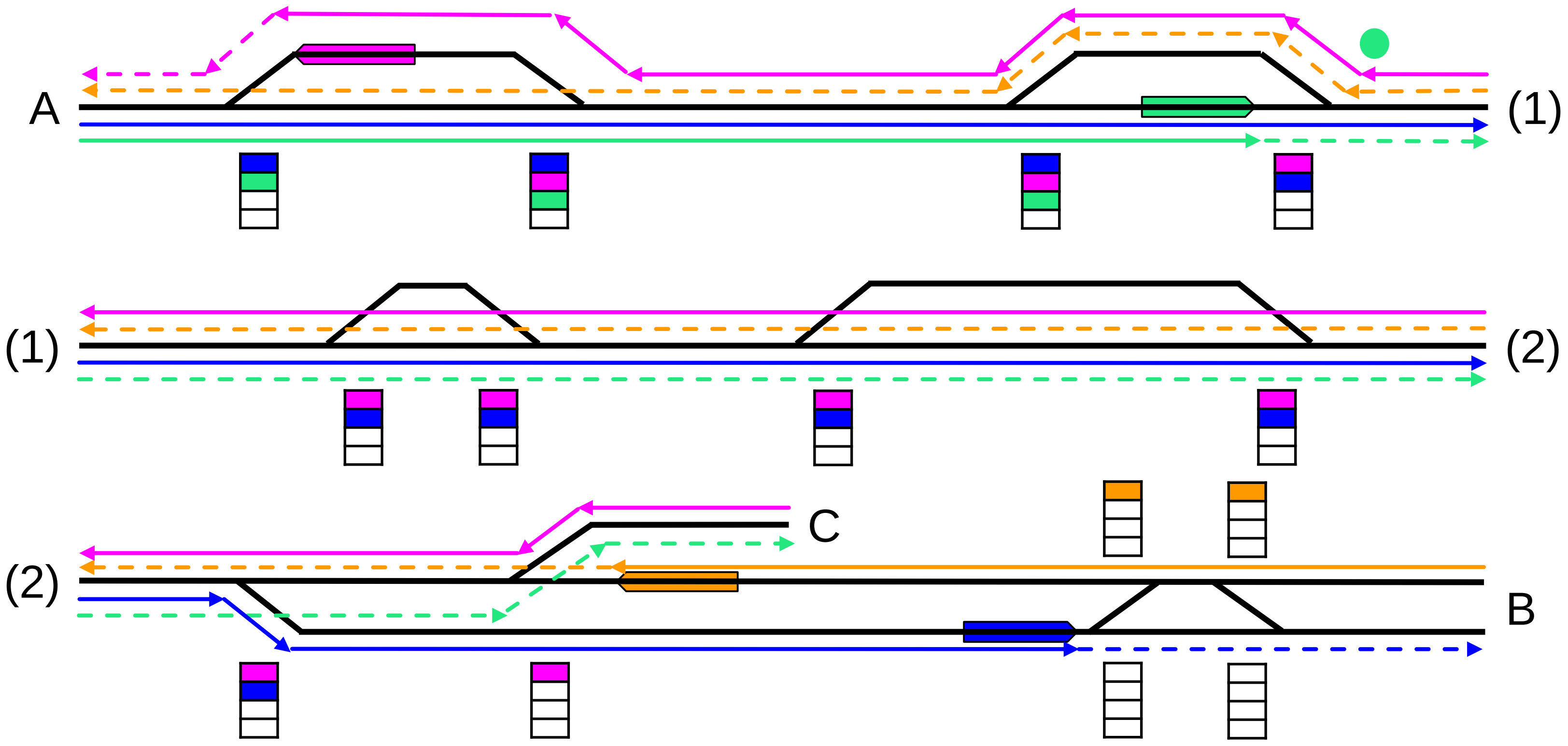
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



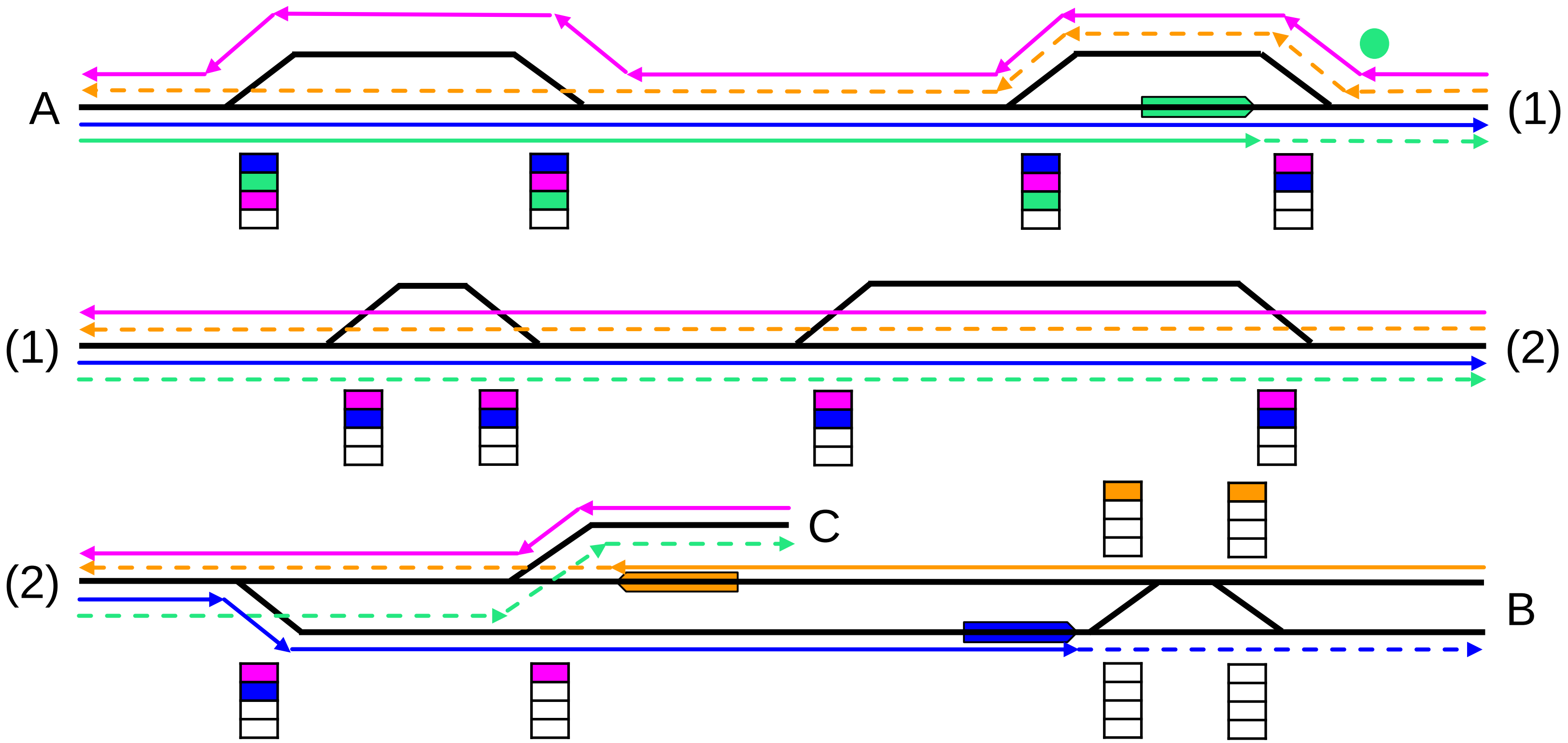
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



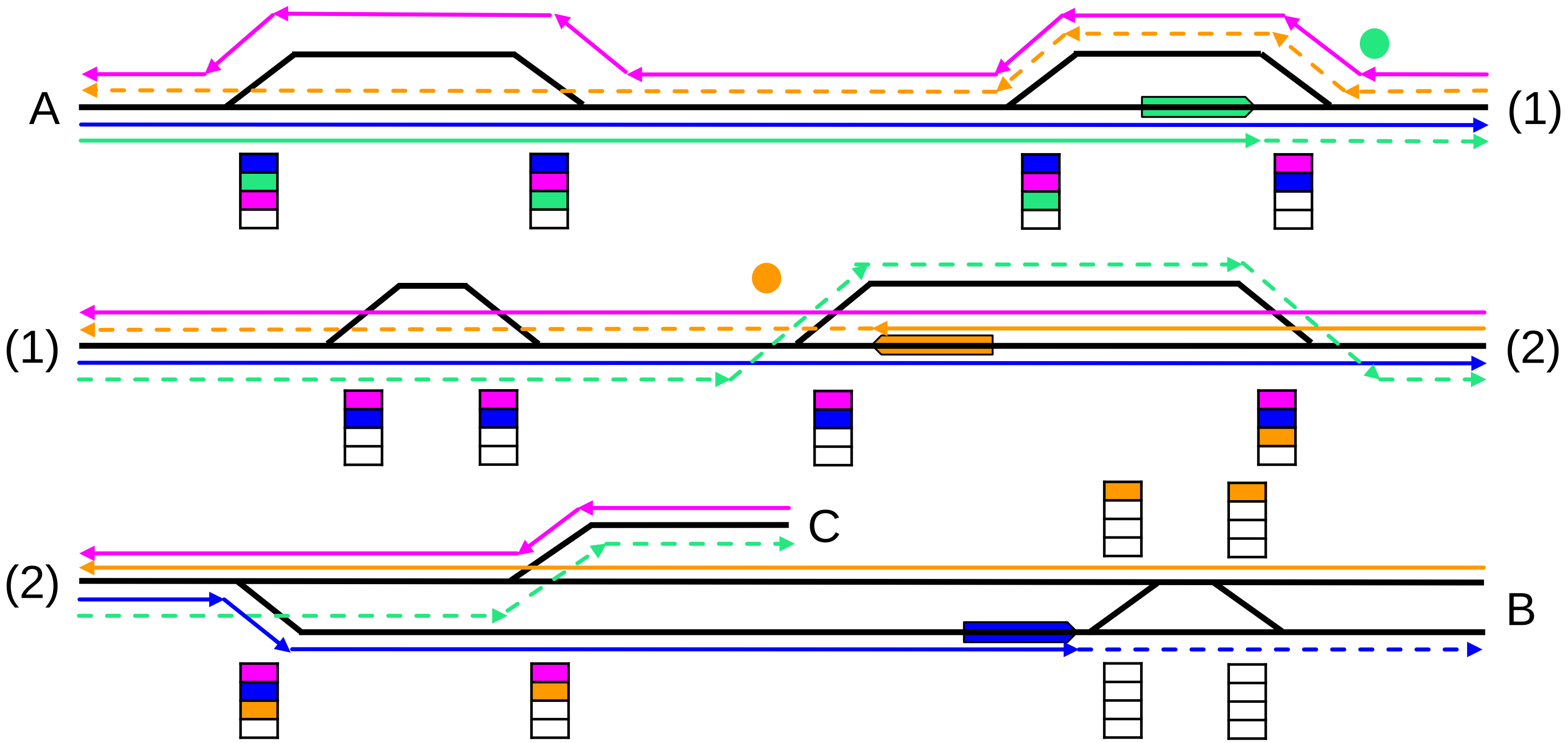
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



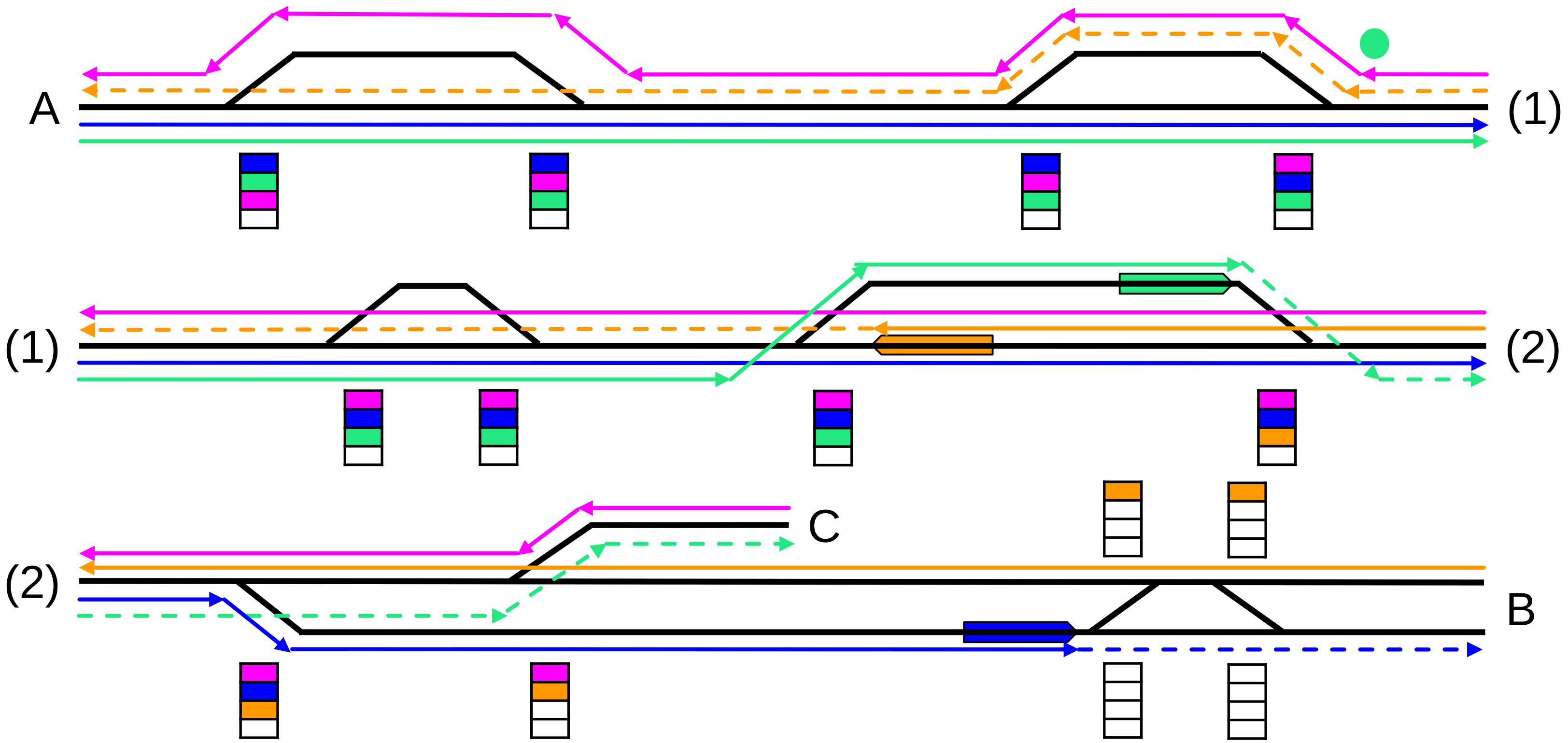
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



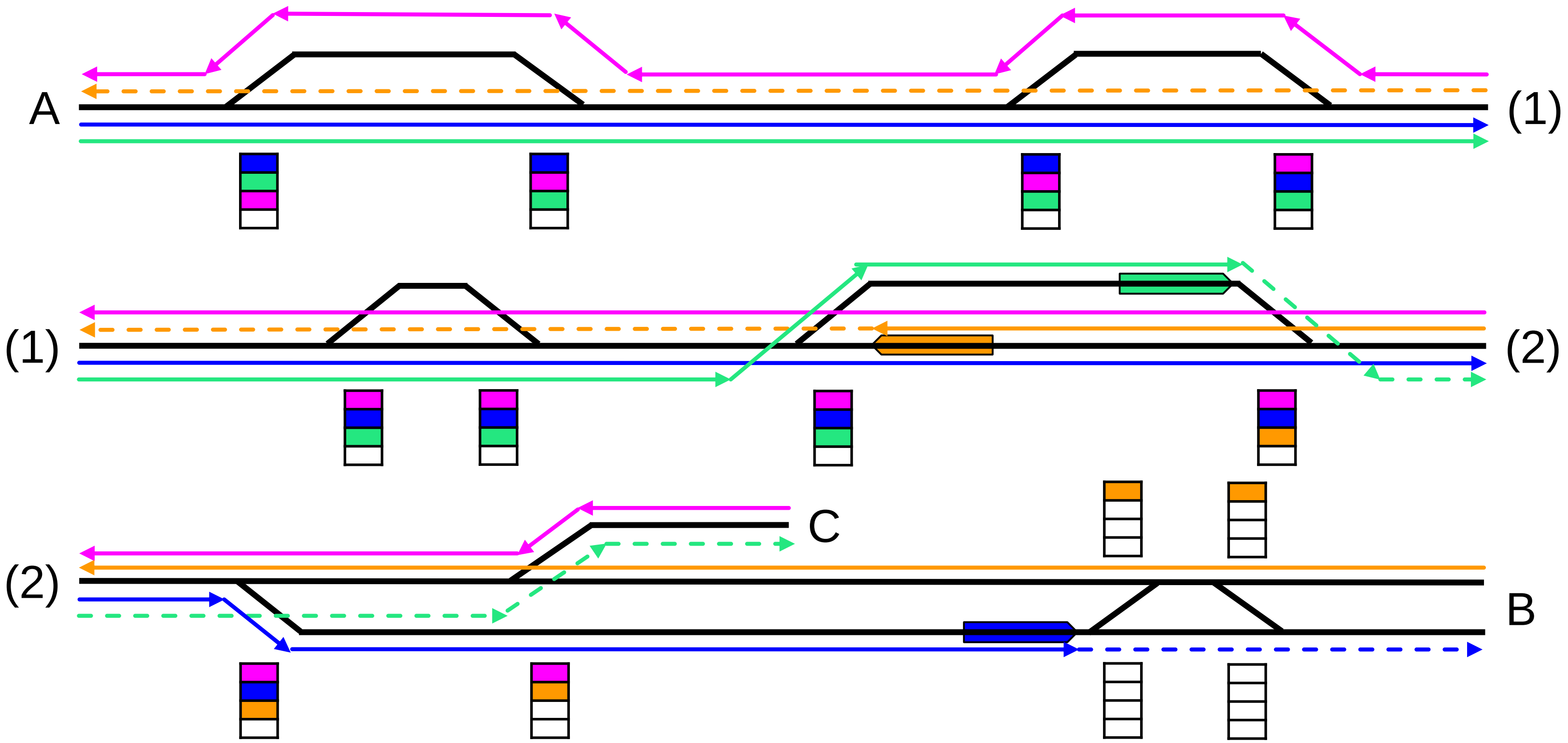
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



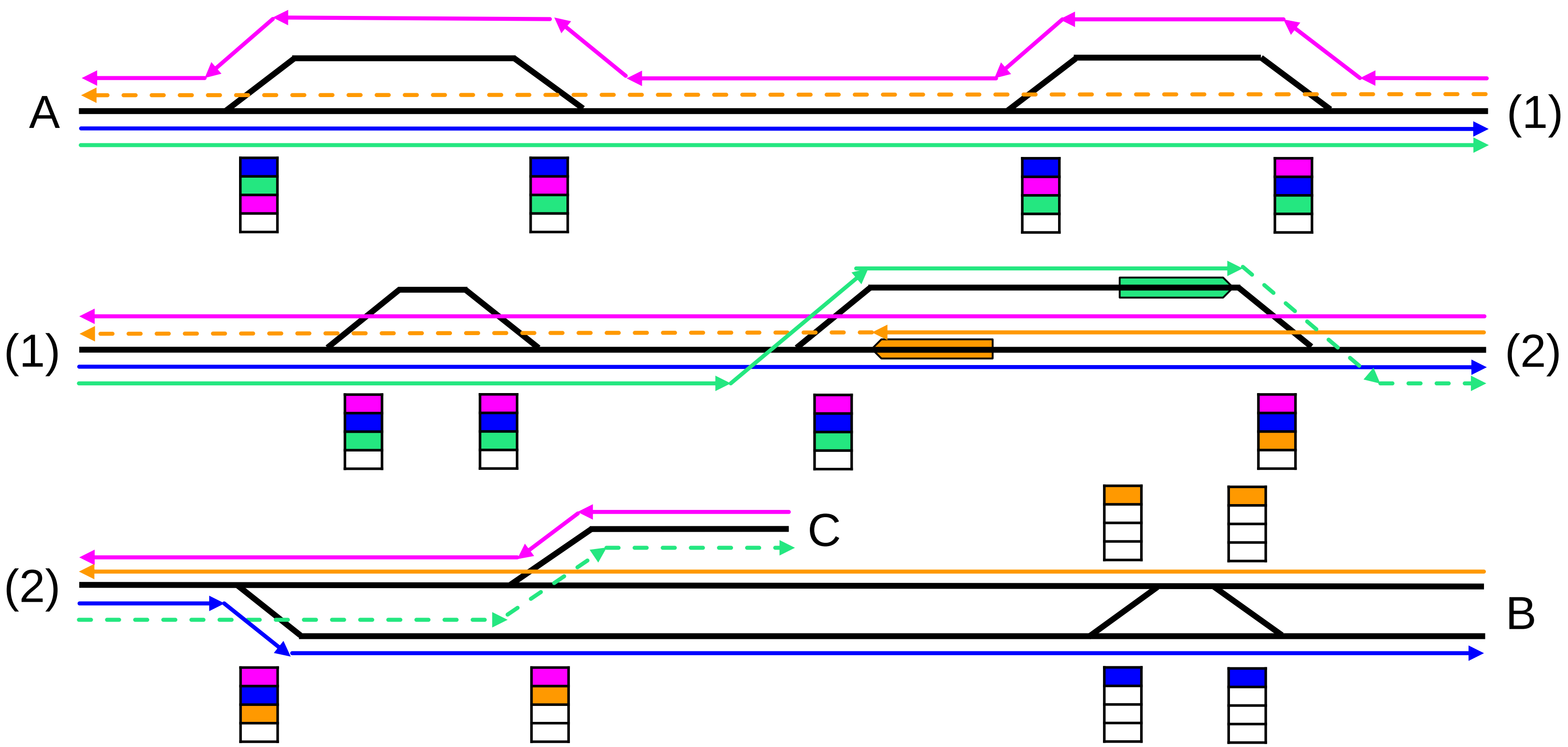
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



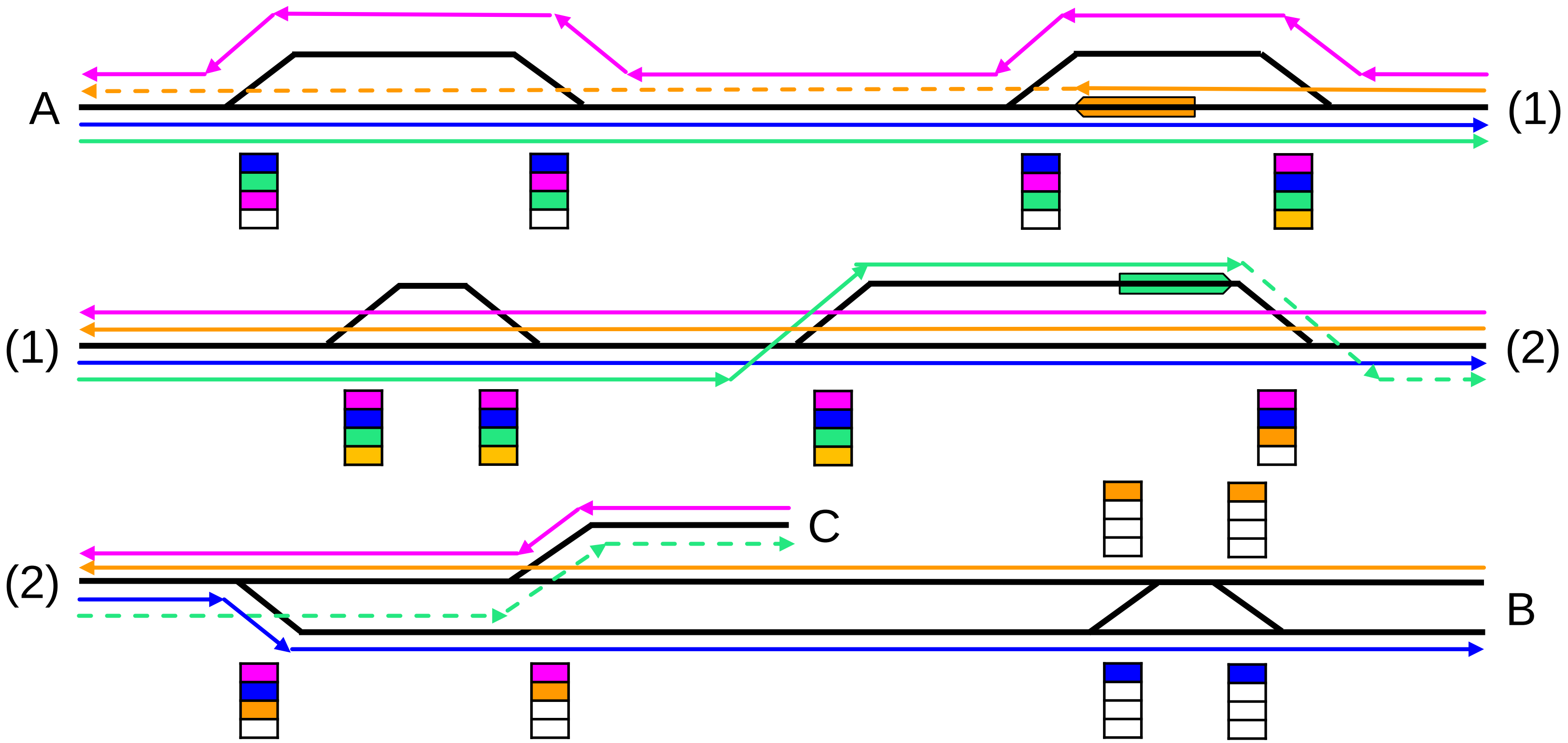
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



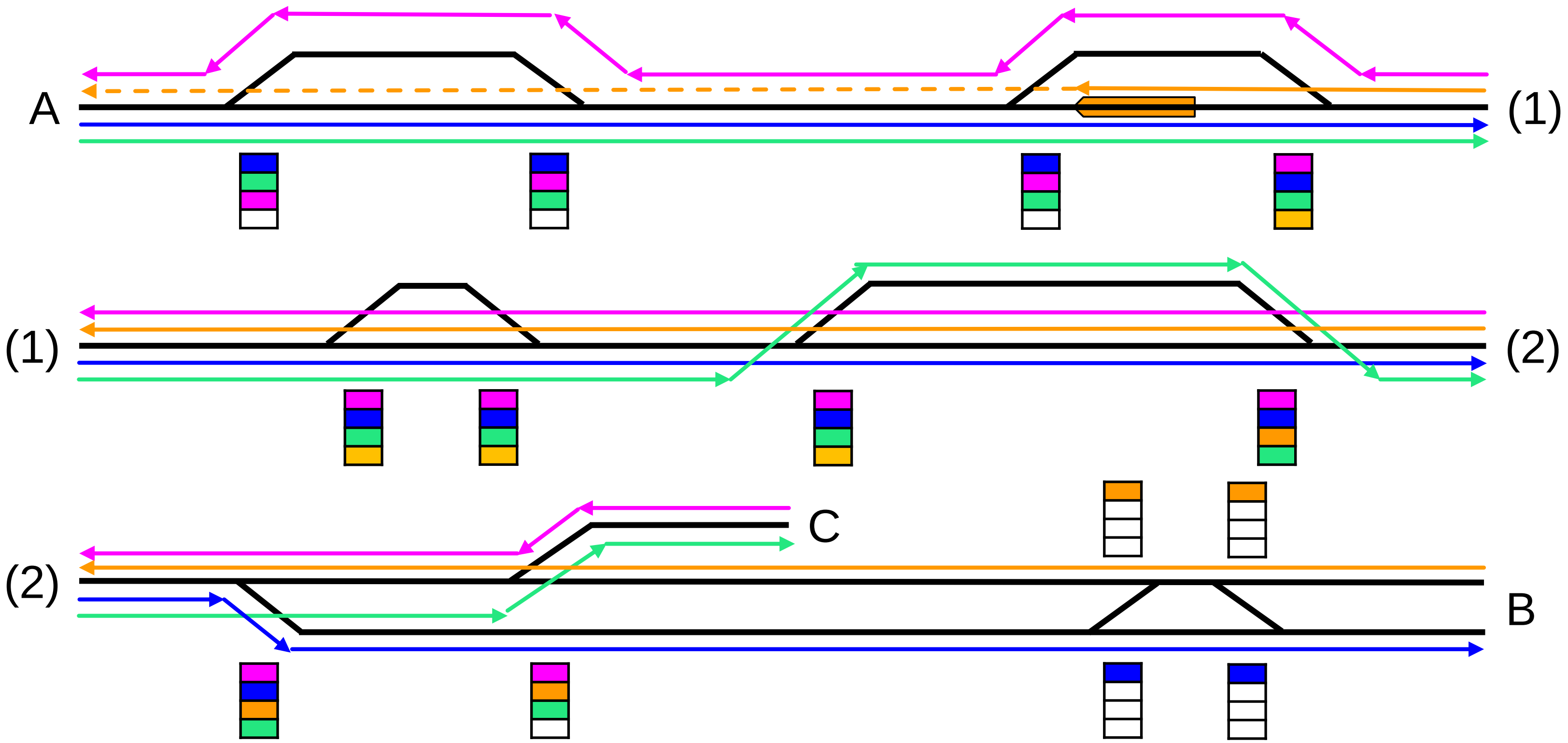
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



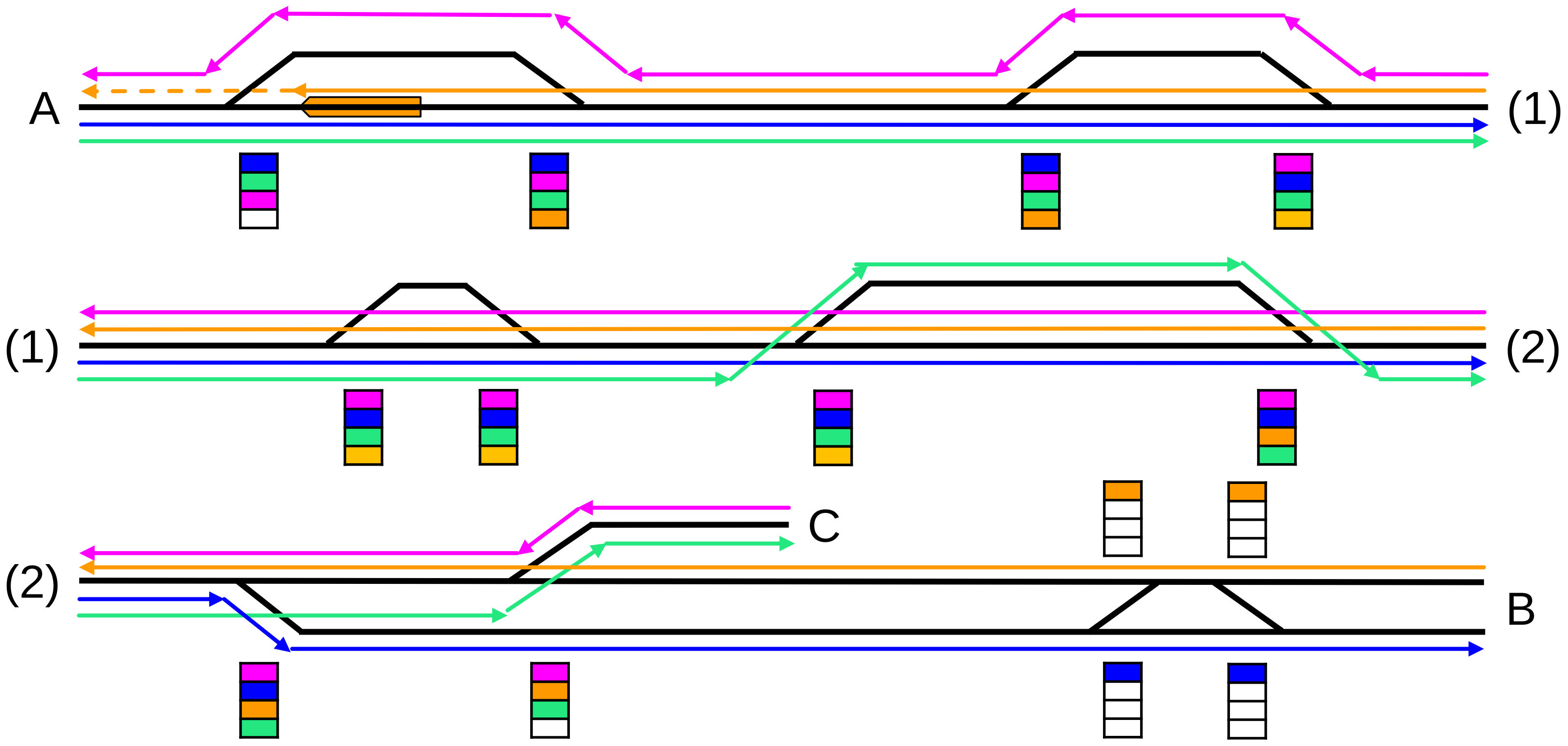
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



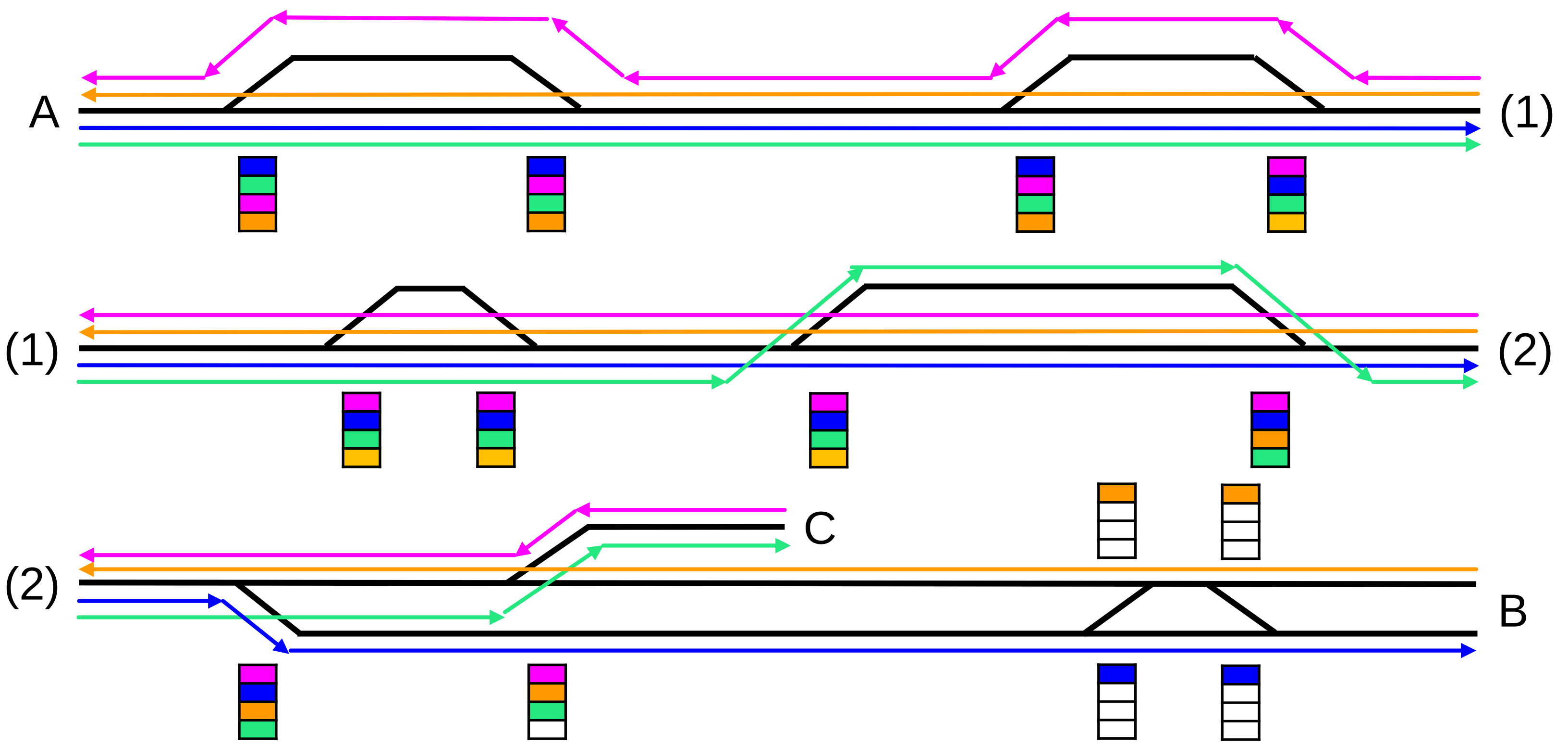
Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain

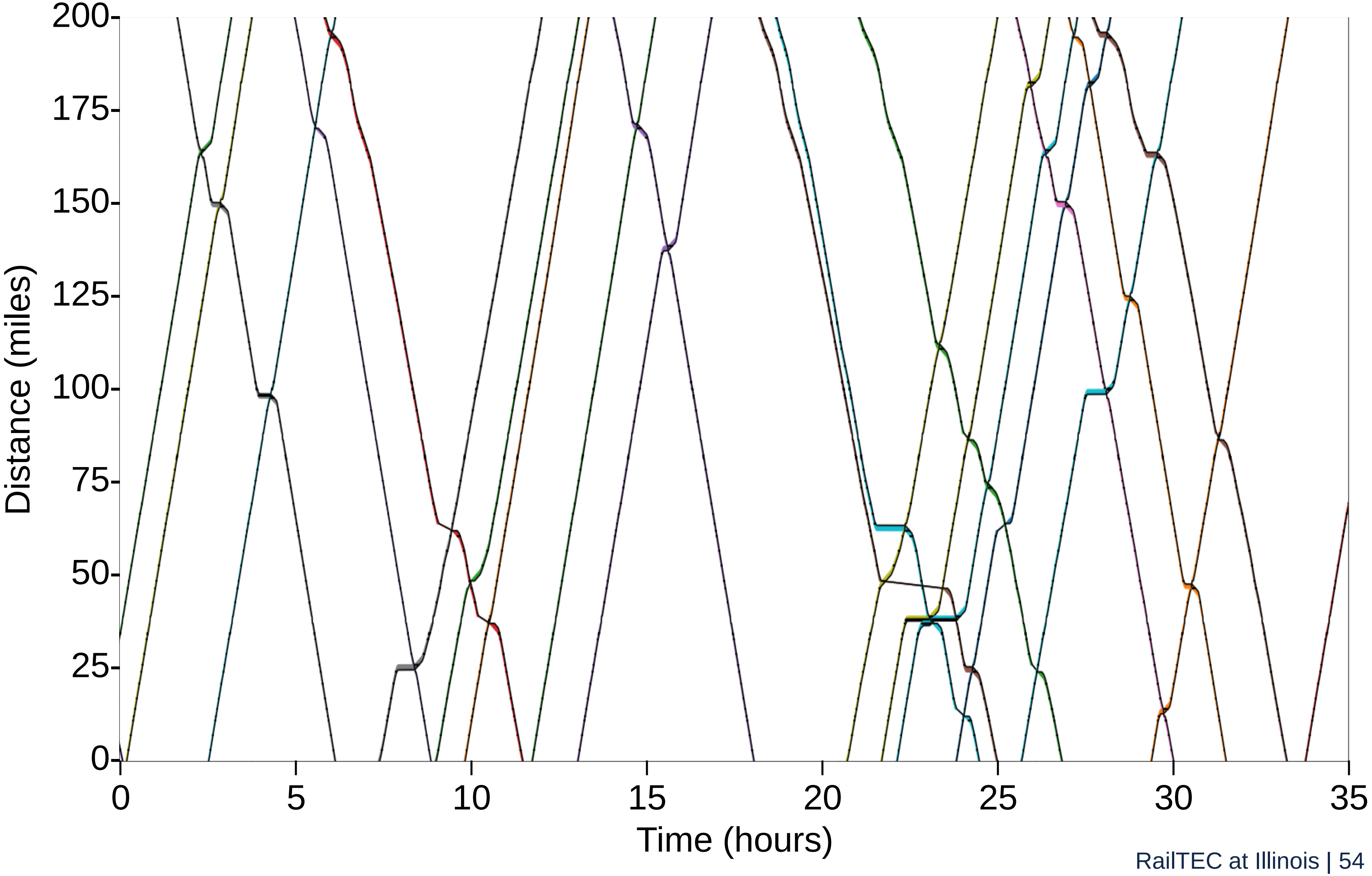


Dispatching Example

- Advance next train until reaching deadlock free position
- If this train gets stuck, rewind it until last known deadlock free position and set next train to be advanced
- Go to first step if any trains remain



Example Results Stringline



Dispatching Algorithm Evaluation

- Advantages
  - Allows incremental approach by providing guaranteed deadlock prevention
  - Slow parts are parallelizable
  - Fully general to any track layout and train plan
  - Overall algorithm based on simple principles
    - Should be robust with less testing

- Disadvantages
  - Does not currently evaluate multiple alternatives for the free path
    - Even if added, best case is finding a local optimum
  - Classifies many deadlock free train configurations as unsafe
    - Optimal solution likely unreachable
    - Starting with trains on the line would require a “translator” algorithm to advance trains from their real position to a deadlock free position

Dispatching Algorithm Approaches

Properties	Greedy Advancing with State Restore	Deadlock Prevention	New Algorithm
Track and Train Generality	In base algorithm	Depends on algorithm	Included
Train Length	Minor effect on speed	Minor effect on speed	Minor effect on speed
Solution Quality	Good	Good	Good
Solution Speed	Very quick to moderate for low to high train density	Quick for all train densities	Very quick for all train densities
Scalability	Very good to moderate for low to high train density	Moderate for all train densities	Very good for all train densities
Intermediate Feasibility	None	Every step	Every step
Example Implementation	RTC	GTMS, Railsys	In development

Thank you for your attention!

Geordie Roscoe
Graduate Research Assistant
Rail Transportation and Engineering Center (RailTEC)
University of Illinois at Urbana-Champaign
groscoe2@illinois.edu

Technical collaboration and assistance by:
Chung-Lin (James) Chan
Jiaxi Zhao
Daniel Holmes
C. Tyler Dick, PhD, P.E.

This project is supported by an industry sponsor
with additional support from the Association of American Railroads