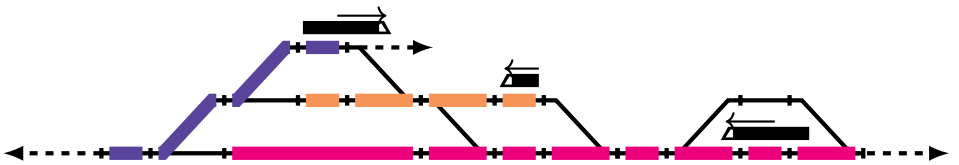


# DISPLIB 2025

## Train dispatching competition

Effective management of dense railway traffic using algorithms has proven to be very hard. The DISPLIB competition challenges you to advance the state-of-the-art in real-time train dispatching and contribute to more environmentally friendly transportation!

This document describes the DISPLIB competition (Sec. 1) and the formal problem definition with the associated file format (Sec. 2).

Any updates to this document will be published on the DISPLIB web page. On the web page you will also find the problem instance files and a Python solution verification program.

**Changelog**

- *2025-02-07*: Added scientific committee members. Added acknowledgement of Wabtec as instance contributor.
- *2024-10-08*: Increased memory limit to 32 GB, added GPU hardware limits, and clarified intentions for hardware limits (Sec. 1.2). Some errors in Phase 1 instances were fixed (see the web page).
- *2024-10-04*: Clarified submission guidelines (Sec. 1.5) and use of multiple resources (Sec. 2.1).
- *2024-10-02*: Clarified rules on multiple algorithms and tuning (Sec. 1.2) and finalist evaluation criteria (Sec. 1.6).
- *2024-10-01*: Added evaluation committee, journal invitation (Sec. 1.6) and disqualification clause (Sec. 1.2).
- *2024-09-06*: First version published on web page.

# Contents


**1 The DISPLIB 2025 Competition** **2**
- 1.1 Spirit . . . 2
- 1.2 General rules . . . 2
- 1.3 Timeline . . . 3
- 1.4 Scoring of submissions . . . 3
- 1.5 Submission instructions . . . 4
- 1.6 Winners and finalists . . . 4

**2 DISPLIB file format** **5**
- 2.1 Conceptual model . . . 5
  - 2.1.1 Objective . . . 6
- 2.2 JSON format . . . 7
  - 2.2.1 Trains . . . 7
  - 2.2.2 Objective . . . 7
  - 2.2.3 Solution format . . . 8
- 2.3 An example problem . . . 8

**3 Acknowledgements** **10**

# 1 The DISPLIB 2025 Competition

## 1.1 Spirit

The DISPLIB competition on train dispatching aims to encourage research into algorithms for optimized train dispatching that are suitable for real-time usage and for integration into in a real-world train dispatching system. We encourage submission of algorithmic ideas that have not been carefully implemented or fine-tuned for real-time usage by focusing on solution quality and instance coverage.

All the instances on which the algorithms will be evaluated are extracted from real-world railway information systems from different countries around the world. Instances with a wide range of characteristics will be used, including passenger-dominated versus freight-dominated railways, and coarse-grained infrastructure (stations and lines) versus fine-grained infrastructure (signals in station areas).

## 1.2 General rules

- Any programming language and computer may be used to solve the problem instances. The use of existing general-purpose optimization software is allowed (including commercial solvers such as CPLEX, Gurobi, Xpress, etc.). The use of multiple unrelated algorithms is discouraged.
- The code is not expected to be delivered to be run by the organizers. However, in order to be considered as a finalist, competitors may be required to show source code to the organizers. This is simply to check that they have followed the rules and the spirit of the competition, and will be treated in the strictest confidence.
- All reported results must be computed using no more than 8 CPU threads and 32 GB of RAM. Teams using GPUs must also limit all reported results to be computed using no more than 1 GPU unit and 24 GB of GPU memory. These limits are put in place to ensure fairness between participants, and to encourage focus on new algorithmic ideas rather than exploiting more computational power. The limits also makes it easier for the organizers to reproduce results if necessary.
- Algorithms suitable for real-time use are strongly encouraged, and all reported reported results must be computed with a **time limit of 10 minutes per instance**.
- Offline tuning or learning phases of the algorithm are allowed, and do not count against the time limits or hardware limits for computing the solution. However, adaptation to specific problem instances are not allowed. The aim is to develop an algorithm that can solve arbitrary instances similar to the competition instances, and, ideally, any train dispatching problem instance.
- Problem instances and solutions will be given in the JSON formats described in Sec. 2. A solution will be considered feasible if the provided Python program for solution verification reports it to be feasible, and the objective value used to evaluate the submission's quality will be the one reported by the program. We have done our best for the verification program to adhere to the problem instance specification in Sec. 2. However, if any bugs are discovered in the verification program, the program may be modified and the new program announced on the DISPLIB web page during the competition. Deliberate exploitation of bugs in this program is not allowed, and such bugs should be reported to the competition organizers.
- The organizers reserve the right to make changes to the rules and the timeline. Violations of the rules or other unfair activity may result in disqualification. In case of conflicts, the

decision of the organizers will be final and binding.

## 1.3 Timeline

The problem instances will be released in two phases:

- **Phase 1**: the first phase will contain a set of varied and challenging problem instances based on real-world use cases. The early phase instances will be publised on the DISPLIB web page on 1st October 2024. We encourage all participants to submit solutions to phase 1 instances as early as possible to become listed on a **Phase 1 scoreboard** on the web page. Submitting the Phase 1 solutions before the final competition deadline is optional but strongly encouraged as it will provide feedback on the progress of the competition as well as a chance to be selected as a finalist.
  - Early submissions (to appear in the Phase 1 scoreboard) must be made at the latest on **31st January 2025** (AoE).

- **Phase 2**: second phase of the competition will contain larger and more challenging problem instances based on real-world use cases. The late phase instances will be published on the DISPLIB web page on 3rd February 2025. A final scoreboard for the competition will be published on the web page after the announcement has been made at ODS 2025. Note that the scores for the final submission will be computed independently of the early submissions, and it is possible to achieve the highest score in the final scoreboard without having submitted anything for the early deadline.
  - Final competition deadline for submissions of solutions to instances from both phases must be made at the latest on **30th April 2025** (AoE).

Please also watch the web page for amendments of the rules, instances, and the verification program. Such amendments will only be made when absolutely necessary to achieve a fair competition based on realistic problem instances in the spirit described in Sec. 1.1.

## 1.4 Scoring of submissions

Points are awarded to each solution based on the position among its competitors and the phase (phase 2 instances are given more points). The top competitors will score points according to the scale in Table 1 for each instance in each phase. When there are two or more solutions tied for the same positions, the points granted by these positions are split evenly between competitors (rounded up in case of fractional points). When a team does not provide any solution for an instance, it is awarded zero points for that instance.

The ordering of all competitors will be based on the sum of points for all instances from both phases. The winner of the competition will be the solver with the highest total number of points. In the event of any ties for a position, the competitor with superior results (based on descending order of lexicographic ordering of points awarded) will gain precedence.

Table 1: Points awarded for an instance.

| Position | Phase 1 | Phase 2 |
|---|---|---|
| 1st | 10 | 15 |
| 2nd | 7 | 11 |
| 3rd | 5 | 8 |
| 4th | 3 | 6 |
| 5th | 2 | 4 |
| 6th | 1 | 3 |
| 7th | | 2 |
| 8th | | 1 |

## 1.5 Submission instructions

Everyone is welcome to participate! Collaboration in teams of any size is encouraged. No registration is needed – any team may submit new solutions to the organizers by email at any point before the deadline. We encourage teams not to send updates more frequently than once a week. A scoreboard for the Phase 1 instances will be maintained on the DISPLIB webpage. Submissions should contain the following:

- The team's name and the team member's names, affiliations and countries.
- A zip file containing at most one solution per problem instance.

During Phase 2, the scoreboard will be kept hidden. Within the final deadline, each team should also submit:

- A short report describing the algorithm. The report should be maximum 6 pages in Springer LNCS format.

The final ranking will be decided by the competition committee based both on the results of the scoreboard and the quality/novelty of the report. Submissions should be sent to the competition organizers within the deadline using the following email address: displib2025competition@gmail.com

## 1.6 Winners and finalists

2-4 of the top teams will be selected as finalists. An international panel of experts of railway optimization will be evaluating the submissions and selecting a group of finalists The panel consists of:

- Giorgio Sartor, SINTEF Digital, Norway (Chair)
- Marcella Samà, Roma Tre University, Italy
- Paolo Ventura, Siemens Mobility, Italy
- Steven Harrod, Technical University of Denmark, Denmark
- Dennis Huisman, Erasmus University Rotterdam, Netherlands

Finalists will be invited to present their algorithms at the International Conference on Optimization and Decision Science (ODS) 2025, and also invited to submit a paper to the Journal of Rail Transport Planning & Management with an expedited review process. The selection of finalists will be based on the points used to compute the scoreboards (see Sec. 1.4) and on the quality

and novelty of the algorithm presented in the report. Teams that have submitted strong results for the Phase 1 scoreboard will also be appreciated by the committee.

Also at ODS 2025, the organizers will present a summary of results of the competition and the ranking of the teams, including which team is the winner, will be announced. Final results and rankings for all teams will be published on the DISPLIB web page after ODS 2025.

# 2 DISPLIB file format

Given a set of trains traveling on a railway, the **Train Dispatching Problem** is the operational problem that occurs when some trains have become delayed with respect to their prescribed timetable, and we want to make routing and scheduling adjustments to minimize the total delay on the railway.

The DISPLIB problem format is a conceptual model and a JSON format for describing instances of the train dispatching problem. The format is designed to be as simple as possible while still being capable of capturing a wide range of real-world dispatching problems.

## 2.1 Conceptual model

Conceptually, the format describes a **set of trains** and an **objective**. Each train consists of a directed acyclic graph of operations, called the **operations graph**. This graph has exactly one node that has no incoming edges, called the **entry operation**, and exactly one node that has no outgoing edges, called the **exit operation**. Each operation has the following properties associated with it:

- A minimum duration, i.e., the shortest allowable time from the start of the operation to the end of the operation. The minimum duration may be zero time units.
- Lower and upper bounds on when the operation may start.
- A set of **resources**, all of which need to be exclusively allocated to the train to perform the operation. Each resource used by the operation also has a release time, which is an additional duration that the operation occupies the resource after the operation has ended. The release time may be zero time units.

Resources may represent anything that can only be used by one train at a time, but typically represent physical sections of the railway track that cannot be shared with other trains according to safety regulations. Note that some operations require more than one resource. For example, in Figure 1, operation 1 requires *both* of the resources $r_1$ and $r_2$, representing the fact that a long train is streching over two consecutive track sections. Note that the set of all resources defined for the problem is given only implicitly, as the union of the sets resources referred to by all the operations.

A **solution** to the problem consists of an ordered sequence of operation **start events** across all trains. Each start event specifies the start time of the corresponding operation. From this global sequence of events, we can extract the subsequence of events for each specific train. In the train's subsequence, any start event (except the first event) is also the **end event** of the train's previous operation. All time and duration values must be given as non-negative integers. This solution sequence is feasible if:

- The sequence of events is given in chronological order, i.e., the time of each event is greater than or equal to the time of all preceding events.

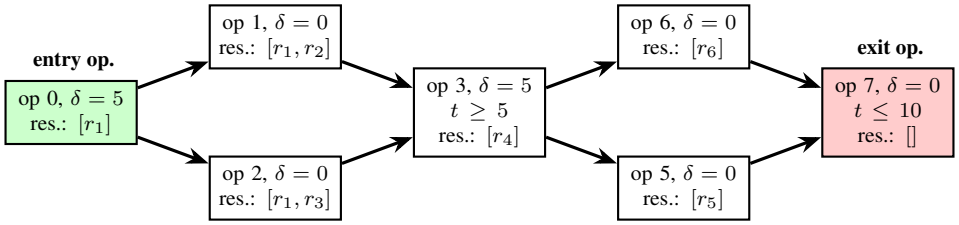



Figure 1: A train operation graph. The boxes represent operations, labeled with a minimum duration $\delta$, and a list of resources $r_1, r_2, \ldots$ to which it needs exclusive access. The green node labeled "op 0" is the entry operation (it has no incoming edges), and the red node labeled "op 7" is the exit operation (it has no outgoing edges). The operations "op 3" and "op 7" have lower and upper bounds, respectively, defined on their start time $t$.

- The subsequence of operations applying to each train forms a path through that train's operations graph, starting at the entry operation and ending at the exit operation.
- The time value of each start event satisfies the lower and upper bounds on the start time of the operation.
- The duration from each start event to the corresponding end event must be greater than or equal to the minimum duration of the corresponding operation.
- For any pair of operations $o_1$ and $o_2$, where:
  - the start event of $o_1$ precedes the start event of $o_2$ in the global sequence, and
  - $o_1$ and $o_2$ belong to different trains, and
  - $o_1$ and $o_2$ use a common resource $r$,

  the following must hold:
  - the end event for $o_1$ precedes the start event for $o_2$ in the global sequence, and
  - the duration from the end event of $o_1$ to the start event of $o_2$ is greater than or equal to the release time of resource $r$ in $o_1$.

  Note that this implies that the global ordering of events is important beyond just the time values. A resource may be released by one train and allocated by another train at the same time value, but this is only feasible if the end event of the previous usage occurs before the next usage in the global ordering (see also the example in Sec. 2.3).

Note that the exit operation has no end event, and so any resources occupied by the exit operation of the train will never be released.

### 2.1.1 Objective

The objective defines the **cost** of a solution, and the goal of solving the DISPLIB problem instances is to find a feasible solution with the minimum cost.

The objective is described as a sum of objective components. Each objective component describes an **operation delay** as a threshold time $\bar{t}$ for when an operation becomes delayed, and the cost associated with the delay. The cost value $v_i$ of an operation delay $i$ is:

$$v_i = c \cdot \max\left\{0, t - \bar{t}\right\} + d \cdot H(t - \bar{t})$$

where $c$ and $d$ are non-negative constants, $t$ is the start time of the referenced operation, and $H$ is the Heaviside step function that takes the value 0 for negative arguments and 1 otherwise. If that operation is not part of the solution, then $v_i = 0$. Note that, typically, either $c$ or $d$ is zero, but the formula contains both to be able to model both step-wise and linear functions of delay.

## 2.2 JSON format

The JSON object for a DISPLIB problem instance contains two keys: `trains` and `objective`, giving the following overall structure of the JSON file, where `...` is a placeholder:

```
{ "trains": [[{ ... operation ... }, ... ], ... ],
  "objective": [{ ... component ... }, ... ] }
```

### 2.2.1 Trains

The top-level `trains` key contains a list of trains, where each train is a list of operations. References to specific trains are given as a zero-based index into the `trains` list, and references to operations are given as a zero-based index into a specific train's list of operations. Each operation is a JSON object with the following keys:

- **`start_lb`** (optional, number): the earliest start time of the operation. Must be a non-negative integer. If the key is not present, defaults to 0.
- **`start_ub`** (optional, number): the latest start time of the operation. Must be a non-negative integer. If the key is not present, defaults to infinity (i.e., no upper bound).
- **`min_duration`** (number): the minimum duration, i.e., the minimum time between the start time and the end time of the operation. Must be a non-negative integer.
- **`resources`** (optional, list): a list of resources used by the train while performing the operation. If the key is not present, defaults to the empty list. Each resource usage is given as a JSON object with the following keys:
  - **`resource`** (string): the name of a resource.
  - **`release_time`** (optional, number): the release time for the resource, i.e., the minimum duration between the end time of this operation and the start time of any subsequent operation (of a different train) using the same resource. If the key is not present, defaults to 0.
- **`successors`** (list): a list of alternative successor operations, given as zero-based indices into the list of this train's operation. The list must be non-empty unless this operation is the *exit operation*.

For example, an operation that starts between time 7 and 8 and has a duration of 5 time units, uses the resource `r1`, and releases it immediately after the end event, and must be succeeded by the operation with index 2, would be formatted as follows:

```
{ "start_lb": 7, "start_ub": 8, "min_duration": 5,
  "resources": [{ "resource": "r1" }], "successors": [2] }
```

For each train, the list of operations is ordered topologically, i.e., all successors for an operation appear after it in the list. Note that this also means that the entry operation will always be at index 0, and the exit operation will always be the last operation in the list.

### 2.2.2 Objective

The top-level `objective` key contains a list of objective components. Each objective component is a JSON object containing the key `type` for determining an objective component type, and other keys depending on the type. Only one type, called **operation delay** (see Sec. 2.1), is defined in DISPLIB 2025 (the `type` key is included for forward compatibility).

The operation delay objective component is decribed as a JSON object with the following keys:

- **`type`** (string): must contain the string `"op_delay"`.
- **`train`** (number): reference to a train as an index into the top-level `trains` list.
- **`operation`** (number): reference to an operation as an index into the list defining the train's operation graph.
- **`threshold`** (number): a time after which this delay component is activated, as defined in the formula above. If the key is not present, defaults to 0.
- **`increment`** (number): the constant $d$ in the formula above. Must be a non-negative integer. If the key is not present, defaults to 0.
- **`coeff`** (number): the constant $c$ in the formula above. Must be a non-negative integer. If the key is not present, defaults to 0.

For example, an objective component that measures the time that train 0's operation 5 is delayed beyond time 10, would be formatted as follows:

```
{ "type": "op_delay", "train": 0, "operation": 5,
  "threshold": 10, "coeff": 1 }
```

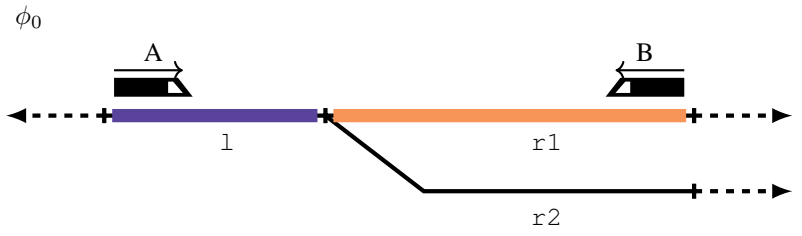
### 2.2.3 Solution format

Solutions to a DISPLIB problem are given as a JSON object (typically in a separate file from the problem instance), containing the following keys:

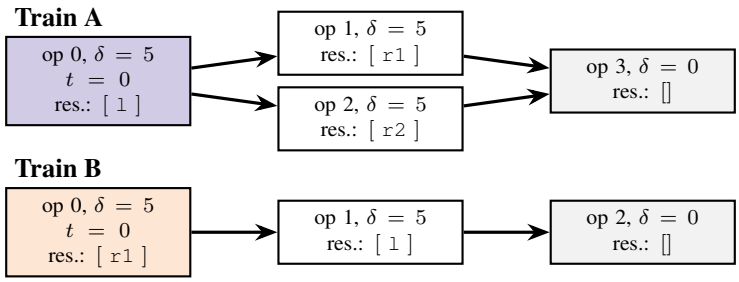
- **`objective_value`** (number): is the objective value of the solution (according to the `objective` from the problem instance).
- **`events`** (list): is an ordered list of start events, i.e. a list of references to operations to be started in the given order. Each start event is described by a JSON object with the following keys:
  - **`time`** (number): the time at which to start the operation. Must be a non-negative integer.
  - **`train`** (number): reference to a train as an index into the top-level `trains` list.
  - **`operation`** (number): reference to an operation as an index into the list defining the train's operation graph.

## 2.3 An example problem

Let's consider two trains meeting at a junction, coming from opposite directions. Train A is currently occupying the left part of the track (resource `l`), and may proceed either to the upper right track (resource `r1`), which train B is currently occupying, or to the lower right track (resource `r2`), which is currently free.



The operation graphs for the two trains would look as follows:




The objective is to minimize the time when train B finishes traveling through `l`. The JSON problem instance for this problem is:

```
{
  "trains": [
    [{ "start_ub": 0,
       "min_duration": 5,
       "resources": [{ "resource": "l" }],
       "successors": [1, 2] },
     { "min_duration": 5,
       "successors": [3],
       "resources": [{ "resource": "r1" }]},
     { "min_duration": 5,
       "successors": [3],
       "resources": [{ "resource": "r2" }]},
     { "min_duration": 5,
       "successors": []}],
    [{ "start_ub": 0,
       "min_duration": 5,
       "resources": [{ "resource": "r1" }],
       "successors": [1]},
     { "min_duration": 5,
       "resources": [{ "resource": "l" }],
       "successors": [2]},
     { "min_duration": 5,
       "successors": []}]],
  "objective": [{ "type": "op_delay",
                  "train": 1,
                  "operation": 2,
                  "coeff": 1}]
}
```

The following JSON object is a feasible (and optimal) solution to the problem instance:

```
{ "objective_value": 10, "events": [
  {"time": 0, "train": 0, "operation": 0},
  {"time": 0, "train": 1, "operation": 0},
  {"time": 5, "train": 0, "operation": 2},
  {"time": 5, "train": 1, "operation": 1},
```

```
{"time": 10, "train": 1, "operation": 2},
{"time": 10, "train": 0, "operation": 3}] }
```

Note that this is an example where the global ordering of events is important beyond the ordering implied by the time values: if we switch the ordering of the two events at time 5, the result is not a feasible solution since operation 1 of train B allocates `1` while it is still in use by train A.

# 3 Acknowledgements

The competition is organized by Bjørnar Luteberget, Giorgio Sartor, Oddvar Kloster, and Carlo Mannino.

Thanks to SINTEF Digital, Siemens Mobility, Wabtec Corporation, and SBB (data.sbb.ch) for providing/publishing raw data for the problem instances.

Thanks to SINTEF Digital for providing funding for the efforts involved in creating this competition.

Thanks to the organizers of the International Conference on Optimization and Decision Science (ODS) for supporting and publicizing the competition.