Optimizing the PDDL domain of TUSP to improve planner performance

Modifying the domain to improve planner execution time, plan quality, and problem solvability

Shu-wing Chiu

Supervisor(s): Sebastijan Dumancic, Issa Hanou

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Shu-wing Chiu
Final project course: CSE3000 Research Project
Thesis committee: Sebastijan Dumancic, Issa Hanou, Rihan Hai

Abstract

The Planning Domain Definition Language (PDDL) is specifically designed to encode various problems. Each PDDL problem consists of an initial state and a goal state, and a domain that defines the constraints of the problem. Planners can be used to find a sequence of actions to achieve the goal state. It is possible to improve the performance of planners by modifying the PDDL domain of a problem. The goal of this research is to implement this to the domain of the Train Unit Shunting Problem (TUSP), which is the problem of moving trains in and out of shunting yards. The research question we attempt to answer is:

To what extent can we improve planner performance by optimizing the PDDL domain of TUSP?

The main contributions of this research are: a formalization of TUSP and its constraints in PDDL terminology, a comprehensive evaluation of the performance of planners on the PDDL domain of TUSP, provide general approaches and techniques that can be used to optimize a PDDL domain, and provide insights into the relationship between planner performance and the domain properties of TUSP. To answer the research question, we measure the performance of the planners in terms of execution time, plan quality, and problem solvability. We modified the domain by combining actions such that the number of computations is decreased and by introducing action costs. With these modifications, we found a decrease in planner execution time and an increase in plan quality. We found no difference in problem solvability. Therefore, we can conclude that we can indeed improve the performance of planners by implementing these modifications to the domain of TUSP.

1 Introduction

The Train Unit Shunting Problem (TUSP) involves moving trains of different types in and out of shunting yards [5]. Railway organizations have to deal with this problem daily, making this a real-world problem [9][11]. How can trains be parked in such a way that, when it needs to depart again, it can do so with as little delay as possible?

To automate this train planning process, we use the Planning Domain Definition Language (PDDL). In classical planning, it is important to select the right planner for a given problem [13]. PDDL is language specifically designed to encode various problems. Each PDDL problem consists of an initial state and a goal state, and a domain that defines the constraints of the problem. Each planner has its properties. Some may execute better or faster on a given problem instance, while other planners are not able to find a solution to the problem. How a problem is defined affects the performance of such planners.

Previous research has found an increase in planner performance [10]. This research provides the definition, characterization, and computation of preconditions and conditional effects for complex actions. However, the question of whether this applies to multiple planners and how it affects performance in terms of plan quality and solvability remains unanswered.

In this research, the aim is to evaluate the performance of planners for TUSP. Specifically, the paper attempts to answer the following question:

To what extent can we improve planner performance by optimizing the PDDL domain of TUSP?

This can be divided into three sub-questions:

1. Is it possible to improve the total execution time of planners?
2. Is it possible to improve the plan quality generated by planners?
3. Is it possible to improve the solvability of problems within the domain?

By answering these questions, the main contributions of this paper are:

1. A formalization of TUSP and its constraints in PDDL terminology.
2. A comprehensive evaluation of the performance of planners in the PDDL domain.
3. Provide general approaches and techniques that can be used to optimize a PDDL domain.
4. Provide insights into the relationship between the performance of planners and PDDL domain properties.

By investigating whether PDDL domains can be modified to improve the performance of different planners, we hope to contribute to this area of research.

2 Background: TUSP, Planners & PDDL

In this section, the topics that will recur in the next chapters are explained. The information described here is a prerequisite to understanding the problem description and contributions.

2.1 TUSP

The Train Unit Shunting Problem (TUSP) is the problem of moving trains in and out of shunting yards [5]. A shunting yard is a train station where trains are allocated and stored outside of servicing hours. An example is shown in Figure 1.

In the figure, the initial state of a simple TUSP instance is shown, with trains 1, 2, and 3 arriving at the shunting yard at their current locations. Each train will move from the arrival path, towards the switch, and be allocated in one of the tracks. After all trains have been parked on track 1 or 2, they can begin to depart the shunting yard (no train can depart if there is another train in the way). In which order they should depart, is defined by the goal state of the problem.

For instance, let train 1 be the first to depart, then train 2, and finally train 3. How should the trains be parked such that

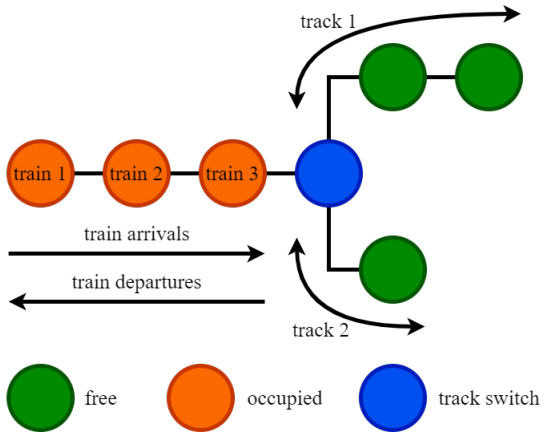

Figure 1: A diagram representing a simple shunting yard, with each node representing a track part. For simplicity, we assume each train has the same length, and each track part has the length of one train. Trains 1, 2, and 3 are arriving at the shunting yard and need to be parked on either track 1 or track 2. This should be done in such a way that each train can depart with as little delay as possible when they need to.

the goal state can be reached? We can see that the best spot for train 1 would be track 2, as it only has one parking spot. That leaves trains 2 and 3 to be parked on track 1. From here, we can see that train 3 needs to be moved out of the way to let train 2 depart. We can do so by re-allocating train 3 to track 2 after train 1 has departed, let train 2 depart next and finally, train 3.

2.2 Planners

Human planners make use of different algorithms to help them create plans that minimize delays in train departures, but planning systems remain mostly unused by commercial organizations [14]. In classical planning, PDDL-based planners can be used to simplify the planning process. These planners are designed to find a sequence of actions that takes the initial state to the goal state. A typical planner uses a translator to turn a PDDL problem into data structures that the planner can use to search through, and a search engine that uses certain heuristics to efficiently search for one or more valid plans [2][4][7][12].

Heuristics are functions used by the planner to estimate the cost between the current state and the goal state [6]. They help the planner search for valid plans more efficiently by making the planner avoid states that either do not lead to the goal state or have a cost that is too large.

Many planners are publicly available, which can be run on the domain and problem instances of the problem in question. Different planners have different properties and their performance also varies per problem instance they are executed on [13]. Hence, selecting the right planner for a problem can be very important in terms of execution time and plan quality. In this research, we will analyse four planners that were submitted to the International Planning Competition 2018 (IPC 2018)[1].

[1]https://ipc2018-classical.bitbucket.io/

2.3 PDDL

To be able to use a planner on a problem, this problem must be defined in code. The Planning Domain Definition Language (PDDL) is a language designed for defining various planning and scheduling domains and problems [8]. It is the standard language used by IPC 2018 for the different domains of the competition. A PDDL problem instance will consist of the following components:

1. A domain definition, which defines the object types, predicates with arguments that can either be true or false, and actions (operations) with each a precondition and effect.
2. A problem definition, which defines the objects, the initial state, and the goal state of the problem within the constraints of the domain.

PDDL domains and problems, together with a planner, can be used to plan the possible solutions to the defined problems.

2.4 Related work

Previous research has found an increase in planner performance, in terms of runtime and number of nodes, after designing a complex action that reduces the search space size [10]. This research provides the definition, characterization, and computation of preconditions and conditional effects for complex actions, where a precondition represents the required state before a domain action and an effect corresponds to the changes to the state after a domain action. However, the question of whether this applies to multiple planners and how it affects performance in terms of plan quality and solvability remains unanswered.

3 Problem Description: Domain of TUSP

Given a set of trains, a set of track parts, and the initial positions of the trains, the goal is to find a sequence of train movements such that the trains are positioned as defined by the goal state. To achieve this, we must first define the domain of TUSP and then define the precondition and effect of the actions.

3.1 Domain definition

In TUSP, the domain consists of a set of objects: {train unit, track, track part}, a set of predicates: {$(at\ x\ y)$, $(parkOn\ x\ y)$, $(hasBeenParked\ x)$, $(nextTo\ x\ y)$, $(free\ x)$, $(onTrack\ x\ y)$, $(onPath\ x)$, $(switch\ x)$}, and a set of actions: {move-on-arrival, move-to-departure, move-to-track, move-from-track, move-along-track}. Each predicate has the following meaning:

- $(at\ x\ y)$ - train unit x is at track part y.
- $(parkOn\ x\ y)$ - train unit x is parked on track y.
- $(hasBeenParked\ x)$ - train unit x has been parked.
- $(nextTo\ x\ y)$ - track part x is next to track part y.
- $(free\ x)$ - track part x is free.
- $(onTrack\ x\ y)$ - track part x is on track y.
- $(onPath\ x)$ - track part x is on the arrival/departure path.

- $(switch\ x)$ - track part x is a switch.

This domain assumes there is exactly one arrival/departure path on which all trains arrive/depart, and exactly one switch. A problem in this domain can only be solvable if there are enough track parts available for all train units to park on.

3.2 Action precondition & effect

Each action contains a precondition, the predicates that must hold before using an action, and effect, the predicates that hold as a result of the action. Below are the five actions of the domain that define what movements a train can make: moving on the arrival path, moving to the departure, moving from the switch to a track, moving from a track to the switch, and moving along a track. The actions of the TUSP domain are visualized in Figure 2 and the complete domain file can be found in Appendix A.1.

```
(:action move-on-arrival
    :parameters (
        ?train      - trainunit
        ?from ?to   - trackpart)
    :precondition (and
        (at ?train ?from)
        (free ?to)
        (nextTo ?from ?to)
        (not (hasBeenParked ?train))
        (onPath ?from))
    :effect (and
        (at ?train ?to)
        (not (at ?train ?from))
        (free ?from)
        (not (free ?to))))
```

Listing 1: move-on-arrival. This action moves a train that is on the arrival path, to a free adjacent track part.

```
(:action move-to-track
    :parameters (
        ?train      - trainunit
        ?from ?to   - trackpart
        ?t          - track)
    :precondition (and
        (at ?train ?from)
        (free ?to)
        (nextTo ?from ?to)
        (onTrack ?to ?t)
        (switch ?from))
    :effect (and
        (at ?train ?to)
        (not (at ?train ?from))
        (free ?from)
        (not (free ?to))
        (hasBeenParked ?train)
        (parkedOn ?train ?t)))
```

Listing 2: move-to-track. This action moves a train that is on the switch, to a free adjacent track part that is on a track.

```
(:action move-along-track
    :parameters (
        ?train      - trainunit
        ?from ?to   - trackpart
        ?t          - track)
    :precondition (and
        (at ?train ?from)
        (free ?to)
        (nextTo ?from ?to)
        (onTrack ?from ?t)
        (onTrack ?to ?t))
    :effect (and
        (at ?train ?to
        (not (at ?train ?from))
        (free ?from)
        (not (free ?to))))
```

Listing 3: move-along-track. This action moves a train that is on a track, to a free adjacent track part that is on the same track.

```
(:action move-from-track
    :parameters (
        ?train      - trainunit
        ?from ?to   - trackpart
        ?t          - track)
    :precondition (and
        (at ?train ?from)
        (free ?to)
        (nextTo ?from ?to)
        (onTrack ?from ?t)
        (switch ?to))
    :effect (and
        (at ?train ?to)
        (not (at ?train ?from))
        (free ?from)
        (not (free ?to))
        (not (parkedOn ?train ?t))))
```

Listing 4: move-from-track. This action moves a train that is on a track, to a free adjacent track part that is a switch.

```
(:action move-to-departure
    :parameters (
        ?train      - trainunit
        ?from ?to   - trackpart)
    :precondition (and
        (at ?train ?from)
        (free ?to)
        (nextTo ?from ?to)
        (onPath ?to)
        (forall
            (?unit - trainunit)
            (hasBeenParked ?unit)))
    :effect (and
        (at ?train ?to)
        (not (at ?train ?from))
        (free ?from)
        (not (free ?to))))
```

Listing 5: move-to-departure. This action moves a train that is on any track part, to a free adjacent track part that is on the path.

3.3 Plan quality

A planner will attempt to find a solution to the problem by providing a sequence of actions. Part of a plan output for this domain will look something like this:

```
(move-on-arrival train1 v1 t0)
(move-to-track train1 t0 t5 track2)
(move-on-arrival train2 v2 v1)
(move-on-arrival train2 v1 t0)
(move-on-arrival train3 v3 v2)
(move-on-arrival train3 v2 v1)
(move-on-arrival train4 v4 v3)
(move-on-arrival train4 v3 v2)
(move-on-arrival train5 v5 v4)
(move-on-arrival train5 v4 v3)
(move-on-arrival train6 v6 v5)
...
```

Listing 6: Part of a plan output. This plan was generated by the DecStar planner of team 2 on the domain.

With $v1$ - $v6$ corresponding to track parts on the arrival path, $t0$ to the switch, and $t1$ - $t5$ to track parts on a track. Note how this plan moves trains 2, 3, 4, 5, and 6 on the arrival path in sequence. In practice, it would be very costly for a driver to constantly switch between trains while moving them one step at a time. A more practical plan would

be to move one train all the way to a track, park it on the track, and then switch to the next train to repeat the same process. Therefore, we determine a plan to be of higher quality if it keeps switching between trains to a minimum. Besides switching between trains, a compact plan with a smaller number of steps is also considered to be of higher quality.

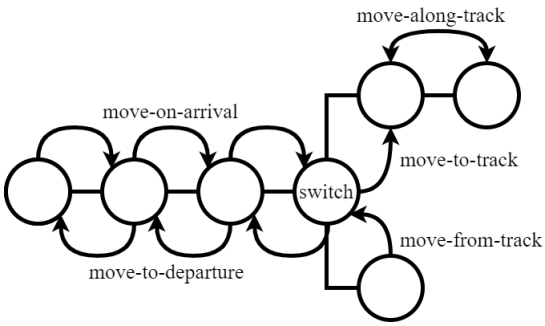

Figure 2: A visualization of the domain actions described in section 3.2. A train can move in a total of five ways: when it moves on the arrival/departure path, when it moves to the arrival/departure path, when it moves to and from the switch, and when it moves along a track.

4 Experimental Work

Given are the domain of TUSP with its objects, predicates, and actions. To decrease execution time, we must find a way to minimize the number of computations made by the planner to select a sequence of actions such that the goal state is reached while satisfying the precondition of all actions. That is, we must minimize the number of times the predicates are called and changed (from true to false, and vice versa). To increase plan quality, we must find a way to minimize switching between trains over the same actions as described in section 3.3, and decrease plan length by combining actions into complex actions such that the goal state is still reachable and the problem can still be solved.

In this section, we provide three modifications to the domain and multiple approaches that are built on top of these modifications to achieve this.

4.1 Domain modification 1: oriented graph

Two track parts that are next to each other, are defined by two predicates in the domain: ($nextTo\ x\ y$), and ($nextTo\ y\ x$).. This symmetric graph representation allows a train to be moved in both directions in the domain. That is, the actions move-from-arrival and move-to-departure can use the same predicate ($nextTo\ from\ to$), even though they move a train in opposite directions. This also applies to move-along-track, which uses this predicate for either direction as well.

As a result, a train can move in either direction through the actions move-on-arrival and move-to-departure, which is different than what is shown in Figure 2. This ambiguity is not desired, as a planner would have to consider these possibilities even if it does not select these actions in the opposite directions. By adding a directional constraint, we can save some computation for the planner. Hence, we want to ensure that these actions can only move a train in the desired direction.

The first modification we made to the domain, is the removal of inverted edges, turning the directed graph into an oriented graph. Suppose track part x and track part y are next to each other, we define this connection with ($nextTo\ x\ y$), but we leave out ($nextTo\ y\ x$). Moving a train in one direction is still possible by checking whether ($nextTo\ from\ to$) holds, and moving a train in the opposite direction is also possible by checking ($nextTo\ to\ from$). The modification is visualized in Figure 3.

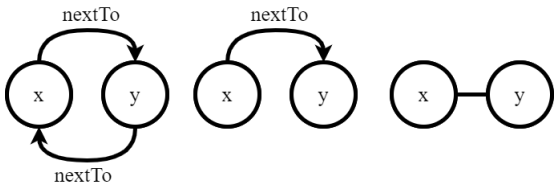

Figure 3: On the left, two track parts are connected through the predicates ($nextTo\ x\ y$) and ($nextTo\ y\ x$). If we want to move a train from x to y and vice versa, we check for ($nextTo\ from\ to$) in the precondition of the action. In the center, two track parts are connected only through the predicate ($nextTo\ x\ y$). If we want to move a train from x to y, we check for ($nextTo\ from\ to$) in the preconditions of the action. If we want to move a train from y to x, we need to check for ($nextTo\ to\ from$) in the preconditions of the action. For simplicity, we omit this predicate in all other figures in this paper as shown on the right.

With this modification, we distinguish two possible directions: forwards, for moving a train into the shunting yard using actions with the predicate ($nextTo\ from\ to$), and backwards, for moving a train out of the shunting yard using actions with the predicate ($nextTo\ to\ from$). Note that we are now required to add an action for moving a train backwards along a track, and change the predicate ($nextTo\ from\ to$) into ($nextTo\ to\ from$) for move-to-departure and move-from-track.

4.2 Domain modification 2: stack

Each action moves a train to an adjacent track part and modifies the predicates as required. Suppose there is only one train that needs to be parked in the shunting yard before it can depart, a planner would provide a plan with as many steps as track parts between the arrival track part and the first track part belonging to a track. In practice, the steps in between moving from the current location to the destination is not useful to the driver of the train as the train can be moved in one direction in one operation. What is important, is knowing that the destination can be reached. Here, an opportunity is presented to combine actions into complex actions that will still allow a planner to find a sequence of actions such that the goal state can be reached from the initial state.

To achieve this, we have introduced a pointer predicate ($trackPointer\ trackPart\ track$) for each track, that points to the furthest reachable track part of that track, including the arrival/departure path for train departures ($pathPointer\ trackPart$). Initially, the pointer of an empty track points to

the last track part of that track since it is the furthest reachable track part. Once a train moves into that track, it will be parked on this last track part and the pointer will move to the next track part. If the track is full, the track will have no pointer as none of its track parts are reachable. When a train moves out of the track (note that this can only be the train that is closest to the start of the track), the pointer will point to the current location of the train. This procedure is visualized in Figure 4.

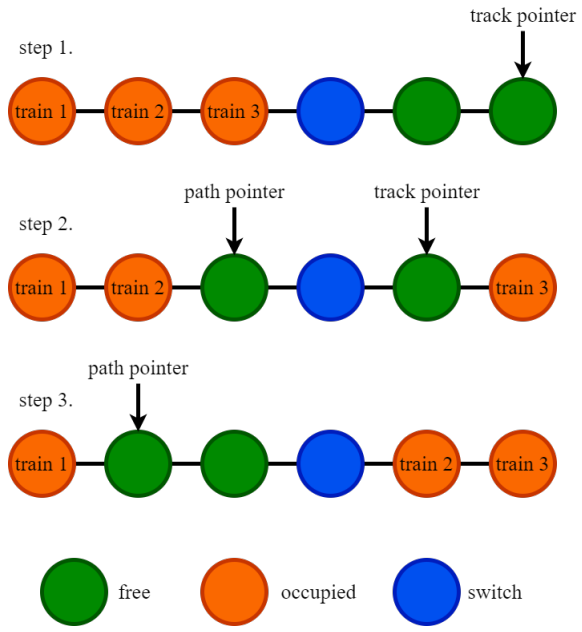

Figure 4: Part of a shunting yard with an arrival/departure path, a switch, and a track. Initially, the track pointer starts at the track part furthest into the shunting yard (step 1) and moves one step backwards each time a train parks on the track (step 2), until the track is full (step 3), after which the pointer disappears. The path pointer behaves in the same manner, but in this case, trains leave the path until it becomes empty, moving the path pointer backwards each time as well.

4.3 Domain modification 3: action costs

A planner will attempt to find a plan with a minimum number of steps. However, each action can be selected at any time without adding cost to the plan. The planner is free to order the selected actions in any way as long as the sequence of actions turns the initial state to the goal state. As a result, the steps included in the plan may not be practical to a human driver and the plan will not be of high quality (remember the example in section 3.3).

To increase plan quality, we have introduced action costs. The domain function $(total\text{-}cost)$ keeps track of the total cost of the plan. Initially, $(total\text{-}cost)$ is equal to 0 and, by default, each action will increase it by 1.

4.4 Approach 1: from path to track (PT)

Using domain modifications 1 and 2, we are now able to move a train from the path or a track to the switch in a single action, and vice versa. This approach removes the need for move-along-track and modifies the actions move-on-arrival, move-to-track, move-from-track, and move-to-departure. The modifications of these actions are described below. Figure 5 shows a visualization of these actions. The complete modified domain can be found in Appendix B.1.

move-on-arrival

Instead of moving a train from the arrival path to a free adjacent track part, this action now checks if the adjacent track part and switch are both free and if this is the case, it moves the train to the furthest free track part and on a track.

move-to-track

Instead of moving a train from the switch to a free adjacent track part that is on a track and on a track, this action now moves the train to the furthest free track part of a track.

move-from-track

Instead of moving a train from a track to the switch if it is adjacent and free, the action now checks if the adjacent track part and switch are both free and if this is the case, moves the train to the switch. This action can only be done after all trains have been parked. This action, together with move-to-track, are only used for reallocating a train to a different track.

move-to-departure

Instead of moving a train from the switch to a free adjacent track part that is on the departure path, this action now checks if the adjacent track part and switch are both free and, if this is the case, moves the train to the furthest free track part of the departure path.

Advantages & disadvantages

This modified set of actions only allows one train to be moved at a time until it is parked or has departed. This means that once a train moves on arrival, the next train can only move when the first train has been parked. The same applies to a train that is parked and wants to depart, it can only do so once the train before it has departed. This is because any operation that can be done without the driver having to physically walk to a different location, is now defined by a single action. The plan can also not contain the same action to the same train in consecutive order (e.g. move-on-arrival cannot be done twice with the same train, consecutively). As a result, the plan will only contain meaningful steps in an order that makes more sense to a human driver.

However, it is now possible to find a solution even if the tracks are not connected to the switch or the path, which should not be the case. Besides this, the actions will not work accurately if there are free track parts between two trains on the arrival path. In this case, a train can move to a switch while there is another train in the way (e.g. move-on-arrival only checks if the next track part and switch are free, but not other track parts in between). The domain now assumes that these properties are correctly defined in the initial state of the problem.

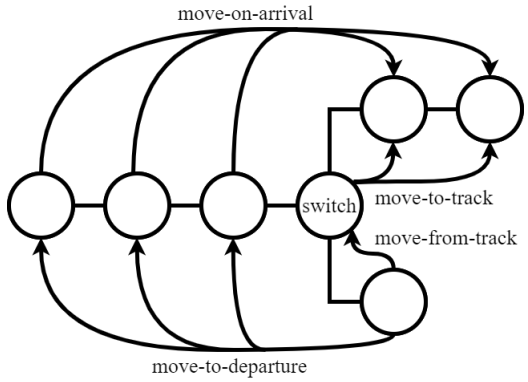

Figure 5: A visualization of the actions in the PT domain. A train can move in a total of four ways: from the arrival path to a track, from the switch to a track, from a track to the switch, and from a track to the departure path. Note that it is not possible to move from the switch to the departure path, as the switch is only used to reallocate trains from one track to another track.

4.5 Approach 2: minimized switching & reallocation (MSR)

This approach uses domain modification 3, which introduces action costs to the domain. By default, each action will increment ($total\text{-}cost$) by 1. The predicate ($currentTrain\ train$) is added, which keeps track of the current train that is being moved. Initially, this is the first train to arrive at the shunting yard.

An additional action switch-to-next-train is added that switches the current train to the next train. Switching between trains will increase the cost of the plan, giving the planner the incentive to minimize train switching. As a result, plans will move a single train to a track and park it there before moving the next train.

Besides train switching, the action move-to-track-hasBeenParked is added that increases ($total\text{-}cost$) by 2. This action moves a train to a track after all trains have been parked. This is done to help planners minimize train reallocation, which is a costly operation. Planners should already minimize this as it adds additional steps to the plan, but adding a higher cost to this operation provides an additional variable to minimize the use of this action. The new actions are briefly described below and shown Figure 6.

switch-to-next-train

This action takes the current train and the next train, and assigns the next train as the current train. Only one train can be the current train. Switching trains increments ($total\text{-}cost$) by 1.

move-to-track-hasBeenParked

This action is the same as move-to-track, but it can only be done after all trains have been parked ($forall$ ($train$ - $trains$) ($haveBeenParked\ train$)), and increments ($total\text{-}cost$) by 2.

Advantages & disadvantages

Planners are encouraged to minimize switching between trains and reallocating trains to a different track.

This domain is only compatible with planners that support the use of action costs. Furthermore, plans for this domain will contain additional steps for switching between trains. This can also be considered an advantage, as it is clear when a driver would need to switch to a different train.

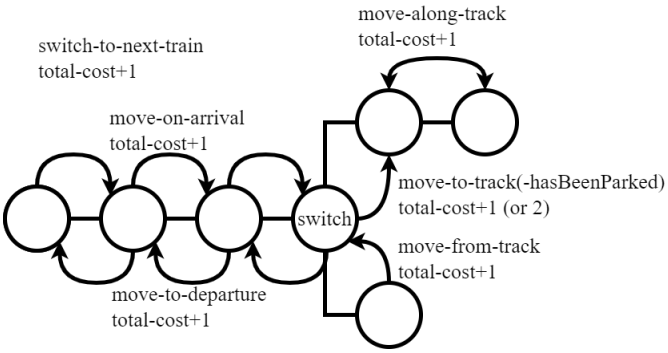

Figure 6: A visualization of the actions in the MSR domain. A train can move in the same manner as in Figure 2, but now only the current train is allowed to move. Each action increases ($total\text{-}cost$) by 1. Moving a train to a track after is was already parked on another track (move-to-track-hasBeenParked) increases ($total\text{-}cost$) by 2.

4.6 Approach 3: PT + MSR

The final approach combines all the domain modifications described in this section, see Figure 7. It contains simplified actions that each increment ($total\text{-}cost$) by 1. In theory, this approach does not differ from approach 1, as with PT we are already ensuring that each train is moved before moving the next train, and reallocating adds additional steps, which planners try to avoid by default. We also do not increment ($total\text{-}cost$) by 2 in move-from-track, as reallocating automatically adds a cost of 2 by using both move-from-track and move-to-track.

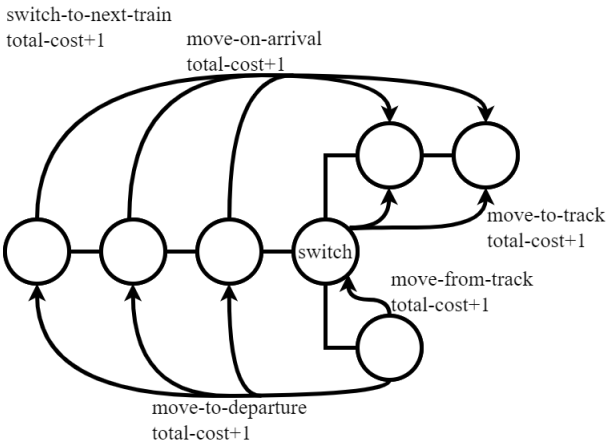

Figure 7: A visualization of the actions in the PT+MSR domains. A train can move in the same manner as shown in Figure 5, but now each action increases ($total\text{-}cost$) by 1.

5 Planner Selection & Analysis

To determine whether our domain modifications make a difference to planner performance, we need to select suitable planners that are compatible with our domain. Since there are many different planners, we have only selected planners from the satisficing track of IPC 2018. After running all the planners that participated in the competition, we only found four planners that were compatible with the domain and were able to find one or more solutions to the problem within our the domain. These planners were: baseline LAMA 2011 [12], DecStar [7], Freelunch-Madagascar [2], and Saarplan [4].

In this section, we will briefly summarize each planner in an attempt to understand how they interpret a problem instance and what operations they perform to solve the problem.

5.1 Baseline: LAMA 2011

LAMA 2011 uses a translator to translate a PDDL problem [12]. It then uses a knowledge compilation module to generate so called landmarks. The final component of the planner, is the search engine, which uses different heuristics to search the data structures generated by the knowledge compilation module.

One of the heuristics used by LAMA 2011 uses, is the landmark heuristic [12]. Landmarks are certain property sets that must hold in every plan that achieves the goal state. In the domain of TUSP, we know that each train must be parked at a track at least once before all trains can depart. The predicate $(hasBeenParked\ train)$ could be considered a landmark of the domain. Landmarks can be seen as sub-goals of a problem, which decomposes the problem into smaller parts [3] They can help the planner avoid landmarks with higher cost. If a no landmark can be reached, then the planner knows that the problem cannot be solved.

5.2 Team 2: DecStar

DecStar is a planner that uses Star-Topology Decoupling (STD) to decompose a problem into smaller components that are independent of each other [7]. It is a technique that reduces the size of the state-space representation by exploiting this independence between components of a planning problem. It partitions the state variables into components, such that the interaction between them takes the form of a star topology, with a single center component and multiple leaf components. It then searches only over action sequences affecting the center component, and enumerates reachable assignments to each leaf component separately. This way, it can avoid exploring many states that are irrelevant or redundant for finding a plan.

However, the planner is not able to find such a split for our domain. So, as a fallback, it switches to the LAMA 2011 planner to find a plan [7].

5.3 Team 4: Freelunch-Madagascar

The Freelunch-Madagascar planner consists of the following components: Freelunch or Madagascar, Incplan, and Lingeling [2].

Freelunch and Madagascar are two different tools used to encodes a PDDL problem in to a SAT (satisfiability) instance [2]. SAT is a logical problem that consists of multiple clauses that contain literals that are either true or false [1]. To solve SAT, we need to find an assignment to each literal such that it satisfies all clauses of the problem instance.

IncPlan is a driver that solves the SAT instance by incrementally calling the final component, Lingeling [2]. Lingeling is a SAT solver that uses inprocessing, a technique that simplifies the SAT instance by removing redundant clauses or literals. The driver gradually adds clauses to the SAT instance and calls the solver until a valid plan is found that achieves the goal state or the instance becomes unsolvable.

5.4 Team 7: Saarplan

The Saarplan planner consists of three components: a similar decoupled search as DecStar, Grey planning, and Refined-HC (RHC) [4].

Grey planning is a technique that finds relaxed plans [4]. A relaxed plan is a plan that ignores (some) of the negative effects of the actions. For instance, if an action normally removes a train from a track part, a relaxed plan may ignore this negation of the predicate. This means that a relaxed plan may involve having multiple trains on the same track part. A relaxed plan is gradually modified by resolving conflicts until it becomes a real plan or it cannot solve a conflict.

RHC is a technique that uses breadth-first search to iteratively find a state with a lower heuristic value and repeats the process from there [4].

Saarplan uses these different techniques to find the best plan in terms of steps or cost [4].

6 Planner Results & Discussion

The performance of the planners described in Section 5 are measured on the different domains. In each domain, the same problem instance is used with six train units and two tracks, with track 1 having four track parts, and track 2 having two track parts, see Appendix A.2 for the code. The problem instance is modified to be compatible with the modified domains, but the objects remain the same. After running the planners, we measure the total execution time in Figure 8, the smallest plan length in Figure 9, number of times switching between trains occurs in the smallest plan in Figure 10, and the total number of plans found in Figure 11.

	initial	PT	MSR	PT+MSR
LAMA 2011	150.50s	0.20s	423.80s	0.75s
DeStar	206.29s	0.34s	591.46s	1.27s
Freelunch-Madagascar	0.01s	0.06s	0.02s	0.11s
Saarplan	7.73s	0.03s	40.84s	0.12s

Figure 8: The total execution time of the planners for each domain, rounded to two decimals.

In Figure 8, the PT domain shows a significant decrease in execution times compared to the initial domain. Combining the actions of the initial domain into more complex actions decreases the number of times predicates are checked in the precondition of actions, and the number of times predicates are changed in the effect of actions. As a result, it decreases

	initial	PT	MSR	PT+MSR
LAMA 2011	70	14	82	26
DeStar	70	14	82	26
Freelunch-Madagascar	74	14	127	27
Saarplan	70	14	127	28

Figure 9: The smallest plan length generated by the planners for each domain. Keep in mind that the plans outputted in MSR and PT+MSR will also contain actions for switching between trains. The complete plans can be found in Appendices E, F, G, and H.

	initial	PT	MSR	PT+MSR
LAMA 2011	38	12	12	12
DeStar	48	12	12	12
Freelunch-Madagascar	56	12	57	13
Saarplan	51	12	51	14

Figure 10: The number of times train switching occurs in the smallest plan of the planners for each domain.

the total execution time. The MSR domain increases execution times as more computations are done compared to the initial domain. The PT+MSR domain decreases computations compared to the initial domain and MSR domain.

In Figure 9, it can be seen that the PT domain allows planners to find a plan with less steps, as many of the steps in the plans for the initial domain are now combined to perform the same operation for the modified domains. The MSR domain increases the steps because of the additional action switch-to-next-train, which is done each time we move a different train than moved in the previous action. Combining these domains in PT+MSR decreases the plan length. The complete plans of the planners for each domain can be found in Appendices E, F, G, and H.

Figures 9 and 10 show that LAMA 2011 and DecStar are able to effectively minimize the total costs by minimizing the number of times switching between trains occurs. DecStar is able to do so as it uses LAMA 2011 as the fallback planner, which is used for these results as explained in Section 5. Interestingly, Freelunch-Madagascar adds a switch-to-next-train at the end of the plan, which is practically redundant.

Figure 11 shows somewhat consistent results with Figure 8. From these two figures, it is clear that the execution time is correlated to the number of plans a planner outputs. If a planner continuously finds a better plan with less steps or cost, it will take more time to execute.

7 Conclusions & Future Work

In this research, we attempt to answer the question: "To what extent can we improve planner performance by optimizing the PDDL domain of TUSP?" Multiple domain modifications were made and selected planners were run on the initial domain and modified domains to achieve the answer to this question.

Combining actions into complex actions and introducing action costs were found to be effective at decreasing the execution time of planners and increasing plan quality by making the plans shorter and less costly to a human driver. They

	initial	PT	MSR	PT+MSR
LAMA 2011	1	1	10	5
DeStar	2	1	8	6
Freelunch-Madagascar	1	1	1	1
Saarplan	2	1	3	1

Figure 11: The number of plans each planner outputs for each domain.

did not affect the solvability of the problem as all planners that were able to solve the problem within the initial domain, were able to so do within the modified domains. Therefore, we can conclude that we have improved the performance of the selected planners in terms of planner execution time and plan quality.

LAMA 2011 shows the best results as it incorporates action costs in the search (unlike Freelunch-Madagascar or Saarplan) and does not make use of other planners that are incompatible with the domains (DecStar).

7.1 Limitations

Unfortunately, this research does not provide results that represent varying problem instances. Due to the limited time period of this research, a single problem instance was used within the different domains, and most of the focus was put on designing and implementing the modifications to the domain. Because of this, we cannot conclude that these modifications will produce the same results for other problem instances that may include a larger shunting yard with more trains.

The complete TUSP domain includes many variables such as varying train unit sizes, different shunting yard types, and time stamps. These variables were not considered in this research, which limits the scope of our findings and contributions.

7.2 Future work

Future research into the domain of TUSP would benefit from the contributions of this research. This includes research into improving planner performance, as well as research focusing on other topics surrounding the same domain. The PDDL domain modifications can be helpful by making planners execute faster and generate plans of higher quality.

Besides TUSP, these modifications can be used as inspiration to optimize other domains with similar action definitions. We believe that any domain that involves the logistics between multiple object types will benefit from the contributions of this research.

8 Responsible Research

We understand the importance of conducting research in a responsible manner. Hence, this section is dedicated to acknowledging the resources used to conduct the research, and providing the steps required to reproduce the same results.

8.1 Resources

To conduct this research, initial PDDL domain and problem instances were provided by our supervisor Issa Hanou, see Appendix A. As well as the remote server to run the planners

on the initial and modified domains[2]. The planners are created by different teams that participated in the classical tracks of IPC 2018[3]. The repositories containing the source code[4] of the planners are publicly available along with the abstracts[5] of each planner.

8.2 Reproducibility

To reproduce the results found in this research, you require the following components: the initial domain, the modified domains, access to the remote server. The source code of the domain and problem instances can be found in Appendices A, B, C, and D.

If read and write access to the server is granted, connect to student-linux.tudelft.nl through port 22 and login with the credentials provided by your administrator. Installing PuTTY is recommended for this[6]. From here, connect to the remote server through ssh:

```
$ssh mapfw.ewi.tudelft.nl
```

Listing 7: Connect to mapfw.ewi.tudelft.nl through ssh. For this, you will need access to student-linux.tudelft.nl.

On the server, you should be able to access your personal directory under /home. Here you will need to pull the repository containing the domains or manually add the domains. After adding the domains, the planners can be executed by calling the image of each planner. Now that we have the necessary components in place, execute:

```
$/data/ipc2018/solvers/sat/team<num>/planner.
    img domain1.pddl problem2.pddl re.out >
    team<num>.log
```

Listing 8: Run the planner on the PDDL domain and problem. This will generate the planner logs and plan outputs.

A log file and plan outputs are generated in the directory this command is executed from. Note that the paths to the planner, domain, and problem files will need to be adjusted depending on their locations on the server.

If you have no access to the server, you will need to setup the planners on your local machine. Documentation is provided to setup a planner to allow running it on your domains locally[3]. This procedure involves installing Singularity and using it to pull the images from the planner repositories[7].

References

[1] Hans van Maaren Armin Biere, Marijn Heule and Toby Walsh. *Handbook of Satisfiability*. IOS Press, 2009.

[2] Toma´s Balyo and Stephan Gocht. The freelunch planning system entering ipc 2018. 2018.

[3] Carmel Domshlak, Michael Katz, and Sagi Lefler. Landmark-enhanced abstraction heuristics. *Artificial Intelligence*, 189:48–68, 2012.

[4] Maximilian Fickert, Daniel Gnad, Patrick Speicher, and Jörg Hoffmann. Saarplan : Combining saarland ' s greatest planning techniques. 2018.

[5] Richard Freling, Ramon M. Lentink, Leo G. Kroon, and Dennis Huisman. Shunting of passenger train units in a railway station. *Transportation Science*, 39(2):261–272, 2005.

[6] Clement Gehring, Masataro Asai, Rohan Chitnis, Tom Silver, Leslie Kaelbling, Shirin Sohrabi, and Michael Katz. Reinforcement learning for classical planning: Viewing heuristics as dense reward generators. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, pages 588–596, 2022.

[7] Daniel Gnad, Alexander Shleyfman, and Jörg Hoffmann. Decstar – star-topology decoupled search at its best. 2018.

[8] Maxence Grand, Humbert Fiorino, and Damien Pellier. Retro-engineering state machines into PDDL domains. In *IEEE International Conference on Tools with Artificial Intelligence*, pages 1186–1193, Baltimore (virtual conference), United States, 2020.

[9] LG. Kroon, RM. Lentink, and A. Schrijver. Shunting of passenger train units: An integrated approach. *Transportation Science*, 42(4):436–449, 2008.

[10] Sheila McIlraith and Ronald Fadel. Planning with complex actions. In *Non-Monotonic Reasoning*, 2002.

[11] Evertjan Peer, Vlado Menkovski, Yingqian Zhang, and Wan-Jui Lee. Shunting trains with deep reinforcement learning. 05 2018.

[12] Silvia Richter, Matthias Westphal, and Malte Helmert. Lama 2008 and 2011. 2011.

[13] Mark Roberts, Adele Howe, and Indrajit Ray. Evaluating diversity in classical planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 24, pages 253–261, 2014.

[14] Dave E. Wilkins. Can ai planners solve practical problems? *Artificial Intelligence Center*, 1990.

[2] mapfw.ewi.tudelft.nl

[3] https://ipc2018-classical.bitbucket.io/

[4] https://bitbucket.org/ipc2018-classical/workspace/repositories

[5] https://ipc2018-classical.bitbucket.io/planner-abstracts/ipc-2018-planner-abstracts-classical-tracks.pdf

[6] https://www.putty.org/

[7] https://sylabs.io/singularity/

A Initial PDDL domain and problem

A.1 Initial Domain

```
(define (domain domain1)

(:requirements :adl)

(:types
    trackpart track trainunit - object
    icm virm sng slt - trainunit ; these are the different types of train units
)

(:predicates
    (nextTo ?x ?y - trackpart) ;track part x next to other track part y
    (onTrack ?x - trackPart ?y - track) ;track part x on track y
    (at ?x - trainunit ?y - trackpart) ;train unit x on track part y
    (hasBeenParked ?x - trainunit) ;true if x is parked on some track
    (free ?x - trackpart) ;trackpart x has nothing parked there
    (parkedOn ?x - trainunit ?y - track) ; indicates x parked on track y
    (onPath ?x) ;trackpart x is on the arrival/departure path L
    (switch ?x) ;trackpart x is a switch
)

; action to move a trainunit to a neighbouring trackpart on a track, to park it
(:action move-to-track
    :parameters (?train - trainunit ?from ?to - trackpart ?t - track)
    :precondition (and (at ?train ?from) (free ?to)
                    (nextTo ?from ?to) (onTrack ?to ?t)
                    (switch ?from))
    :effect (and (at ?train ?to) (not (at ?train ?from))
                    (free ?from) (not (free ?to))
                    (hasBeenParked ?train) (parkedOn ?train ?t))
)

; action to move a trainunit to out of a track, and reset the parkedOn predicate
(:action move-from-track
    :parameters (?train - trainunit ?from ?to - trackpart ?t - track)
    :precondition (and (at ?train ?from) (free ?to)
                    (nextTo ?from ?to) (onTrack ?from ?t)
                    (switch ?to))
    :effect (and (at ?train ?to) (not (at ?train ?from))
                    (free ?from) (not (free ?to))
                    (not (parkedOn ?train ?t)))
)

; action to move a trainunit along a track
(:action move-along-track
    :parameters (?train - trainunit ?from ?to - trackpart ?t - track)
    :precondition (and (at ?train ?from) (free ?to)
                    (nextTo ?from ?to) (onTrack ?from ?t)
                    (onTrack ?to ?t))
    :effect (and (at ?train ?to) (not (at ?train ?from))
                    (free ?from) (not (free ?to)))
)

; Can only move back to departure if all trains have been parked.
(:action move-to-departure
    :parameters (?train - trainunit ?from ?to - trackpart)
    :precondition (and (at ?train ?from) (free ?to)
                    (nextTo ?from ?to) (onPath ?to)
                    (forall (?unit - trainunit) (hasBeenParked ?unit)))
    :effect (and (at ?train ?to) (not (at ?train ?from))
                    (free ?from) (not (free ?to)))
)

; Action to move train unit over the arrival path towards the shunting yard
(:action move-on-arrival
    :parameters (?train - trainunit ?from ?to - trackpart)
    :precondition (and (at ?train ?from) (free ?to)
                    (nextTo ?from ?to) (not (hasBeenParked ?train))
                    (onPath ?from))
    :effect (and (at ?train ?to) (not (at ?train ?from))
                    (free ?from) (not (free ?to)))
)
)
```

Listing 9: Initial domain

A.2 Initial Problem

```
(define (problem problem2) (:domain domain1)
(:objects
   train1 - sng
   train2 - sng
   train3 - icm
   train4 - virm
   train5 - slt
   train6 - slt
   v1 v2 v3 v4 v5 v6 t0 t1 t2 t3 t4 t5 t6  - trackpart
   track1 track2  - track
)
(:init
   (at train1 v1)
   (nextTo v1 v2)
   (nextTo v2 v1)
   (at train2 v2)
   (nextTo v2 v3)
   (nextTo v3 v2)
   (at train3 v3)
   (nextTo v3 v4)
   (nextTo v4 v3)
   (at train4 v4)
   (nextTo v4 v5)
   (nextTo v5 v4)
   (at train5 v5)
   (nextTo v5 v6)
   (nextTo v6 v5)
   (at train6 v6)
   (nextTo v1 t0)
   (nextTo t0 v1)
   (nextTo t0 t1)
   (nextTo t1 t0)
   (free t0)
   (free t1)
   (nextTo t1 t2)
   (nextTo t2 t1)
   (free t2)
   (nextTo t2 t3)
   (nextTo t3 t2)
   (free t3)
   (nextTo t3 t4)
   (nextTo t4 t3)
   (free t4)
   (nextTo t0 t5)
   (nextTo t5 t0)
   (free t5)
   (nextTo t5 t6)
   (nextTo t6 t5)
   (free t6)
   (onTrack t1 track1)
   (onTrack t2 track1)
   (onTrack t3 track1)
   (onTrack t4 track1)
   (onTrack t5 track2)
   (onTrack t6 track2)
   (onPath v1)
   (onPath v2)
   (onPath v3)
   (onPath v4)
   (onPath v5)
   (onPath v6)
   (switch t0)
```

```
)
(:goal (and
   (forall (?t - slt) (or (at ?t v1) (at ?t v4)))
   (forall (?t - sng) (or (at ?t v2) (at ?t v5)))
   (forall (?t - virm) (or (at ?t v3)))
   (forall (?t - icm) (or (at ?t v6)))
   (forall (?t - trainunit) (hasBeenParked ?t))
)))
```

Listing 10: Initial problem

B PT domain and problem

B.1 PT Domain

```
(define (domain domain1)

(:requirements :adl)

(:types
    trackpart track trainunit - object
    icm virm sng slt - trainunit ; these are the different types of train units
)

(:predicates
    (nextTo ?x ?y - trackpart) ;track part x next to other track part y
    (onTrack ?x - trackPart ?y - track) ;track part x on track y
    (at ?x - trainunit ?y - trackpart) ;train unit x on track part y
    (hasBeenParked ?x - trainunit) ;true if x is parked on some track
    (free ?x - trackpart) ;trackpart x has nothing parked there
    (parkedOn ?x - trainunit ?y - track) ; indicates x parked on track y
    (onPath ?x) ;trackpart x is on the arrival/departure path L
    (switch ?x) ;trackpart x is a switch
    (trackHeader ?x - trackpart ?y - track) ; last free trackpart x of track y
    (pathHeader ?x - trackpart) ; last free trackpart x of path L
)

(:action move-from-arrival-to-track
    :parameters (?train - trainunit ?from ?next ?toprev ?to - trackpart ?t - track)
    :precondition (and
        (not (parkedOn ?train ?t))
        (at ?train ?from)
        (onPath ?from)
        (free ?next)
        (free ?to)
        (trackHeader ?to ?t)
        (nextTo ?from ?next)
        (nextTo ?toprev ?to)
        (onTrack ?to ?t)
        (not (hasBeenParked ?train))
        (exists (?switch - trackpart) (and (switch ?switch) (free ?switch))))
    :effect (and
        (at ?train ?to) (not (at ?train ?from))
        (free ?from) (not (free ?to))
        (pathHeader ?from) (not (pathHeader ?next))
        (when (not (switch ?toprev)) (trackHeader ?toprev ?t)) (not (trackHeader ?to ?t))
        (hasBeenParked ?train)
        (parkedOn ?train ?t))
)

(:action move-from-track-to-departure
    :parameters (?train - trainunit ?from ?next ?toprev ?to - trackpart ?t - track)
    :precondition (and
        (parkedOn ?train ?t)
        (at ?train ?from)
        (onTrack ?from ?t)
        (free ?next)
        (free ?to)
        (nextTo ?next ?from)
        (nextTo ?to ?toprev)
        (onPath ?to)
        (pathHeader ?to)
        (forall (?unit - trainunit) (hasBeenParked ?unit))
        (exists (?switch - trackpart) (and (switch ?switch) (free ?switch))))
    :effect (and
        (at ?train ?to) (not (at ?train ?from))
        (free ?from) (not (free ?to))
        (trackHeader ?from ?t) (not (trackHeader ?next ?t))
        (when (not (switch ?toprev)) (pathHeader ?toprev))
        (not (pathHeader ?to))
        (not (parkedOn ?train ?t)))
)

(:action move-from-switch-to-track
    :parameters (?train - trainunit ?from ?toprev ?to - trackpart ?t - track)
    :precondition (and
        (not (parkedOn ?train ?t))
        (at ?train ?from)
        (free ?to)
        (trackHeader ?to ?t)
        (nextTo ?toprev ?to)
        (onTrack ?to ?t)
        (switch ?from)
        (forall (?unit - trainunit) (hasBeenParked ?unit))
    )
    :effect (and
        (at ?train ?to)         (not (at ?train ?from))
        (free ?from)            (not (free ?to))
        (when (not (switch ?toprev)) (trackHeader ?toprev ?t))
        (not (trackHeader ?to ?t))
        (parkedOn ?train ?t))
)

(:action move-from-track-to-switch
    :parameters (?train - trainunit ?from ?next ?to - trackpart ?t - track)
    :precondition (and
        (parkedOn ?train ?t)
        (at ?train ?from)
        (onTrack ?from ?t)
        (nextTo ?next ?from)
        (free ?next)
        (free ?to)
        (switch ?to)
        (forall (?unit - trainunit) (hasBeenParked ?unit))
    )
    :effect (and
        (at ?train ?to)         (not (at ?train ?from))
        (free ?from)            (not (free ?to))
        (trackHeader ?from ?t)  (not (trackHeader ?next ?t))
        (not (parkedOn ?train ?t)))
)
)
```

Listing 11: PT domain

B.2 PT Problem

```
(define (problem problem2) (:domain domain1)
(:objects
   train1 - sng
   train2 - sng
   train3 - icm
   train4 - virm
   train5 - slt
   train6 - slt
   v1 v2 v3 v4 v5 v6 t0 t1 t2 t3 t4 t5 t6 - trackpart
   track1 track2 - track
)
(:init
```

```
   (at train1 v1)
   (nextTo v2 v1)
   (at train2 v2)
   (nextTo v3 v2)
   (at train3 v3)
   (nextTo v4 v3)
   (at train4 v4)
   (nextTo v5 v4)
   (at train5 v5)
   (nextTo v6 v5)
   (at train6 v6)
   (nextTo v1 t0)
   (nextTo t0 t1)
   (free t0)
   (free t1)
   (nextTo t1 t2)
   (free t2)
   (nextTo t2 t3)
   (free t3)
   (nextTo t3 t4)
   (free t4)
   (nextTo t0 t5)
   (free t5)
   (nextTo t5 t6)
   (free t6)
   (onTrack t1 track1)
   (onTrack t2 track1)
   (onTrack t3 track1)
   (onTrack t4 track1)
   (onTrack t5 track2)
   (onTrack t6 track2)
   (trackHeader t4 track1)
   (trackHeader t6 track2)
   (onPath v1)
   (onPath v2)
   (onPath v3)
   (onPath v4)
   (onPath v5)
   (onPath v6)
   (switch t0)
)
(:goal (and
   (forall (?t - slt) (or (at ?t v1) (at ?t v4)))
   (forall (?t - sng) (or (at ?t v2) (at ?t v5)))
   (forall (?t - virm) (or (at ?t v3)))
   (forall (?t - icm) (or (at ?t v6)))
   (forall (?t - trainunit) (hasBeenParked ?t))
)))
```

Listing 12: PT problem

C MSR domain and problem

C.1 MSR Domain

```
(define (domain domain1)

(:requirements :adl :action-costs :conditional-effects)

(:types
    trackpart track trainunit - object
    icm virm sng slt - trainunit ; these are the different
      types of train units
)

(:predicates
    (nextTo ?x ?y - trackpart) ;track part x next to other
      track part y
    (onTrack ?x - trackPart ?y - track) ;track part x on
     track y
    (at ?x - trainunit ?y - trackpart) ;train unit x on
     track part y
    (hasBeenParked ?x - trainunit) ;true if x is parked on
      some track
    (free ?x - trackpart) ;trackpart x has nothing parked
     there
    (parkedOn ?x - trainunit ?y - track) ; indicates x
     parked on track y
    (onPath ?x - trackpart) ;trackpart x is on the arrival
     /departure path L
    (switch ?x - trackpart) ;trackpart x is a switch
    (currtrain ?x - trainunit) ; current train unit x that
      is being moved
)

(:functions
    (total-cost)
)

(:action switch-to-next-train
    :parameters (?prev ?next - trainunit)
    :precondition (and
        (currtrain ?prev)
        (not (currtrain ?next)))
    :effect (and
        (currtrain ?next)
        (not (currtrain ?prev))
        (increase (total-cost) 1))
)

; action to move a trainunit to a neighbouring trackpart
     on a track, to park it
(:action move-to-track
    :parameters (?train - trainunit ?from ?to - trackpart
     ?t - track)
    :precondition (and (at ?train ?from) (free ?to)
                    (nextTo ?from ?to) (onTrack ?to ?t)
                    (switch ?from)
                    (not (hasBeenParked ?train))
                    (currtrain ?train))
    :effect (and (at ?train ?to) (not (at ?train ?from))
                    (free ?from) (not (free ?to))
                    (hasBeenParked ?train) (parkedOn ?
     train ?t)
                    (increase (total-cost) 1))
)

(:action reallocate-to-track
    :parameters (?train - trainunit ?from ?to - trackpart
     ?t - track)
    :precondition (and (at ?train ?from) (free ?to)
                    (nextTo ?from ?to) (onTrack ?to ?t)
                    (switch ?from)
                    (hasBeenParked ?train)
                    (currtrain ?train))
    :effect (and (at ?train ?to) (not (at ?train ?from))
                    (free ?from) (not (free ?to))
                    (parkedOn ?train ?t)
                    (increase (total-cost) 2))
)

; action to move a trainunit to out of a track, and reset
     the parkedOn predicate
(:action move-from-track
    :parameters (?train - trainunit ?from ?to - trackpart
     ?t - track)
    :precondition (and (at ?train ?from) (free ?to)
                    (nextTo ?from ?to) (onTrack ?from ?t)
                    (switch ?to)
                    (currtrain ?train))
    :effect (and (at ?train ?to) (not (at ?train ?from))
                    (free ?from) (not (free ?to))
                    (not (parkedOn ?train ?t))
                    (increase (total-cost) 1))
)

; action to move a trainunit along a track
(:action move-along-track
    :parameters (?train - trainunit ?from ?to - trackpart
     ?t - track)
    :precondition (and (at ?train ?from) (free ?to)
                    (nextTo ?from ?to) (onTrack ?from ?t)
                    (onTrack ?to ?t)
                    (currtrain ?train))
    :effect (and (at ?train ?to) (not (at ?train ?from))
                    (free ?from) (not (free ?to))
                    (increase (total-cost) 1))
)

```

```
; Can only move back to departure if all trains have been parked.
(:action move-to-departure
    :parameters (?train - trainunit ?from ?to - trackpart)
    :precondition (and (at ?train ?from) (free ?to)
                       (nextTo ?from ?to) (onPath ?to)
                       (forall (?unit - trainunit) (
     hasBeenParked ?unit))
                       (currtrain ?train))
    :effect (and (at ?train ?to) (not (at ?train ?from))
                       (free ?from) (not (free ?to))
                       (increase (total-cost) 1))
)

; Action to move train unit over the arrival path towards the shunting yard
(:action move-on-arrival
    :parameters (?train - trainunit ?from ?to - trackpart)
    :precondition (and (at ?train ?from) (free ?to)
                       (nextTo ?from ?to) (not (hasBeenParked ?train))
                       (onPath ?from)
                       (currtrain ?train))
    :effect (and (at ?train ?to) (not (at ?train ?from))
                       (free ?from) (not (free ?to))
                       (increase (total-cost) 1))
)
)
```

Listing 13: MSR domain

C.2 MSR Problem

```
(define (problem problem2) (:domain domain1)
(:objects
   train1 - sng
   train2 - sng
   train3 - icm
   train4 - virm
   train5 - slt
   train6 - slt
   v1 v2 v3 v4 v5 v6 t0 t1 t2 t3 t4 t5 t6  - trackpart
   track1 track2  - track
)
(:init
   (= (total-cost) 0)
   (currtrain train1)
   (at train1 v1)
   (nextTo v1 v2)
   (nextTo v2 v1)
   (at train2 v2)
   (nextTo v2 v3)
   (nextTo v3 v2)
   (at train3 v3)
   (nextTo v3 v4)
   (nextTo v4 v3)
   (at train4 v4)
   (nextTo v4 v5)
   (nextTo v5 v4)
   (at train5 v5)
   (nextTo v5 v6)
   (nextTo v6 v5)
   (at train6 v6)
   (nextTo v1 t0)
   (nextTo t0 v1)
   (nextTo t0 t1)
   (nextTo t1 t0)
   (free t0)
   (free t1)
   (nextTo t1 t2)
   (nextTo t2 t1)
   (free t2)
   (nextTo t2 t3)
   (nextTo t3 t2)
   (free t3)
   (nextTo t3 t4)
   (nextTo t4 t3)
   (free t4)
   (nextTo t0 t5)
   (nextTo t5 t0)
   (free t5)
   (nextTo t5 t6)
   (nextTo t6 t5)
   (free t6)
   (onTrack t1 track1)
   (onTrack t2 track1)
   (onTrack t3 track1)
   (onTrack t4 track1)
   (onTrack t5 track2)
   (onTrack t6 track2)
   (onPath v1)
   (onPath v2)
   (onPath v3)
   (onPath v4)
   (onPath v5)
   (onPath v6)
   (switch t0)
)
(:goal (and
   (forall (?t - slt) (or (at ?t v1) (at ?t v4)))
   (forall (?t - sng) (or (at ?t v2) (at ?t v5)))
   (forall (?t - virm) (or (at ?t v3)))
   (forall (?t - icm) (or (at ?t v6)))
   (forall (?t - trainunit) (hasBeenParked ?t))
))
(:metric minimize (total-cost)))
```

Listing 14: MSR problem

D PT+MSR domain and problem

D.1 PT+MSR Domain

```
(define (domain domain1)

(:requirements :adl :action-costs)

(:types
    trackpart track trainunit - object
    icm virm sng slt - trainunit ; these are the different types of train units
)

(:predicates
    (nextTo ?x ?y - trackpart) ;track part x next to other track part y
    (onTrack ?x - trackPart ?y - track) ;track part x on track y
    (at ?x - trainunit ?y - trackpart) ;train unit x on track part y
    (hasBeenParked ?x - trainunit) ;true if x is parked on some track
    (free ?x - trackpart) ;trackpart x has nothing parked there
    (parkedOn ?x - trainunit ?y - track) ; indicates x parked on track y
    (onPath ?x) ;trackpart x is on the arrival/departure path L
    (switch ?x) ;trackpart x is a switch
    (trackHeader ?x - trackpart ?y - track) ; last free trackpart x of track y
    (pathHeader ?x - trackpart) ; last free trackpart x of path L
    (currtrain ?x - trainunit) ; current train unit x that is being moved
)

(:functions
    (total-cost)
)

(:action switch-to-next-train
    :parameters (?prev ?next - trainunit)
    :precondition (and
        (currtrain ?prev)
        (not (currtrain ?next)))
    :effect (and
        (currtrain ?next)
        (not (currtrain ?prev))
```

```
        (increase (total-cost) 1))
)

(:action move-from-arrival
    :parameters (?train - trainunit ?from ?next ?toprev ?to - trackpart ?t - track)
    :precondition (and
        (not (parkedOn ?train ?t))
        (at ?train ?from)
        (onPath ?from)
        (free ?next)
        (free ?to)
        (trackHeader ?to ?t)
        (nextTo ?from ?next)
        (nextTo ?toprev ?to)
        (onTrack ?to ?t)
        (not (hasBeenParked ?train))
        (exists (?switch - trackpart) (and (switch ?switch) (free ?switch)))
        (currtrain ?train))
    :effect (and
        (at ?train ?to) (not (at ?train ?from))
        (free ?from) (not (free ?to))
        (pathHeader ?from) (not (pathHeader ?next))
        (when (not (switch ?toprev)) (trackHeader ?toprev ?t)) (not (trackHeader ?to ?t))
        (hasBeenParked ?train)
        (parkedOn ?train ?t)
        (increase (total-cost) 1))
)

(:action move-to-departure
    :parameters (?train - trainunit ?from ?next ?toprev ?to - trackpart ?t - track)
    :precondition (and
        (parkedOn ?train ?t)
        (at ?train ?from)
        (onTrack ?from ?t)
        (free ?next)
        (free ?to)
        (nextTo ?next ?from)
        (nextTo ?to ?toprev)
        (onPath ?to)
        (pathHeader ?to)
        (forall (?unit - trainunit) (hasBeenParked ?unit))
        (exists (?switch - trackpart) (and (switch ?switch) (free ?switch)))
        (currtrain ?train))
    :effect (and
        (at ?train ?to) (not (at ?train ?from))
        (free ?from) (not (free ?to))
        (trackHeader ?from ?t) (not (trackHeader ?next ?t))
        (when (not (switch ?toprev)) (pathHeader ?toprev)) (not (pathHeader ?to))
        (not (parkedOn ?train ?t))
        (increase (total-cost) 1))
)

(:action move-to-track
    :parameters (?train - trainunit ?from ?toprev ?to - trackpart ?t - track)
    :precondition (and
        (not (parkedOn ?train ?t))
        (at ?train ?from)
        (free ?to)
        (trackHeader ?to ?t)
        (nextTo ?toprev ?to)
        (onTrack ?to ?t)
        (switch ?from)
        (forall (?unit - trainunit) (hasBeenParked ?unit))
        (currtrain ?train))
    :effect (and
        (at ?train ?to)         (not (at ?train ?from))
        (free ?from)            (not (free ?to))
        (when (not (switch ?toprev)) (trackHeader ?toprev ?t))
        (not (trackHeader ?to ?t))
        (parkedOn ?train ?t)
        (increase (total-cost) 1))
)

(:action move-from-track
    :parameters (?train - trainunit ?from ?next ?to - trackpart ?t - track)
    :precondition (and
        (parkedOn ?train ?t)
        (at ?train ?from)
        (onTrack ?from ?t)
        (nextTo ?next ?from)
        (free ?next)
        (free ?to)
        (switch ?to)
        (forall (?unit - trainunit) (hasBeenParked ?unit))
        (currtrain ?train))
    :effect (and
        (at ?train ?to)         (not (at ?train ?from))
        (free ?from)            (not (free ?to))
        (trackHeader ?from ?t)  (not (trackHeader ?next ?t))
        (not (parkedOn ?train ?t))
        (increase (total-cost) 1))
)
)
```

Listing 15: PT+MSR domain

D.2 PT+MSR Problem

```
(define (problem problem2) (:domain domain1)
(:objects
   train1 - sng
   train2 - sng
   train3 - icm
   train4 - virm
   train5 - slt
   train6 - slt
   v1 v2 v3 v4 v5 v6 t0 t1 t2 t3 t4 t5 t6 - trackpart
   track1 track2 - track
)
(:init
   (= (total-cost) 0)
   (currtrain train1)
   (at train1 v1)
   (nextTo v2 v1)
   (at train2 v2)
   (nextTo v3 v2)
   (at train3 v3)
   (nextTo v4 v3)
   (at train4 v4)
   (nextTo v5 v4)
   (at train5 v5)
   (nextTo v6 v5)
   (at train6 v6)
   (nextTo v1 t0)
   (nextTo t0 t1)
   (free t0)
   (free t1)
   (nextTo t1 t2)
   (free t2)
   (nextTo t2 t3)
   (free t3)
   (nextTo t3 t4)
   (free t4)
   (nextTo t0 t5)
   (free t5)
   (nextTo t5 t6)
   (free t6)
   (onTrack t1 track1)
   (onTrack t2 track1)
   (onTrack t3 track1)
   (onTrack t4 track1)
   (onTrack t5 track2)
   (onTrack t6 track2)
   (trackHeader t4 track1)
   (trackHeader t6 track2)
   (onPath v1)
   (onPath v2)
   (onPath v3)
   (onPath v4)
   (onPath v5)
```

```
   (onPath v6)
   (switch t0)
)
(:goal (and
   (forall (?t - slt) (or (at ?t v1) (at ?t v4)))
   (forall (?t - sng) (or (at ?t v2) (at ?t v5)))
   (forall (?t - virm) (or (at ?t v3)))
   (forall (?t - icm) (or (at ?t v6)))
   (forall (?t - trainunit) (hasBeenParked ?t))
))
(:metric minimize (total-cost)))
```

Listing 16: PT+MSR problem

E LAMA 2011 Output Plans

The plans generated by LAMA 2011 in each domain. Only the plans with the shortest plan length are included.

E.1 Initial domain

```
(move-on-arrival train1 v1 t0)
(move-to-track train1 t0 t5 track2)
(move-on-arrival train2 v2 v1)
(move-on-arrival train2 v1 t0)
(move-to-track train2 t0 t1 track1)
(move-on-arrival train3 v3 v2)
(move-on-arrival train3 v2 v1)
(move-on-arrival train3 v1 t0)
(move-on-arrival train4 v4 v3)
(move-on-arrival train4 v3 v2)
(move-on-arrival train4 v2 v1)
(move-on-arrival train5 v5 v4)
(move-on-arrival train5 v4 v3)
(move-on-arrival train5 v3 v2)
(move-on-arrival train6 v6 v5)
(move-on-arrival train6 v5 v4)
(move-on-arrival train6 v4 v3)
(move-along-track train1 t5 t6 track2)
(move-to-track train3 t0 t5 track2)
(move-on-arrival train4 v1 t0)
(move-on-arrival train5 v2 v1)
(move-on-arrival train6 v3 v2)
(move-along-track train2 t1 t2 track1)
(move-to-track train4 t0 t1 track1)
(move-on-arrival train5 v1 t0)
(move-on-arrival train6 v2 v1)
(move-along-track train2 t2 t3 track1)
(move-along-track train4 t1 t2 track1)
(move-to-track train5 t0 t1 track1)
(move-on-arrival train6 v1 t0)
(move-along-track train2 t3 t4 track1)
(move-along-track train4 t2 t3 track1)
(move-along-track train5 t1 t2 track1)
(move-to-track train6 t0 t1 track1)
(move-from-track train3 t5 t0 track2)
(move-to-departure train3 t0 v1)
(move-to-departure train3 v1 v2)
(move-to-departure train3 v2 v3)
(move-to-departure train3 v3 v4)
(move-to-departure train3 v4 v5)
(move-to-departure train3 v5 v6)
(move-along-track train1 t6 t5 track2)
(move-from-track train1 t5 t0 track2)
(move-to-departure train1 t0 v1)
(move-to-departure train1 v1 v2)
(move-to-departure train1 v2 v3)
(move-to-departure train1 v3 v4)
(move-to-departure train1 v4 v5)
(move-from-track train6 t1 t0 track1)
(move-to-departure train6 t0 v1)
(move-along-track train5 t2 t1 track1)
(move-to-departure train6 v1 v2)
(move-to-departure train6 v2 v3)
(move-to-departure train6 v3 v4)
(move-along-track train4 t3 t2 track1)
(move-along-track train2 t4 t3 track1)
(move-from-track train5 t1 t0 track1)
(move-along-track train4 t2 t1 track1)
(move-to-track train5 t0 t5 track2)
(move-along-track train2 t3 t2 track1)
(move-from-track train4 t1 t0 track1)
(move-to-departure train4 t0 v1)
(move-along-track train2 t2 t1 track1)
(move-to-departure train4 v1 v2)
(move-to-departure train4 v2 v3)
(move-from-track train2 t1 t0 track1)
(move-to-departure train2 t0 v1)
(move-to-departure train2 v1 v2)
(move-from-track train5 t5 t0 track2)
(move-to-departure train5 t0 v1)
; cost = 70 (unit cost)
```

Listing 17: LAMA 2011 output plan in the initial domain.

E.2 PT domain

```
(move-from-arrival-to-track train1 v1 t0 t3 t4 track1)
(move-from-arrival-to-track train2 v2 v1 t5 t6 track2)
(move-from-arrival-to-track train3 v3 v2 t0 t5 track2)
(move-from-arrival-to-track train4 v4 v3 t2 t3 track1)
(move-from-arrival-to-track train5 v5 v4 t1 t2 track1)
(move-from-arrival-to-track train6 v6 v5 t0 t1 track1)
(move-from-track-to-departure train3 t5 t0 v5 v6 track2)
(move-from-track-to-departure train2 t6 t5 v4 v5 track2)
(move-from-track-to-departure train6 t1 t0 v3 v4 track1)
(move-from-track-to-switch train5 t2 t1 t0 track1)
(move-from-switch-to-track train5 t0 t5 t6 track2)
(move-from-track-to-departure train4 t3 t2 v2 v3 track1)
(move-from-track-to-departure train1 t4 t3 v1 v2 track1)
(move-from-track-to-departure train5 t6 t5 t0 v1 track2)
; cost = 14 (unit cost)
```

Listing 18: LAMA 2011 output plan in PT.

E.3 MSR domain

```
(move-on-arrival train1 v1 t0)
(move-to-track train1 t0 t1 track1)
(move-along-track train1 t1 t2 track1)
(move-along-track train1 t2 t3 track1)
(move-along-track train1 t3 t4 track1)
(switch-to-next-train train1 train2)
(move-on-arrival train2 v2 v1)
(move-on-arrival train2 v1 t0)
(move-to-track train2 t0 t5 track2)
(move-along-track train2 t5 t6 track2)
(switch-to-next-train train2 train3)
(move-on-arrival train3 v3 v2)
(move-on-arrival train3 v2 v1)
(move-on-arrival train3 v1 t0)
(move-to-track train3 t0 t5 track2)
(switch-to-next-train train3 train4)
(move-on-arrival train4 v4 v3)
(move-on-arrival train4 v3 v2)
(move-on-arrival train4 v2 v1)
(move-on-arrival train4 v1 t0)
(move-to-track train4 t0 t1 track1)
(move-along-track train4 t1 t2 track1)
(move-along-track train4 t2 t3 track1)
(switch-to-next-train train4 train5)
(move-on-arrival train5 v5 v4)
(move-on-arrival train5 v4 v3)
(move-on-arrival train5 v3 v2)
(move-on-arrival train5 v2 v1)
(move-on-arrival train5 v1 t0)
(move-to-track train5 t0 t1 track1)
(move-along-track train5 t1 t2 track1)
(switch-to-next-train train5 train6)
(move-on-arrival train6 v6 v5)
(move-on-arrival train6 v5 v4)
(move-on-arrival train6 v4 v3)
(move-on-arrival train6 v3 v2)
(move-on-arrival train6 v2 v1)
(move-on-arrival train6 v1 t0)
(move-to-track train6 t0 t1 track1)
```

```
(switch-to-next-train train6 train3)
(move-from-track train3 t5 t0 track2)
(move-to-departure train3 t0 v1)
(move-to-departure train3 v1 v2)
(move-to-departure train3 v2 v3)
(move-to-departure train3 v3 v4)
(move-to-departure train3 v4 v5)
(move-to-departure train3 v5 v6)
(switch-to-next-train train3 train2)
(move-along-track train2 t6 t5 track2)
(move-from-track train2 t5 t0 track2)
(move-to-departure train2 t0 v1)
(move-to-departure train2 v1 v2)
(move-to-departure train2 v2 v3)
(move-to-departure train2 v3 v4)
(move-to-departure train2 v4 v5)
(switch-to-next-train train2 train6)
(move-from-track train6 t1 t0 track1)
(move-to-departure train6 t0 v1)
(move-to-departure train6 v1 v2)
(move-to-departure train6 v2 v3)
(move-to-departure train6 v3 v4)
(switch-to-next-train train6 train5)
(move-along-track train5 t2 t1 track1)
(move-from-track train5 t1 t0 track1)
(reallocate-to-track train5 t0 t5 track2)
(switch-to-next-train train5 train4)
(move-along-track train4 t3 t2 track1)
(move-along-track train4 t2 t1 track1)
(move-from-track train4 t1 t0 track1)
(move-to-departure train4 t0 v1)
(move-to-departure train4 v1 v2)
(move-to-departure train4 v2 v3)
(switch-to-next-train train4 train1)
(move-along-track train1 t4 t3 track1)
(move-along-track train1 t3 t2 track1)
(move-along-track train1 t2 t1 track1)
(move-from-track train1 t1 t0 track1)
(move-to-departure train1 t0 v1)
(move-to-departure train1 v1 v2)
(switch-to-next-train train1 train5)
(move-from-track train5 t5 t0 track2)
(move-to-departure train5 t0 v1)
; cost = 83 (general cost)
```

Listing 19: LAMA 2011 output plan in MSR.

E.4 PT+MSR domain

```
(move-from-arrival train1 v1 t0 t5 t6 track2)
(switch-to-next-train train1 train2)
(move-from-arrival train2 v2 v1 t3 t4 track1)
(switch-to-next-train train2 train3)
(move-from-arrival train3 v3 v2 t0 t5 track2)
(switch-to-next-train train3 train4)
(move-from-arrival train4 v4 v3 t2 t3 track1)
(switch-to-next-train train4 train5)
(move-from-arrival train5 v5 v4 t1 t2 track1)
(switch-to-next-train train5 train6)
(move-from-arrival train6 v6 v5 t0 t1 track1)
(switch-to-next-train train6 train3)
(move-to-departure train3 t5 t0 v5 v6 track2)
(switch-to-next-train train3 train1)
(move-to-departure train1 t6 t5 v4 v5 track2)
(switch-to-next-train train1 train6)
(move-from-track train6 t1 t0 t0 track1)
(move-to-track train6 t0 t5 t6 track2)
(switch-to-next-train train6 train5)
(move-to-departure train5 t2 t1 v3 v4 track1)
(switch-to-next-train train5 train4)
(move-to-departure train4 t3 t2 v2 v3 track1)
(switch-to-next-train train4 train2)
(move-to-departure train2 t4 t3 v1 v2 track1)
(switch-to-next-train train2 train6)
(move-to-departure train6 t6 t5 t0 v1 track2)
; cost = 26 (unit cost)
```

Listing 20: LAMA 2011 output plan in PT+MSR.

F DecStar Output Plans

The plans generated by DecStar in each domain. Only the plans with the shortest plan length are included.

F.1 Initial domain

```
(move-on-arrival train1 v1 t0)
(move-to-track train1 t0 t5 track2)
(move-on-arrival train2 v2 v1)
(move-on-arrival train2 v1 t0)
(move-on-arrival train3 v3 v2)
(move-on-arrival train3 v2 v1)
(move-on-arrival train4 v4 v3)
(move-on-arrival train4 v3 v2)
(move-on-arrival train5 v5 v4)
(move-on-arrival train5 v4 v3)
(move-on-arrival train6 v6 v5)
(move-to-track train2 t0 t1 track1)
(move-on-arrival train3 v1 t0)
(move-along-track train1 t5 t6 track2)
(move-on-arrival train4 v2 v1)
(move-on-arrival train5 v3 v2)
(move-on-arrival train6 v5 v4)
(move-to-track train3 t0 t5 track2)
(move-on-arrival train6 v4 v3)
(move-on-arrival train4 v1 t0)
(move-on-arrival train5 v2 v1)
(move-on-arrival train6 v3 v2)
(move-along-track train2 t1 t2 track1)
(move-to-track train4 t0 t1 track1)
(move-on-arrival train5 v1 t0)
(move-on-arrival train6 v2 v1)
(move-along-track train2 t2 t3 track1)
(move-along-track train4 t1 t2 track1)
(move-to-track train5 t0 t1 track1)
(move-on-arrival train6 v1 t0)
(move-along-track train2 t3 t4 track1)
(move-along-track train4 t2 t3 track1)
(move-along-track train5 t1 t2 track1)
(move-to-track train6 t0 t1 track1)
(move-from-track train3 t5 t0 track2)
(move-to-departure train3 t0 v1)
(move-to-departure train3 v1 v2)
(move-to-departure train3 v2 v3)
(move-to-departure train3 v3 v4)
(move-to-departure train3 v4 v5)
(move-to-departure train3 v5 v6)
(move-along-track train1 t6 t5 track2)
(move-from-track train1 t5 t0 track2)
(move-to-departure train1 t0 v1)
(move-to-departure train1 v1 v2)
(move-to-departure train1 v2 v3)
(move-to-departure train1 v3 v4)
(move-to-departure train1 v4 v5)
(move-from-track train6 t1 t0 track1)
(move-to-departure train6 t0 v1)
(move-along-track train5 t2 t1 track1)
(move-along-track train4 t3 t2 track1)
(move-from-track train5 t1 t0 track1)
(move-along-track train2 t4 t3 track1)
(move-along-track train4 t2 t1 track1)
(move-to-track train5 t0 t5 track2)
(move-along-track train2 t3 t2 track1)
(move-from-track train4 t1 t0 track1)
(move-along-track train2 t2 t1 track1)
(move-to-departure train6 v1 v2)
(move-to-departure train4 t0 v1)
(move-to-departure train6 v2 v3)
(move-to-departure train4 v1 v2)
(move-to-departure train6 v3 v4)
(move-to-departure train4 v2 v3)
(move-from-track train2 t1 t0 track1)
(move-to-departure train2 t0 v1)
(move-to-departure train2 v1 v2)
(move-from-track train5 t5 t0 track2)
(move-to-departure train5 t0 v1)
; cost = 70 (unit cost)
```

Listing 21: DecStar output plan in the initial domain.

F.2 PT domain

```
(move-from-arrival-to-track train1 v1 t0 t3 t4 track1)
(move-from-arrival-to-track train2 v2 v1 t5 t6 track2)
(move-from-arrival-to-track train3 v3 v2 t0 t5 track2)
(move-from-arrival-to-track train4 v4 v3 t2 t3 track1)
(move-from-arrival-to-track train5 v5 v4 t1 t2 track1)
(move-from-arrival-to-track train6 v6 v5 t0 t1 track1)
(move-from-track-to-departure train3 t5 t0 v5 v6 track2)
(move-from-track-to-departure train2 t6 t5 v4 v5 track2)
(move-from-track-to-departure train6 t1 t0 v3 v4 track1)
(move-from-track-to-switch train5 t2 t1 t0 track1)
(move-from-switch-to-track train5 t0 t5 t6 track2)
(move-from-track-to-departure train4 t3 t2 v2 v3 track1)
(move-from-track-to-departure train1 t4 t3 v1 v2 track1)
(move-from-track-to-departure train5 t6 t5 t0 v1 track2)
; cost = 14 (unit cost)
```

Listing 22: DecStar output plan in PT.

F.3 MSR domain

```
(move-on-arrival train1 v1 t0)
(move-to-track train1 t0 t1 track1)
(move-along-track train1 t1 t2 track1)
(move-along-track train1 t2 t3 track1)
(move-along-track train1 t3 t4 track1)
(switch-to-next-train train1 train2)
(move-on-arrival train2 v2 v1)
(move-on-arrival train2 v1 t0)
(move-to-track train2 t0 t5 track2)
(move-along-track train2 t5 t6 track2)
(switch-to-next-train train2 train3)
(move-on-arrival train3 v3 v2)
(move-on-arrival train3 v2 v1)
(move-on-arrival train3 v1 t0)
(move-to-track train3 t0 t5 track2)
(switch-to-next-train train3 train4)
(move-on-arrival train4 v4 v3)
(move-on-arrival train4 v3 v2)
(move-on-arrival train4 v2 v1)
(move-on-arrival train4 v1 t0)
(move-to-track train4 t0 t1 track1)
(move-along-track train4 t1 t2 track1)
(move-along-track train4 t2 t3 track1)
(switch-to-next-train train4 train5)
(move-on-arrival train5 v5 v4)
(move-on-arrival train5 v4 v3)
(move-on-arrival train5 v3 v2)
(move-on-arrival train5 v2 v1)
(move-on-arrival train5 v1 t0)
(move-to-track train5 t0 t1 track1)
(move-along-track train5 t1 t2 track1)
(switch-to-next-train train5 train6)
(move-on-arrival train6 v6 v5)
(move-on-arrival train6 v5 v4)
(move-on-arrival train6 v4 v3)
(move-on-arrival train6 v3 v2)
(move-on-arrival train6 v2 v1)
(move-on-arrival train6 v1 t0)
(move-to-track train6 t0 t1 track1)
(switch-to-next-train train6 train3)
(move-from-track train3 t5 t0 track2)
(move-to-departure train3 t0 v1)
(move-to-departure train3 v1 v2)
(move-to-departure train3 v2 v3)
(move-to-departure train3 v3 v4)
(move-to-departure train3 v4 v5)
(move-to-departure train3 v5 v6)
(switch-to-next-train train3 train2)
(move-along-track train2 t6 t5 track2)
(move-from-track train2 t5 t0 track2)
(move-to-departure train2 t0 v1)
(move-to-departure train2 v1 v2)
(move-to-departure train2 v2 v3)
(move-to-departure train2 v3 v4)
(move-to-departure train2 v4 v5)
(switch-to-next-train train2 train6)
(move-from-track train6 t1 t0 track1)
(move-to-departure train6 t0 v1)
(move-to-departure train6 v1 v2)
(move-to-departure train6 v2 v3)
(move-to-departure train6 v3 v4)
(switch-to-next-train train6 train5)
(move-along-track train5 t2 t1 track1)
(move-from-track train5 t1 t0 track1)
(reallocate-to-track train5 t0 t5 track2)
(switch-to-next-train train5 train4)
(move-along-track train4 t3 t2 track1)
(move-along-track train4 t2 t1 track1)
(move-from-track train4 t1 t0 track1)
(move-to-departure train4 t0 v1)
(move-to-departure train4 v1 v2)
(move-to-departure train4 v2 v3)
(switch-to-next-train train4 train1)
(move-along-track train1 t4 t3 track1)
(move-along-track train1 t3 t2 track1)
(move-along-track train1 t2 t1 track1)
(move-from-track train1 t1 t0 track1)
(move-to-departure train1 t0 v1)
(move-to-departure train1 v1 v2)
(switch-to-next-train train1 train5)
(move-from-track train5 t5 t0 track2)
(move-to-departure train5 t0 v1)
; cost = 83 (general cost)
```

Listing 23: DecStar output plan in MSR.

F.4 PT+MSR domain

```
(move-from-arrival train1 v1 t0 t3 t4 track1)
(switch-to-next-train train1 train2)
(move-from-arrival train2 v2 v1 t5 t6 track2)
(switch-to-next-train train2 train3)
(move-from-arrival train3 v3 v2 t0 t5 track2)
(switch-to-next-train train3 train4)
(move-from-arrival train4 v4 v3 t2 t3 track1)
(switch-to-next-train train4 train5)
(move-from-arrival train5 v5 v4 t1 t2 track1)
(switch-to-next-train train5 train6)
(move-from-arrival train6 v6 v5 t0 t1 track1)
(switch-to-next-train train6 train3)
(move-to-departure train3 t5 t0 v5 v6 track2)
(switch-to-next-train train3 train2)
(move-to-departure train2 t6 t5 v4 v5 track2)
(switch-to-next-train train2 train6)
(move-from-track train6 t1 t0 t0 track1)
(move-to-track train6 t0 t5 t6 track2)
(switch-to-next-train train6 train5)
(move-to-departure train5 t2 t1 v3 v4 track1)
(switch-to-next-train train5 train4)
(move-to-departure train4 t3 t2 v2 v3 track1)
(switch-to-next-train train4 train1)
(move-to-departure train1 t4 t3 v1 v2 track1)
(switch-to-next-train train1 train6)
(move-to-departure train6 t6 t5 t0 v1 track2)
; cost = 26 (unit cost)
```

Listing 24: DecStar output plan in PT+MSR.

G Freelunch-Madagascar Output Plans

The plans generated by Freelunch-Madagascar in each domain. Only the plans with the shortest plan length are included.

G.1 Initial domain

```
0 : (move-on-arrival train1 v1 t0)
1 : (move-on-arrival train2 v2 v1)
2 : (move-to-track train1 t0 t5 track2)
3 : (move-on-arrival train2 v1 t0)
4 : (move-on-arrival train3 v3 v2)
5 : (move-along-track train1 t5 t6 track2)
6 : (move-on-arrival train3 v2 v1)
```

```
7 : (move-on-arrival train4 v4 v3)
8 : (move-to-track train2 t0 t1 track1)
9 : (move-along-track train2 t1 t2 track1)
10 : (move-on-arrival train3 v1 t0)
11 : (move-on-arrival train4 v3 v2)
12 : (move-on-arrival train5 v5 v4)
13 : (move-along-track train2 t2 t3 track1)
14 : (move-on-arrival train4 v2 v1)
15 : (move-on-arrival train5 v4 v3)
16 : (move-on-arrival train6 v6 v5)
17 : (move-to-track train3 t0 t5 track2)
18 : (move-on-arrival train4 v1 t0)
19 : (move-on-arrival train5 v3 v2)
20 : (move-on-arrival train6 v5 v4)
21 : (move-along-track train2 t3 t4 track1)
22 : (move-on-arrival train5 v2 v1)
23 : (move-on-arrival train6 v4 v3)
24 : (move-to-track train4 t0 t1 track1)
25 : (move-along-track train4 t1 t2 track1)
26 : (move-on-arrival train5 v1 t0)
27 : (move-on-arrival train6 v3 v2)
28 : (move-along-track train4 t2 t3 track1)
29 : (move-on-arrival train6 v2 v1)
30 : (move-to-track train5 t0 t1 track1)
31 : (move-along-track train5 t1 t2 track1)
32 : (move-on-arrival train6 v1 t0)
33 : (move-to-track train6 t0 t1 track1)
34 : (move-from-track train3 t5 t0 track2)
35 : (move-along-track train1 t6 t5 track2)
36 : (move-to-departure train3 t0 v1)
37 : (move-from-track train1 t5 t0 track2)
38 : (move-to-departure train3 v1 v2)
39 : (move-to-departure train1 t0 v1)
40 : (move-to-departure train3 v2 v3)
41 : (move-from-track train6 t1 t0 track1)
42 : (move-to-departure train1 v1 v2)
43 : (move-to-departure train3 v3 v4)
44 : (move-along-track train5 t2 t1 track1)
45 : (move-to-departure train1 v2 v3)
46 : (move-to-departure train3 v4 v5)
47 : (move-to-track train6 t0 t5 track2)
48 : (move-along-track train4 t3 t2 track1)
49 : (move-along-track train6 t5 t6 track2)
50 : (move-from-track train5 t1 t0 track1)
51 : (move-to-departure train1 v3 v4)
52 : (move-along-track train2 t4 t3 track1)
53 : (move-along-track train4 t2 t1 track1)
54 : (move-to-departure train1 v4 v3)
55 : (move-to-departure train5 t0 v1)
56 : (move-along-track train2 t3 t2 track1)
57 : (move-from-track train4 t1 t0 track1)
58 : (move-to-departure train1 v3 v4)
59 : (move-to-departure train3 v5 v6)
60 : (move-to-departure train5 v1 v2)
61 : (move-along-track train2 t2 t1 track1)
62 : (move-to-departure train1 v4 v5)
63 : (move-to-departure train4 t0 v1)
64 : (move-to-departure train5 v2 v3)
65 : (move-from-track train2 t1 t0 track1)
66 : (move-to-departure train4 v1 v2)
67 : (move-to-departure train5 v3 v4)
68 : (move-along-track train6 t6 t5 track2)
69 : (move-to-departure train2 t0 v1)
70 : (move-to-departure train4 v2 v3)
71 : (move-from-track train6 t5 t0 track2)
72 : (move-to-departure train2 v1 v2)
73 : (move-to-departure train6 t0 v1)
```

Listing 25: Freelunch-Madagascar output plan in the initial domain.

G.2 PT domain

```
0 : (move-from-arrival-to-track train1 v1 t0 t5 t6 track2)
1 : (move-from-arrival-to-track train2 v2 v1 t3 t4 track1)
2 : (move-from-arrival-to-track train3 v3 v2 t0 t5 track2)
3 : (move-from-arrival-to-track train4 v4 v3 t2 t3 track1)
4 : (move-from-arrival-to-track train5 v5 v4 t1 t2 track1)
5 : (move-from-arrival-to-track train6 v6 v5 t0 t1 track1)
6 : (move-from-track-to-departure train3 t5 t0 v5 v6
    track2)
7 : (move-from-track-to-departure train1 t6 t5 v4 v5
    track2)
8 : (move-from-track-to-switch train6 t1 t0 t0 track1)
9 : (move-from-switch-to-track train6 t0 t5 t6 track2)
10 : (move-from-track-to-departure train5 t2 t1 v3 v4
    track1)
11 : (move-from-track-to-departure train4 t3 t2 v2 v3
    track1)
12 : (move-from-track-to-departure train2 t4 t3 v1 v2
    track1)
13 : (move-from-track-to-departure train6 t6 t5 t0 v1
    track2)
```

Listing 26: Freelunch-Madagascar output plan in PT.

G.3 MSR domain

```
0 : (move-on-arrival train1 v1 t0)
1 : (switch-to-next-train train1 train2)
2 : (move-on-arrival train2 v2 v1)
3 : (switch-to-next-train train2 train3)
4 : (move-on-arrival train3 v3 v2)
5 : (switch-to-next-train train3 train4)
6 : (move-on-arrival train4 v4 v3)
7 : (switch-to-next-train train4 train1)
8 : (move-to-track train1 t0 t1 track1)
9 : (switch-to-next-train train1 train2)
10 : (move-on-arrival train2 v1 t0)
11 : (move-to-track train2 t0 t5 track2)
12 : (move-along-track train2 t5 t6 track2)
13 : (switch-to-next-train train2 train1)
14 : (move-along-track train1 t1 t2 track1)
15 : (switch-to-next-train train1 train3)
16 : (move-on-arrival train3 v2 v1)
17 : (switch-to-next-train train3 train4)
18 : (move-on-arrival train4 v3 v2)
19 : (switch-to-next-train train4 train3)
20 : (move-on-arrival train3 v1 t0)
21 : (switch-to-next-train train3 train5)
22 : (move-on-arrival train5 v5 v4)
23 : (move-on-arrival train5 v4 v3)
24 : (switch-to-next-train train5 train4)
25 : (move-on-arrival train4 v2 v1)
26 : (switch-to-next-train train4 train3)
27 : (move-to-track train3 t0 t5 track2)
28 : (switch-to-next-train train3 train5)
29 : (move-on-arrival train5 v3 v2)
30 : (switch-to-next-train train5 train6)
31 : (move-on-arrival train6 v6 v5)
32 : (switch-to-next-train train6 train4)
33 : (move-on-arrival train4 v1 t0)
34 : (move-to-track train4 t0 t1 track1)
35 : (switch-to-next-train train4 train5)
36 : (move-on-arrival train5 v2 v1)
37 : (switch-to-next-train train5 train6)
38 : (move-on-arrival train6 v5 v4)
39 : (move-on-arrival train6 v4 v3)
40 : (switch-to-next-train train6 train1)
41 : (move-along-track train1 t2 t3 track1)
42 : (switch-to-next-train train1 train4)
43 : (move-along-track train4 t1 t2 track1)
44 : (switch-to-next-train train4 train1)
45 : (move-along-track train1 t3 t4 track1)
46 : (switch-to-next-train train1 train4)
47 : (move-along-track train4 t2 t3 track1)
48 : (switch-to-next-train train4 train6)
49 : (move-on-arrival train6 v3 v2)
50 : (switch-to-next-train train6 train5)
51 : (move-on-arrival train5 v1 t0)
52 : (move-to-track train5 t0 t1 track1)
53 : (switch-to-next-train train5 train6)
54 : (move-on-arrival train6 v2 v1)
55 : (move-on-arrival train6 v1 t0)
56 : (switch-to-next-train train6 train5)
57 : (move-along-track train5 t1 t2 track1)
58 : (switch-to-next-train train5 train6)
59 : (move-to-track train6 t0 t1 track1)
60 : (switch-to-next-train train6 train3)
61 : (move-from-track train3 t5 t0 track2)
62 : (move-to-departure train3 t0 v1)
```

```
63 : (move-to-departure train3 v1 v2)
64 : (switch-to-next-train train3 train2)
65 : (move-along-track train2 t6 t5 track2)
66 : (move-from-track train2 t5 t0 track2)
67 : (move-to-departure train2 t0 v1)
68 : (switch-to-next-train train2 train3)
69 : (move-to-departure train3 v2 v3)
70 : (switch-to-next-train train3 train2)
71 : (move-to-departure train2 v1 v2)
72 : (switch-to-next-train train2 train6)
73 : (move-from-track train6 t1 t0 track1)
74 : (switch-to-next-train train6 train5)
75 : (move-along-track train5 t2 t1 track1)
76 : (switch-to-next-train train5 train4)
77 : (move-along-track train4 t3 t2 track1)
78 : (switch-to-next-train train4 train3)
79 : (move-to-departure train3 v3 v4)
80 : (switch-to-next-train train3 train6)
81 : (reallocate-to-track train6 t0 t5 track2)
82 : (switch-to-next-train train6 train5)
83 : (move-from-track train5 t1 t0 track1)
84 : (switch-to-next-train train5 train4)
85 : (move-along-track train4 t2 t1 track1)
86 : (switch-to-next-train train4 train1)
87 : (move-along-track train1 t4 t3 track1)
88 : (switch-to-next-train train1 train3)
89 : (move-to-departure train3 v4 v5)
90 : (switch-to-next-train train3 train2)
91 : (move-to-departure train2 v2 v3)
92 : (switch-to-next-train train2 train5)
93 : (move-to-departure train5 t0 v1)
94 : (switch-to-next-train train5 train1)
95 : (move-along-track train1 t3 t2 track1)
96 : (switch-to-next-train train1 train3)
97 : (move-to-departure train3 v5 v6)
98 : (switch-to-next-train train3 train2)
99 : (move-to-departure train2 v3 v4)
100 : (move-to-departure train2 v4 v5)
101 : (switch-to-next-train train2 train5)
102 : (move-to-departure train5 v1 v2)
103 : (switch-to-next-train train5 train4)
104 : (move-from-track train4 t1 t0 track1)
105 : (switch-to-next-train train4 train1)
106 : (move-along-track train1 t2 t1 track1)
107 : (switch-to-next-train train1 train4)
108 : (move-to-departure train4 t0 v1)
109 : (switch-to-next-train train4 train1)
110 : (move-from-track train1 t1 t0 track1)
111 : (switch-to-next-train train1 train5)
112 : (move-to-departure train5 v2 v3)
113 : (switch-to-next-train train5 train4)
114 : (move-to-departure train4 v1 v2)
115 : (switch-to-next-train train4 train5)
116 : (move-to-departure train5 v3 v4)
117 : (switch-to-next-train train5 train1)
118 : (move-to-departure train1 t0 v1)
119 : (switch-to-next-train train1 train4)
120 : (move-to-departure train4 v2 v3)
121 : (switch-to-next-train train4 train1)
122 : (move-to-departure train1 v1 v2)
123 : (switch-to-next-train train1 train6)
124 : (move-from-track train6 t5 t0 track2)
125 : (move-to-departure train6 t0 v1)
126 : (switch-to-next-train train6 train5)
```

Listing 27: Freelunch-Madagascar output plan in MSR.

G.4 PT+MSR domain

```
0 : (move-from-arrival train1 v1 t0 t5 t6 track2)
1 : (switch-to-next-train train1 train2)
2 : (move-from-arrival train2 v2 v1 t3 t4 track1)
3 : (switch-to-next-train train2 train3)
4 : (move-from-arrival train3 v3 v2 t0 t5 track2)
5 : (switch-to-next-train train3 train4)
6 : (move-from-arrival train4 v4 v3 t2 t3 track1)
7 : (switch-to-next-train train4 train5)
8 : (move-from-arrival train5 v5 v4 t1 t2 track1)
9 : (switch-to-next-train train5 train6)
10 : (move-from-arrival train6 v6 v5 t0 t1 track1)
11 : (switch-to-next-train train6 train3)
12 : (move-to-departure train3 t5 t0 v5 v6 track2)
13 : (switch-to-next-train train3 train1)
14 : (move-to-departure train1 t6 t5 v4 v5 track2)
15 : (switch-to-next-train train1 train6)
16 : (move-from-track train6 t1 t0 t0 track1)
17 : (move-to-track train6 t0 t5 t6 track2)
18 : (switch-to-next-train train6 train5)
19 : (move-to-departure train5 t2 t1 v3 v4 track1)
20 : (switch-to-next-train train5 train4)
21 : (move-to-departure train4 t3 t2 v2 v3 track1)
22 : (switch-to-next-train train4 train2)
23 : (move-to-departure train2 t4 t3 v1 v2 track1)
24 : (switch-to-next-train train2 train6)
25 : (move-to-departure train6 t6 t5 t0 v1 track2)
26 : (switch-to-next-train train6 train1)
```

Listing 28: Freelunch-Madagascar output plan in PT+MSR.

H Saarplan Output Plans

The plans generated by Saarplan in each domain. Only the plans with the shortest plan length are included.

H.1 Initial domain

```
(move-on-arrival train1 v1 t0)
(move-on-arrival train2 v2 v1)
(move-on-arrival train3 v3 v2)
(move-on-arrival train4 v4 v3)
(move-on-arrival train5 v5 v4)
(move-on-arrival train6 v6 v5)
(move-to-track train1 t0 t5 track2)
(move-on-arrival train2 v1 t0)
(move-on-arrival train3 v2 v1)
(move-on-arrival train4 v3 v2)
(move-on-arrival train5 v4 v3)
(move-to-track train2 t0 t1 track1)
(move-along-track train1 t5 t6 track2)
(move-on-arrival train3 v1 t0)
(move-on-arrival train4 v2 v1)
(move-on-arrival train5 v3 v2)
(move-on-arrival train6 v5 v4)
(move-on-arrival train6 v4 v3)
(move-to-track train3 t0 t5 track2)
(move-on-arrival train4 v1 t0)
(move-on-arrival train5 v2 v1)
(move-along-track train2 t1 t2 track1)
(move-to-track train4 t0 t1 track1)
(move-on-arrival train6 v3 v2)
(move-along-track train2 t2 t3 track1)
(move-on-arrival train5 v1 t0)
(move-on-arrival train6 v2 v1)
(move-along-track train4 t1 t2 track1)
(move-along-track train2 t3 t4 track1)
(move-to-track train5 t0 t1 track1)
(move-along-track train4 t2 t3 track1)
(move-on-arrival train6 v1 t0)
(move-along-track train5 t1 t2 track1)
(move-to-track train6 t0 t1 track1)
(move-from-track train3 t5 t0 track2)
(move-to-departure train3 t0 v1)
(move-along-track train1 t6 t5 track2)
(move-to-departure train3 v1 v2)
(move-to-departure train3 v2 v3)
(move-to-departure train3 v3 v4)
(move-to-departure train3 v4 v5)
(move-to-departure train3 v5 v6)
(move-from-track train1 t5 t0 track2)
(move-to-departure train1 t0 v1)
(move-to-departure train1 v1 v2)
(move-to-departure train1 v2 v3)
(move-to-departure train1 v3 v4)
(move-to-departure train1 v4 v5)
(move-from-track train6 t1 t0 track1)
(move-to-departure train6 t0 v1)
(move-along-track train5 t2 t1 track1)
(move-to-departure train6 v1 v2)
```

```
(move-to-departure train6 v2 v3)
(move-to-departure train6 v3 v4)
(move-along-track train4 t3 t2 track1)
(move-along-track train2 t4 t3 track1)
(move-from-track train5 t1 t0 track1)
(move-along-track train4 t2 t1 track1)
(move-to-track train5 t0 t5 track2)
(move-along-track train2 t3 t2 track1)
(move-from-track train4 t1 t0 track1)
(move-to-departure train4 t0 v1)
(move-along-track train2 t2 t1 track1)
(move-to-departure train4 v1 v2)
(move-to-departure train4 v2 v3)
(move-from-track train2 t1 t0 track1)
(move-to-departure train2 t0 v1)
(move-to-departure train2 v1 v2)
(move-from-track train5 t5 t0 track2)
(move-to-departure train5 t0 v1)
; cost = 70 (unit cost)
```

Listing 29: Saarplan output plan in the initial domain.

H.2 PT domain

```
(move-from-arrival-to-track train1 v1 t0 t3 t4 track1)
(move-from-arrival-to-track train2 v2 v1 t5 t6 track2)
(move-from-arrival-to-track train3 v3 v2 t0 t5 track2)
(move-from-arrival-to-track train4 v4 v3 t2 t3 track1)
(move-from-arrival-to-track train5 v5 v4 t1 t2 track1)
(move-from-arrival-to-track train6 v6 v5 t0 t1 track1)
(move-from-track-to-departure train3 t5 t0 v5 v6 track2)
(move-from-track-to-departure train2 t6 t5 v4 v5 track2)
(move-from-track-to-departure train6 t1 t0 v3 v4 track1)
(move-from-track-to-switch train5 t2 t1 t0 track1)
(move-from-switch-to-track train5 t0 t5 t6 track2)
(move-from-track-to-departure train4 t3 t2 v2 v3 track1)
(move-from-track-to-departure train1 t4 t3 v1 v2 track1)
(move-from-track-to-departure train5 t6 t5 t0 v1 track2)
; cost = 14 (unit cost)
```

Listing 30: Saarplan output plan in PT.

H.3 MSR domain

```
(move-on-arrival train1 v1 t0)
(move-to-track train1 t0 t1 track1)
(switch-to-next-train train1 train2)
(move-on-arrival train2 v2 v1)
(move-on-arrival train2 v1 t0)
(switch-to-next-train train2 train3)
(move-on-arrival train3 v3 v2)
(move-on-arrival train3 v2 v1)
(switch-to-next-train train3 train4)
(move-on-arrival train4 v4 v3)
(move-on-arrival train4 v3 v2)
(switch-to-next-train train4 train5)
(move-on-arrival train5 v5 v4)
(move-on-arrival train5 v4 v3)
(switch-to-next-train train5 train6)
(move-on-arrival train6 v6 v5)
(move-on-arrival train6 v5 v4)
(switch-to-next-train train6 train2)
(move-to-track train2 t0 t5 track2)
(switch-to-next-train train2 train1)
(move-along-track train1 t1 t2 track1)
(switch-to-next-train train1 train3)
(move-on-arrival train3 v1 t0)
(switch-to-next-train train3 train4)
(move-on-arrival train4 v2 v1)
(switch-to-next-train train4 train5)
(move-on-arrival train5 v3 v2)
(switch-to-next-train train5 train6)
(move-on-arrival train6 v4 v3)
(switch-to-next-train train6 train1)
(move-along-track train1 t2 t1 track1)
(switch-to-next-train train1 train2)
(move-along-track train2 t5 t6 track2)
(switch-to-next-train train2 train3)
(move-to-track train3 t0 t5 track2)
(switch-to-next-train train3 train4)
(move-on-arrival train4 v1 t0)
(switch-to-next-train train4 train1)
(move-along-track train1 t1 t2 track1)
(switch-to-next-train train1 train5)
(move-on-arrival train5 v2 v1)
(switch-to-next-train train5 train6)
(move-on-arrival train6 v3 v2)
(switch-to-next-train train6 train2)
(switch-to-next-train train2 train4)
(move-to-track train4 t0 t1 track1)
(switch-to-next-train train4 train5)
(move-on-arrival train5 v1 t0)
(switch-to-next-train train5 train1)
(move-along-track train1 t2 t3 track1)
(switch-to-next-train train1 train4)
(move-along-track train4 t1 t2 track1)
(switch-to-next-train train4 train5)
(move-to-track train5 t0 t1 track1)
(move-from-track train5 t1 t0 track1)
(switch-to-next-train train5 train4)
(move-along-track train4 t2 t1 track1)
(switch-to-next-train train4 train6)
(move-on-arrival train6 v2 v1)
(switch-to-next-train train6 train1)
(move-along-track train1 t3 t4 track1)
(switch-to-next-train train1 train4)
(move-along-track train4 t1 t2 track1)
(move-along-track train4 t2 t3 track1)
(switch-to-next-train train4 train5)
(reallocate-to-track train5 t0 t1 track1)
(move-along-track train5 t1 t2 track1)
(switch-to-next-train train5 train6)
(move-on-arrival train6 v1 t0)
(move-to-track train6 t0 t1 track1)
(switch-to-next-train train6 train3)
(move-from-track train3 t5 t0 track2)
(move-to-departure train3 t0 v1)
(switch-to-next-train train3 train2)
(move-along-track train2 t6 t5 track2)
(switch-to-next-train train2 train3)
(move-to-departure train3 v1 v2)
(move-to-departure train3 v2 v3)
(move-to-departure train3 v3 v4)
(move-to-departure train3 v4 v5)
(move-to-departure train3 v5 v6)
(switch-to-next-train train3 train1)
(switch-to-next-train train1 train2)
(move-from-track train2 t5 t0 track2)
(move-to-departure train2 t0 v1)
(move-to-departure train2 v1 v2)
(move-to-departure train2 v2 v3)
(move-to-departure train2 v3 v4)
(move-to-departure train2 v4 v5)
(switch-to-next-train train2 train1)
(switch-to-next-train train1 train6)
(move-from-track train6 t1 t0 track1)
(switch-to-next-train train6 train5)
(move-along-track train5 t2 t1 track1)
(switch-to-next-train train5 train4)
(move-along-track train4 t3 t2 track1)
(switch-to-next-train train4 train1)
(move-along-track train1 t4 t3 track1)
(switch-to-next-train train1 train6)
(reallocate-to-track train6 t0 t5 track2)
(switch-to-next-train train6 train5)
(move-from-track train5 t1 t0 track1)
(move-to-departure train5 t0 v1)
(switch-to-next-train train5 train4)
(move-along-track train4 t2 t1 track1)
(switch-to-next-train train4 train1)
(move-along-track train1 t3 t2 track1)
(switch-to-next-train train1 train5)
(move-to-departure train5 v1 v2)
(move-to-departure train5 v2 v3)
(move-to-departure train5 v3 v4)
(switch-to-next-train train5 train1)
(switch-to-next-train train1 train4)
(move-from-track train4 t1 t0 track1)
(move-to-departure train4 t0 v1)
```

```
(switch-to-next-train train4 train1)
(move-along-track train1 t2 t1 track1)
(switch-to-next-train train1 train4)
(move-to-departure train4 v1 v2)
(move-to-departure train4 v2 v3)
(switch-to-next-train train4 train1)
(move-from-track train1 t1 t0 track1)
(move-to-departure train1 t0 v1)
(move-to-departure train1 v1 v2)
(switch-to-next-train train1 train6)
(move-from-track train6 t5 t0 track2)
(move-to-departure train6 t0 v1)
; cost = 129 (general cost)
```

Listing 31: Saarplan output plan in MSR.

H.4 PT+MSR domain

```
(move-from-arrival train1 v1 t0 t5 t6 track2)
(switch-to-next-train train1 train2)
(move-from-arrival train2 v2 v1 t3 t4 track1)
(switch-to-next-train train2 train3)
(move-from-arrival train3 v3 v2 t0 t5 track2)
(switch-to-next-train train3 train4)
(move-from-arrival train4 v4 v3 t2 t3 track1)
(switch-to-next-train train4 train5)
(move-from-arrival train5 v5 v4 t1 t2 track1)
(switch-to-next-train train5 train6)
(move-from-arrival train6 v6 v5 t0 t1 track1)
(switch-to-next-train train6 train3)
(move-to-departure train3 t5 t0 v5 v6 track2)
(switch-to-next-train train3 train1)
(move-to-departure train1 t6 t5 v4 v5 track2)
(switch-to-next-train train1 train2)
(switch-to-next-train train2 train6)
(move-to-departure train6 t1 t0 v3 v4 track1)
(switch-to-next-train train6 train2)
(switch-to-next-train train2 train5)
(move-from-track train5 t2 t1 t0 track1)
(move-to-track train5 t0 t5 t6 track2)
(switch-to-next-train train5 train4)
(move-to-departure train4 t3 t2 v2 v3 track1)
(switch-to-next-train train4 train2)
(move-to-departure train2 t4 t3 v1 v2 track1)
(switch-to-next-train train2 train5)
(move-to-departure train5 t6 t5 t0 v1 track2)
; cost = 28 (unit cost)
```

Listing 32: Saarplan output plan in PT+MSR.