Temporal Task and Motion Plans: Planning and Plan Repair
– Repairing Temporal Task and Motion Plans Using Replanning with Temporal Macro Operators

Temporal uppgifts- och rutt-planering och planreparation

Erik Hansson

Supervisor : Mikael Nilsson
Examiner : Jonas Kvarnström

External supervisor : Lars Rundqwist
Upphovsrätt


Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances. The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Erik Hansson
Abstract

This thesis presents an extension to the Temporal Fast Downward planning system that integrates motion planning in it and algorithms for generating two types of temporal macro operators expressible in PDDL2.1. The extension to the Temporal Fast Downward planning system includes, in addition to the integration of motion planning itself, an extension to the context-enhanced additive heuristic that uses information from the motion planning part to improve the heuristic estimate. The temporal macro operators expressible in PDDL2.1 are, to the author’s knowledge, an area that is not studied within the context of plan repair before. Two types of temporal macro operators are presented along with algorithms for automatically constructing and using them when solving plan repair problems by replanning. Both the heuristic extension and the temporal macro operators were evaluated in the context of simulated unmanned aerial vehicles autonomously executing reconnaissance missions to identify targets and avoiding threats in unexplored areas. The heuristic extension was proved to be very helpful in the scenario. Unfortunately, the evaluation of the temporal macro operators indicated that the cost of introducing them is higher than the gain of using them for the scenario.
Acknowledgements

To begin with I would like to express my gratitude to Jonas Kvarnström and Mikael Nilsson for the feedback on the report. The comments and required changes did in general make the report better. Lars Rundqwist, my supervisor at Saab, proved to be just as important for the thesis by listening to my ideas and asking questions that I had overlooked. I doubt the thesis would have worked out the way it did without that support. Naturally, it has been a wonderful opportunity to write the thesis at Saab and to get some insight in the company.

I like to give my thanks to one expected and some unexpected sources for reviewing the methodology that I have used in the thesis. I asked my brother (the expected source, a statistician) to take a quick look at the methodology chapter to verify that it was sound (it is after all not a standard methodology). In addition to providing feedback, he said he would run it by a couple of colleagues since it was not a standard methodology. Before I knew it, the methodology (not the report) had been verified by some of his unnamed colleagues at Statistiska Centralbyrån (en. Statistics Sweden) and an unnamed professor in statistics at Uppsala University. Unfortunately, I am unable to quote them to strengthen the claim about the methodology (which is already grounded in literature) due to the second hand nature and lack of name. Nevertheless, you all have my thanks for verifying the methodology, even if you probably will never read this.

Finally, to my family, friends and colleagues who have supported me, knowingly or unknowingly, by forcing me to switch context or to refocus. Naturally, both cases are just as important and were much appreciated.

Linköping, October 2018
Erik Hansson
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iv</td>
</tr>
<tr>
<td>Contents</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Relevant Terminology</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Scope</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Aim</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Research Questions</td>
<td>3</td>
</tr>
<tr>
<td>1.5 Delimitations</td>
<td>4</td>
</tr>
<tr>
<td>1.6 Thesis Outline</td>
<td>4</td>
</tr>
<tr>
<td>2 Background</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Scenario Overview</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Agents</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Threats</td>
<td>6</td>
</tr>
<tr>
<td>2.4 Malfunction Events</td>
<td>7</td>
</tr>
<tr>
<td>2.5 Mission Policies</td>
<td>7</td>
</tr>
<tr>
<td>3 Theory</td>
<td>9</td>
</tr>
<tr>
<td>3.1 Task Planning</td>
<td>9</td>
</tr>
<tr>
<td>3.2 Motion Planning</td>
<td>18</td>
</tr>
<tr>
<td>3.3 Mutual Exclusion</td>
<td>22</td>
</tr>
<tr>
<td>3.4 Metrics</td>
<td>22</td>
</tr>
<tr>
<td>4 Related Work</td>
<td>24</td>
</tr>
<tr>
<td>4.1 Temporal Fast Downward</td>
<td>24</td>
</tr>
<tr>
<td>4.2 Integration of Task and Motion Planning</td>
<td>27</td>
</tr>
<tr>
<td>4.3 Task Plan Repair</td>
<td>29</td>
</tr>
<tr>
<td>5 Theory Extension</td>
<td>35</td>
</tr>
<tr>
<td>5.1 Path Integration in Temporal Macro Operators</td>
<td>35</td>
</tr>
<tr>
<td>5.2 Temporal Macro Operator Composition</td>
<td>37</td>
</tr>
<tr>
<td>5.3 Temporal Macro Operator Generation</td>
<td>47</td>
</tr>
<tr>
<td>6 Design</td>
<td>50</td>
</tr>
<tr>
<td>6.1 Planner</td>
<td>50</td>
</tr>
<tr>
<td>6.2 Problem Generator</td>
<td>55</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>6.3 Simulator</td>
<td>62</td>
</tr>
<tr>
<td>7 Evaluation Method</td>
<td>65</td>
</tr>
<tr>
<td>7.1 Evaluation Overview</td>
<td>65</td>
</tr>
<tr>
<td>7.2 Generating Problems</td>
<td>66</td>
</tr>
<tr>
<td>7.3 System Configurations</td>
<td>67</td>
</tr>
<tr>
<td>7.4 Population and Sampling</td>
<td>67</td>
</tr>
<tr>
<td>7.5 Values for Analysis</td>
<td>68</td>
</tr>
<tr>
<td>7.6 Analysis Methods</td>
<td>69</td>
</tr>
<tr>
<td>7.7 Analysis</td>
<td>73</td>
</tr>
<tr>
<td>8 Results</td>
<td>75</td>
</tr>
<tr>
<td>8.1 Plan Repair</td>
<td>75</td>
</tr>
<tr>
<td>8.2 Heuristic Extension</td>
<td>79</td>
</tr>
<tr>
<td>9 Discussion</td>
<td>82</td>
</tr>
<tr>
<td>9.1 Integration of Motion Planning</td>
<td>82</td>
</tr>
<tr>
<td>9.2 Temporal Macro Operator in Plan Repair</td>
<td>82</td>
</tr>
<tr>
<td>9.3 Heuristic extension</td>
<td>85</td>
</tr>
<tr>
<td>9.4 Evaluation Method</td>
<td>86</td>
</tr>
<tr>
<td>9.5 Ethical Aspects</td>
<td>89</td>
</tr>
<tr>
<td>10 Conclusion</td>
<td>90</td>
</tr>
<tr>
<td>11 Future Work</td>
<td>92</td>
</tr>
<tr>
<td>11.1 Impact of Motion Planners</td>
<td>92</td>
</tr>
<tr>
<td>11.2 Geometric Backtracking</td>
<td>92</td>
</tr>
<tr>
<td>11.3 Multiple Repair Strategies</td>
<td>92</td>
</tr>
<tr>
<td>11.4 Temporal Macro Operators in PDDL2.1</td>
<td>93</td>
</tr>
<tr>
<td>11.5 Temporal Macro Operators</td>
<td>93</td>
</tr>
<tr>
<td>11.6 Preprocessing on What If Basis</td>
<td>93</td>
</tr>
<tr>
<td>A Typical scenarios</td>
<td>94</td>
</tr>
<tr>
<td>A.1 Scenario 1</td>
<td>94</td>
</tr>
<tr>
<td>A.2 Scenario 2</td>
<td>96</td>
</tr>
<tr>
<td>A.3 Scenario 3</td>
<td>97</td>
</tr>
<tr>
<td>A.4 Scenario 4</td>
<td>98</td>
</tr>
<tr>
<td>A.5 Scenario 5</td>
<td>99</td>
</tr>
<tr>
<td>B Formalism Examples</td>
<td>100</td>
</tr>
<tr>
<td>B.1 Classical Planning with Resources</td>
<td>100</td>
</tr>
<tr>
<td>B.2 Temporal Planning</td>
<td>102</td>
</tr>
<tr>
<td>C Extensions to PDDL2.1</td>
<td>106</td>
</tr>
<tr>
<td>C.1 Common Grammar</td>
<td>106</td>
</tr>
<tr>
<td>C.2 Work Space and Configuration Space</td>
<td>107</td>
</tr>
<tr>
<td>C.3 Motion Planners</td>
<td>107</td>
</tr>
<tr>
<td>C.4 Motion Planning Resources and State Variables</td>
<td>108</td>
</tr>
<tr>
<td>C.5 Operator Extension</td>
<td>109</td>
</tr>
<tr>
<td>C.6 Initial Values</td>
<td>111</td>
</tr>
<tr>
<td>C.7 Collision Zones</td>
<td>112</td>
</tr>
<tr>
<td>D Plan Visualisation</td>
<td>113</td>
</tr>
<tr>
<td>E Evaluation Data</td>
<td>116</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Synthetic-aperture radar overview</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Surface-to-air units weapon range</td>
<td>7</td>
</tr>
<tr>
<td>3.1</td>
<td>A example DTG for the variable <em>is at uav1</em>. The first parameter in all the actions is <em>uav1</em>. However, this parameter has been left out in the graph to reduce the size in the graph.</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Dubins paths to configuration</td>
<td>21</td>
</tr>
<tr>
<td>3.3</td>
<td>Mutex-lock protection of critical section</td>
<td>23</td>
</tr>
<tr>
<td>5.1</td>
<td>Example of an ideal temporal macro operator</td>
<td>38</td>
</tr>
<tr>
<td>5.2</td>
<td>Example of a sequential temporal macro operator</td>
<td>38</td>
</tr>
<tr>
<td>5.3</td>
<td>Example of a parallel temporal macro operator</td>
<td>39</td>
</tr>
<tr>
<td>6.1</td>
<td>System overview</td>
<td>51</td>
</tr>
<tr>
<td>6.2</td>
<td>Dubins paths to location</td>
<td>54</td>
</tr>
<tr>
<td>6.3</td>
<td>The problem generator</td>
<td>57</td>
</tr>
<tr>
<td>6.4</td>
<td>Architecture for temporal macro operator generation</td>
<td>59</td>
</tr>
<tr>
<td>8.1</td>
<td>Mean search time comparison for temporal macro operators</td>
<td>76</td>
</tr>
<tr>
<td>8.2</td>
<td>Mean search time and standard deviation all problems, configurations 0, 1 and 6</td>
<td>77</td>
</tr>
<tr>
<td>8.3</td>
<td>Mean search time and standard deviation same problems, configurations 0 and 6</td>
<td>78</td>
</tr>
<tr>
<td>8.4</td>
<td>Mean search time and standard deviation same problems, configurations 1 and 6</td>
<td>78</td>
</tr>
<tr>
<td>8.5</td>
<td>Mean search time comparison for heuristic extension</td>
<td>80</td>
</tr>
<tr>
<td>8.6</td>
<td>Mean search time and standard deviation all problems, configurations 0 and 13</td>
<td>81</td>
</tr>
<tr>
<td>8.7</td>
<td>Mean search time and standard deviation same problems, configurations 0 and 13</td>
<td>81</td>
</tr>
<tr>
<td>A.1</td>
<td>Overview of scenario 1</td>
<td>95</td>
</tr>
<tr>
<td>A.2</td>
<td>Overview of scenario 2</td>
<td>96</td>
</tr>
<tr>
<td>A.3</td>
<td>Overview of scenario 3</td>
<td>97</td>
</tr>
<tr>
<td>A.4</td>
<td>Overview of scenario 4</td>
<td>98</td>
</tr>
<tr>
<td>A.5</td>
<td>Overview of scenario 5</td>
<td>99</td>
</tr>
<tr>
<td>D.1</td>
<td>Snapshot 1</td>
<td>114</td>
</tr>
<tr>
<td>D.2</td>
<td>Snapshot 2</td>
<td>115</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Parameters to the reconnaissance mission scenario ............................................... 5
2.2 Specifications for the UAVs in the scenario ........................................................ 6
2.3 The different types of malfunction events in the scenarios ..................................... 8
6.1 Designs for integrating motion planning ................................................................. 51
7.1 System configurations for evaluation ........................................................................ 67
7.2 Summary of comparisons in the evaluation .............................................................. 74
7.3 Hypotheses for evaluation ........................................................................................ 74
8.1 Descriptive results for temporal macro operators .................................................... 76
8.2 Descriptive results for evaluating the heuristic extension ........................................ 79
A.1 Parameters for scenario 1 ....................................................................................... 94
A.2 Parameters for scenario 2 ....................................................................................... 96
A.3 Parameters for scenario 3 ....................................................................................... 97
A.4 Parameters for scenario 4 ....................................................................................... 98
A.5 Parameters for scenario 5 ....................................................................................... 99
Autonomous agents have the potential to solve problems where humans would be exposed to unacceptable risks. For example, humans should not enter an area with high radiation due to the risk of radiation sickness. However, autonomous agents, for example unmanned aerial vehicles (UAVs), do not necessarily have any problems with it. Therefore, they are able to survey the area to find and identify different objects (it could, for example, be leakage spreading the radioactive substance). Similarly, there are military scenarios where one wants to avoid sending personnel into hostile territory but still needs to scan the area for targets. By sending UAVs, one could find the targets without having personnel at risk.

Both of the above mentioned scenarios provide three challenges (among others that are not covered in this thesis). Firstly, making the UAVs cooperate and to plan ahead. This can be done by generating plans where the UAVs divide the different tasks (in the military scenario, identify the targets) between them. Secondly, to find out which route a UAV must take to move from one point to another. Thirdly, all threats and targets might not be known beforehand, which requires that the system must be able to adapt when threats and targets are detected.

These three challenges are well studied problems or properties of problems. The first is a task planning problem, sometimes referred to as a symbolic (planning) problem. The second is a motion planning problem, which is also called a geometrical (planning) problem. Finally, there is the property that the problem may change during execution (e.g. new threats can be discovered).

There are techniques for handling the problems individually: Generating plans can be done by using a task planner \[^1\]; finding paths (also called motion plans) for how to reach a point in a complex environment is something that motion planners do \[^2\]; and handling changes in the problem is something that has been done by plan repair algorithms (repairing a task plan, for example Scala’s algorithm \[^3\], and repairing a motion plan, for example Ferguson, Kalra and Stentz’s algorithm \[^4\]). Moreover, work has been done on solving the motion planning and task planning problems as a combined problem (Bidot et al. identified three different categories of solvers for the combined problem \[^5\]).

To the author’s knowledge, there is no publicly available work on repair of the combined task and motion planning problem. However, the description of DARPA’s Collaborative

---

\[^1\] DARPA stands for “Defense Advanced Research Projects Agency” and is an agency in USA under the United States Department of Defense.
Operations in Denied Environment program indicates that this problem is being researched at the moment. Moreover, a similar problem has been studied in the industry. This problem is to repair an existing plan through rescheduling and update paths with an integrated motion planner. The rescheduling reallocates which agent does which tasks and when they are executed. Therefore, it does not permit that new tasks are introduced as they can be in a task and motion plan repair problem. However, it can handle new threats since they require new strategies for finishing a task (e.g. a new flight path) rather than handling a new task.

If the combined task and motion planning repair problem can be solved efficiently, then it is possible to decrease the time an autonomous agent requires to resume deliberated acting instead of reactive acting after a non-predicted event occur. The meaning of deliberate acting is in the thesis acting after ensuring that the acting eventually achieves the goal. Furthermore, reactive acting is the opposite: Acting before ensuring that the acting eventually achieves the goal. Naturally, deliberate acting is preferred over reactive acting as long as it is fast enough because it decreases the risk of unwanted effects of the acting that hinders that the goal is achieved (e.g. the agent runs out of fuel or it enters an area which it cannot leave). Hence, it is beneficial to decrease the time to repair the plan so that deliberate acting can be resumed faster after a non-predicted event occur. That being said, reactive acting is important as well because it is faster and can therefore handle sudden unexpected problems (e.g. another agent changes direction so that the current course will result in a crash within a short time frame).

1.1 Relevant Terminology

This section gives a short introduction to two terms that are required to understand the more technical part of the introduction: Macro operators; and heuristic functions.

Macro Operators

To understand what macro operators (and macro actions) are, one needs to understand what operators and actions are. The actions within task planning are controlled changes to the world. These, normally, correspond to something an agent can do, for example “aircraft ABC123 fly from London to New York”. If one replaces all the objects of an action with variables, then one gets the corresponding operator. That is, an action is a grounded (meaning that it has no free variables) operator. In the case of the fly action, the operator is “aircraft ?aircraft fly from ?from to ?to”. Based on this, a macro operator is simply a sequence of operators and a macro action is a sequence of actions.

Macro operators is an old concept that dates back to 1985 or earlier. In more recent years, macro operators have been used to speed up the search process in the Fast Forward planner and for task plan repair. The basic idea behind macro operators is to provide the planner with shortcuts that guide it towards the goal. To understand this, one has to know that state-space task planning algorithms (in many cases) solve the planning problem by searching a graph where the nodes correspond to states (all relevant information about the world at a point in time) and the edges correspond to actions. Hence, a macro action, \( m = \langle a_0, a_1, a_2 \rangle \) is equivalent to introducing an extra edge \( A \xrightarrow{a_0} D \) for each node \( A \) and \( D \) in the graph that is connected by a path \( A \xrightarrow{a_0} B \xrightarrow{a_1} C \xrightarrow{a_2} D \) where \( A, B, C \) and \( D \) are nodes in the graph. Thereby, introducing shortcuts in the search space. Unfortunately, adding edges increase the number of choices at each step in the search process (i.e. increases the average branching factor). This in turn can increase the search time for the planning algorithm.

Heuristic Functions

A heuristic function is an estimation of another function. In the context of planning, it is commonly used as an estimation of the distance between a node and the closest goal node in task planning. The heuristic function can be used in the search process to select which
edge to follow. There is a more in-depth description of heuristic functions and some specific heuristic functions later in the thesis (see section 3.1). However, knowing that it is a function that is used by the planner to find a solution faster is enough to understand the research questions related to heuristic functions.

1.2 Scope

There has been a lot of work done within the different areas presented in the first section. Unfortunately, it is outside the scope of the thesis to cover all this work in detail. Instead, two techniques have been selected to start from and the focus has been to extend and integrate them to solve the integrated task and motion planning problem and its corresponding plan repair problem. These selections of techniques are based on the focus of the thesis and the main scenario of the thesis, which is a reconnaissance mission (i.e. find and identify targets within an area) performed by multiple UAVs.

The first technique that was selected to be used in the thesis is the strategy for repairing task plans called replanning as repair. This technique is based on repairing a plan by finding a whole new plan by using an enhanced version of the planning problem that contains macro operators generated from the plan that is to be repaired \[2, 3\]. Earlier works on macro operators have been focused on non-temporal planning and nothing has, to the author’s knowledge, been done regarding macro operators in a temporal context that are expressible in PDDL2.1. This means that parts of the thesis covers how macro operators can be extended to a temporal version of them, temporal macro operator (TMO), expressible in PDDL2.1.

The second technique that was selected is the task planner, Temporal Fast Downward (TFD). For now it is enough to know that TFD \[12\]: uses a search method to find a plan; uses a heuristic function to guide the search; and is a temporal task planner that commits all actions it adds to the plan to an exact time point (using this type of commitment will be called early commitment in time throughout the thesis).

1.3 Aim

The aim of this thesis is to investigate the restrictions a temporal planner that does early commitment in time have when integrating motion planning in it. Another aim is to explore whether TMOs can be used to decrease the repair time when using the replanning as repair strategy to repair temporal task plans with integrated motion plans. The final aim is to investigate if extending the heuristics in the task planner with geometric information can be beneficial.

1.4 Research Questions

Based on the aim of the thesis the following six research questions will be answered in the thesis, where RQ1 is related to the integration of task and motion planning; RQ2, RQ3 and RQ4 concern repairing plans by replanning with TMOs; and RQ5 and RQ6 concerns the extending the heuristic function of TFD with geometric information. The research questions are as follows:

RQ1 How can motion planning be integrated in a temporal planner that does early commitment in time?

RQ2 Can TMOs expressed using PDDL2.1, be generated automatically?

\[2\] Planning domain definition language version 2.1, a standard language for expressing temporal planning problems \[11\].

\[3\] It should be noted that macro operators (or composite actions as they are sometimes called) in a temporal context that are not expressed in PDDL (any versions) have been studied earlier.
1.5 Delimitations

In this thesis, the models of the scenarios have the following simplifications:

- It is assumed that all actions are deterministic.
- The geometrical model is restricted to two continuous dimensions, latitude and longitude, and one discrete dimension, altitude. This delimitation was done to avoid shifting the focus from integrating motion planning in task planning to effective motion planners (simpler motion planning problems can be solved quite fast by ineffective motion planners).
- Changing altitude can only be done with predetermined actions and paths.
- It is assumed that all UAVs have a constant velocity when flying.
- The only external events that are considered in the thesis are those that are predetermined by the scenario.
- Ensuring that two agents do not collide with each other is not considered.

In addition to the simplifications to the models, the thesis does not cover whether different task or motion planning algorithms would give a better result than the selected. Similarly, only the replanning as repair strategy will be investigated in the thesis. Finally, the thesis does not consider how the agent should react or how the world changes while the system finds a new plan. These are undeniable interesting questions but outside the scope of the thesis. Hence, it is assumed the world is frozen while the plan repair algorithm executes.

1.6 Thesis Outline

This thesis is divided as follows. Chapter 2 is a detailed description of the scenario. Following that is chapter 3, which provides the fundamental knowledge about task and motion planning that is needed to understand the chapters after it. Chapter 4 presents the recent research that this thesis is built upon. Chapter 5 is a presentation of the theoretical work that has been done that together with the theory and related work makes up the basis for the design of the system that has been implemented and is presented in chapter 6. The rest of the chapters are: chapter 7, method for evaluating the system; chapter 8, results; chapter 9, discussion of the results; chapter 10, conclusions drawn in the thesis; and chapter 11, future work.
This chapter describes the main scenario that the thesis presents a solution to. The first section is an overview of the scenario and the rest cover different parts of the scenario in more detail.

2.1 Scenario Overview

The scenario for this thesis is a reconnaissance mission. Essentially, there are a few agents that together have to find and identify all targets within an area. This is done by scanning the area with low resolution from a large distance to locate where the potential targets are located and then identify the potential targets by scanning them with high resolution from a short distance. In addition, the area can contain threats that the agents have to avoid. Finally, when all the targets have been identified, the agents have to return to the base. This description allows for quite some variants to the scenario depending on some parameters, which are shown in table 2.1. A few typical scenarios that will be used for evaluation are presented in appendix A.

Table 2.1: Parameters to the reconnaissance mission scenario.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent count</td>
<td>The number of agents that are available in the scenario.</td>
</tr>
<tr>
<td>Area size</td>
<td>The length and width of the area.</td>
</tr>
<tr>
<td>Base distance</td>
<td>The distance to the base.</td>
</tr>
<tr>
<td>Risk policies</td>
<td>Policies for which risks the agents may take concerning the threats.</td>
</tr>
<tr>
<td>Threats</td>
<td>The number of threats and their locations (relative to the area).</td>
</tr>
<tr>
<td>Targets</td>
<td>The number of targets and their locations (relative to the area).</td>
</tr>
<tr>
<td>Malfunction events</td>
<td>The type of events and when they occur.</td>
</tr>
</tbody>
</table>
Table 2.2: Specifications for the UAVs in the scenario.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cruise speed</td>
<td>130 km/h</td>
</tr>
<tr>
<td>Fuel capacity</td>
<td>105 kg</td>
</tr>
<tr>
<td>Fuel consumption$^1$</td>
<td>5 kg/h</td>
</tr>
<tr>
<td>Landing fuel margin</td>
<td>21 kg</td>
</tr>
<tr>
<td>Climb rate</td>
<td>4.5 m/s</td>
</tr>
<tr>
<td>Turn radius</td>
<td>47 m</td>
</tr>
<tr>
<td>EO/IR flight altitude</td>
<td>1.5 km</td>
</tr>
<tr>
<td>SAR flight altitude</td>
<td>5.5 km</td>
</tr>
</tbody>
</table>

$^1$ When flying at cruise speed.

2.2 Agents

All the agents in the scenarios are modelled after the Elbit Hermes 450 (a UAV). The flight specifications of the model are shown in table 2.2. Note that the specifications may deviate from the specifications of the platform in some cases. These deviations are there to ensure that the model conforms to the delimitations of the thesis.

To avoid being identified by any enemy radar, the agents can use their own radar equipment to identify any active radars within 30 km. More importantly, this enables a UAV to locate a threat before the UAV is within range of the threat’s radar (which range is 19 km). In addition to the radars, the UAVs are equipped with two types of sensors, presented below, for identifying objects.

The first sensor for identifying objects is a synthetic-aperture radar (SAR). This sensor can be used to do a low resolution survey, which locates all targets, of an area left or right of the UAV while flying at an altitude of 5.5 km. However, the data does not provide enough details to identify the targets. The sensor requires that the UAV moves in a straight line 10 km before a point to 10 km after the point to get enough information about the point. Hence, a UAV that uses the SAR and flies 20+x km in a straight line will scan an area that is x km long. Orthogonal to the travel direction, the coverage of the scanner is 5-25 km left or right of the UAV. Continuing with the same example, this means that the UAV will have scanned an area that is x km x 20 km after flying 20 + x km with the SAR. Figure 2.1 shows a top down view of the example.

The second sensor for identifying objects is an electro-optical, i.e. daylight, and infrared (EO/IR) sensor. This sensor is used to scan the targets in greater detail. However, it requires that the distance to the target is 3 km or less to get a resolution that is high enough. A full scan is assumed to take 5 s per target.

2.3 Threats

In the scenarios, all threats are short ranged surface-to-air missiles (SAMs). They are assumed to have a weapon range of 7 km with a maximum altitude of 5 km, an acceleration of 290 m/s$^2$ and a maximum speed of 3700 m/s (roughly Mach 3). In addition, the SAMs have radars that can find any UAV within 19 km. It is assumed that if a UAV is within the weapon range, then the UAV will be shot down. However, all UAVs that are surveying with the SAR do so at an altitude that is above the weapons maximum height, which means that they cannot be shot down. Figure 2.2 shows the weapon range of a threat.

---

$^1$http://www.israeli-weapons.com/weapons/aircraft/uav/hermes_450/Hermes_450.html
2.4 Malfunction Events

During the execution of a mission, agents may have malfunctions in their sensor equipment. These events are defined as a tuple \((t, x)\) where \(t\) is a time stamp for when an event occur and \(x\) is the event. All event types are specified in table 2.3.

2.5 Mission Policies

Reconnaissance missions can have different polices that describe what a UAV is allowed to do. In this thesis, there are two mission policies, no UAV losses and no UAV detection, and each mission must use exactly one of them. The former of the mission policies means that no UAV may be lost and the latter means that no UAV that flies on the lower altitude (1.5 km) may be detected by the SAMs. Naturally, the no UAV detection policy also implies that no UAVs may be lost (the SAM units must be able to detect a UAV to shoot it down).
Table 2.3: The different types of malfunction events in the scenarios.

<table>
<thead>
<tr>
<th>Event name</th>
<th>Require agent</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE_SAR_MALFUNCTION</td>
<td>No</td>
<td>The SAR scanner starts to malfunction for the agent that is the next to use its SAR scanner.</td>
</tr>
<tr>
<td>ACTIVE_EO/IR_MALFUNCTION</td>
<td>No</td>
<td>The EO/IR starts to malfunction for the agent that is the next to use its EO/IR sensor.</td>
</tr>
</tbody>
</table>
This chapter is an introduction to the relevant background knowledge that is needed to understand the related works in the next chapter. The covered areas are (in order): Task planning; motion planning; mutual exclusion; and metrics.

### 3.1 Task Planning

Task planning, also known as automatic planning or simply planning, is a sub-field within artificial intelligence that deals with how to combine actions to achieve a goal \[1\]. In other words, task planning is typically about finding a partially or totally ordered set of actions that ensures that a goal is reached. Hence, systems based on task planning are deliberative systems (i.e. they find out how to achieve their goal before they start acting). Unfortunately, being deliberative through task planning (in the most simple but still interesting case) generally comes with the cost of its PSPACE-complete complexity\[1\].

A lot of work has been done in the field of task planning, though not all is relevant for this thesis. The following sections give an overview of the relevant parts for this thesis. First is a section describing the definition of task planning with resources. Thereafter, is a section covering forward chaining planning algorithms. Following that is a short introduction of some relevant heuristics for this thesis. Finally, a section covering the temporal extension to task planning.

#### Classical Task Planning With Resources

Classical task planning can formally be defined with the so called classical formalism \[1\]. This section covers the definition of this formalism and an extension of it that includes resources. There are no examples of using the formalism in this section. Instead, one example can be found in appendix \[B\].

The classical formalism is expressed with a language based on first-order logic\[2\] with the restriction that there may be no function symbols and only a finite amount of predicate and

---

1 PSPACE is the set of all problems that has a polynomial space complexity (how much memory it needs). Note that this includes problem that has a non-polynomial time complexity.

2 It is assumed that the reader have knowledge about first-order logic and nothing but a short informal explanation of some terminology will be given as footnotes. For more information about first-order logic see appendix B in Automated Planning Theory and Practice \[1\], or any literature on first-order logic.
constant symbols can exist. It follows that this language cannot express arithmetic constraints unless it is restricted to a finite set of numbers 3. The rest of this section covers how the language is extended and how classical planning with resource problems can be defined using the language. In the definitions, object term is used as either a constant or a variable (the variable can be instantiated with a constant).

The first extension to the language enables it to express arithmetic constraints by extending the language to include a finite number of so called resource expressions (or resources for short) defined as follows 1:

\[\text{Definition 3.1} \quad \text{A resource expression is an expression } r(x_0, x_1, \ldots, x_n) \mapsto \mathbb{R} \text{ where } r \text{ is a unique name for the resource expression and } x_i \text{ is an object term for all } i \in \{0, 1, \ldots, n\}.\]

In addition to adding the resources, the language is extended with a set of numeric expressions (the numerical operators in the numerical expressions can vary but multiplication, addition, subtraction and division are common):

\[\text{Definition 3.2} \quad \text{A numeric expression is an arithmetic expression } f(x_0, x_1, \ldots, x_n) \mapsto \mathbb{R} \text{ where } x_i \text{ is a resource, a numeric expression or a constant real number for all } i \in \{0, 1, \ldots, n\}.\]

The numeric condition (restricted to <, ≤, =, ≥ and >), as defined below, is added to the language so that constraints based on resources can be expressed 1:

\[\text{Definition 3.3} \quad \text{A numeric condition is an Boolean function } f(x_0, x_1) \mapsto \{\text{true, false}\} \text{ where } x_0 \text{ and } x_1 \text{ are numeric expressions.}\]

Finally, resource assignments are defined as follows to allow that the value of resources can change 1:

\[\text{Definition 3.4} \quad \text{A resource assignment is tuple } (r, x) \text{ where: } r \text{ is the resource that is getting a new value; and } x \text{ is a numeric expression.}\]

Based on the extended language definition, a state is defined as (extending the classical definition of Ghallab, Nau and Traverso 1):

\[\text{Definition 3.5} \quad \text{A state, } s, \text{ is a tuple } (L^+, R) \text{ where: } L^+ \text{ is a set of ground atoms } 4; \text{ and } R \text{ is a function from ground resource expressions to real numbers. The closed world assumption is done for } L^+ \text{ so an atom } a \text{ is assumed to be true if } a \in L^+ \text{ and otherwise false.}\]

The two final components in the language are the operators and the actions. In the classical form they only consists of conditions and effects based on literals 5. However, the extended form includes the resources as well and is defined as follows (extending the version of Ghallab, Nau and Traverso 1):

\[\text{Definition 3.6} \quad \text{An operator is a tuple } (n, P_{lit}, P_{num}, E_{lit}, E_{num}) \text{ where:}\]

- \(n\) is \(o(x_0, x_1, \ldots, x_n)\) where \(x_i\) is a variable symbol for all \(i \in \{0, 1, \ldots, n\}\) and \(o\) is a unique object symbol.
- \(P_{lit}\) is a set of literals.
- \(P_{num}\) is a set of numeric conditions.

3With a finite set of numbers one can express all numbers with one constant symbol per number and one predicate symbol per instantiated relation and function.
4An atom is a predicate \(p(t_0, t_1, \ldots, t_n)\) where \(p\) is a predicate symbol and \(t_i\) is an object term for all \(i\). Moreover, an atom is grounded if \(t_i\) is grounded for all \(i\).
5A literal is either an atom or a negated atom (i.e. \(a\) and \(\neg a\) are literals if and only if \(a\) is an atom).
3.1. Task Planning

- $E_{lit}$ is a set of literals.
- $E_{num}$ is a set of numeric assignments.
- $\forall e (e \in E_{lit} \implies \neg e \notin E_{lit})$.
- $\forall (r, x)((r, x) \in E_{num} \implies \neg \exists (r, y)(x \neq y \land (r, y) \in E_{num}))$.

**Definition 3.7** An action is a ground instance of an operator.

$P$ with subscripts or superscripts are conditions to an operator or action in this definition and in the rest of the thesis unless otherwise state. Similarly, $E$ with subscripts or superscripts are the effects of the operator or action. Sometimes properties of actions and operators will be referred to using the following dot notation: $a:conditions$ refers to $(P_{lit}, P_{num})$ and $a:effects$ refers to $(E_{lit}, E_{num})$ for action or operator $a$.

For simplicity, the definition below defines an entailment operator that shows when a set of literals and resource conditions are true in a state or when the effects of an action makes them true. In the definitions, the following are notation is used: $atoms^+(L)$ is the set of all atoms in $L$, a set of literals; $atoms^-(L)$ is the set of all atoms occurring negated in $L$, a set of literals; and $evaluate(c, R)$ evaluates numeric condition $c$ using $R$ (the resource part of the state or the resource effects of the action).

**Definition 3.8** $(L^+, R) \models (C_{lit}, C_{num})$ if and only if $atoms^+(C_{lit}) \subseteq L^+ \land atoms^-(C_{lit}) \cap L^+ = \emptyset \land \forall e (e \in C_{num} \implies evaluate(e, R) = true)$

**Definition 3.9** $(n, P_{lit}, P_{num}, E_{lit}, E_{num}) \models (C_{lit}, C_{num})$ if and only if $C_{lit} \subseteq E_{lit} \land \forall e (e \in C_{num} \implies evaluate(e, E_{num}) = true)$.

To increase the readability in the thesis, the following abbreviations will be used: $(L^+, R) \models l$ abbreviates $(L^+, R) \models (\{l\}, \emptyset)$ when it is obvious that $l$ is a literal; and $(L^+, R) \models c$ abbreviates $(L^+, R) \models (\emptyset, \{c\})$ when it is obvious that $c$ is a resource constraint.

Based on the actions, a state transition function, $\gamma$, for the language can be defined as follows (extending the version of Ghallab, Nau and Traverso [1]):

**Definition 3.10** The state transition function $\gamma$ is defined for each pair of state $s = (L^+, R)$ and action $a = (n, P_{lit}, P_{num}, E_{lit}, E_{num})$ where $s \models (P_{lit}, P_{num})$ as $\gamma(s, a) = (L^+, R')$ where:

$L^+ = (L^+ \setminus atoms^-(E_{lit})) \cup atoms^+(E_{lit})$

$R' = \{(r, v)|(r, v) \in R \land \forall y ((r, y) \notin E_{num}) \cup \{(r, evaluate(y, R))|(r, y) \in E_{num}\}$

and otherwise undefined.

Using these definitions one can define a planning domain as a state transition system $\Sigma$.

**Definition 3.11** A planning domain is a state transition system $\Sigma = (S, A, \gamma)$ where: $S$ is the set of all states that can be constructed using the literals and resources in the language; $A$ is the set of all actions in the language; and $\gamma$ is the state transition function (defined above).

A planning problem is then defined as [1]:

---

6One should note that the classical version, without resources, is a finite state transition system [1]. However, introducing resources generates an infinite number of states since each resource can have an infinite number of values.
Definition 3.12 A planning problem is a tuple \((\Sigma, s_0, g)\) where: \(\Sigma\) is the planning domain \(\Sigma = (S, A, \gamma)\); \(s_0\) is the initial state \((s_0 \in S)\); and the goal is \(g = (G_{\text{lit}}, G_{\text{num}})\) where \(G_{\text{lit}}\) is a set of literals and \(G_{\text{num}}\) is a set of numeric conditions.

Finally, a classical task plan is defined as:

Definition 3.13 A plan is a sequence of actions \(<a_0, a_1, \ldots, a_n>\) where each \(a_i\) is an action in the language.

The definition of a classical task plan allows for all sorts of plans, a plan is after all simply a sequence of actions. However, there is one category of classical task plans that are of interest for a classical planning problem. This category consists of the classical task plans that are applicable in the initial state (i.e. it is possible to use all actions in the plan in order when starting from the initial state) and that results in a state where the goal holds when the whole plan has been applied to the initial state.

Forward-Chaining Planning

Forward-chaining (also known as forward search) planning is a planning technique that searches through the state space\(^7\) starting from the initial state\(^8\). The technique uses a list of open states that are yet to be visited, initially the only item in the list is the initial state. From this list, states are selected and visited until a goal state has been selected or there are no more states in the list. When visiting a state \(s\), the algorithm expands \(s\) using all actions \(a\) for which \((s, a)\) is defined. The resulting states \((s, a)\) for all \(a\) is then added to the list of yet to be visited states. Pseudo-code for the technique is shown in algorithm 3.1.

---

**Algorithm 3.1: FORWARD_SEARCH**

```
input : Initial state \(s_0\), Goal \(g\), Actions \(A\)
output : Plan or failure

begin
1 | \(s_i \leftarrow \text{SEARCH\_NODE}()\)
2 | \(s_i, \text{state} \leftarrow s_0\)
3 | \(s_i, \pi \leftarrow \text{LIST}()\)
4 | \(\text{open} \leftarrow \text{APPEND(LIST(), } s_i)\)
5 | while \(\neg \text{EMPTY(open)}\) do
6 |   | \(s \leftarrow \text{SELECT(open)}\)
7 |   | if \(s, \text{state} \vDash g\) then
8 |   |     | return \(s, \pi\)
9 |   end
10 | for \(a \in A\) do
11 |   | if \(\text{IS\_DEFINED}(\gamma(s, a, \text{state}))\) then
12 |     | \(s_{\text{new}} \leftarrow \text{SEARCH\_NODE}()\)
13 |     | \(s_{\text{new, state}} \leftarrow \gamma(s, \text{state}, a)\)
14 |     | \(s_{\text{new, } \pi} \leftarrow \text{APPEND}(s, \pi, a)\)
15 |     | \(\text{open} \leftarrow \text{APPEND(open, } s_{\text{new}})\)
16 |     end
17 end
18 end
19 return failure
20 end
```

One of the main benefits of this technique is that it always has complete knowledge about a state when visiting it\(^9\). This allow a forward search algorithm to use heuristics to estimate

---

\(^7\)Remember that a state space is a graph where each node is a state and each edge corresponds to an action. A path in the state space is then a plan going leading from the state that is the first node of the path to the state that is the last node of the path.

\(^8\)A classical planning problem.

\(^9\)A state when visiting it.
how far a state is from the goal and to use this knowledge to select which state to visit next. Moreover, it is simple to adapt the unguided forward search algorithm to utilise heuristics as shown in algorithm 3.2. Furthermore, multiple planning systems, for example Fast Forward [13] and Fast Downward [14], has shown that forward chaining planning can be effective.

Algorithm 3.2: HEURISTIC_FORWARD_SEARCH

```plaintext
input : Initial state s₀, Goal g, Actions A
output : Plan or failure

begin
  s₁ ← SEARCH_NODE()
  s₁.state ← s₀
  s₁.π ← LIST()
  open ← APPEND(LIST(), s₁)
  while ¬EMPTY(open) do
    s ← HEURISTICALLY_SELECT(open)
    if s.state ⊨ g then
      return s.π
    end
    for a ∈ A do
      if IS_DEFINED(γ(a, s.state)) then
        snew ← SEARCH_NODE()
        snew.state ← γ(s.state, a)
        snew.π ← APPEND(s.π, a)
        open ← APPEND(open, snew)
      end
    end
  end
  return failure
end
```

Heuristics

Heuristics, within forward chaining planning, usually refer to functions that decide which state the planning algorithm should visit next when searching [1]. One example, is line 7 in the HEURISTIC_FORWARD_SEARCH algorithm (algorithm 3.2) where the heuristic function is used to select the state that is estimated to be closest to the goal. The heuristic function that is used in the example belongs to a category of heuristic functions that map a state and a goal to a value. Mathematically this is defined as function \( h: (L^+, R); (G_{lit}, G_{num}) \mapsto R \) where \((L^+, R)\) is a state and \((G_{lit}, G_{num})\) is a goal. This type of heuristic function is common in planning and there exists many such heuristic functions. Two are of specific interest for this thesis because the heuristic function that Temporal Fast Downward (TFD) uses is based on them: The additive heuristic \( h_{add} \) [1]; and the causal graph heuristic \( h_{cg} \) [15].

Additive Heuristic

The additive heuristic \( h_{add} \) is based on solving a simpler planning problem where resource constraints and the interaction between actions are ignored [1]. Computing the value for the heuristic is done according to equation 3.1. To get the estimated cost of reaching the goal, \( g = (G_{lit}, G_{num}) \), from a state, \( s = (L^+, R) \), one calculates \( h_{add}(s, g) \) using equation 3.1 where \( U \) is the set of all atoms.

\[
h_{add}(L^+, R); (G_{lit}, G_{num})) =
\text{satisfy}_\text{act}(\text{"goal"}; G_{lit}, G_{num}; \emptyset; \emptyset); (L^+ \cup \{\neg a | a \notin L^+ \land a \in U\}; R))
\] (3.1)
In equation 3.1, functions $satisfy\_act$ and $satisfy\_lit$ are defined as follows (where $cost(a)$ is the cost of applying action $a$ and $A$ is the set of all actions):

$$satisfy\_act(n, P_{lit}, P_{num}, E_{lit}, E_{num}, (L, R)) = \sum_{l \in P_{lit}} satisfy\_lit(l, (L, R))$$

$$satisfy\_lit(l, (L, R)) = \begin{cases} 0 & \text{if } (L, R) \models l \\ \arg\min_{a \in A_{uat1}} (cost(a) + satisfy\_act(a, (L, R))) & \text{else} \end{cases}$$

### Causal Graph Heuristic

The causal graph heuristic ($h_{cg}$) is like $h_{add}$ based on solving a simpler problem. However, this heuristic takes some interaction between actions into consideration but resource constraints are still ignored \[14, 15\]. Calculating $h_{cg}$ requires another formalism, called SAS*, than the classic formalism presented earlier in the chapter. In a few words, SAS* uses variables with finite domains instead of predicate symbols. For example, the location of an agent could be defined with a predicate ($is\_at\ agent\ ?location$) in the classical formalism. Using SAS* this would be represented by a variable $is\_at\ agent = location$ for each agent. Disregarding which formalism that is used, the number of reachable states in the planning problem is the same \[15\].

It is easy to generate a graph, called a domain transition graph (DTG) for each variable when using the SAS* formalism \[15\]. The DTGs describe how the value of the variables can change. Each node in a DTG for variable $v$ represents one of the values that $v$ can have and an edge labelled from node $n_i$ to node $n_j$ represents action $a$ that changes the value of $v$ from the
Temporal Planning

Temporal planning is an extension to standard task planning that introduces the notion of time. Naturally, there are many ways of handling this extension. However, in this thesis the approach described in this section is used. Hence, the claims and arguments about temporal planning applies to this approach and not temporal planning in general.

The first difference between non-temporal planning and temporal planning is that a plan is no longer a sequence of actions but a set of actions where every action has a start time and a duration. An implication of this is that the plan can contain concurrent actions. One way of handling this extension is to discretise time, as done in the TFD planning system. Based on the discretisation, one can extend the definition of classical planning with resources from section 3.1 to include the notion of time based on the source code of TFD (the planner described in Eyerich, Mattmüller and Röger’s paper) and the definition of PDDL2.1. An example for using the formalism can be found in appendix 3.1.1.

One starting point for extending task planning with time is to define active conditions (conditions that must hold now and from now to a specified amount of time) and scheduled effects (effects that will happen after a specified amount of time) as follows:

**Definition 3.14** An active condition is a tuple \((L, C, d)\) where: \(L\) is a set of literals; \(C\) is a set of numeric conditions; and \(d (d \in \mathbb{R}_{\geq 0})\) is a duration for how long the condition should hold. Note that an active condition where \(d = 0\) is an instantaneous condition, i.e. it must hold when the conditions becomes active conditions but not thereafter.

**Definition 3.15** A scheduled effect is a tuple \((L, R, t)\) where: \(L\) is a set of literals; \(R\) is a set of numeric assignments; and \(t\) is the time until the effect happens \((t \in \mathbb{R})\).

**Definition 3.16** A scheduled condition is a tuple \((L, C, t, d)\) where: \(L\) is a set of literals; \(C\) is a set of numeric conditions; \(t\) is the time until the condition has to be true \((t \in \mathbb{R}_{\geq 0})\); and \(d (d \in \mathbb{R}_{\geq 0})\) is a duration for how long the condition should hold.

Based on these three definitions, temporal state is defined as follows:
A temporal state is a tuple \((t, s, C^a, E^s, C^s)\) where: \(t\) is time point \((t \in \mathbb{R}_{\geq 0})\); \(s\) is a state; \(C^a\) is a set of active conditions; \(E^s\) is a set of scheduled effects; and \(C^s\) is a set of scheduled conditions.

To increase the readability \(s = C^a\) is used as an abbreviation for \(s = (L, R)\) when \(C^a = (L, R, t)\). Furthermore, \(s^\tau = x\) where \(s^\tau\) is the temporal state \((t, s, C^a, E^s, C^s)\) is used to abbreviate \(s = x\).

For each temporal state a progression function, \(\sigma\), is defined. The purpose of \(\sigma\) is the progress a temporal state \(s^\tau\) according to the active conditions, scheduled conditions and scheduled effects to a new temporal state \(s_{\tau}^\prime\). The time point of \(s_{\tau}^\prime\) is later than the time point of \(s^\tau\) and it is the earliest possible time point for which the active conditions, scheduled conditions or scheduled effects are not the same in \(s_{\tau}^\prime\) as in \(s^\tau\). Using the new time point it is possible to calculate state \(s_{\tau}^\prime\). Formally, this is done as follows:

The progression function \(\sigma\) for a temporal state \(s^\tau = (t, (L, R), C^a, E^s, C^s)\) is defined as \(\sigma(s^\tau) = (t', (L', R'), C'^a, E'^s, C'^s)\) where:

\[
\begin{align*}
t' &= t - \Delta \\
L' &= \{L \setminus \{\text{atoms}(E_{lit})(E_{lit}, E_{num}, \Delta) \in E^s\}\} \cup \\
& \quad \{\text{atoms}(E_{lit})(E_{lit}, E_{num}, \Delta) \in E^s\}\} \\
R' &= \{(r, x)|(r, x) \in R \land \exists y((r, y) \in \bigcup_{(E_{lit}, E_{num}, \Delta) \in E^s} E_{num})\} \cup \\
& \quad \{(r, \text{evaluate}(x, R))(r, x) \in \bigcup_{(E_{lit}, E_{num}, \Delta) \in E^s} E_{num}\} \\
C'^a &= \{(C_{lit}, C_{num}, d)|(C_{lit}, C_{num}, d + \Delta) \in C^a \land d \geq 0\} \cup \\
& \quad \{(C_{lit}, C_{num}, d)|(C_{lit}, C_{num}, t_r + \Delta, d) \in C^s \land t_r = 0\} \\
E'^s &= \{(E_{lit}, E_{num}, t_r)|(E_{lit}, E_{num}, t_r + \Delta) \in E^s \land t_r > 0\} \\
C'^s &= \{(C_{lit}, C_{num}, t_r, d)|(C_{lit}, C_{num}, t_r + \Delta, d) \in C^s \land t_r > 0\}
\end{align*}
\]

and \(\Delta = \min_{t_o \in \{t_{\text{num}}\}, (E_{lit}(E_{num}, t_{\text{num}}))(t_{\text{num}})(C_{lit}(C_{num}, t_{\text{num}}, d) \in C^s)} t_u\)

if and only if \(E^s \neq \emptyset \lor C^a \neq \emptyset\) and otherwise undefined.

Using the progression function, a consistent state (i.e. a state \(s^\tau\) for which all active conditions are true and for which \(\sigma(s^\tau)\) is either undefined or returns a consistent state) is defined as \[11\] and \[12\]:

A temporal state \(s^\tau = (t, s, C^a, E^s, C^s)\) is consistent, denoted \(\omega(s^\tau)\), if and only if:

- \(s = C^a\)
- \(E^s \neq \emptyset \land C^a = \emptyset\) \(\lor \omega(\sigma(s^\tau))\)

Following that a temporal operator is defined as \[11\]:

A temporal operator is a tuple \((n, s, d, P_{\text{pre, lit}}^{\text{pre}}, P_{\text{pre, num}}^{\text{pre}}, P_{\text{over, lit}}^{\text{over}}, P_{\text{over, num}}^{\text{over}}, P_{\text{post, lit}}^{\text{post}}, P_{\text{post, num}}^{\text{post}}, E_{\text{start, lit}}^{\text{start}}, E_{\text{start, num}}^{\text{start}}, E_{\text{end, lit}}^{\text{end}}, E_{\text{end, num}}^{\text{end}})\) where: \(n\) is a unique name \(\{x_0, x_1, \ldots, x_n\}\) where \(x_i\) is a variable symbol; \(s\) is a variable that is the start time of the operator; \(d\) is a numeric expression that is the duration of the temporal operator; \(P_{\text{pre, lit}}^{\text{pre}}, x \in \{\text{pre, over, post}\}\) are sets of literals; \(P_{\text{pre, num}}^{\text{pre}}, x \in \{\text{pre, over, post}\}\) are sets of numeric conditions; \(E_{\text{start, lit}}^{\text{start}}, x \in \{\text{start, end}\}\) are sets of literals; and \(E_{\text{start, num}}^{\text{start}}, x \in \{\text{start, end}\}\) are sets of numeric assignments.

\[11\text{Note that it is possible to get a consistent state from an inconsistent state.}\]
A temporal action is defined as \[ a = (n, t, d, P^\text{pre}_\text{lit}, P^\text{num}_\text{lit}, P^\text{pre}_\text{num}, P^\text{num}_\text{num}, t^\text{start}_\text{lit}, t^\text{start}_\text{num}, t^\text{end}_\text{lit}, t^\text{end}_\text{num}) \]

where \( \gamma^t(s^t, a) = (t', (L', R'), C'^a, E'^a, C'^s) \) where

\[
\begin{align*}
t' &= t \\
L' &= (L \setminus \text{atoms}^s(E^\text{start}_\text{lit})) \cup \text{atoms}^s(E^\text{start}_\text{num}) \\
R' &= \{ (r, x) | (r, x) \in R \land \neg \exists y (y, r, y) \in E^\text{start}_\text{num} \} \\
E'^s &= E^s \cup \{ (E^\text{end}_\text{lit}, E^\text{end}_\text{num}, \text{evaluate}(d, R)) \} \\
C'^a &= C^a \cup \{ (\text{evaluate}(d, R)) \} \\
C'^s &= C^s \cup \{ (\text{evaluate}(d, R), 0) \}
\end{align*}
\]

if and only if \((L, R) = (P^\text{start}_\text{lit}, P^\text{start}_\text{num})\) and otherwise undefined.

A planning problem is defined as:

\[
\Sigma^p = (S^p, A^p, \gamma^p)
\]

where \( S^p \) are the set of all temporal states in the language; \( A^p \) are all temporal actions in the language; and \( \gamma^p \) is the state transition function for temporal state and actions.

And a temporal plan is defined as:

\[
\pi = (s_0^t, g)
\]

where \( \Sigma^t \) is the planning domain \( \Sigma^t = (S^t, A^t, \gamma^t) \); \( s_0^t \) is the initial state (\( s_0^t \in S^t \)); and the goal is \( g = (G^\text{lit}, G^\text{num}) \) where \( G^\text{lit} \) is a set of literals and \( G^\text{num} \) is a set of numeric conditions.
3.2 Motion Planning

Motion planning is the task of finding a path free from collisions that an agent can travel between two points in an n-dimensional space \( \mathbb{R}^n \). Unfortunately, a complete search of all possible paths is intractable since the physical world is continuous. Therefore, there are an infinite number of possible paths. Moreover, it has been shown that a discretisation of the problem is PSPACE-hard in the general case \[19\]. Naturally, techniques have been proposed to tackle these problems. However, they have problems with handling many dimensions at a high resolution \[18\]. Fortunately, there are other ways than discretisation to handle the continuity of the dimensions and the many dimensions. One of those ways is to use sampling based planning. Essentially, these methods draws randomised positions that the agent can move between. These positions are then used to find a complete path \[18\].

The following sections cover: some common terminology for motion planning; motion planning using rapidly-exploring random tree (RRT); motion planning using probabilistic roadmap (PRM); Dubins path, a method for finding the shortest path between two nodes for agents that can move forward and turn in arcs; and Path pruning, an algorithm for reducing the length of a non-optimal path.

Common Terminology

Within motion planning there are two concepts that one should recognise and understand. These two are: \textbf{Work space (WS)} and \textbf{configuration space (CS)}. WS is the world that the robot exists in and CS describes the pose of the robot \[18\]. They are most easily clarified with a couple of examples.

A car, ignoring height difference on the ground, has a two dimensional WS and a three dimensional CS. The WS consists of a plane with x- and y-coordinates (i.e. the ground). Looking to the CS of the car, two of its dimensions are the same as those in the WS. However, it has a third dimension that is the orientation of the car, usually is denoted with \( \theta \).

Continuing with an example using a fixed wing aircraft. The fixed wing aircraft moves in a WS that has three dimensions (a room). On the other hand, the CS of the fixed wing aircraft has six dimensions. These are the x, y and z, which describes the location in the room (the same as in the WS) and the Euler angles (roll, pitch and yaw), which describe the orientation in the WS.

Two more important concepts, mostly for sampling based planning, are \textbf{local planner} and \textbf{collision checker}. A local planner is a less powerful (i.e. it can only find simple plans) but faster motion planner \[18\]. Collision checkers are what they sound like, algorithms that take a path and all objects as input and answers if the path is free from collisions or not \[18\].

Rapidly-exploring Random Tree

\textbf{Rapidly-exploring random tree (RRT)} is a \textbf{sampling based planning} method \[20\]. In the original form, it generates a search tree with a given number of nodes. All the nodes in the tree are configurations in the CS and each edge is a path between two configurations in the CS. Hence, a node \( n \) in the tree can be reached by following the paths (i.e. motion plans) associated with the edges that makes up the path (i.e. a path in graph theory) from the root node to \( n \). An algorithm for creating a RRT is presented in algorithm 3.3. In a few words, the algorithm draws a random configuration \( c_{\text{rand}} \) from the CS using a uniform distribution. Thereafter, it finds the closest configuration \( c_{\text{near}} \) in the search tree using a heuristic function. A new configuration \( c_{\text{new}} \) is selected that is a configurable distance, \( \Delta \), away from \( c_{\text{near}} \) towards...
3.2. Motion Planning

$c_{rand}$ along a possible path for the agent. The step of finding a path between $c_{near}$ and $c_{new}$ is done by using a local planner and a collision checker.

Algorithm 3.3: GENERATE_RRT

| input | : Initial configuration $c_{init}$, Node count in tree $N$, Distance $\Delta$ |
| output | : New search tree |

begin
1. $T$ ← TREE()
2. $T$ ← ADD_VERTEX($T, c_{init}$)
3. for $i \in \{1 \ldots N\}$ do
4.   $c_{rand}$ ← RANDOM_CONFIGURATION()
5.   $c_{near}$ ← GET_CLOSEST($c_{rand}, T$)
6.   $c_{new}$ ← STEP_TOWARDS($c_{near}, c_{rand}, \Delta$)
7.   $T$ ← ADD_VERTEX($T, c_{new}$)
8.   $T$ ← ADD_EDGE($T, c_{near}, c_{new}$)
9. end
10. return $T$
end

The algorithm for generating the general RRT has been modified since it was first designed so that $c_{new}$ is selected to be the closest configuration to $c_{rand}$ before any potential collisions. From this follows that $c_{rand}$ will be added if the local planner can find a collision free path between $c_{rand}$ and $c_{near}$ because $c_{new}$ will then be $c_{rand}$.

Finally, the RRT algorithm generates a tree and does therefore not find the solution by itself. However, one simple adaptation of the algorithm that can be used to find a solution is to use the goal configuration instead of $c_{rand}$ once in a while, for example every 100th node. Unfortunately, using this creates a bias towards the goal node which can lead to a higher risk of the algorithm getting stuck on some obstacles. Combining this with the above mentioned modification gives algorithm 3.4 (where GET_PATH extrapolates the path by moving from the leaf node $c_{goal}$ to the root $c_{init}$ in the tree).

Probabilistic Roadmap

The probabilistic roadmap (PRM) motion planning is one of the most common of sampling based planning algorithms within motion planning. PRM algorithms are divided into two different phases, a learning phase and a query phase. During the learning phase, a roadmap of the $CS$, essentially a set of graphs, is created (this is explained in more details in the following paragraphs). This roadmap is later used to connect the start configuration and the end configuration in the query phase. The following paragraphs provide a more detailed description of the two phases.

In the learning phase, configurations are randomised from the $CS$ using a uniform distribution and added to the roadmap as new nodes in the graphs. If a node is not free in the $CS$ (i.e. there is some other object that, partly, occupies the space that the agent is in when it has the configuration associated with the node), then it is discarded instead of added. Assuming that a node is in the free region, the next step is to connect the node to other nodes in the roadmap. This is done by using a local planner, a collision checker and a function that estimates the distance between two configurations. Essentially, the distance function is used to find a set of nodes, $N_{close}$, that are within a given distance (a parameter to the planner). Thereafter, the local planner is used to find valid paths between the new node and the nodes in $N_{close}$ and the collision checker is used to validate that they are free of

---

12 Note that the original implementation was based on a time step instead of a configured distance. However, that means that the velocity of the agent must be known when solving the motion planning problem. In this description, distance is used instead of time since distance is independent of the agent.
Algorithm 3.4: RRT\_SEARCH

\begin{algorithm}
\begin{algorithmic}[1]
\algstore{alg1}
\Statex \textbf{input} : Initial configuration \(c_{\text{init}}\), Goal configuration \(c_{\text{goal}}\), Maximum node count in tree \(N\), Goal check every \(G\)
\Statex \textbf{output} : Path or failure
\begin{algorithmic}
\begin{align*}
1 & \text{begin} \\
2 & \quad T \leftarrow \text{TREE()} \\
3 & \quad T \leftarrow \text{ADD\_VERTEX}(T, c_{\text{init}}) \\
4 & \quad r \leftarrow G \\
5 & \quad \text{for } i \in \{1 \ldots N\} \text{ do} \\
6 & \quad \quad \text{if } r = 0 \text{ then} \\
7 & \quad \quad \quad c_{\text{next}} \leftarrow c_{\text{goal}} \\
8 & \quad \quad \quad r \leftarrow G \\
9 & \quad \quad \text{else} \\
10 & \quad \quad \quad c_{\text{next}} \leftarrow \text{RANDOM\_CONFIGURATION()} \\
11 & \quad \quad \quad r \leftarrow r - 1 \\
12 & \quad \end{algorithmic}
13 & \quad \text{c}_{\text{near}} \leftarrow \text{GET\_CLOSEST}(c_{\text{next}}, T) \\
14 & \quad \text{c}_{\text{new}} \leftarrow \text{GET\_CLOSEST\_FREE\_CONFIGURATION}(c_{\text{near}}, c_{\text{next}}) \\
15 & \quad T \leftarrow \text{ADD\_VERTEX}(T, c_{\text{new}}) \\
16 & \quad T \leftarrow \text{ADD\_EDGE}(T, c_{\text{near}}, c_{\text{new}}) \\
17 & \quad \text{if } c_{\text{new}} = c_{\text{goal}} \text{ then} \\
18 & \quad \quad \text{return } \text{GET\_PATH}(T, c_{\text{goal}}, c_{\text{init}}) \\
19 & \quad \end{algorithmic}
20 & \text{end} \\
21 & \text{return } \text{failure} \\
\end{algorithmic}
\end{algorithm}
\end{algorithm}

obstacles. If a path free of obstacles is found for the new node and one of the nodes in \(N_{\text{close}}\) \((n_{\text{close}})\) and \(n_{\text{close}}\) belongs to a graph that the new node is not connected to, then an edge is created between the two nodes and it is added to the roadmap. The path is discarded if the local planner is deterministic and otherwise stored with the edge \(22\). This is repeated until a predefined coverage is reached or a predefined number of nodes has been added.

The result from the first step of the learning phase is a set of graphs that are a roadmap. Unfortunately, the number of disconnected graphs might be large \(22\). Hence, it is preferred to connect them so that it is possible to find paths that spans over more than one, currently, disconnected graphs. Connecting graphs is done by heuristically selecting nodes that are in hard to connect areas (for example, a small passage). Once a node has been selected a random walk is conducted from the node for a predefined time \(22\). That means, select a random direction and move in that direction from the node until a collision occur or the time runs out. If a collision occur, select a new random direction and continue. The resulting configuration is then added to the roadmap and edges are added as before (note that the path is explicitly stored since it is not generated by a deterministic algorithm).

The query phase begins after the learning phase is finished and a roadmap has been created. During the query phase one or more queries are answered. Answering a query is done in two steps: Connecting the start and the goal configuration to the roadmap; and finding a path using the roadmap \(22\). Connecting the start and the goal configuration is similar to connecting a new configuration in the learning phase. The main difference is that the initial and goal configuration has to be connected to the same graph in the roadmap. Therefore, one of the graphs of the roadmap is heuristically selected. Thereafter, the local planner and collision checker tries to connect the initial and goal configuration to the selected graph. If they fail to connect a configuration, then a random walk is done as described above. The algorithm then tries to connect the resulting node to the selected graph. If the algorithm fails to connect the new node, then a new graph from the roadmap is heuristically selected \(22\). Finally, when the two nodes are connected to the same graph a standard graph search algorithm is executed.
3.2. Motion Planning

(a) Dubins path RSR (right turn, straight, right turn).

(b) Dubins path LSL (left turn, straight, left turn).

(c) Dubins path RSL (right turn, straight, left turn).

(d) Dubins path LSR (left turn, straight, right turn).

(e) Dubins path RLR (right turn, left turn, right turn).

(f) Dubins path LRL (left turn, right turn, left turn).

Figure 3.2: The six different Dubins paths that can be constructed to connect two points. In the figures, $r$ is the minimum turn radius of the agent, the blue arrow is the agent’s current position and orientation and the green arrow is the goal position and orientation.

to find the path between the initial configuration and the goal configuration. Thereafter, the local planner is applied to the edges in the path that does not have an associated path to generate a motion plan. Note that no collision checking has to be done since the paths were verified to be free of obstacles in the learning step and the local planner is deterministic if no paths are stored.

Dubins Path

Dubins path is a method for finding the shortest path between two points with orientations in a two dimensional Euclidean space when there is a minimal turn radius for the path \( r \). Using the terminology from motion planning, Dubins path can be used to find the shortest path between two configurations in a $\mathbb{CS}$ that consists of $\{x, y, \theta\}$ for an agent that has two degrees of freedom, velocity $\mathbb{R}_{\geq 0}$ and turn angle, and that has a minimum turn radius. An example for such a scenario is a car that may only drive forward and steer. There are six different types of Dubins path that can be created for one problem, see figure 3.2, and one of them is guaranteed to be the shortest \( 23 \). Hence, finding the shortest is the simple task of calculating the six paths and then select the shortest of them.

Path Pruning

The path pruning algorithm is an algorithm with polynomial time complexity\( 13 \) that shortens a path consisting of waypoints by removing unnecessary waypoints \( 24 \). This is done by removing waypoint $c_{i+1}$ from the path if waypoint $c_i$ can be connected with waypoint $c_{i+2}$.

\[ 13 \text{The time complexity is a polynomial function.} \]

21
3.3 Mutual Exclusion

Mutual exclusion (sometimes abbreviated as mutex) is a term used in concurrency among processes in operative systems. It means, while a process $p_i$ is executing a critical section (i.e. a section of the code), then no other process $p_j$ may execute a critical section that is mutually exclusive with the critical section $p_i$ is executing [25]. There are various methods for protecting the critical sections of a program but, the mutex-lock is of most interest in this thesis.

A mutex-lock is used to protect mutual exclusive critical sections by requiring that a process acquire a so called lock before executing its critical section [25]. When the process is finished with executing its critical section, it will release the lock. Combining this with that only one process may hold a lock at the same time means that only one process can execute a section of the code protected by the lock. This leads to that a critical section is protected from inference by a lock as long as all mutual exclusive critical sections requires the same lock. An example for two processes is shown in figure 3.3.

3.4 Metrics

Comparing task planning systems with each other is something that has been done for a long time [14]. Naturally, this have resulted in that different metrics for comparing the systems have been developed. Some of them, that are relevant for this thesis, are explained in the following sections.

Search Time Score

The time it takes to find a plan is a common variable when evaluating a task planning system or task plan repair system, either as a measurement on itself or to test solved tasks as a function of search time [2, 6, 12, 27]. In the two latest International Planning Competitions

\[\text{Algorithm 3.5: PATH\_PRUNING}\]

\begin{verbatim}
input : Path p
output : Pruned path
1 begin
2     i ← 0
3     while i < LENGTH(p) - 2 do
4         if FIND_COLLISION_FREE_PATH(GET(p, i), GET(p, i + 2)) then
5             REMOVE(p, i + 1)
6         else
7             i ← i + 1
8         end
9     end
10 return p
end
\end{verbatim}

\[\text{Algorithm 3.5: PATH\_PRUNING}\]

\begin{verbatim}
input : Path p
output : Pruned path
1 begin
2     i ← 0
3     while i < LENGTH(p) - 2 do
4         if FIND_COLLISION_FREE_PATH(GET(p, i), GET(p, i + 2)) then
5             REMOVE(p, i + 1)
6         else
7             i ← i + 1
8         end
9     end
10 return p
end
\end{verbatim}
(IPC), search time has been used to calculate a score for a planning system for the so-called agile track\footnote{The agile track only consider how fast a plan is found and not the quality of the plan}. In addition, the score has been used to evaluate plan repair algorithms\footnote{The agile track only consider how fast a plan is found and not the quality of the plan}. The score $s$ is calculated using the following equation (where $T$ is the time it took for the planner to solve the problem and $T_{\text{max}}$ is the maximum allowed search time for the planner, 300 seconds in the IPC):\footnote{The agile track only consider how fast a plan is found and not the quality of the plan}

$$s = \begin{cases} 
1 & \text{if } T \leq 1 \\
0 & \text{else if } T > T_{\text{max}} \\
1 - \frac{\log(T)}{\log(T_{\text{max}})} & \text{else}
\end{cases} \tag{3.2}$$

**Coverage**

Coverage is a metric for evaluating how many problems a task planner can solve\footnote{The agile track only consider how fast a plan is found and not the quality of the plan}. This is a measurement that has been used to compare optimal task planners and task planning problems with bounded cost (all plans with a cost lower or equal to a limit is treated as equally good solutions)\footnote{The agile track only consider how fast a plan is found and not the quality of the plan}. The essence of the measurement is to count all the solved problems within a specified time limit (the limit is required since solving a problem is quite easy while doing it fast is not).
Related Work

This chapter covers the work that is the basis for integrating motion planning in task planning and generating temporal macro operators. It starts with a description of the temporal task planner that is used in the thesis. Thereafter, follows a description of different designs for integrating motion planning in task planning. The final section covers generation and filtering of non-temporal macro operators.

4.1 Temporal Fast Downward

Temporal Fast Downward (TFD) is an extension to the Fast Downward planner (see Helmert’s paper for more information about the Fast Downward planner \[14\]) that adds support for temporal planning \[12\]. This extension can be divided into two main categories:

- Changes to the forward chaining search algorithm.
- Adapting the heuristic to work with temporal problems.

Search Procedure

The search procedure that TFD uses is a standard forward-chaining A*-search \[1\] (see subsection Forward-Chaining Planning in section 3.1). The main difference is that the search is adapted to the definition of a temporal state (see subsection Temporal Planning in section 3.1). In particular, a state is expanded with all applicable actions and by progressing the state using the progression function for temporal states when it is visited. Hence, a plan is created, as shown in more details in algorithm \[4.1\], during the search by: adding all actions that should start at the time point 0; progress the temporal state to the next interesting time point by using the progression function \(\sigma\) (see definition \[3.18\] in subsection Temporal Planning in section 3.1); add all actions that should start at that time point; progress the temporal state; add all actions that should start at the new time point; et cetera. The add-actions-then-progress cycle is repeated until the whole plan is constructed.

\[1\] The A*-search means that the function HEURISTICALLY_SELECT returns the state that minimises \(h + g\), where \(h\) is the heuristic value of the state and \(g\) is the cost for reaching the state. TFD can also be configured to use a forward-chaining greedy best-first search which means that the function HEURISTICALLY_SELECT returns the state that has the lowest heuristic value.
Algorithm 4.1: TFD_SEARCH

\begin{verbatim}
input : Initial temporal state \( s_0 \), Goal \( g \), Temporal Actions \( A \)
output : Plan or failure
begin
\hspace{0.5em} s_i ← SEARCH_NODE()
\hspace{0.5em} s_i.state ← s_0
\hspace{0.5em} open ← LIST()
\hspace{0.5em} while ~(EMPTY(open)) do
\hspace{1em} s ← HEURISTICALLY_SELECT(open)
\hspace{1em} if s.state.scheduled_effects = ∅ ∧ s.state.scheduled_conditions = ∅ ∧ s.state ⊨ g then
\hspace{1em} \hspace{1em} return s
\hspace{1em} end
\hspace{1em} for a ∈ A do
\hspace{1.5em} if IS_DEFINED((a; s.state)) then
\hspace{2em} s_new ← SEARCH_NODE()
\hspace{2em} s_new.state ← γ^+(s.state, a)
\hspace{2em} s_new.π ← APPEND(s.π, a)
\hspace{2em} if ω(s_new.state) then
\hspace{3em} open ← APPEND(open, s_new)
\hspace{1em} end
\hspace{1em} end
\hspace{1em} end
\hspace{1em} if IS_DEFINED(σ(s.state)) then
\hspace{1.5em} s_pro ← SEARCH_NODE()
\hspace{1.5em} s_pro.state ← σ(s.state)
\hspace{1.5em} s_pro.π ← s.π
\hspace{1.5em} if ω(s_pro.state) then
\hspace{2em} open ← APPEND(open, s_pro)
\hspace{1em} end
\hspace{1em} end
\hspace{1em} end
return failure
end
\end{verbatim}

In a bit more details, the algorithm used in TFD to search through the state space is based on visiting and expanding temporal states that have not been visited yet. This is done by keeping a list of temporal states that are not visited and from which the algorithm selects temporal states to visit until it visits a goal state, i.e., a temporal state with no scheduled conditions, no scheduled effects and that fulfills the goal (line 8, algorithm 4.1). The selection of which temporal state to visit next (line 7, algorithm 4.1) is done by selecting the temporal state closest to the goal according to the heuristic function that TFD uses (which is explained in the next section). If the temporal state that is visited is a goal state, then the plan that leads to it is returned (line 9, algorithm 4.1). Otherwise, the state is expanded and new temporal states are generated by:

- Applying the state transition function \( γ^+ \) to the temporal state \( s \) that is being visited and temporal action \( a \) for all temporal actions that \( γ^+(a, s) \) is defined for (line 11 to line 20, algorithm 4.1).

- Applying the progression function \( σ \) to the temporal state \( s \) that is being visited if \( σ(s) \) is defined (line 21 to line 28, algorithm 4.1).

All the temporal states that are generated are added to the list containing temporal states that are not visited. It should be noted that the planner ignores the states that it has already visited. However, this is left out of algorithm 4.1 to reduce the size of the pseudocode.
A final note regarding the Temporal Fast Downward’s search procedure is that it is trying to find a plan that takes as short time as possible to execute. This means that the cost of a plan is made equivalent to the time it takes to execute the plan. In case that two plans have the exact same time to execute, then the standard cost associated with an action is used to determine the cheapest plan. Similarly the costs of an action is considered to be the duration of the action if nothing else is specified.

It should be noted that Temporal Fast Downward adds actions to specific time points (i.e. the same time as in sections 4.2 and 4.4 where h heuristics does early commitment in time. Moreover, it does calculate the duration of the action as soon as it is added. Hence, Temporal Fast Downward does early commitment in time.

### Context-Enhanced Additive Heuristic

The context-enhanced additive heuristic \( h'_{cea} \) is a heuristic function that is combining the \( h_{cea} \) and \( h_{add} \) heuristics\(^2\). It considers the interaction between literals and is based on the same formalism, SAS\(^*\), as in \( h_{cea} \). Moreover, \( h'_{cea} \) is calculated recursively similarly as when calculating \( h_{add} \). Essentially, one decides the best action, \( a \), of making a condition true by recursively estimate the cost to make all the conditions for \( a \) true. The estimate is then the cost of \( a \) plus the estimated cost of making its preconditions true. Of course, the best way is to not use an action at all if the literal is already true in the current state. The heuristic value is the sum of this calculation for all conditions in the goal state.

There are two extension to the original version of \( h_{cea} \) (called \( h'_{cea} \) in this thesis): Resources and time. Both the original version and the extension will be covered in this section. It should be noted that all references to \( h_{cea} \) outside this section refers to the version with both extensions. Moreover, it should be noted that the extended version of \( h_{cea} \) is a non-admissible heuristic.

Calculating \( h'_{cea} \) is done according to equations 4.1, 4.2 and 4.3 where \( s \) is a state and \( c_g \) is the goal conditions \([12, 27]\). The following notation is used in the equation: val(\( X, s \)) means the value of variable \( X \) in state or set of conditions \( s \); vars(\( s \)) are all the variables in state or set of conditions \( s \); cond(\( a \)) are all the conditions of action \( a \); cond(\( X,a \)) is the condition of variable \( X \) in the conditions of action or set of conditions \( a \); effect(\( a \)) are all the effects of action \( a \); cost(\( a \)) is the cost of action \( a \); achieves(\( s,X,x \)) is the state that is the result of applying the sequence of actions that achieves \( X = x \) to state \( s \); \( A \) is the set of all actions.

\[
h'_{cea}(s,c_g) = \sum_{X \in \text{vars}(c_g)} f'(s,X,\text{val}(X,c_g))
\] (4.1)

\[
f'(s,X,x) = \begin{cases} 
0 & \text{if } \text{val}(X,s) = x \\
\arg\min_{a \in \{a_i|a_i \in A, \text{val}(X,\text{effect}(a_i)) = x\}} g'(s,X,x,a) & \text{else}
\end{cases}
\] (4.2)

\[
g'(s,X,x,a) = \text{cost}(a) + f'(s,X,\text{val}(X,\text{cond}(a)))+ \sum_{Y \in \text{vars(\text{cond}(a))} \setminus \{X\}} f'(\text{achieves}(s,X,x,Y,\text{val}(Y,\text{cond}(a))))
\] (4.3)

Extending \( h'_{cea} \) to support resources is done by handling the resource conditions similarly to the literals \([12]\). Calculating the extended version for state \( s \) and goal \( c_g \) is done with equations 4.4, 4.5, 4.6 and 4.7 where: cond(\( a \)) is the set of all preconditions (resource and literal) in action \( a \); cond(\( X,a \)) is the condition of variable \( X \) in the conditions of action or set of conditions \( a \); is_resource(\( x \)) is true if and only if \( x \) is a resource condition; vars(\( s \)) are the variables in the state if \( s \) is a state or the variables in a set of conditions if \( s \) is a set of conditions; prom(\( X,s,E,x \)) is true if there is an effect in the set of effects \( E \) that is making

---

\(^2\)See subsection Heuristics in section 3.1 for a description of the \( h_{cea} \) and \( h_{add} \) heuristics.
the value of $X$ in state $s$ closer to fulfilling resource condition $x$; $\text{fulfil}(X,s,c)$ is true if the value of variable $X$ in state $s$ fulfils condition $c$; and $A$ is the set of all actions.

$$h_{cea}(s,c_g) = \sum_{X\invars(c_g)} f(s, X, \text{cond}(X, c_g))$$

(4.4)

$$f(s, X, c) = \begin{cases} 0 & \text{if } \text{fulfil}(X, s, c) \\ \arg \min_{a \in \{a[A], a[A].\text{prom}(X,s,\text{effect}(a), c)\}} g_r(s, X, c, a) & \text{else if } \text{is\_resource}(X) \\ \arg \min_{a \in \{a[A],[a[A].\text{fulfil}(X,\text{effect}(a), c)\}} g_l(s, X, c, a) & \text{else} \end{cases}$$

(4.5)

$$g_r(s, X, c, a) = \text{cost}(a) + \sum_{Y\invars(\text{cond}(a))\setminus\{X\}} f(\text{achieves}(s, X, c), Y, \text{cond}(Y, a))$$

(4.6)

$$g_l(s, X, c, a) = \text{cost}(a) + f(s, X, \text{cond}(X, a)) + \sum_{Y\invars(\text{cond}(a))\setminus\{X\}} f(\text{achieves}(s, X, c), Y, \text{cond}(Y, a))$$

(4.7)

There are two interesting things in equation [13]. Firstly, the heuristic accepts any action that brings the state closer to fulfilling a resource condition when calculating $f(s, X, c)$ and $c$ is a resource condition [12]. Secondly, it does not consider if the resource condition is fulfilled or not since it does not recursively calculate the cost for the condition after applying an action [12].

The final extension to $h'_{cea}$ is the temporal extension. Handling the temporal aspect is done by transforming the temporal actions and then counting the heuristic function as in the non-temporal case. The transformation is done by transforming each temporal action into two non-temporal actions, a start action and a compressed action, which both have a cost that is the duration of the temporal action [12]. Both of the non-temporal actions have the same conditions. These are: the preconditions of the temporal actions; and all the over all conditions and post-conditions that does not follow from the start effect of the temporal action. The effects are where the two non-temporal actions differ. They are the start effects of the temporal action for the start action. For the compressed action, the effects are all start effects of the temporal action that is not made false by the end effects of the temporal action and the end effects of the temporal action. For a temporal action $a = (n, s, d, \text{pre}_\text{lit}, \text{pre}_\text{num}, \text{pre}_\text{power}, \text{pre}_\text{pov}, \text{pre}_\text{post}, \text{post}_\text{pov}, \text{post}_\text{num}, \text{post}_\text{power}, \text{post}_\text{lit}, \text{post}_\text{end}, \text{start}_\text{num}, \text{start}_\text{lit}, \text{end}_\text{num}, \text{end}_\text{lit})$ this means that the start action is

$$(n + \text{"start"}, \text{start}_\text{num}, \text{start}_\text{lit}, \text{start}_\text{power}, \text{start}_\text{pre}, \text{start}_\text{pov}, \text{start}_\text{post}, \text{end}_\text{num}, \text{end}_\text{lit})$$

and that the compressed action is

$$(n + \text{"compressed"}, \text{start}_\text{num}, \text{start}_\text{lit}, \text{start}_\text{power}, \text{start}_\text{pre}, \text{start}_\text{pov}, \text{start}_\text{post}, \text{end}_\text{num}, \text{end}_\text{lit})$$

4.2 Integration of Task and Motion Planning

There are three main designs that have been identified for solving the integrated problem of task planning and motion planning [6]. These are motion planning guided by task planning, task
4.2. Integration of Task and Motion Planning

The integration of task and motion planning is crucial in robotics and autonomous systems. This chapter will cover three main designs: task planning and then motion planning, motion planning guided by task planning, and task planning querying motion planning. These three designs are described in the following subsections. After that is a section explaining geometric backtracking and the problem it tries to solve.

**Motion Planning Guided by Task Planning**

Motion planning guided by task planning is slightly different from the other designs because task planning is the main component and not motion planning \[4\]. Basically, task planning is used to guide the search in the motion planning. This design will not be covered further in this thesis since the task planning is the main part of the thesis. The interested reader is referred to the paper by Bidot et al. \[4\].

**Task Planning Querying Motion Planning**

This design for the integration is based on the task planner querying a motion planner \[4\]. The basic idea is that some actions in the task planning problem have motion planning problems associated with them. For example, action $(\text{fly uav1 base target1})$ has the motion planning problem of finding a path from the configuration of uav1 at location base to the position of target1. One should note that the motion planning problem is to move from a configuration in the CS to a position in the WS. Naturally, an action can have a motion planning problem that requires finding a path to a configuration in the CS instead of to a position in the WS. When the task planner uses an action with a motion planning problem attached to it, then it queries a motion planner for a path. An important aspect of this category is that all backtracking is done on the task planning level \[4\].

**Task Planning and Then Motion Planning**

The task planning and then motion planning design is quite similar to the querying design. However, there are two main differences with this design. Firstly, the task planner is not aware of the integration \[4\]. Secondly, the task planner cannot backtrack based on the information from the motion planner \[4\]. The structure for solving the problem is that the task planner finds a solution to the task planning part of the problem. Thereafter, the motion planner finds paths for all the actions in the task plan that have a motion planning component. If a path is successfully found for each action, then a solution has been found. Otherwise, the task planner is in invalid and a new task plan has to be found. Preventing the task planner from finding the same task plan again is done by adding extra conditions to the actions in the task planning problem \[4\].

**Geometric Backtracking**

When using the task planning querying motion planning design, there is an issue with backtracking that needs to be considered \[30\]. This originates from the fact that there are one or more valid motion plans for every partial task plan that is constructed by the task planner. Some task planners ignore this completely and the motion plan will therefore not change \[4\]. As a result, these planners might fail to find a solution even though there exist one. For example, assume that there is a partial task plan with a valid associated motion plan, $(\pi_0, \pi_0^p)$. However, when the planner adds a new action, $a$, to the task plan, it fails to find a valid motion plan, $p$. I.e. it fails to find $p$ in $(\pi_0^t + a, \pi_0^p + p)$. However, there might still exist a valid plan $(\pi_0^t + a, \pi_0^p + p)$ (the previous motion plan $\pi_0^p$ is replaced with a new motion plan $\pi_1^p$). Ignoring this option means that a system may incorrectly classify a task plan $\pi_0^t + a$ as invalid and therefore fails to find a solution. This is the problem that geometric backtracking aims to solve.

Geometric backtracking is an extension to the backtracking mechanism in a task planner \[4, 30\]. Usually, the backtracking in a task planner removes the last choice (adding an action
to the plan) and tries another choice instead (adding another action instead). However, with geometric backtracking, the task planner considers other alternative motion plans as well. If the task planner was unable to add an action due to geometric constraints, it will then try to find a new motion plan (i.e. it backtracks the choice of motion plans) first. In case that it is unable to find a new motion plan, the task planner backtracks in the task plan (i.e. remove the last added action and try another). As a result, all possible associated motion plans are tried before the task planner removes the action that was added last and tries another.

4.3 Task Plan Repair

Task plan repair is the problem of finding a new task plan given an old task plan that is no longer valid. There are two main reasons to why a plan is no longer valid:

- A state during execution of the plan differs from the expected so that at least one action in the plan cannot be executed or the goal will not be reached.
- The goal has changed.

Naturally, both these problems require that the plan is repaired if the goal is to be achieved. Multiple solutions have been proposed to solve this problem (e.g. Scala’s usage of macro actions and the plan adaption algorithm from Fox et al.). However, only the subset that is solving the task plan repair problem by replanning using macro actions or macro operators are considered in this thesis. Essentially, solving a task plan repair problem with replanning is to use a standard task planner to create a completely new plan from the current state. The macro operators and macro actions part are that the operators in the planning problem are extended with macro operators and macro actions before replanning.

The following subsections describe macro operators, techniques for generating macro operators and techniques for reducing the amount of generated operators. A distinction is made between algorithms for online generation of macro operators and algorithms for offline generation of macro operators. Where online generation is defined as generation of macro operators that must be done after the event that triggers a replanning problem; and offline generation is defined as the complement, i.e. all generation algorithms that can run before the event.

Macro Operators

A macro operator is a sequence of operators that are concatenated to a single operator. This means that the planner sees a macro operator as a single entity but the semantic meaning of the macro operator is the same as if all operators in the sequence are applied one after the other. In other words, a macro operator is a sub-part of a whole plan. For example, a macro operator could mean fly to ?target, scan ?target, fly to base and then land. Adding this macro operator to a plan has the same effects as if it had added the four actions in the order specified in the macro operator (i.e. fly, scan, fly and then land). In other words, the macro operator has provided a shortcut in the search. Of course, a macro operator is only useful if it actually achieves something. For example, a macro operator meaning fly to ?target, fly to base, fly to ?target and then fly to base is in most problems useless since its effects are the same as a single operator meaning fly to base.

Macro actions are related to macro operators in the same way as actions are related to operators, i.e. they are the grounded form of the macro operators. Throughout the rest of the thesis, a macro operator consisting of operators $a_i$ through $a_j$, $i < j$, will be written as $<a_i, a_{i+1}, \ldots, a_j>$. Similarly, a macro action consisting of actions $a_i$ through $a_j$ will be written as $<a_i, a_{i+1}, \ldots, a_j>$.

---

Testing all possible motion plans is theoretical since there can be infinitely many paths between two points. However, the algorithm can have a stop condition that is triggered when it is x% (e.g. 99.9%) sure that no path exists.
Constructing a macro operator from actions can be done with algorithm 4.2, adapted from Scala \[2\]. In the algorithm, \textsc{EMPTY\_OPERATOR} is a function that returns an unnamed operator with preconditions and effects that are empty sets. Moreover, \textsc{LIFT} is a function that replaces constants in the temporal operator with a variables. The constants that are replaced with variables are those that were variables in an operator that the macro operator was constructed from and that was not unified with a constant. For example, if the not-yet lifted macro operator \((\text{fly} \ uav\ \text{base} \ target1)\) was created from actions \((\text{fly} \ uav \ \text{base} \ target1)\) and \((\text{scan} \ uav \ target1)\) that are instants of operators \((\text{fly} \ ?uav \ ?from \ ?to)\) and \((\text{scan} \ ?uav \ target1)\), then the lifted version of the macro operator is \((\text{fly} \sim \text{scan} \ ?uav \ ?from \ target1)\). In the example, \(target1\) was not lifted to variable \(?\) to in the \(fly\) operator because \(?to\) was unified with constant \(target1\) in the \(scan\) operator.

Algorithm 4.2: CONSTRUCT\_MACRO

\[
\begin{align*}
\text{input} &: \text{list of actions } A \\
\text{output} &: \text{Macro operator} \\
\begin{align*}
1 & \text{begin} \\
2 & \quad o \leftarrow \text{EMPTY\_OPERATOR}() \\
3 & \quad \text{for } a \in A \text{ do} \\
4 & \quad \quad o.name \leftarrow o.name + a.name \\
5 & \quad \quad \text{for } p \in a.precond \text{ do} \\
6 & \quad \quad \quad \text{if } p \notin o.effect \text{ then} \\
7 & \quad \quad \quad \quad \text{ADD}(o.precond, p) \\
8 & \quad \quad \quad \text{end} \\
9 & \quad \quad \text{end} \\
10 & \quad \text{for } e \in a.effect \text{ do} \\
11 & \quad \quad \text{if } \neg e \in o.effect \text{ then} \\
12 & \quad \quad \quad \text{REMOVE}(o.effect, \neg e) \\
13 & \quad \quad \text{end} \\
14 & \quad \quad \text{ADD}(o.effect, e) \\
15 & \quad \text{end} \\
16 & \quad \text{return LIFT}(o) \\
17 & \text{end} \\
18 & \end{align*}
\]

Macro operators, once generated, can be used by any planner that supports the level of expressiveness used by the original domain \[2, 32\]. This means that the generation of the macro operators can be decoupled from the usage of them. Naturally, macro operators can also be used by a planner that recognises macro operators and use that information when planning \[8\]. Moreover, it has been shown that using macro operators can decrease the search time for a plan \[2, 8, 11, 32\]. The decrease in search time comes from decreasing the depth of the search \[32\]. Unfortunately, introducing more operators of any kind increases the branching factor, which can lead to longer search times \[2, 32\]. However, there are procedures that can be used to limit the increased branching factor for the generated macro operators \[2, 8, 11, 32\].

Online Generation

The following subsections cover algorithms that only can generate macro operators during a plan repair process.

\footnote{Constructing macro operators from operators can be done similarly as constructing macro actions from actions. First, variables in the operators are joined (i.e. variable \(?loc\) in operator \(o0\) must be instantiated with the same constant as variable \(?to\) in operator \(o1\), according to some strategy. Second, algorithm 4.2 is executed (note that \textsc{LIFT} will not have any effect since every term is a variable).}
Plan Suffix

The suffix macro operator generation is an algorithm that uses all suffixes of a task plan, π, as macro operators. This means that if \( \pi = \langle a_0, a_1, \ldots, a_i \rangle \), then the following macro operators are generated \( M = \{ \langle a_0, a_1, \ldots, a_i \rangle, \langle a_1, a_2, \ldots, a_i \rangle, \ldots, \langle a_{i-1}, a_i \rangle \} \) where \( a_j \) is an instantiated action of operator \( o_j \). The motivation for using suffix macro operator is that they can all reach the goal state, as long as it remains the same. Hence, if the planner can find a state where a suffix macro operator is applicable, then it can reach the goal in one step by adding the suffix macro operator (assuming that the goal remains the same).

Plan Prefix

The prefix macro operator generation is an algorithm that uses all prefixes of task plan \( \pi \) as macro operators. This means that if \( \pi = \langle a_0, a_1, \ldots, a_i \rangle \), then the following macro operators are generated \( M = \{ \langle a_0, a_1, \ldots, a_i \rangle, \langle a_0, a_1, \ldots, a_{i-1} \rangle, \ldots, \langle a_0, a_1 \rangle \} \) where \( a_j \) is an instantiation of operator \( o_j \). The motivation for using prefix macro operator is that they are easy to execute in the current state, as long as it remains the same. Hence, deordering the action can provide more useful macro operators than if the total ordering was enforced. Calculating the macro operators are done in three steps. Where the first step of the algorithm is to create two validation structure for the old plan, \( \pi \). These validation

4.3. Task Plan Repair

Causal Links

The causal link algorithm for generating macro operators is inspired by partial order planning in that it is based on finding actions in the old plan, \( \pi \), that threatens the new goal, \( g' \). In addition to finding the actions that threatens \( g' \), it finds the actions that does not have their preconditions satisfied (called open preconditions). Based on these actions, an ordered list of splitting indices for the plan are found. Essentially, if action \( a_i \) threatens \( g' \), then \( i + 1 \) is inserted in the splitting indices. Furthermore, if \( a_i \) has one or more open preconditions then, \( i \) is inserted in the splitting indices. In addition, 0 and the index of the last action in \( \pi \) are always splitting indices. Finally, a macro operator is created for each pair of consecutive splitting indices \( a \) and \( b \) for which \( b - a > 2 \). Hence, it follows that each macro operator that is created contains at least three operators. For example, if \( \pi = \langle a_0, a_1, a_2, a_3, a_4 \rangle \) and the splitting indices are \( < 0, 3, 4 > \) then only macro operator \( < a_0, a_1, a_2, a_3 > \) would be generated using indices 0 and 3 (indices 3 and 4 have a difference of 1 and is therefore not considered when generating macro operators).

Deordered Causal Link

The deordered causal link algorithm for generating macro operators is an extension to the causal link algorithm. The motivation for the algorithm is that even though two actions are totally ordered in a sequential plan, it is not necessary that they must be used in that order. For example, if \( \pi = \langle \text{unlock door}, \text{(start-motor)}, \text{(move room1 dest)} \rangle \), then it might be that \( \text{unlock door} \) and \( \text{(start-motor)} \) has to be executed before \( \text{(move room1 dest)} \). However, there might not be a requirement that \( \text{unlock door} \) is executed before \( \text{(start-motor)} \). Hence, deordering the action can provide more useful macro operators than if the total ordering in \( \pi \) was enforced. Calculating the macro operators are done in three steps. Where the first step of the algorithm is to create two validation structure for the old plan, \( \pi \). These validation

---

5 Scala’s algorithms and Scala and Torasso’s algorithm for generation macro operators are in the original implementations generating macro actions. However, in this thesis, this is considered a filtering technique since it is decreasing the increased branching factor for the planner. Hence, the generation is presented in terms of macro operators.

6 One should note that if the plan becomes invalid because the current state is significantly different from what was assumed then this might not hold.

7 For information about partial order planning please see Weld’s paper from 1994.
structures are Propositional Validation Structure (PVS) for the literals in the action and Numeric Validation Structure (NVS) for the resources.

The PVS is calculated with the algorithm presented by Kambhampati. In a few words, the algorithm iterates over all the ordering relations among the actions in the task plan and checks if they are required. If an ordering relation is not required, then it is removed. A required ordering relation $a_i, a_j$ according to the algorithm, fulfills one of these requirements: $a_i$ is a fake action corresponding to the initial state; $a_j$ is a fake action corresponding to the goal; $a_i$ has literal $l$ as an effect and $a_j$ has $l$ as a condition; $a_j$ has literal $l$ as an effect and $a_i$ has $\neg l$ as a condition; or $a_i$ has a literal $l$ as an effect, $a_j$ has $\neg l$ as an effect and there exists a $a'$, $a_j < a'$, that has $\neg l$ as a condition.

Creating the NVS is done by calculating a set of actions, Numeric Dependency Set (NDS), for each action in $\pi$. Calculating NDS for action $a_j$ is done as follows: for each resource condition $c$ among the conditions of $a_j$, greedily add $a_i$, $i < j$, that decreases the difference of the value of the resource in the current state and what is required by $c$. This is done until all $c$ is fulfilled or until an action $a_j$ is added so that $\exists a \in a_i, a_i+1, \ldots, a_j$ where $a$ is an action that cannot be executed in its position in $\pi$ (this can happen since $\pi$ no longer valid).

In the second step of the algorithm, the total ordering of $\pi$ is relaxed by removing all orderings that are not essential according to the calculated NVS or PVS. This results in one or more separate Directed Acyclic Graphs (DAGs).

In the final step of the algorithm, a macro operator is generated from each possible total ordering for each DAG. In the DAG, a node represent an action. This means that a total ordering of a DAG results in a macro action from which a macro operator can be extracted by replacing all constants with variables.

### Offline Generation

The following subsections cover algorithms for generating macro operators that has not been used during a plan repair process.

#### Plateau-Escaping

This method for generating macro operators is based on so called plateaus that are encountered when using the enforced hill climbing (EHC) search strategy. In a few words, EHC is a greedy search strategy that visits nodes in a depth-first manner while the nodes have a lower heuristic value than its parent. When a visited node does not have any children nodes with a lower heuristic value, i.e. a local optimum, a plateau is encountered. Escaping a plateau is done with a breadth-first search from the current node until a node, at any depth counted from the current node, with a lower heuristic value is encountered. Unfortunately, this escape procedure can be a quite expensive. Based on the sequence of actions that was used to escape a plateau during an EHC search, a macro operator is created. This macro operator is simply the sequence of operators that corresponds to the sequence of actions that was used to escape the plateau.

#### Component Abstraction

Component abstraction generation of macro operators is built on the fact that there are planning problems for which there are de facto static literals. In a few words, based on a planning domain and problem, it analyses which literals are static, i.e. those that never change. That is all literals in $\{l|s \models l, \forall(n,P_{lit},P_{num},E_{lit},E_{num})(n,P_{lit},P_{num},E_{lit},E_{num}) \in A \land l \notin E_{lit} \land \neg l \notin E_{lit}\}$, where $s$ is the initial state and $A$ are all the actions, are static literals.

---

\(^a\) $a_i < a_j$ means that $a_i$ happens before $a_j$ in the plan.

\(^b\) See Weld’s paper about partial ordering planning for reasoning to why the initial state and the goal are represented as fake actions.
The static literals are thereafter grouped together in components where all literals shares at least one constant. The next step is to filter the components so that there are no components that are equal if the constants are replaced with variables. For example, replacing the constants with variables in \{(move room1 DEST), (open room1)\} gives \{(move X Y), (open X)\} which also is a variable replacement for \{(move room2 corridor), (open room2)\}. Hence, only one of them would be kept. Finally, macro operators are constructed from each component by taking possible combinations of operators that use literals in the component \[32\]. Moreover, each constructed macro operator must fulfil the following:

- There are no cycles regarding the effects in the macro operator.
- All operators except the first in the macro operator are related in the sense that an operator always has at least one literal, \(l\), as its condition such that \(o_p \models l\) where \(o_p\) is the previous operator.
- The macro operator does not affect literals in any other component than the one it is based on.
- The macro operator has a maximum number of component operators (a parameter to the generator).

**Filtering**

The following subsections present methods for filtering macro operators. Filtering is done after macro operators have been generated but before they are added to the domain. Hence, the filtered macro operators are never a part of the search. Therefore, the filtering process is not a type of pruning.

**Usage Filtering**

Usage filtering is a simple offline filtering method based on how often a macro operator is selected by the planner when solving planning problems \[32\]. Essentially, a set of training problems, extended with the generated macro operators, are solved with the planner. The macro operators are thereafter ranked based on the found plans, \(\Pi\), according to their weights calculated as follows:

\[
w_i = \sum_{\pi \in \Pi} \left( \text{count}(m_i, \pi) + \text{exists}(m_i, \pi) \right)
\]

Where \(\text{count}(m, \pi)\) is a function that returns the number of times macro operator \(m\) is used in plan \(\pi\); \(\text{exists}(m, \pi)\) is a function that returns a constant, ten in the original implementation, if \(m\) is used in \(\pi\) otherwise zero; and \(w_i\) is the weight of \(m_i\). A predetermined number of macro operator with the highest weight (two to nine in the original papers \[8, 32\]), are selected and the rest are discarded \[8, 32\].

**Applicability and Goal Distance**

The applicability and goal distance filter uses an estimation of how hard it is to execute a macro operator and how far away the planner is from finding the goal after it has been executed to select macro operators \[2\]. Similar to the usage filter, this filter also selects a constant number of macro operators that has the best estimation, three in the original paper. However, it differs in that a higher value means that the macro operator is bad. Hence, it selects the macro operators with the lowest estimated values.

The estimations for this filter can be divided into two parts \[2\]: The part that deals with literals and the part that deals with resources. The estimation for the literals is a simple count of how many literals in the precondition that is not fulfilled in the current state and how
many goals that are not fulfilled by the effects of the macro operator. Hence, the literal part takes both the goal and the current state into consideration. The resource part, unlike the literal part, only considers the preconditions. It is calculated by summing the Euclidean distance between the current state and the requirements in the precondition for all resources in a precondition that is a linear expression. When both parts are calculated, they are added together to create an overall estimation. The macro operators with the lowest estimations are kept and the rest are discarded.

Using the definitions in the previous chapter, this calculation can be described by equation 4.8 for a macro operator \( o = (\mathcal{P}_{lit}, \mathcal{P}_{num}, \mathcal{E}_{lit}, \mathcal{E}_{num}) \), the initial state \( s \) and the goal \( g = (G_{lit}, G_{num}) \). In the equation \( \text{euclidean}(r, x, s) \) calculates the Euclidean distance between the required value, \( x \), of resource \( r \) and the value of \( r \) in state \( s \).

\[
w = |G_{lit} \setminus \mathcal{E}_{lit}| + |\{l \in \mathcal{P}_{lit}, s \not\models l \}| + \sum_{(r, x) \in \mathcal{P}_{num}} \text{euclidean}(r, x, s) \tag{4.8}
\]

**Grounding**

This filtering method is based on the idea that macro operators yield too many alternatives for the planner since there are, in the general case, many constants that can replace each variable in an operator. However, if the macro operators are grounded, i.e. one macro operator is instantiated to one macro action, then they will not introduce as many alternatives to the planner since there are no variables. Hence, using macro actions instead of macro operators yields a lower branching factor and thereby filters the result. This filtering technique has been used in the context of plan repair before. In those cases, the macro operators were generated from the original plan and their variables were replaced according to the constants of the actions they originated from. Naturally, this is usually done by never lifting the actions to operators in the first place.

---

\[10\] It is assumed that the original paper has a slight fault in the equation since it does not match the description in the original paper.
5 Theory Extension

This chapter presents two extensions to macro operators: path integration; and time. In addition, it covers how macro operators with the extensions can be constructed and generated. Finally, a method for evaluating the new macro operators is presented.

5.1 Path Integration in Temporal Macro Operators

This section covers how the motion planning part can be integrated in temporal macro operators (TMOs). There are two main points to why the integration has to be handled: The need for valid paths; and to utilise as much information from the old plan as possible. The goal of the integration is the same as with TMOs in general. Namely, to speed up the search procedure by providing as much useful information as possible.

There are three main approaches for how to handle path integration in a TMO generated from a task and path plan. Firstly, the TMO caches all paths that was used in the old plan. Secondly, the TMO caches some paths from the old plan, e.g. those that are still valid. Thirdly, the TMO does not cache any paths. All these three has benefits and drawbacks that are presented in the following subsections. However, before going into details about the three approaches, it should be noted that it can be impossible to cache a path if the path comes from an action that was lifted to an operator. For example, if \( p \) is the path that is associated with action \((\text{fly } \text{uav}1 \text{base target}1)\) and the action was lifted to \((\text{fly } \text{uav}1 \ ?\text{from} \ ?\text{to})\), then it would not be possible to cache \( p \) as a valid path because \( p \) is a path from path to target1 and not a path between two general locations.

In order to understand the following subsections, one must be aware of a major difference between TMOs and non-temporal macro operators. Namely, a TMO is a construct in a temporal planning context and can therefore, in the general case, consist of operators that are executed in parallel. Hence, a TMO is, unlike the non-temporal macro operators, not restricted to a sequence of operators.

\(^1\)There is the edge case where the only possible variable binding is that \(?\text{from}\) is replaced by \text{base} and the \(?\text{to}\) by \text{target}1. However, this is an edge case (i.e. it only works for a selected few problems) and is therefore not of interest for the general case.
5.1. Path Integration in Temporal Macro Operators

Full Caching

Using this method means that the TMO stores all paths used in the old plan. Naturally, this can be problematic if any of the paths are invalid. Another part that needs to be considered is that the paths have specific starting points in the CS. Unfortunately, it might be the case (and is in this thesis) that the operator’s associated motion planning problems are to find paths to a point in the WS. A result of this is that the task planner can find a temporal state, $s$, which fulfills the conditions of the TMO and has an associated configuration $\beta$ but the paths associated with the TMO require a start configuration $\alpha$, $\alpha \neq \beta$. For example, it could be that $uav1$ is at location base with configuration $(0, 0, z)$ and that TMO $fly uav1$ from base to target1, $scan$ target1, $fly$ back to base has an associated path that starts at configuration $(0, 0, \frac{z}{2})$. If the planner recognises that the configurations are different, then it risks to backtrack a lot while trying to find a new path that ensures that $uav1$ has configuration $(0, 0, z)$ by chance (remember that the operators associated motion planning problems are to find a path to a point in the WS). However, if it does not recognise that the configurations are different, then it risks generating a plan with invalid paths. Naturally, none of these options are a good solution to the problem.

Not everything with caching all paths in the TMO is bad. The method for integration has one significant benefit. This is that the TMO has, in addition to being a shortcut the task planning part, a solution to motion planning problems associated with the TMO (assuming that the paths are valid and that the $\alpha \neq \beta$ problem is solved).

Handling the problem with invalid path can be done by either discarding a TMO that contains invalid paths or by pruning the actions that are connected to the invalid paths $\beta$. These two solutions are the opposite of each other when it comes to the effect they have on the plan. Keeping the TMO and pruning the actions with the invalid paths introduces a risk of keeping a TMO that is not possible to execute. Hence, there is a risk of providing worthless and potentially misleading information to the planner. However, there is also a chance that a useful, though slightly different, TMO is introduced. Discarding the TMO guarantees that no wasteful and potentially misleading information is introduced. However, it risks discarding a useful TMO.

The second issue with full caching, requiring that the agents have specific configurations in the CS instead of the WS (the $\alpha \neq \beta$ problem), can be solved by injecting extra motion planning problems to the TMO. The extra motion planning problems consist of moving an agent from its current configuration to the configuration that is the starting point of its cached path in the TMO. The drawback with this solution is that the TMO takes slightly longer to execute (potentially longer than the component operators would have required if they were added one after another).

Partial Caching

This method for integrating the paths in a TMO only differs from full caching in how it handles the invalid paths. With partial caching, the invalid paths are discarded and replaced by new motion planning problems. Of course, the issue with connecting the agent’s configuration to the one required by the cached paths is also a problem with this method. However, it can be handled in the same way as for the full caching method.

Finding new paths instead of those that are now invalid has one main problem, the new paths must not only connect two configurations but they also have to ensure that the ordering

---

2There is a third option, to repair all invalid paths. However, this is the same as partial caching except that any new paths are found when generating macros instead of during the planning. Moreover, both the macro generation and the planning are done, in this thesis, when the plan repair problem occur. Hence, it does not matter (or the impact is highly likely the same) if the computation is done when generating macros or when planning.

3Valid paths can be discarded from a TMO in addition to the invalid paths with the aim to find better paths. However, this reduces the gain of information in the TMO.
between the operators the $TMO$ is made up from remains the same. This is because the paths that are found can have different length and can therefore take different time to travel. Hence, different paths can lead to different durations of the operators in the $TMO$. Naturally, when the duration of an operator changes, the ordering of operators in the $TMO$ can also change. For example, operator $o_1$ is scheduled between $[0, 3]$, $o_2$ between $[0, 2]$ and $o_3$ between $[2, 4]$ in a $TMO$. If the path of $o_2$ changes so that it would take 3 time units for the agent to travel the path instead of 2, then the ordering of the operators in the $TMO$ is no longer the same since the end of $o_2$ is no longer before the end of $o_1$ and the end of $o_2$ is no longer the same as the start of $o_3$. As a result, the $TMO$ made up from these three operators might not be executable any more.

One way to ensure that the ordering between the operators in the $TMO$ remains the same is to add constraints on the length of the paths. For example, a constraint that requires that the length of a path is equal to the speed of the agent times the wanted duration of the operator that the path is associated with in the $TMO$. Naturally, this assumes that the average speed when travelling a path is constant. Unfortunately, the number of ordering constraints grows as the $TMO$ increases in size (i.e. it is created from more operators). Moreover, the solution space to the motion planning problem can decrease when more constraints are added. This can in turn increase the search time.

The benefits of using partial caching is that no useful $TMO$ is discarded. However, it comes at the cost, as mentioned earlier, of risking longer search times when finding a path that satisfies the extra constraints.

No Caching

The final method for integrating the paths in the $TMO$s is the same as partial caching but it discards all old paths and not just a subset of them. Doing this means that the constraints that are enforced on the paths increase in number since every discarded path yields a set of constraints. Therefore, the planner will, in general, have more problem finding paths that fulfil the constraints and thereby risk having even longer search time. Otherwise, the two methods are equal.

Using no caching has the advantage over partial caching and full caching that it provides better opportunities for finding better paths in the $TMO$. Naturally, it comes at the cost of having to find those paths.

5.2 Temporal Macro Operator Composition

As written in chapter 4, there are algorithms for constructing a macro operator from a sequence of operators. However, there seems to be a lack of algorithms for constructing $TMO$s using the expression level of PDDL2.1 from a set of temporal operators. Naturally, constructing a $TMO$ for a planner that is aware of them is a simple case of providing a set of actions and temporal offsets for when they should be applied. Unfortunately, TFD and other planners that only supports standard PDDL2.1 are not aware of $TMO$s (i.e. they can only handle normal temporal operators). Hence, an algorithm had to be created to construct $TMO$s that can be expressed in PDDL2.1 (i.e. using normal temporal operators). The motivation is, as expressed in the introduction, that many temporal planners supports PDDL2.1. Hence, expressing the $TMO$s in PDDL2.1 results in that many planners is able to utilise them.

The parallelism that comes with temporal planning makes a $TMO$ slightly more complex than non-temporal macro operators. Take the three operators, of which two are executed in parallel, in figure 5.1 as an example. The following types of $TMO$s that are expressible in PDDL2.1 and that preserves the ordering of the operators they consist of, are presented in this thesis:
5.2. Temporal Macro Operator Composition

Figure 5.1: An example of an ideal temporal macro operator, \textit{tmo}, consisting of operators $o_0$, $o_1$ and $o_2$ where $o_1$ and $o_2$ are executed in parallel. Green boxes are conditions, blue boxes are effects and jagged parts on the timeline means that $\lim_{\epsilon \to 0} \epsilon$ time passes there. $T$ is used to denote a condition that is true in all states.

Figure 5.2: An example of an \textit{STMO}, \textit{stmo}, consisting of operators $o_0$, $o_1$ and $o_2$. Green boxes are conditions, blue boxes are effects, red boxes are mutex-locks (no other operator may use the mentioned resources and predicates when the \textit{STMO} executes) and jagged parts on the timeline means that $\lim_{\epsilon \to 0} \epsilon$ time passes there. Finally, the dashed boxes corresponds to conditions and effects in the component operators that are no longer used in the \textit{stmo}. $T$ is used to denote a condition that is true in all states.

- **Sequential temporal macro operator (STMO)** that preserves a subset of the opportunities for parallelism, illustrated in figure 5.2. The advantage with this type of TMOs is that it does not add a lot of overhead for the planner when it uses a TMO of this type.

- **Parallel temporal macro operator (PTMO)** that preserves all opportunities for parallelism, illustrated in figure 5.3. The disadvantage with this type of TMOs is that it adds a lot of overhead for the planner when it uses a TMO of this type.

Before explaining the two types of TMOs in detail, there are some terminology to introduce: A \textit{primitive (temporal) operator} is an operator that is not a (temporal) macro operator;
and a (temporal) macro operator consists of (temporal) component operators, the (temporal) operators it was constructed from.

**Sequential Temporal Macro Operator**

Sequential temporal macro operators (STMOs) are similar to non-temporal macro operators in that they are both abstracting all their component operators to a single operator. Because of that, the construction of a STMO is similar to the one for non-temporal macro operators. However, the parallelism of temporal planning adds some extra conditions to the STMO to preserve correctness of the plan in the event of other actions that are executed in parallel in the plan. These extra conditions will be covered in the two following sections, the first describes the conditions that are theoretically sufficient and the second describes the conditions that are used due to practical reasons. Following that is a section covering an algorithm for constructing STMOs.

**Theoretical Conditions for Correctness**

As stated above, a STMO requires extra conditions to ensure that a plan using it is correct. The problem comes from that a resource may have different values, a literal may change between true and false and conditions on literals and resources may change during the execution of the STMO. In addition, the planner is unaware of these changes when it creates the plan. For example, literal $r$ can be true or false at the start of the STMO in figure 5.2, but it becomes true when component operator $o_0$ has executed and must remain so until the STMO has executed. Because the planner does not know about the composition of the STMO, it can neither assume

---

\*\*Note that this definition allows for a (temporal) macro operator to be a (temporal) component operator to another (temporal) macro operator. However, it should be noted that this never happens in the thesis because the (temporal) macro operators are expanded to their (temporal) component operators when a plan is found. Hence, there are no (temporal) macro operators in any of the expanded plans and thereby among the (temporal) operators that the (temporal) macro operators are constructed from.

\*\*Remember that an operator, in PDDL2.1, is restricted to conditions that must hold at the start, at the end or during the execution of the operator and that all effects happens at the start or at the end of the operator.

---

39
that \( r \) is true (or false) for the duration of the \( \text{STMO} \) nor can it add an action that changes \( r \) to true or false at a point in time when the \( \text{STMO} \) is planned to execute to the plan without risking that the plan becomes invalid. As a result, some extra conditions must be added to ensure that all actions that are executed in parallel with the \( \text{STMO} \) does not conflict with these, to the task planner, unseen changes and conditions. The extra conditions can be divided into two categories of protection that affects the search and the resulting plan differently.

The first category of protection consists of using extra over all conditions. This category can be used when protecting something for which it is possible to use an over all condition to ensure correctness during the duration of the \( \text{STMO} \). These extra conditions originate from that there is one condition, \( c \), in a component operator that must hold at a point in time after the start and before the end of the \( \text{STMO} \). Moreover, \( c \) is not fulfilled by the effects of any component action that happens earlier in the \( \text{STMO} \). As a result, \( c \) must be fulfilled by an action outside the \( \text{STMO} \). Using this type of protection means that \( c \) must hold during the whole duration of the \( \text{STMO} \). Hence, there may not be any other unseen condition or effect in the \( \text{STMO} \) that conflicts with that \( c \) holds.

The second category of conditions is used when the first category cannot be used because the condition cannot hold during the duration of the \( \text{STMO} \). As a result, the planner cannot deduce whether a literal or resource condition, \( c \), is true or false at a specific time point during the execution of the \( \text{STMO} \). For example, literal \( c \) has to be true during the first half of the \( \text{STMO} \) and false during the second half of the \( \text{STMO} \). If this is the case, then neither \( c \) nor \( \neg c \) can be deduced to be true for the whole duration of the \( \text{STMO} \) when planning. Therefore, no over all condition that ensures correctness can be added. Note that this does not affect if the \( \text{STMO} \) can execute since they are, as described later on, formed so that they are internally valid (i.e. if the precondition are met, then they can execute). However, it does mean that no action except the \( \text{STMO} \) may modify \( c \) or rely on that \( c \) or \( \neg c \) holds during the duration of the \( \text{STMO} \). As a result, the opportunities for parallelism decreases.

There are similarities between the conditions in the second category and the critical section problem in concurrent process execution (see section 3.3). In concurrent process execution, two processes may not execute a mutual exclusive critical section at the same time. In the case of \( \text{STMOs} \), no concurrent action may modify or rely on a literal or resource that is protected using the second category. This similarity gives a solution for how to ensure that this does not happen. Just like mutex-locks can be used in concurrent processes, they can be used to protect \( x \), a literal or resource.

A mutex-lock for temporal task planning is a new predicate symbol with no terms, \( x_{\text{lock}} \), that is used as a mutex-lock to protect a critical section that is the duration of the \( \text{STMO} \). This is achieved by acquiring the mutex-lock when the \( \text{STMO} \) starts to execute (i.e. add \( x_{\text{lock}} \) to the start effects of the \( \text{STMO} \)) and releasing the mutex-lock when the \( \text{STMO} \) ends (i.e. add \( \neg x_{\text{lock}} \) to the end effect of the \( \text{STMO} \)). Moreover, to ensure that no other operator has acquired the lock the \( \text{STMO} \) has to have a precondition \( \neg x_{\text{lock}} \). Finally, all other operators that uses \( x \) are injected with an over all condition \( \neg x_{\text{lock}} \) that ensures that they respect that the mutex-lock is taken.

One of the drawbacks with using mutex-locks to protect a literal or a resource in a \( \text{STMO} \) is that it might require a lot of mutex-locks when the \( \text{STMO} \) grows larger. Thereby, increasing the size of a state which in turn can have a negative impact on the planner performance. Fortunately, the number of mutex-locks can be reduced by connecting a mutex-lock to a \( \text{STMO} \) instead of a literal or resource (i.e. each \( \text{STMO} \) that has one or more mutex-locks gets one mutex-lock instead). This means that the mutex-lock is used to protect all literals and resources that have a critical section in the \( \text{STMO} \). It is easy to see that this works because all locks from the same \( \text{STMO} \) are acquired and released at the same time. Of course, this mutex-lock has to be injected in all operators that uses any of the literals or resources that the mutex-lock protects.

When proving which type of protections that are needed in a \( \text{STMO} \), \( o_{\text{stmo}} \), to ensure the correctness of the found plan, the following rules applies:
1. If a literal, $p$, can be inferred to be true between $o_{stmo}.time_{start}$ and $o_{stmo}.time_{end}$, there is a condition that $p$ is true between those points in time and there are no effects regarding $p$, then $o_{stmo}$ must have $p$ as an over all condition to ensure correctness.

2. If a resource $r$ is used in an effect or condition in $o_{stmo}$, it can be proven $r \in d$ during the execution of the $o_{stmo}$, that all conditions related to $r$ in $o_{stmo}$ holds if $r \in d$ and $d$ is the largest domain fulfilling this, then $o_{stmo}$ must have $r \in d$ as an over all condition to ensure correctness.

3. If a literal $p$ is used in a condition in $o_{stmo}$ that cannot be inferred to always be true then the mutex-lock for $o_{stmo}$ must protect the literal to ensure that nothing else changes or relies on $p$. This is to model that the planner does not know if $p$ is true or false at time point $t$, $o_{stmo}.time_{start} < t < o_{stmo}.time_{end}$.

4. If a literal $p$ is an effect of a component operator in $o_{stmo}$ at time point $t$, $o_{stmo}.time_{start} < t < o_{stmo}.time_{end}$, and $p$ is not protected by an over all condition, then the mutex-lock for $o_{stmo}$ must be used to protect $p$. This protects $p$ from that an external action changes $p$ at a time point between $t$ and $o_{stmo}.time_{end}$.

5. If a resource $r$ is used in $o_{stmo}$ at time point $t$, $o_{stmo}.time_{start} < t < o_{stmo}.time_{end}$, but there is no domain $d$ such that $r \in d$ during the whole execution and all conditions related to $r$ in $o_{stmo}$ holds when $r \in d$, then the mutex-lock for $o_{stmo}$ must be used to protect the resource to ensure that nothing else changes or relies on $r$. This is to model that the planner does not know the value of $r$ at time point $t_u$, $o_{stmo}.time_{start} < t_u < o_{stmo}.time_{end}$.

6. If rule 1, 3 and 4 for a literal or rule 2 and 5 for a resource does not apply, then no extra condition is needed for that literal or resource.

**Practical Conditions for Correctness**

The two categories of protection works well in practice as well as in theory. However, the rules for when to use which type of condition does not work as well for all cases. Those concerning literals works since they can be computed fast. Unfortunately, finding an allowed domain for a resource is computationally harder. The problem is equal to finding all the possible solutions to the constraint satisfaction problem (CSP) that consists of all effects and conditions regarding the resource in the component operators. These constraints can be extracted as follows:

1. Initialise a resource binding $B$ with all resources mentioned in the $STMO$ to itself (i.e. $B[r] = r$, where $B[r]$ is the value for resource $r$ in $B$).

2. Sort all effects, preconditions, postconditions and over all conditions (based on the start time of the over all conditions) concerning resources by the time they happen or must be true, respectively. If two effects, conditions or an effect and a condition is scheduled to happen (or hold) at the same time, then the order in the plan is used. The plans are presented in a textual manner where each row is an action so an effect or condition from the action at row $x$ that happens (or holds) is sorted before an effect or condition from action at row $y$ that happens (or holds) at the same point in time if $y > x$.

3. Initialise an empty set of constraints, $C$.

4. Iterate through all the effects and conditions and do the following:

---

6Checking which type of protection a literal needs can be done by iterating over the conditions and effects in the component operators in the order they occur and keep track of if the literal is true, false and if it changes value. Hence, a literal can be checked in linear time to the number of effects and conditions in the component operators. Based on this a complete check for the macro can be done in $O(n \cdot m)$ where $n$ is the number of literals in the $STMO$ and $m$ is the number of effects and conditions in the $STMO$. 

---
a) If it is an effect, replace all resources in the right hand side of the assignment with their current value in $B$ and update the left hand side in $B$ (i.e. a resource $r$ in $B$, $B[r]$) to be the resulting right hand side in $B$. For example, if the effect is $r_i = s_3$ and the resource binding is $B[r_i] = r_i + 2$ then the new binding is $B[r_i] = r_i + 5$.

b) If it is an effect, for each over all condition that is active at the time that the effect happens: Copy the condition; replace all resources in the condition with their values in $B$; and add the resulting constraint to $C$.

c) If it is a condition, replace all resources in the condition with their values in $B$ and add the resulting constraint to $C$.

Finding all possible solution to the CSP is at least as hard as finding one of the solutions (which is NP-hard [1]). Because of the complexity of the CSP, the second and fifth rule for ensuring the correctness of a STMO is replaced by a new rule when used in practice. The new rules says that the mutex-lock of a STMO is used to protect every resource that is used in the STMO.

Constructing the STMO

Constructing a STMO can be done with algorithm 5.1. The algorithm starts by finding all resources that occur in the component actions and adds them to the set of resources and atoms that will be protected by a mutex (lines 18). Thereafter, the start and end time points of the component actions are calculated (lines 19-20) and stored in a list sorted on the time points. The order used in the plan that the STMO is created from (TFD presents all actions in a plan in a total order) is used to break any ties. Then the algorithm initialises an empty operator (stmo, line 17), a set that will contain all over all conditions that are currently active (O, line 21) and a knowledge representation of what is known to at the moment ($K$, line 14). At the start, the knowledge representation does not contain any information about literals and each resource that occurs in the component actions maps to the resource itself. Next, the algorithm iterates over all the start and end points of the component actions (lines 22-48) and updates the effects and the conditions of stmo, the knowledge representation and the set of atoms and resources that the mutex protects. In the iteration, the following sub-procedures are used:

- UPDATE_CONDITIONS($K, M, stmo, C$): This procedure updates the conditions of stmo, the set of atoms and resources that are protected by a mutex ($M$) and the current knowledge representation with the conditions $C$.

The numeric conditions in $C$ are the most simple since they are modified and then added as a precondition to stmo. Modifying the numeric conditions is done by replacing all resources in the numeric conditions with what they are mapped to in $K$. Using the mapped value means that the condition considers any changes made to the resources between the start of stmo and when the condition must be true. In other words, a condition that must be true at time point $t_1$ is equivalent to a condition at $t_0$, $t_0 < t_1$, where all resources have been updated with all the changes between $t_0$ and $t_1$. Naturally, this relies on the fact that the resource has not been changed by anything else between $t_0$ and $t_1$. Fortunately, this is assured since the mutex for the stmo hinders any other action from using the resources that occur in the stmo during the execution of the stmo.

The conditions in $C$ that concerns literals are in general a bit more complex than the resource conditions. However, there is a simple case when a condition concerning a literal is known to be true using the current knowledge $K$. In this case, the condition can safely be ignored. This is because it is known that the literal is true when the condition must hold and that there is either an over all condition or a mutex that prevents the literal from being made false by an action outside the STMO. Unfortunately, if the atom of a literal ($a$ if the literal is $a$ and $\neg a$ if the literal is $\neg a$) is not known in $K$ (it could be true
5.2. Temporal Macro Operator Composition

Algorithm 5.1: STMO

\textbf{input}: Temporal actions $A$

\textbf{output}: Required mutex-lock, Temporal macro operator

\begin{algorithmic}
\State $M \leftarrow \text{SET}()$
\For{$a \in A$}
\State $\Delta \leftarrow \arg \min_{a \in A} a.t_{\text{start}}$
\State $d \leftarrow \arg \max_{a \in A} a.t_{\text{end}} - \Delta$
\EndFor
\State $T \leftarrow \text{LIST\_ORDERED\_BY\_TIME\_POINT}()$
\For{$a \in A$}
\State $\text{INSERT}(T, <a.t_{\text{start}} - \Delta, \text{“start”}, a>)$
\State $\text{INSERT}(T, <a.t_{\text{end}} - \Delta, \text{“end”}, a>)$
\EndFor
\State $\text{stmo} \leftarrow \text{EMPTY\_TEMPORAL\_OPERATOR}()$
\State $K \leftarrow \text{KNOWLEDGE\_REPRESENTATION}()$
\State $O \leftarrow \text{SET}()$
\For{$<t, \text{type}, a> \in T$}
\For{$<\text{conditions}, to> \in O$}
\If{$to < t$}
\State $\text{UPDATE\_CONDITIONS}(K, \text{stmo}, \text{conditions})$
\Else
\State $\text{REMOVE}(O, <\text{conditions}, to>)$
\EndIf
\EndFor
\If{$t = 0$}
\State $\text{EXTEND}(\text{stmo}.\text{conditions}_{\text{pre}}, \text{a.conditions}_{\text{pre}})$
\State $\text{EXTEND}(K, \text{AS\_FACTS}(\text{a.conditions}_{\text{pre}}))$
\State $\text{APPLY}(K, \text{stmo}, \text{“start”}, \text{a.effects}_{\text{start}})$
\State $\text{ADD}(O, <\text{a.conditions}_{\text{over\_all}}, \text{a.time}_{\text{end}}>)$
\Else\If{$t = d$}
\State $\text{EXTEND}(\text{stmo}.\text{conditions}_{\text{post}}, \text{a.conditions}_{\text{post}})$
\State $\text{APPLY}(K, \text{stmo}, \text{“end”}, \text{a.effects}_{\text{end}})$
\Else
\If{$\text{type} = \text{start}$}
\State $\text{UPDATE\_CONDITIONS}(K, \text{M}, \text{stmo}, \text{a.conditions}_{\text{pre}})$
\State $\text{UPDATE\_EFFECTS}(K, \text{M}, \text{stmo}, \text{“start”}, \text{a.effects}_{\text{start}})$
\Else
\State $\text{UPDATE\_EFFECTS}(K, \text{M}, \text{stmo}, \text{“end”}, \text{a.effects}_{\text{end}})$
\EndIf
\EndIf
\EndIf
\State $\text{UPDATE\_CONDITIONS}(K, \text{M}, \text{stmo}, \text{a.conditions}_{\text{post}})$
\EndFor
\EndFor
\State $\text{ADD\_RESOURCE\_EFFECTS}(\text{stmo}, K)$
\State $\text{return} <\text{stmo}, \text{MUTEX\_LOCK}(M)>$
\EndAlgorithmic
5.2. Temporal Macro Operator Composition

or false), then the literal is added as a precondition to \textit{stmo}, \( K \) is updated to know that the literal is true and the \textit{stmo} gets the literal as an extra over all condition if the atom of the literal is not in \( M \).

There is a final case for conditions concerning literals: If the literal is known to be false in \( K \) but is required to be true, then the \textit{STMO} relies on that an action that is not a part of the \textit{STMO} makes the condition true. This means that the \textit{STMO} is internally invalid and the construction of it will therefore be aborted. Hence, no internally invalid \textit{STMO} will be created due to unfulfilled conditions.

- \textit{AS\_FACT(conditions)}: This function takes a set of conditions and returns all the literals.

- \textit{UPDATE\_EFFECTS(}\( K; M; \textit{stmo}; \textit{target}; \textit{effects} \textit{)}\textit{)}: This procedure updates the start or end effects of \( \textit{stmo} \) (depending on what is passed as the parameter \textit{target}) as well as the current knowledge \( K \) with \textit{effects}. The literal and the resource assignments are handled separately.

Resource assignments are handled by updating the current knowledge \( K \) with the effect as follows. Each resource assignment \((r, x)\) is transformed by replacing every resource in the numeric expression \( x \) with what it is mapped to in \( K \) yielding a new numeric expression \( y \). When all resource assignments are updated, \( K \) updates its mapping for each \((r, y)\) so that \( r \) is now mapped to \( y \) in \( K \).

The effects concerning literals are handled as follows: If a literal is known to be true in \( K \), then no changes are made at all since the effect does not change anything. However, if the atom in the literal is not known in \( K \) or the literal is known to be false in \( K \), then the effects of \( \textit{stmo} \) and \( K \) is updated. \( K \) is updated by setting the literal to be known as true and the negated literal to be known as false. \( \textit{stmo} \) is updated by removing all effects concerning the atom in the literal from the current start and end effects. Thereafter, the literal is added as a start effect of \( \textit{stmo} \) if \textit{target} is “start” and otherwise as an end effect of \( \textit{stmo} \).

- \textit{APPLY(}\( K; \textit{stmo}; \textit{target}; \textit{effects} \textit{)}\textit{)}: This procedure is similar to the UPDATE\_EFFECTS procedure and only differs in one aspect. The procedure never adds anything to what the mutex will protect. This simplification is possible because the procedure is only called for start effects of component actions that starts at the beginning of \( \textit{stmo} \) and end effects of component actions that ends at the end of \( \textit{stmo} \). Hence, \( \textit{stmo} \) models them at the correct time and no extra protection is therefore needed.

One can see that the iteration over the start and end points of the component actions consist of two main part: Handling the over all conditions that are currently active; and adding the new effects and conditions associated with the time point. The over all conditions are quite straight forward. If they are still active, then the \textit{STMO} must be updated so that the conditions are still valid at the new point in time (line 29). If they are not active, then they can be removed from the currently active over all conditions (line 27). The second part is to update the \textit{STMO} according to the new conditions and effects associated with the time point. This is divided into three different cases: The time point is the start of the \textit{STMO} (line 30); the time point is the end of the \textit{STMO} (line 35); and the time point is somewhere in between (line 38). This division is done because there is no need to create extra over all conditions or protect anything with the mutex to ensure the correctness of the \textit{STMO} when the effects and conditions happens at the same time point that they would in the component actions.

The final part of the algorithm is to add the resource effects to the \textit{STMO}. This is done with the ADD\_RESOURCE\_EFFECTS(\( \textit{stmo}, K \) \textit{)} procedure (line 43). The procedure works by adding all the resource mappings (resource \( r \) maps to numeric expression \( x \)) in \( K \) as a resource assignment to the end effects of \( \textit{stmo} \). During the iteration over all start and end
5.2. Temporal Macro Operator Composition

Parallel Temporal Macro Operator

Parallel temporal macro operators (PTMOs) are a type of TMOs that preserves all potential to parallelism. This is done by using extra literals and operators that forces the planner to add the component operators at specific points in time, as shown earlier in figure 5.3. The hardest part with PTMO is to include as much information (in the form of conditions and effects) as possible in the PTMO itself to get as accurate heuristic value and pruning opportunities as possible. Unfortunately, only the following are possible to prove for PTMO $o_{ptmo}$:

- **Preconditions:** The preconditions are all conditions $c$ in the component operators that can be traced to hold at $o_{ptmo}.time_{start}$ using the over all conditions in the component operators and no effects in the component operators makes $c$ not hold between $o_{ptmo}.time_{start}$ and the time $c$ is active. For example, if $c$ is a precondition of a component operator that starts at 5, then $c$ is a precondition of $o_{ptmo}$ if: There are over all conditions that requires that $c$ holds and that covers the interval $[0, 5]$; and there are no effects in the component operators in the interval $[0, 5]$ that make $c$ not hold.

- **Start effects:** The start effects of $o_{ptmo}$ are all start effects $e$ in the component operators that can be proven to hold between $o_{ptmo}.time_{start}$ and when its component operator starts. Moreover, it must be possible to apply $e$ multiple times with the same result. For example, a resource assignment that depends on the resource itself cannot be applied repeatedly with the same result. However, a literal effect can be applied multiple times with the same effect.

- **Over all conditions:** The over all conditions $c$ of $o_{ptmo}$ are all over all conditions in the component operators that: Can be traced to hold between $o_{ptmo}.time_{start}$ and $o_{ptmo}.time_{end}$ by using the over all conditions in the component operators; and there are no effects in the component operator happening in the interval $[o_{ptmo}.time_{start}, o_{ptmo}.time_{end}]$ that make $c$ not hold.

- **End effects:** The end effects of $o_{ptmo}$ are all the end effects in the component operators that can be proven to hold from the end of the component operator to the end of $o_{ptmo}$. For example, if $e$ is an end effect scheduled to happen at 10 and $o_{ptmo}$ has the duration 15, then $e$ is an end effect of $o_{ptmo}$ if $e$ can be proven to hold during $[10, 15]$.

- **Postconditions:** The postconditions of $o_{ptmo}$ are all conditions $c$ in all component operators in $o_{ptmo}$ that can be traced to hold until the end of $o_{ptmo}$ using the over all conditions in the component operators and no effects in the component operators make $c$ not hold between the time $c$ is active and the end of $o_{ptmo}$. For example, if $c$ is an over all condition between $[7, 13]$ and $o_{ptmo}$ has the duration 15, then $c$ is a postcondition of $o_{ptmo}$ if: There are over all conditions in the component operators that $c$ must hold during $[13, 15]$; and no effects in the component operators in the interval $[7, 15]$ that make $c$ not hold.

Analogous to the extra conditions when constructing a STMO, proving that an effect or a condition using resources is true over a time period is NP-hard in the general case. Therefore, only the provable literals are added to the PTMO in practice.
Generating a PTMO can be done with algorithm 5.2. This algorithm goes through all the temporal actions it is created from, the component actions, and adds conditions and effects according to the rules specified above. In the algorithm, the following procedures are used:

- **MAKE_NAME_UNIQUE(a)**: This procedure makes the name of operator a unique by appending a unique entity (for example an id) to its current name.
- **FIND_TRUE_CONDITIONS_BETWEEN(A, C, from, to)**: This function returns all the conditions c in C for which: there are over all conditions in the actions in A that ensures that c holds and that covers the interval ]from, to[; and there are no effects in A happening between ]from, to[ that ensures that c does not hold.
- **FIND_TRUE_EFFECTS_BETWEEN(A, E, from, to)**: This function works similar as the FIND_TRUE_CONDITIONS_BETWEEN function. However, it works on effects instead of conditions.
- **FIND_OR_GENERATE_START_PROVIDER(a, ptmo, component_operators, extra_goals, p_con)**: This procedure finds an operator (a provider) that ends at the same time as a starts and connects the two with a new unique atom. Moreover, the provider is marked as having provided a start criteria. There are three cases to how a provider is found and the first possible is used: If a starts at the same time as ptmo, then a is connected to ptmo: if a starts at the same time as any action a_c in component_operators, then a is connected to a_c: an if none of the earlier works, then a new action a_fill = {n, s, start, start, {p_con}, ¯c, c, c, c, c, c}, where n is a unique name and s is a variable, is created, added to the component_operators, connected to a and finally ptmo is connected to a_fill.

When connecting an action a_0 to a_1, a new unique atom p_con is used. The atom, p_con is added as an end effect to a_0 (if a_0 is ptmo, then p_con is added as a start effect instead of an end effect), p_con is added as a precondition to a_1 and the negated atom, ¯p_con, is added as a start effect of a_1. This ensures that a_1 is added to the plan after a_0. Moreover, it also ensures that only one instance of a_1 can be added per instance a_0. Finally, ¯p_con is added to the set extra_goals. The extra goals ensure that planner cannot generate a plan that has a PTMO unless all component operators are in the plan as well.

- **HAS_NOT_PROVIDED_START(component_operators)**: This function returns all the actions in component_operators that are not marked as having provided a start criteria.
- **FIND_OR_GENERATE_END_PROVIDER(a, ptmo, component_operators, extra_goals, p_con)**: This procedure is similar to the FIND_OR_GENERATE_START_PROVIDER but it looks at the end of action a instead of the start. This also means that: The end of ptmo is considered instead of the start; the start of the component_operators are considered instead of their ends; that action is created to be placed between the end of a to the end of ptmo if a new action is required because the two above were not possible; and that no actions are marked as start provider here. In all other aspects, the two procedures work in the same way.

A more detailed explanation of how algorithm 5.2 works is the following. Most of the work in the algorithm is done in an iteration over all the component actions (lines 14-20) ordered by the start time (as when constructing STMOs, the order of the plan the component actions come from is used to break a tie). In the iteration, a copy of the component action is created (line 19) and then added to a set of actions that is used in the PTMO (line 19). Moreover, the component action is also used to extract new preconditions (line 19), start effects (line 14),
over all conditions (line 15), end effects (line 16) and postconditions (line 17) that are added to the PTMO during the iteration. The final part of the iteration is to add a connection to the copy of the component action so that the planner cannot add the copy to a plan at a time point before it is supposed to be used in the PTMO (line 18). The last part of the algorithm is to iterate over all the component operators that have been added and add connections to them so that the planner cannot add them at a time point after they are to be used in the PTMO (lines 21-23).

Algorithm 5.2: PTMO

<table>
<thead>
<tr>
<th>input</th>
<th>: Temporal actions A, Duration d</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>: Temporal macro operator, Components, Extra goals</td>
</tr>
</tbody>
</table>

begin

ptmo ← EMPTY_TEMPORAL_OPERATOR()
component_actions ← SET()
extra_goals ← SET()

pin_macro ← GENERATE_UNIQUE_ATOM()
ADD(ptmo:effects_start, pin_macro)
ADD(ptmo:effects_end, ¬pin_macro)
ADD(extra_goals, pin_macro)

for action ∈ SORTED_ON_START(A) do

a ← COPY(action)
MAKE_NAME_UNIQUE(a)
ADD(a:conditions_over_all, pin_macro)

EXTEND(ptmo:conditions_pre, FIND_TRUE_CONDITIONS_BETWEEN(A, a:conditions_pre ∪ a:conditions_over_all ∪ a:conditions_post, 0, a:time_start))

EXTEND(ptmo:effects_start, FIND_TRUE_EFFECTS_BETWEEN(A, a:effects_start, 0, a:time_start))

EXTEND(ptmo:conditions_over_all, FIND_TRUE_CONDITIONS_BETWEEN(A, a:conditions_over_all, 0, d))

EXTEND(ptmo:effects_end, FIND_TRUE_EFFECTS_BETWEEN(A, a:effects_end, a:time_end, d))

EXTEND(ptmo:conditions_post, FIND_TRUE_CONDITIONS_BETWEEN(A, a:conditions_pre ∪ a:conditions_over_all ∪ a:conditions_post, a:time_end, d))

FIND_OR_GENERATE_START_PROVIDER(a, A, component_actions, extra_goals, pin_macro)
ADD(component_actions, a)

end

for a ∈ HAS_NOT_PROVIDED_START(component_actions) do

FIND_OR_GENERATE_END_PROVIDER(a, component_actions, extra_goals, pin_macro)

end

return < ptmo, component_actions, extra_goals >

end

5.3 Temporal Macro Operator Generation

Generating and filtering TMOs from temporal plans requires some changes to the algorithms for generating and filtering non-temporal macro operators from non-temporal plans. The changes that were done to the generation used in this thesis are presented below. In addition, a new evaluator used to evaluate STMOs and an evaluator for TMOs that is based on the applicability and goal distance filtering method are presented here. Both of the evaluators can be used to filter a set of TMOs by selecting the TMOs that are the best according to the evaluators. Exactly how the evaluators are used are described in more details in the next chapter.
Prefix and Suffix Generation

A requirement for using prefix and suffix based generation of TMOs are that there exist definitions for suffix and prefix. Unfortunately, a temporal plan is not totally ordered which means that the standard definition of prefix and suffix is not applicable for the temporal plans. Hence, definitions for prefix and suffix for temporal plans are needed to extend these types of generation of macro operators to handle TMOs. In this thesis, prefix and suffix of a temporal plan is defined by using a total order of a temporal plan as follows (omitting the conditions and effects of the temporal actions):

Definition 5.1 A total order of a temporal plan \( \pi \) consisting of \( n \) temporal actions that is presented as a sequence of temporal actions is 
\[
\text{total\_order}(\pi) = < (name_0, t_0, d_0, \ldots), (name_1, t_1, d_1, \ldots), \ldots, (name_{n-1}, t_{n-1}, d_{n-1}, \ldots) >
\]
where \( t_i \leq t_j \) for all \( i < j \leq n \). In addition, if \( t_i = t_j \), then the temporal action at index \( i \) is presented before the temporal action at index \( j \) in the presentation of the temporal plan.

The presentation order of temporal plan \( \pi \) is, in this thesis, the order that \( \text{TFD} \) presents the temporal plan, \( \pi \), that it found. This order is deterministic for a search, meaning that the presentation order is the same if the search was repeated. Moreover, all temporal plans that are used in the thesis are found by \( \text{TFD} \). Hence, the definition of a total order, using this presentation order to break ties, is sufficient for this thesis. Throughout the rest of the thesis, the fact that the temporal plan is presented as a sequence of temporal actions is implied when a total order of the plan is required.

Based on the total order of a temporal plan, a prefix of a temporal plan is defined as:

Definition 5.2 A prefix of a temporal plan \( \pi \) is a set consisting of all temporal actions in a prefix of total\_order(\( \pi \)).

Similarly a suffix of a temporal plan is defined as:

Definition 5.3 A suffix of a temporal plan \( \pi \) is a set consisting of all temporal actions in a suffix of total\_order(\( \pi \)).

Applicability and Goal Evaluator

The applicability and goal evaluator is a way of evaluating a TMO that is based on the applicability and goal distance filtering technique for non-temporal macro operators (near the end of section 4.3). Essentially, the evaluator is an extension of the equation in the applicability and goal distance filtering technique so that it can handle the time aspect. The extension is done similarly to how the \( b_{过} \) was extended to a temporal context. That is, by considering all the conditions that is not made true by the action itself and all the effects that the action does not negate itself. Essentially, it means that the evaluator is calculated as shown in the following equation for temporal action \( a = (n, t, d, lit, num, F_{\text{lit}}, F_{\text{num}}, P_{\text{pre}}, P_{\text{post}}, E_{\text{lit}}, E_{\text{num}}, E_{\text{over}}) \), goal \( g = (G_{\text{lit}}, G_{\text{num}}) \) and state \( s \):

\[
w(a,s,g) = |G_{\text{lit}} \times (E_{\text{lit}}^{end}) \cup \{ l \in E_{\text{lit}}^{start} \land l \notin E_{\text{lit}}^{end} \}| + \\
|\{ l \in P_{\text{pre}}^{start} \cup \{ l \in P_{\text{pre}}^{over} \land l \notin E_{\text{lit}}^{start} \} \cup \{ l \in P_{\text{post}}^{start} \land l \notin E_{\text{lit}}^{end} \cup E_{\text{start}} \} \land s \notin l \}| + \\
\sum_{(r,x) \in P_{\text{num}}} \text{euclidean}(r, x, s)
\]

Where \( \text{euclidean}(r, x, s) \) is the Euclidean distance between value \( x \) and the value of resource \( r \) in state \( s \).

As for the original equation in the applicability and goal distance filter, the lower value is, the better TMO is estimated to be by the applicability and goal evaluator.
5.3. Temporal Macro Operator Generation

Mutex Evaluator

The mutex-locks that are introduced with the STMOs needs to be injected in the operators that the planner uses. Naturally, this injection does not come for free in terms of time. Moreover, the time to inject a mutex-lock is dependent on the number of terms in the literals and resources it protects. Hence, a STMO that has a mutex-lock that protects fewer literals and resources is more likely to require less time to inject the operators. The mutex-lock evaluator is defined to capture the cost of injecting a STMO when selecting which TMOs to use. The calculation of the mutex evaluator for a TMO is done by summing the number of terms in all the literals and resources that the mutex-lock for the TMO is protecting. A low value means that the evaluator estimates that the cost for introducing the mutex-lock is small and a high value means that the evaluator estimates the cost to be high. Hence, a low value is preferred.
The system that was designed to solve the scenario consists of four main parts: A planner; a planning problem generator; a plan executor; and a primitive extractor. In this thesis, the plan executor is replaced with a simulator for the scenario. Moreover, the primitive extractor is simply replacing all TMOs with their component operators until no TMOs remains in the plan (i.e. it is more or less a regular expression). Hence, it will not be covered in the thesis. However, the other components are covered in this chapter.

A simplified overview (where some parallelism that are ignored) of how the system works is shown as a finite state machine in figure 6.1. The finite state machine and the overall workings of the system can be summarised as:

1. Generate a task and motion planning problem for the scenario given the current knowledge.
2. Solve the task and motion planning problem.
3. Replace all TMOs with their the instantiated component operators (called components in the paragraph). For a STMO, this is done by simply replacing the STMO with its components. For a PTMO, this is done by removing the PTMO operator (the operator called optmo in figure 5.3 and ptmo in algorithm 5.2 for constructing PTMOs) and all the extra actions that are used to ensure that its components are aligned in time. The remaining parts are then the copies of the PTMO’s components, which can easily be replaced with the components.
4. Execute the plan that was generated until the whole plan has been executed or some new information has been gained.
5. If new information was gained, go to step one (i.e. repair the problem by replanning with TMOs).

6.1 Planner

The planner that the system uses is a version of TFD that have been extended by integrating motion planning in it. This extension can be divided into three parts that are covered in the
6.1. Planner

Table 6.1: Effects and requirement on different integration designs (see section 4.2) for TFD in the context of the scenario in this thesis.

<table>
<thead>
<tr>
<th>Integration Design</th>
<th>Effects and requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion planning guided by task planning</td>
<td>This design mainly focuses on solving a motion planning problem which is not the case for the scenario in the thesis.</td>
</tr>
<tr>
<td>Task planning first, then Motion planning</td>
<td>The plans that TFD generates contain temporal actions with specific start and end times. As a result, the motion planning problems are restricted to finding plans that takes as long time to travel as the duration of the temporal action that uses the path.</td>
</tr>
<tr>
<td>Task planning querying motion planning $h_{cea}$</td>
<td>$h_{cea}$ uses the duration of temporal actions to estimate how far away a temporal state is from the goal. Hence, the motion planner must be able to estimate the duration of a path to make $h_{cea}$'s estimation of the duration of a temporal action that have an associated motion problem somewhat accurate.</td>
</tr>
</tbody>
</table>

following subsections: The choice of which type of integration design that was used and why; a description of the motion planning algorithm that was integrated in TFD; and how TFD's search procedure was updated to include the integration of motion planning.

Integration Design

The choice for how to integrate motion planning in TFD is restricted by two main factors. Firstly, the main part of the scenario is a task planning problem. This rules out the motion planning guided by task planning design since its main part is motion planning. Secondly, the early commitment in time that TFD does affects the designs for the integration. A short summary of the integration designs for TFD in this thesis are shown in table 6.1 and are covered in more details in the following paragraphs.

As one can see in the summary (table 6.1), task planning first, then motion planning is not suitable design when integrating motion planning in TFD. This is because TFD commits to a duration for an action when it plans. Hence, the motion planner will have to find a plan that takes a specific time to travel. As a result, the solution space for each motion planning problem is reduced and there is therefore a risk that the search time for a path increases. In addition,

$^1$The considered designs are those presented earlier in the thesis (see section 4.2).
the risk that the motion planner does not find a path at all increases when the solution space is reduced. Hence, the risk of having to replan also increases. Finally, this integration design either forces \( TFD \) to accurately estimate how long time a temporal action takes to execute or that the duration is accurately modelled by hand\(^2\).

Continuing with the task planning querying motion planning design, this does not have the drawback that the task planner forces a specific duration on a path. However, it has one main drawback when used in \( TFD \). Namely, \( h_{cea} \) uses the duration of temporal actions. This means that \( h_{cea} \) depends on the motion planner to get the duration for all temporal actions that have an associated motion planning problem. There are three main ways of handling this dependency:

1. Path estimation: Defining a heuristic function that estimates the duration of a path. This means that \( h_{cea} \) depends on another heuristic function to estimate the duration of a temporal action that have an associated motion planning problem. Naturally, the new heuristic function has to be defined.

2. Handwritten estimation: Let the model specify estimated durations for the temporal actions that have associated motion planning problems.

3. Blind estimation: Do not provide an accurate cost for the actions\(^3\) that requires a path. Naturally, this decreases the information gained by the heuristic. This can be implemented as setting the same cost to all actions that requires a path. Moreover, this is a non-admissible estimation.

When using the task planning querying motion planning design, there is the option to use geometric backtracking. Unfortunately, geometric backtracking clashes a bit with the early commitment in time that \( TFD \) does. Due to the commitment in time, changing the duration of an action can violate the ordering of the action and thereby invalidate the plan. Moreover, backtracking is not done in the type of search that \( TFD \) does. However, one can still do a type of geometric backtracking by extending the search in \( TFD \) to expand a visited node by finding new associated paths in addition to letting time pass and adding temporal actions to the plan.

Solving the problem of early commitment in time when finding new paths for an already existing task plan can, for example, be done by:

1. Find the ordering constraints between the temporal actions.
2. Find new paths for the relevant temporal actions.
3. Check that the ordering constraints hold.

The two last steps are then iterated until new valid paths are found for which the ordering constraints holds. If it is concluded that there are no more combinations of valid paths that can be tried, then there are no more possible paths for that partial task plan. This means that it is not possible to do a geometric backtrack in the temporal state.

The integration design that was selected for this thesis was the task planning querying motion planning. There are two main reasons to why this was selected: It can be extended to include geometric backtracking and it works with early commitment in time. In addition, two ways for handling the integration with \( h_{cea} \) was implemented: path estimation using the Euclidean distance in the \( WS \) (making it an admissible heuristic); and blind estimation.

\(^2\)The modelling can be done with PDDL2.1 syntax which means that some calculation can be done by the planner and the duration can therefore depend on the parameters of the action. Nevertheless, it is still an educated guess, for the scenario in this thesis.

\(^3\)All temporal actions used in \( TFD \) must have a duration so a complete void of cost is not possible. However, assigning the same estimation to all works.
(implemented by setting the estimated duration to 1 s for all temporal operators with a motion planning part). The motivation is that the path estimation is the only one of the three found strategies that semi-automatically provide information to the planner (the heuristic function still has to be defined). Moreover, the blind estimation was implemented to provide a baseline for comparison (as needed to answer RQ5 and RQ6).

**Motion Planner**

There are two main methods among the sampling based planning methods to select from when implementing the motion planner: PRM (probabilistic roadmap) and RRT (rapidly-exploring random tree). The PRM has the benefit that it can quickly answer multiple queries after a roadmap has been created. However, it does have the drawback that the roadmap has to be updated as more knowledge is gained about the world. The RRT has the benefit that it does not require any precomputations before a query is asked. Its main drawback is that it does not store any information between the queries. In addition to the restrictions of the different motion planners, the aim of this thesis is not to implement an effective motion planner. Hence, all the derivations and most extensions to the two types of motion planners was ruled out.

In the end, the RRT was selected because it does not require a preprocessing phase. The RRT planner that was implemented is similar to the original RRT planner described in section 3.2 and is shown in algorithm 6.1. As can be seen in the algorithm, there are two extensions to the basic RRT planner: Path pruning; and path caching. The path pruning is the one described in section 3.2. Path caching is a simple caching of paths so that a query to find a path between configuration α and position (or configuration) β is never calculated twice for the same α and β. The caching is needed since the task planner may add the same action at different points in time during the search. That is, the same query can be repeated multiple times. Using caching ensures that the motion planner has a deterministic behaviour for a query. As a result, the task planner will not prefer adding an action at a worse point in time because the path that was found (based on randomisation) happened to be better. Note that the caching is possible since the obstacles (i.e. objects that the agents can collide with) are static (i.e. they are never moved).

It should also be noted that the RRT planner that was used does not have any stop conditions except for finding a path. This is because it is assumed that if the start and the end are not within an object, then there exists a valid path. Naturally, this is not true in the general case. However, the assumption holds for the typical scenarios in this thesis. A RRT planner needs a local planner to connect two configurations. In this implementation, the local planner is a planner based on Dubins path that can either find a path to a point in the CS, or to a point in the WS. The former of the two is using the standard Dubins path method and the latter is not. However, finding a path to a point in the WS is a simplification of the standard problem since it does not restrict the orientation of the agent in the goal position. Hence, one can find the path by starting with the two circles at the start position, as in the standard method, and calculate the tangent from the circles (respecting directions) that passes through the goal position as shown in figure 6.2. Then it is a simple case of selecting the shorter of the two resulting paths.

The final component that the RRT planner needs is the collision checker. For this, a collision checker that relies on the fact that the scenario only has circles and squares as collision zones and that the path is made up of arcs and straight lines was used. It detects a collision by finding if there are any intersections between the geometrical objects that are the path and those that are the collision zones. A final note about the collision checker is that it is global, so all path segments are checked against all collision zones.

---

4 None of the extensions are presented in this thesis. However, there exist quite a lot of them.

5 Naturally, this could give better results in terms of plan quality. However, it would be better to use a motion planner that finds better paths instead.
Algorithm 6.1: RRT_AS_QUERY

**input**: Current configuration $c_{from}$, Goal position or configuration $to$, Collision objects $C$

**output**: Path or failure

1. begin
2.  if $COLLIDES(c_{from}, C) \lor COLLIDES(to, C)$ then
3.    return failure
4.  end
5.  cache ← GET_GLOBAL_CACHE()
6.  if $<c_{from}, to> \notin cache$ then
7.    return GET(cache, $<c_{from}, to>$)
8.  end
9.  $T \leftarrow$ TREE()
10. while true do
11.    $c_{rand} \leftarrow$ RANDOM_CONFIGURATION()
12.    $c_{near} \leftarrow$ GET_CLOSEST_WITH_VALID_PATH($c_{rand}, T$)
13.    if $c_{near} \neq$ null then
14.      ADD_VERTEX($T; c_{rand}$)
15.      ADD_EDGE($T; c_{near}; c_{rand}$)
16.    end
17.    if VALID_PATH($c_{rand}; to$) then
18.      ADD_VERTEX($T; to$)
19.      ADD_EDGE($T; c_{rand}; to$)
20.      path ← PATH_PRUNE(GET_PATH($T, to, c_{from}$))
21.      SET(cache, $<c_{from}, to>$, path)
22.      return path
23.  end
24. end
25. end

(a) Using a LS (left turn, straight) path to reach the goal.

(b) Using a RS (right turn, straight) path to reach the goal.

Figure 6.2: Finding the optimal path to a point in the WS using Dubins path.
6.2 Problem Generator

Generating a planning problem from the current knowledge of the scenario is in the simplest form done by translating the internal representation of the current knowledge of the scenario to the extended version of PDDL2.1 that the planner understands. However, one of the main parts of this thesis is to extend the planning problem with extra information in the form of TMOs, which is also done in this part of the system. Therefore, the problem generator does more than just a translation. In fact, it is easier to divide the problem generator to multiple parts as shown in figure 6.3.

The problem generator as a whole takes two inputs: the current knowledge about the world (including the goal of the scenario); and the old plan, if one exists. Based on the current


Algorithm 6.2: FORWARD_SEARCH_PATH

input : Initial temporal state \( s_0 \), Goal proposition \( g \), Temporal Actions \( A \)
output : Plan and associated path or failure

begin
    \( s_i:state \leftarrow s_0 \)
    \( s_i:path \leftarrow \text{LIST}() \)
    \( \text{expanded} \leftarrow \text{APPEND}() \)
    while \( \neg \text{EMPTY}(\text{expanded}) \) do
        \( s \leftarrow \text{HEURISTICALLY SELECT}(\text{expanded}) \)
        if \( s:state:scheduled\_effects = \emptyset \land s:state:scheduled\_conditions = \emptyset \land s \models g \) then
            return \( s:; s:path \)
        end
        for \( a \in A \) do
            if \( \text{APPLICABLE}(a; s:state) \) then
                \( \text{path} \leftarrow \text{FIND\_PATH}(s; a) \)
                if \( \text{IS\_VALID\_PATH}(\text{path}) \) then
                    \( s_{\text{new}}:state \leftarrow \gamma'(s:state; a) \)
                    \( s_{\text{new}}:\pi \leftarrow \text{APPEND}(s:\pi; a) \)
                    \( s_{\text{new}}:path \leftarrow \text{APPEND}(s: path; path) \)
                    if \( \omega(s_{\text{new}}:state) \) then
                        \( \text{expanded} \leftarrow \text{APPEND}(\text{expanded}; s_{\text{new}}) \)
                    end
                end
            end
        end
        \( s_{\text{pro}}:state \leftarrow \sigma(s:state) \)
        \( s_{\text{pro}}:\pi \leftarrow s:\pi \)
        \( s_{\text{pro}}:path \leftarrow s: path \)
        for \( a \in \text{PASSIVE\_AGENTS}(s) \) do
            \( \text{DECREASE\_FUEL}(s_{\text{pro}}; a; s_{\text{pro}}:time - s:time) \)
        end
        if \( \omega(s_{\text{pro}}:state) \) then
            \( \text{expanded} \leftarrow \text{APPEND}(\text{expanded}; s_{\text{pro}}) \)
        end
    end
return \( \text{failure} \)

knowledge of the world a minimal planning problem is created and the old plan is used to
generate a set of TMOs (the set is empty if there is no old plan) that are injected in the
minimal planning problem. The resulting problem is then sent to the translator to continue
the main computation and to a TMO preprocessing block that can be computed in parallel
with the plan executor. In the TMO preprocessing block, some internal data structures that
are used for generating TMOs are computed and passed to the TMO generator for the next
call to the problem generator. Finally, the translator translates the planning problem to the
extended PDDL2.1 and sends it on from the problem generator to the planner.

The following subsections will cover three parts of the problem generator in greater details:
The minimal problem generator; the TMO generator; and the TMO injector.
6.2. Problem Generator

Figure 6.3: An overview of the problem generator. The solid arrows represent data and execution transfer between the different parts of the problem generator. Dashed arrows are data transfer and forks, i.e. they execute in parallel with the rest of the system.

**Minimal Problem Generator**

The minimal problem generator is responsible for creating a minimal planning problem to solve the scenario as it is currently known. This includes translating the knowledge about the scenario to an initial state and a goal. Moreover, based on the initial state and the goal, the minimal problem generator selects all primitive temporal operators that may be needed to solve the problem.

Translating the world knowledge to the initial state and the goal is in most cases a simple transformation in the representation. However, there are two important steps when doing this. The first is to handle any temporal operators that are currently executing. This can happen since a replanning can be triggered at any time when the plan is executing. In most cases, it is enough to update the current position of the UAV. However, a few actions, for example scanning with the EO/IR and lifting, have to be finished. To solve this, a new temporal operator and a set of predicates are introduced to the planning problem. The temporal operator models the effects of finishing the execution of the temporal action. The predicates ensure that the planner adds that temporal operator with the correct agent at the start of the plan.

---

6Remember that it is a delimitation of the thesis that the world is frozen while the plan repair algorithm executes.
The second interesting part of translating the world knowledge to a planning problem is to introduce relevant discrete locations. Even though the agents can be at every single position allowed by the mission policies (i.e., it might not be allowed to be within the range of the SAMs’ scanners), all positions are not of interest when finding a plan. In fact, all the interesting positions for a planning problem are:

- The location of the home base.
- The location of all targets.
- The backup locations.
- The waypoints for the SAR scanning.
- The waypoints for the UAV following the UAV doing the SAR scanning.
- The current position of an agent that is at a different location than one of the above.

Solving the planning problem becomes significantly easier if only the relevant locations according to the current knowledge of the world are introduced. Doing this means that TFD never has to consider moving an agent to a location that is not of interest. For example, if the two first strips of SAR scanning has been completed, then the planner does not have to consider flying to the start of the first SAR strip because that location is not important to solve the problem. Therefore, the location can be removed from the task planning problem.

Selecting the set of primitive temporal operators that are in the planning problem is done by removing those that are unnecessary. For example, if there are no targets in the planning problem, then there is no need to include a temporal operator that scans a target with an EO/IR scanner. In this case, no task planner would ever be able to use that temporal operator because it requires a constant that is a target and no such constant exist in the planning problem. However, the task planner still has to figure this out. Hence, the time it takes to find a plan is reduced by not including the temporal operator.

### Temporal Macro Operator Generation

The generation of TMOs is designed according to the pipe-and-filter architecture as shown in figure 6.4. In the figure, the generation of the temporal macro operators can be divided into three layers: The input layer; the macro generation layer; and the filter layer. These layers will be covered in the following subsections.

Integrating the motion planning in the TMOs is done with the full caching method where all temporal actions with invalid paths are pruned, as described in section 5.1. This method was selected since the composition of STMOs fails if the resulting STMO would be internally invalid. Hence, the STMO’s composition nullifies the drawback with full caching. Unfortunately, the PTMOs are not ensured to be internally consistent, which means that there is a risk of generating internally invalid PTMOs. However, an internally invalid PTMO never results in an invalid plan since nothing is hidden from the planner in the PTMO. Finally, all TMOs have one extra motion planning problem per agent that has a cached path to ensure that the paths in the found plan are correct.

### Input Layer

The input layer consists of two filters. The first filter, valid and not executed, removes all executed actions and those with invalid paths. Removing executed actions is similar to how

---

7 The improvement can vary a lot depending on the planner. TFD have a preprocessing phase that instantiates temporal operators to all possible temporal actions. This limits the impact of having temporal operators that cannot be executed but it is still an improvement when they are not part of the planning problem.
6.2. Problem Generator

Figure 6.4: The pipe-and-filter architecture of the temporal macro operator generation.

macro operators have previously been used for plan repair. However, it differs in what is considered an executed temporal action. During the execution of a temporal plan, a temporal action can either be executed, partially executed or not executed. To handle this, partially executed temporal actions are treated like executed temporal actions when generating TMOs. The motivation for this is that the model does not describe what will happen if a temporal action is aborted during its execution. Hence, it is not safe to assume anything. Therefore, it is safer to ignore all partially executed temporal actions completely when generating TMOs.

The second filter, agent splitter, takes a temporal plan and knowledge about which agents that exist and splits the temporal plan into one temporal plan for each agent and one extra containing all temporal actions that are not connected to any of the agents.

**Generation Layer**

The generation layer consists of two parts: Generating sets of temporal actions that the TMOs are created from; and create TMOs from the sets of temporal actions. The generation of the sets of temporal actions is done by using the prefix and the suffix generation methods specified.

---

*A planning problem can, for example, contain temporal actions that model delayed effects. In these cases, there are no agents that execute the action and they do therefore not belong to any agent.*
in the previous chapter (see section 5.3). Constructing TMOs from the sets of temporal actions is thereafter done with the algorithms for creating STMOs and PTMOs (see section 5.2).

Filter Layer

The role of the filter layer is to select the most interesting TMOs and to remove all other. This process is done in four steps. Firstly, each TMO is instantiated with the constants that come from the temporal actions the TMO was created from. Secondly, all TMOs are evaluated using up to two evaluators (depending on how the planner is configured) that evaluates how good a TMO is. The first evaluator is the applicability and goal evaluator for TMOs and the second is the mutex evaluator (see section 5.3). Thirdly, the values of the used filters is combined with a summation evaluator that sums the values. Finally, the three STMOs and the PTMO with the lowest value are selected from each generator.

Temporal Macro Operator Injector

The TMO injector injects the TMOs in the planning problem and modifies all other temporal operators to respect the mutex-locks. Injecting the TMOs in the planning problem is done by simply adding them to it. However, injecting mutex-locks in the relevant temporal operators is trickier. Naturally, injecting mutex-locks is only needed if STMOs are used since the PTMOs does not use mutex-locks.

A mutex-lock is used to protect a set of predicates and resources (remember that the number of mutex-locks is reduced by only using one mutex-lock per STMO to protect the literals and resources that needs protection in the STMO). This means that a temporal operator o requires a mutex-lock m to be free during its execution if its parameters are bound to any combination of constants that results in that at least one of the predicates or resources that m protects is used in o. Hence, each potential variable binding of a temporal operator can be mapped to a set of mutex-locks that are required to be free for the duration of the temporal operator. Unfortunately, the number of possible variable bindings is combinatorial. Hence, listing all of them would result in a planning problem that takes significantly more space to express. A planning problem that takes more space to express takes longer time for the planner to parse.

Fortunately, many variable bindings require the same set of mutex-locks which means that they can be grouped together to reduce the required space to express the planning problem.

Pseudocode for the algorithm that was implemented to inject mutex-locks in a temporal operator is shown in algorithm 6.3. In the algorithm, the following procedures are used:

- GET_INTERESTING_BINDINGS(o, M): This function gets a mapping from partial variable bindings (0 or more but not all the variables are bound to constants) to a set of mutex-locks that must be respected. This is done by going through all the symbols (literals and resources) that are protected by any of the mutex-locks in M. If a symbol protected by mutex-lock m exists in operator o when o has a partial variable binding, then m must be free when the partial variable binding is used to instantiate o. To keep this in memory, a mapping from the partial variable binding to a set of mutex-locks the

---

9 In the implementation, this is done by never lifting the variables in the temporal actions that the TMOs were created from.
10 In the implementation, this is done by never lifting the variables in the temporal actions that the TMOs were created from.
11 In the implementation, this is done by never lifting the variables in the temporal actions that the TMOs were created from.
12 In the implementation, this is done by never lifting the variables in the temporal actions that the TMOs were created from.
operator must respect when it is instantiated with the partial variable binding. After all symbols in all mutex-locks have been checked, the algorithm has a mapping from partial variable bindings to set of mutex-locks.

Next, the algorithm expands the partial variable bindings to complete variable bindings (variable bindings where all variables are bound to constants) is done by finding all complete variable bindings that are interesting and then map them to a set of mutex-locks. This is done by checking all the partial variable bindings to find the constants that are of interest for the variables in o (i.e. if ?uav is bound to uav1 in any of the partial variable bindings, then uav1 is a interesting constant for variable ?uav). In addition, for each variable, a wild card (a constant) that represents all constants that is not bound to the variable in any of the partial variable bindings is also considered interesting. A set of complete variable bindings are thereafter computed as all combinations of the interesting bindings for all variables.

Each of the complete variable bindings are then mapped to a set of mutex-locks that applies to the complete variable binding. This is easily done for each complete variable binding by iterating over all the partial variable bindings and extend the set of mutex-locks that the complete variable binding maps to with the set of mutex-locks the partial variable binding maps to if the partial variable binding is a subset of the complete variable binding. The result is a mapping from complete variable bindings to a set of mutex-locks.

- **GROUP_ON_MUTEX_LOCKS(binding_to_mutex_locks)**: This function takes a mapping from a complete set of variable bindings to mutex-locks and reverse it. Essentially, a new map is created using the values in the bindings_to_mutex_locks map as keys and sets of complete variable bindings as values. Then for each key value pair in bindings_to_mutex_locks, the key is added to the set of complete variable bindings that the value maps to in the new mapping. The resulting map is then returned.

- **REDUCE(mutex_locks_to_binding)**: The reduce procedure performs a simple grouping on the values (sets of complete variable bindings) in the mutex_locks_to_bindings map. For each complete variable binding in a value of the mutex_locks_to_bindings map, the algorithm tries to merge the complete variable binding with all other complete variable bindings in the value of the mutex_locks_to_bindings map. This is called a pass and the algorithm continuously repeats doing passes on a value in the mutex_locks_to_bindings map until it does a pass that does not merge two variable bindings. The algorithm finishes when it has reduced all the values in mutex_locks_to_bindings.

A merging occurs between two complete variable bindings if they only differ on one variable. For example, complete variable bindings (?x = 1, ?y = 2, ?z = 3) and (?x = 1, ?y = 2, ?z = 2) only differs in variable ?z and they are therefore combined to form a variable binding (?x = 1, ?y = 2, ?z ∈ {2, 3}). Naturally, result is no longer a complete variable binding but a representation of a set of complete variable bindings. However, these are merged in the same way as for the complete variable bindings. E.g. (?x = (1, 2), ?y = 2, ?z ∈ {2, 3}) is merged with (?x = 3, ?y = 2, ?z ∈ {2, 3}) to form (?x = {1, 2, 3}, ?y = 2, ?z ∈ {2, 3}).

The result of this procedure is not necessarily the best grouping but it provided good results in terms of reduction capabilities. More importantly, it reduced the problem size so that TFTD was able to parse the problem in a reasonable time.

- **TO_LOGIC_FORMULA(binding, bindings_to_mutex_locks)**: This function takes a binding that is a set of variable bindings on the form ?x ∈ D and returns a logic formula that expresses this. This is done by transforming each ?x ∈ D to a condition (?x = x1 ∨ ?x = x2 ∨ ⋅ ⋅ ⋅ ∨ ?x = xn) where the xi are the values in D and join these clauses with logical and
operators. Moreover, if \( D \) contains a wild card then the wild card equality is expanded to \( (?x ≠ x_1 ∧ ?x ≠ x_2 ∧ \cdots ∧ ?x ≠ x_n) \) where \( x_1, x_2, \ldots, x_n \) are all the constants that are bound to \(?x\) in any of the partial variable that are the keys in \( \text{bindings\_to\_mutex\_locks} \).

In a few words, the algorithm (6.3) that injects mutex-locks into an operator does so by:

1. Finding a mapping from all the interesting partial variable bindings to a set of mutex-locks (line 2).
2. Reverse to order of the map and expand the partial variable bindings so that each set of mutex-locks maps to a set of complete variable bindings (line 3).
3. Reduce the size of the complete variable bindings by merging complete variable bindings (line 4).
4. For each set of mutex-locks: Create a copy of the operator (line 7); add conditions that requires that the variables are bound to any of the complete variable bindings that the set of mutex-lock maps to as a precondition to the copy (line 8-12); and add conditions that all the mutex-locks are false as preconditions and an overall conditions to the copy (line 13-16).
5. Return all the copies of the operator (line 19). These operators, unlike the input, now respects the mutex-locks and are therefore used in the plan repair problem.

**Algorithm 6.3: INJECT_MUTEX_LOCKS**

```plaintext
input : Temporal operator o, Mutex-locks M
output : Temporal operators with injected mutex-locks
begin
  bindings_to_mutex_locks ← GET_INTERESTING_BINDINGS(o, M)
  mutex_locks_to_bindings ← GROUP_ON_MUTEX_LOCKS(bindings_to_mutex_lock)
  REDUCE(mutex_lock_to_binding)
  O ← SET()
  for <mutex_locks, bindings> ∈ mutex_locks_to_bindings do
    o_copy ← COPY(o)
    cond ← LOGIC_EXPR(false)
    for binding ∈ bindings do
      cond ← LOGIC_EXPR(TO_LOGIC_FORMULA(binding, bindings_to_mutex_locks) ∨ cond)
    end
    APPEND(o_copy:condition_pre, cond)
    for mutex_lock ∈ mutex_locks do
      APPEND(o_copy:condition_pre, ¬mutex_lock)
      APPEND(o_copy:condition_over_alt, ¬mutex_lock)
    end
    O ← APPEND(O, o_copy)
  end
  return O
end
```

6.3 Simulator

A scenario in this thesis specifies a situation that has to be solved. Naturally, it is costly to set up the scenario and run everything on real platforms when something has to be tested
or evaluated. Therefore, a simulator was implemented for testing and evaluating the system. The simulator simulates the plan executor and checks for events that trigger replanning (e.g. finding a new SAM unit). In addition, the simulator is responsible for updating the current knowledge of the world as the plan is executed and events are triggered.

The following sections cover the four main parts of the simulator: Executing a plan that was generated to solve the scenario; all the events that can happen during the simulation; visualisation of the scenario; and measurements of how long time the replanning phases take.

Plan Execution

The execution of a plan is done by stepping through the plan using discrete time steps (each step is 1 second simulated time) until an event is encountered or the whole plan is executed. One step is done by applying all effects that are triggered during the step and update the positions of all UAVs that are moving during the step. Between two steps, the simulator checks if any event has been triggered. If that happens, a replanning phase is triggered.

A scenario is finished when a plan has been executed without triggering any events or the system failed to find a plan. If the plan was executed without triggering any events, then the system solved the scenario. This is because the plan is guaranteed to solve the scenario unless something externally changes, i.e. an event occurs. Hence, the scenario is solved if the whole plan is executed. However, if the system failed to find a plan, then it also failed to solve the scenario.

Events

There are multiple types of events that can trigger a replanning phase. These types of events can be divided into three categories, which all are handled in the same way in this thesis. The categories are: Threatening events, events that render the current plan invalid; opportunistic events, events that might make it possible find a better plan; and peripheral events, events that neither affect the current plan nor gives opportunities to improve the current plan.

The threatening events that the simulator finds are:

- A new SAM is found and at least one agent is planned to fly within the SAM’s weapon range (risk policy states that flying within SAM radar range is acceptable) or radar range when having an altitude of 1.5km (risk policy states that no UAV may be scanned by a SAM radar when flying at an altitude of 1.5km).
- A new target has been encountered.
- A sensor that will be used in the plan starts to malfunction.

The opportunistic events are:

- The last target is scanned. This is an opportunistic event since there is a possibility that the UAV that is scanning targets and following the SAR scanning can remove some of its waypoints because the SAR has passed them long ago.
- The SAR scanning is completed. There is no need for any UAV to follow the one doing the SAR scanning any longer.

Finally, the peripheral events are:

---

\[13\] It is most likely that handling the three categories differently would achieve a better result. Moreover, more categories could be created to handle the different situations even better. For example, an extra category could be created for path violations. The plan could then be repaired by finding new paths, if possible, instead of a new task plan with new paths when an event of that type happens.
6.3. Simulator

- A new SAM is found and it does not threaten the execution of the current plan.
- A sensor that is not used in the current plan starts to malfunction.

Visualisation

Visualisation of a simulation is done by generating snapshots of the scenario. Each snapshot shows the current knowledge of the world and the current plan. All this is shown through an overview of the current scenario and two images for each UAV that shows the UAV’s part of the current plan on the different altitudes. A few of the snapshots from the visualisation are shown and explained in appendix D.

The purpose of the visualisation is not to provide the best, or even remotely good, visualisation of the plan. Instead, the purpose of the visualisation is to visualise the paths that was found by the motion planner to help find bugs about the paths that was found.

Measurements

An important part of the simulator is the ability to measure the time the plan repair takes. This is an essential part for measuring the performance of different planner configurations. For the purpose of measurements, the plan repair is divided into three phases: a preprocessing phase (currently, this only consists of the problem generator reading the SAS+ variables, and their domains, that was used in the previous planning problem from a file created by TFD); a TMO generation phase; and a planning phase. This division is important because the preprocessing phase can run during the execution of the old plan. Hence, this part only affects the time it takes to repair the plan if the preprocessing time exceeds the time that the old plan is executed. This means that the simulator measures the plan repair time as follows:

\[
 t_{\text{repair}} = t_{\text{tmo}} + t_{\text{planning}} + \max(0, t_{\text{preprocess}} - t_{\text{execution}})
\]  

(6.1)

Where:

- \( t_{\text{repair}} \) is the time it takes to repair the plan.
- \( t_{\text{preprocess}} \) is the time the repair algorithm is in the preprocessing phase (in the problem generation part of the system).
- \( t_{\text{tmo}} \) is the time the repair algorithm spends on generating TMOs and injecting them in the planning problem (based on the current knowledge).
- \( t_{\text{planning}} \) is the time the system is in the planner part. I.e. the time it takes to find a new plan for the planning problem that is extended with the TMOs.
- \( t_{\text{execution}} \) is how long the old plan was executed before a repair was triggered.
Research questions RQ3 through RQ6 concern the performance of the system. The research questions are on the form “can X achieve Y?”. Therefore, they can be answered by providing a single case where X achieves Y. However, a structured evaluation can show that the effect is more than an isolated case. Conducting a structured evaluation requires a set of problems that the system can be evaluated on and a method for analysing the data.

Earlier studies have evaluated task plan repair algorithms on randomised problems based on de facto benchmark task planning problems \[2, 31\]. Unfortunately, these benchmarks cannot be used in this thesis because the system is built to solve plan repair problems for combined task and motion plans and not for task plans. Moreover, to the author’s knowledge there are no other studies investigating the combined task and motion plan repair problem. The lack of benchmarks and other studies to compare with makes it harder to do a comparative study, though not impossible. One can use a set of problems based on the typical scenarios in this thesis and a baseline configuration of the system to make a comparison. Naturally, this means that all conclusions from the evaluation are restricted to the typical scenarios.

The rest of the chapter covers how the evaluation of the system was done, starting with an overview of the evaluation (section 7.1). Following that is the generation of the problems that the system was evaluated on (section 7.2). After that is section 7.3 about the different configurations of the system that were evaluated. Next is a description of the populations and how they were sampled (section 7.4). Following that is section 7.5 covering how the raw data was refined to the values that were used for the evaluation. The two final sections in the chapter covers how the data were analysed (7.6) and the analyses that were done (7.7) to answer research questions RQ3 through RQ6.

### 7.1 Evaluation Overview

The method that was used to evaluate the system can be divided into 6 steps:

- Generating problems: This consisted of generating plan repair problems that the system could be evaluated on.
- Defining treatments and baselines: The treatments in this study were different configurations of the system. Similarly, the baselines (there were different baselines for the different research questions) were also configurations.
• Defining analysis methods: This consisted of defining which analysis methods that were to be used to analyse the data.

• Sampling data: Data was sampled using the generated problems, the treatments and the baselines.

• Computing variables: The raw data in itself was not directly used in the evaluation. Instead, variables that were calculated from the raw data was used.

• Analyse the data: The analyses that were used to answer the research questions.

This overview shows the steps that was done when conducting the analysis and it can be beneficial to keep them in mind when reading the rest of the chapter. It should be noted that the analysis methods are described in the second last section of this chapter to create a better flow. However, the analysis methods were defined before the data were inspected.

7.2 Generating Problems

Plan repair problems have earlier been generated by randomisation using the de facto benchmark task planning problems [2, 31]. Naturally, using randomisation has the benefit that one can draw more general conclusions as long as the gathered data is unbiased. However, the possible conclusions in this study were already restricted by the population (the reconnaissance mission that the thesis originated from), which means that not as much could be gained from randomising problems. Hence, the simulator that was implemented and the typical scenarios that were defined, see appendix A, were used to generate repair problems. The generation was done by running the simulator for each scenario using a baseline configuration (no macro extension and the path estimation extension of heur was used) and no time limit. Each event that was triggered using the simulation resulted in one repair problem. The problem generation method can be summarised with the following steps:

1. For each of the five typical scenarios do the following:
   a) Formulate a plan repair problem based on the scenario (note that the plan repair problem is a standard planning problem with an empty old plan).
   b) Save the plan repair problem.
   c) Find a plan using the baseline configuration.
   d) Simulate the scenario using the plan until an event is triggered.
   e) Do the following while the end of the scenario has not been reached:
      i. Formulate a plan repair problem based on the current state of the scenario.
      ii. Save the plan repair problem.
      iii. Find a solution using the baseline configuration.
      iv. Continue the simulation of the scenario using the found plan until an event is triggered or the end of the scenario is reached.

2. Return the saved plane repair problems as the result of the problem generation.

The result from the problem generation was a set of 66 randomised repair problems based on the typical scenarios. It should be noted that the randomisation comes from the randomised parts in the motion planner that is integrated in the system. Hence, the resulting repair problems were a biased set of all the possible repair problems from the typical scenarios. As a result, the conclusions were restricted from being about the typical scenarios to the generated problems. However, using this method for generating planning problems had the benefit over randomised mutation of a state and the goal (the method used in earlier studies [2, 31]) that the repair problems are more realistic for the typical scenarios because they originate from an actual (simulated) execution.
### 7.3 System Configurations

The system is designed so that it can use different configurations when solving a repair problem. The different configurations come from combining the following options:

- Strategies for how to generate TMOs: no macro operators; STMO; PTMO; and STMO and PTMO.
- Filters for how to select which TMOs to use: the applicability and goal evaluator; the mutex-lock evaluator; and the sum of the applicability and goal evaluator and the mutex-lock evaluator.
- The input to the TMO generators (the prefix and suffix generators) can be: the old temporal plan (the agent splitter component, see figure 6.4, is not used); or the old temporal plan and the old temporal plan split on agents (the agent splitter component, see figure 6.4, is used).

Fortunately, not all combinations of these three options are valid configurations. In fact, only 13 of them are valid configurations. In addition, there is a configuration, for evaluating the path extension to \( h_{cea} \), that is using blind estimation for extending \( h_{cea} \) and no TMOs.

All 14 configurations are presented in table 7.1.

<table>
<thead>
<tr>
<th>Configuration id</th>
<th>TMO generation</th>
<th>TMO filter</th>
<th>Agent splitter</th>
<th>( h_{cea} ) extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>None</td>
<td>No</td>
<td>path estimation</td>
</tr>
<tr>
<td>1</td>
<td>STMO</td>
<td>Mutex</td>
<td>No</td>
<td>path estimation</td>
</tr>
<tr>
<td>2</td>
<td>STMO</td>
<td>Mutex</td>
<td>Yes</td>
<td>path estimation</td>
</tr>
<tr>
<td>3</td>
<td>STMO</td>
<td>A&amp;G</td>
<td>No</td>
<td>path estimation</td>
</tr>
<tr>
<td>4</td>
<td>STMO</td>
<td>A&amp;G</td>
<td>Yes</td>
<td>path estimation</td>
</tr>
<tr>
<td>5</td>
<td>STMO</td>
<td>A&amp;G+Mutex</td>
<td>No</td>
<td>path estimation</td>
</tr>
<tr>
<td>6</td>
<td>STMO</td>
<td>A&amp;G+Mutex</td>
<td>Yes</td>
<td>path estimation</td>
</tr>
<tr>
<td>7</td>
<td>PTMO</td>
<td>A&amp;G</td>
<td>No</td>
<td>path estimation</td>
</tr>
<tr>
<td>8</td>
<td>PTMO</td>
<td>A&amp;G</td>
<td>Yes</td>
<td>path estimation</td>
</tr>
<tr>
<td>9</td>
<td>STMO &amp; PTMO</td>
<td>A&amp;G</td>
<td>No</td>
<td>path estimation</td>
</tr>
<tr>
<td>10</td>
<td>STMO &amp; PTMO</td>
<td>A&amp;G</td>
<td>Yes</td>
<td>path estimation</td>
</tr>
<tr>
<td>11</td>
<td>STMO &amp; PTMO</td>
<td>A&amp;G+Mutex</td>
<td>No</td>
<td>path estimation</td>
</tr>
<tr>
<td>12</td>
<td>STMO &amp; PTMO</td>
<td>A&amp;G+Mutex</td>
<td>Yes</td>
<td>path estimation</td>
</tr>
<tr>
<td>13</td>
<td>None</td>
<td>None</td>
<td>No</td>
<td>blind estimation</td>
</tr>
</tbody>
</table>

### 7.4 Population and Sampling

Combining each generated problem with all configurations gave the populations that had to be sampled. In total, this resulted in 924 populations for which the underlying distribution for the samples are unknown. However, if the sample size is larger or equal to 30, then the central limit theorem (CLT) says that the mean can be approximated with a normal distribution \[ 35, 36 \]. The CLT also requires that the data points are independent (i.e. the probability function for a data point does not change depending on the earlier drawn data points). This holds for this evaluation because all random factors (mostly the random component in the motion planning but also the unavoidable noise introduced by the operative system on the computer) have infinite populations with uniform distributions (this ensures that the probability mass function remains the same regardless of the data points that has been drawn). Unfortunately,
the worst case estimation for the sampling time when using the sample size that \text{CLT} requires and a time limit of 300 seconds, the same as in the \text{IPC} \cite{28,29}, is slightly more than 96 days. Therefore, it was concluded that it was impossible to gather 30 data points for each for the 924 samples within the time limits of this thesis. Hence, some reduction to the population or the number of gathered data points had to be done.

One of the drawbacks with using a time limit provides an opportunity to do the reduction. The drawback is that it is not possible to estimate the mean and standard deviation for a sample using the \text{CLT} without introducing a bias if a data point in the sample exceeds the time limit \cite{35}. However, this means that it does not matter if one or 30 data points of a sample are gathered if at least one of them exceeds the time limit when requiring that the sample is unbiased. Hence, there was no need to sample more data points for a sample after a data point that exceeded the time limit was sampled. Doing this made the sampling time manageable.

7.5 Values for Analysis

There are four values that are used in the analyses in the evaluation. Each of them are explained in this section. Moreover, an example will be shown for how the values are calculated for a sample using two samples \(X_0 = \{150, 200, 175\}\) and \(X_1 = \{250, \infty\}\) consisting of up to three data points. Through the rest of the report, \(\infty\) is used in a sample to indicate that the data point exceeded the time limit.

The two first values that were calculated were the mean and the standard deviation of a sample. They are calculated as usual (see equation 7.1 and 7.2 in which \(X\) is the sample and \(n\) is the sample size for how to calculate the mean and the standard deviation, respectively). However, they were only calculated for the samples that did not contain any data points that exceeded the time limit.

\[
\mu = E(X) = \frac{\sum x}{n} \quad (7.1)
\]
\[
\sigma = \sqrt{V(X)} = \sqrt{\frac{\sum (x - E(X))^2}{n - 1}} \quad (7.2)
\]

Calculating the mean and the standard deviation for the example sample \(X_0\) is done as follows (they are not calculated for \(X_1\) since \(\infty \in X_1\)):

\[
\mu_{X_0} = \frac{150 + 200 + 175}{3} = 175
\]
\[
\sigma_{X_0} = \sqrt{\frac{-25^2 + 25^2 + 0^2}{2}} = 25
\]

One value that had to be adapted is the binary solved-or-not-solved value that the coverage metric uses (i.e. it is 1 if the planner solved the problem within the time limit and otherwise 0). Of course, this value can be modified in many ways to define it for a sample rather than a single execution. Two ways was considered: A threshold (e.g. if the planner solves a problem 90% of the time, then the solved-or-not-solved value is 1 and otherwise 0); and a percentage (i.e. the solved-or-not-solved variable is the number of times the planner found a solution within the time limit divided by the total sample size). Unfortunately, the latter requires that the whole sample is gathered. As stated in the previous section, this was not possible to do within the time frame of the thesis. Hence, the solved-or-not-solved value had to be adapted.

\footnote{All program executions executed as a single thread process on a MacBook Air with 1.6 GHz Intel Core i5 and 8 GB 1600 MHz DDR3 memory.}
to work on samples that are not fully sampled if one data point exceeds the time limit. This
was done by using a threshold of 100% as follows (for a sample $X$):

$$\text{solved}(X) = \begin{cases} 0 & \text{if } \infty \in X \\ 1 & \text{else} \end{cases}$$  \hspace{1cm} (7.3)

Calculating the solved value for samples $X_0$ and $X_1$ is done as follows:

\begin{align*}
\text{solved}(X_0) &= 1 \\
\text{solved}(X_1) &= 0
\end{align*}

Similar to the coverage metric, the search time score (see section 3.4) is defined in terms of
how well a planner handles a problem. Naturally, there is no problem to calculate this when
the planner is deterministic. However, complications arise when there are components using
randomisation in the planning procedure because this can, and does in this case, cause the
search time to be non-deterministic. Hence, the search time score was modified to take this
into account as follows (for sample $X$):

$$\text{sts}(X) = \begin{cases} 0 & \text{if } \infty \in X \lor E(X) > 300 \\ 1 & \text{else if } E(X) \leq 1 \\ 1 - \frac{\log(E(X))}{\log(300)} & \text{else} \end{cases}$$  \hspace{1cm} (7.4)

Calculating the search time score for samples $X_0$ and $X_1$ is done as follows:

\begin{align*}
\text{sts}(X_0) &= 1 - \frac{\log(175)}{\log(300)} \approx 0.094 \\
\text{sts}(X_1) &= 0
\end{align*}

7.6 Analysis Methods

The structured evaluation used five analysis methods to analyse the data. Two of the analysis
methods, coverage and search time score, are standard metrics for task planning and sometimes
for task plan repair \[2, 9, 28, 29\]. The other three are not, to the author’s knowledge, as
common in the field. One of them is a statistical comparison of how fast two configurations
solves the problem. The two last are a ranking of the standard deviation and a comparison
of the standard deviation between two configurations. All five analyses are covered in further
details in the following sections.

In general, the results from the analyses can only give indications for the scenarios that
was used to generate the problems. This is because the problems were generated from the
scenarios with a biased method to increase the realism. Hence, the conclusions that can be
drawn are restricted to be about the problems. However, the results can be seen as indications
for the scenario and the most general case (all task and motion plan (repair) problems).

Coverage

The coverage value is a simple measurement of which configuration that manages to solve
most problems. The original version of it simply counts the number of problems solved within
the time limit. However, this is not reasonable when some samples may miss data points.
Instead, the coverage was calculated by summing up the solved values (see equation 7.3) for
each configuration.

Interpreting the results was done by comparing the coverage score for the configurations
with each other. A higher coverage score for one configuration than for another indicated a
higher likelihood that the first configuration will solve a problem within the time limit.
7.6. Analysis Methods

Search Time Score

The search time score is a measurement of which configuration that is the fastest on solving the problems. Just like the coverage score, this measurement was adopted to handle samples instead of data points. This was done by summing up the $sts$ values (see equation 7.4) for each sample related to a configuration to calculate the search time score for that configuration. It should be noted that the handling of time outs in the $sts$ introduced a bias towards configurations that have a lower standard deviation. This is because a configuration with mean $\mu_i$ and standard deviation $\sigma_i$ for a problem is more likely to yield a sample without data points above the time limit than a configuration with mean $\mu_j$ and standard deviation $\sigma_j$ for the same problem when $\mu_i = \mu_j \leq 300$ and $\sigma_i < \sigma_j$.

Interpreting the results was done by comparing the search time score for the configurations with each other. A higher search time score for one configuration than for another configuration is an indication that the first configuration will solve the problems faster.

Mean Analysis

The two previous analysis methods are based on descriptive statistics of the mean search time for each configuration and problem. Drawing a conclusion based on them would be risky due to the fact that the search time for solving a problem using different configurations have different distributions that overlaps. Instead, one can use the normal distributions and draw a conclusion ($H_1$) by rejecting the null hypothesis ($H_0$) if the results are significant ($p < .05$ is normally treated as significant [36]). This procedure considers the fact that the search time for a configuration and a problem has a distribution.

Comparing all configurations for all problems will increase the risk of doing a type I error (i.e. rejecting $H_0$ when $H_0$ is true [35]). The huge amount of comparisons comes from that there are many configurations and many problems in the evaluation. Fortunately, the number of configurations to test can be reduced by only testing the configurations that performed best according to the coverage or the search time score metrics against the baseline. This gives at most two tests to perform per problem. Unfortunately, there are still a lot of tests and therefore also a high risk of performing a type I error.

Reducing the number of comparisons even further can be done by comparing the sum of the search times for two configurations instead of comparing each pair of mean search time for the two configurations for each problem. This can be done summing the normal distributions that are the mean search times for each configuration. Doing this means that only one comparison is done per pair of configurations that are compared. Hence, only up to two comparison is done per hypothesis to test.

Summing a set of normal distributions is a special case of linear combination of normal distributions where the weights are 1. The linear combination of normal distributions results in a new normal distribution [37] and is calculated as shown in equation 7.5 where $cov(X_i, X_j)$ is the covariance for samples $X_i$ and $X_j$:

$$a_0 * N(\mu_0, \sigma_0) + a_1 * N(\mu_1, \sigma_1) + \cdots + a_m * N(\mu_m, \sigma_m) = N\left(\sum_{i=0}^{m} a_i \mu_i, \sqrt{\sum_{i=0}^{m} a_i \sigma_i^2 + \sum_{i=0}^{m} \sum_{j=0}^{m} a_i a_j \text{cov}(X_i, X_j)}\right)$$  \hspace{1cm} (7.5)

As stated above, the new normal distribution provides information about the sum of the search time for all problems used in to calculated the summed distribution and not the individual problem. However, this is enough to state that one configuration solves the set of problems faster than another configuration, given that the comparison shows it.

---

2Two normal distributions [31] allows for approximating the mean value of an unknown distribution as a normal distribution for large sample sizes) do always have an overlap [35].
Testing if one random value from one normal distribution \(N(\mu_a, \sigma_a)\) is lower than a random value from another normal distribution \(N(\mu_b, \sigma_b)\) can be done by rejecting the null hypothesis that it is larger or equal to the other (i.e. doing a one-tailed comparison). If \(H_0\) is rejected, then \(H_a\) is true (i.e. it is lower). Rejecting \(H_0\) can be done by calculating the probability that a random value from \(N(\mu_a, \sigma_a) - N(\mu_b, \sigma_b)\) is larger or equal to 0. This is done by firstly calculating the difference (a special case of linear combinations of normal distributions) as shown in equation (7.6)\(^3\) where \(\text{cov}(X_i, X_j)\) is the covariance for samples \(X_i\) and \(X_j\):

\[
\text{cov}(X_i, X_j) = \frac{\sum_{i=1}^{n} (X_i - E(X))(Y_i - E(Y))}{n-1} 
\]

Thereafter, the probability \(p\) that the random value is larger or equal to 0 can be read from a table for the standard normal distribution \(N\) using the \(z\)-value calculated with equation (7.7) (simply replace \(\mu\) with \((\mu_a - \mu_b)\) and \(\sigma\) with \(\sqrt{\sigma_a^2 + \sigma_b^2 - 2 \times \text{cov}(X_a, X_b)}\)). If the resulting probability is significant (i.e. \(p \leq .05\)), then \(H_0\) can be rejected and \(H_a\) accepted.

\[
z = \frac{0 - (\mu_a - \mu_b)}{\sqrt{\sigma_a^2 + \sigma_b^2 - \text{cov}(X_a, X_b)}} 
\]

\[
p = P(z \geq 0)
\]

The last part that is needed to calculate the \(z\)-value is a formula for calculating the covariance. There are three different ways of calculating \(\text{cov}(X, Y)\) that are relevant for this thesis:

1. When \(X\) and \(Y\) are independent from each other \(\text{cov}(X, Y) = 0\) (7.8)

2. When \(X\) and \(Y\) is the same distribution \(\text{cov}(X, Y) = V(X) = V(Y)\) (7.9)

3. In general the covariance of distributions \(X\) and \(Y\) can be estimated from samples (of size \(n\)) \(\text{cov}(X, Y) = \frac{\sum_{i=1}^{n} (X_i - E(X))(Y_i - E(Y))}{n-1}\) (7.10)

Finally, not all problems can be used in a comparison because the two configurations in the comparison may not have solved all problems within the time limit. As a result, there is not enough data to construct a normal distribution for all problems for the two configurations. Hence, only the problems that both configurations manages to solve within the time limit are used. Naturally, this means that any conclusions regarding two configurations are limited to the problems that both configurations manage to solve.

Based on all the above, calculating the \(p\)-value for two configurations, \(x\) and \(y\), was done by first calculating the set of problems that both configurations managed to solve and the size of the set as follows (\(\Psi\) is the set of all problems, \(X_i\) is the sample for solving problem \(i\) with configuration \(x\) and \(Y_i\) is the sample for solving problem \(i\) with configuration \(y\)):

\(^3\)It might be counter intuitive that the \(z\)-value is higher than 0 when \(\mu_a\) is lower than \(\mu_b\) (though the difference might be insignificant) but it is correct. When testing the hypotheses in this thesis, the probability that \(H_0\) does not hold increases as the \(z\)-value increases. More importantly, when the \(z\)-value is large enough, then \(H_0\) can be rejected and it can be claimed that \(\mu_a\) is significantly lower than \(\mu_b\).
Thereafter, the normal distribution describing the difference was calculated (using equation 7.8):

\[
N(\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
\]

This was simplified because the mean search time when solving a problem is independent from the mean search time when solving another problem by using equations 7.8 and 7.9:

\[
N(\mu_x, \sigma_x) = N(\mu, \sigma) - N(\mu_y, \sigma_y)
\]

Next, the two resulting normal distributions was combined using the estimated covariance (equations 7.6 and 7.10) by using the mean search times for the different problems to estimate the covariance where \( X = \{X_i|i \in S\} \) and \( Y = \{Y_i|i \in S\} \):

\[
N(\mu_x, \sigma_x) = N(\mu_x, \sigma_x) - N(\mu_y, \sigma_y)
\]

\[
N(\mu_x, \sigma_x) = N\left(\mu_x - \mu_y, \sqrt{\sigma_x^2 + \sigma_y^2 - 2\sigma_x\sigma_y \text{cov}(X,Y)}\right)
\]

Where \( \bar{x} \) and \( \bar{y} \) were defined as follows:

\[
\bar{x} = \frac{\sum_{i \in S} E(X_i)}{n}
\]

\[
\bar{y} = \frac{\sum_{i \in S} E(Y_i)}{n}
\]

Based on this, the p-value was calculated as follows (using equation 7.7 and a table for normal distributions):

\[
z = \frac{0 - \left(\sum_{i \in S} E(X_i) - \sum_{i \in S} E(Y_i)\right)}{\sqrt{\sum_{i \in S} V(X_i) + \sum_{i \in S} V(Y_i) - 2 \sum_{i \in S} (E(X_i) - \bar{x})(E(Y_i) - \bar{y})/n - 1}}
\]

\[
p = P(z \geq 0)
\]

Finally, \( H_0 \) (search time for solving the problem when using \( x \) is lower than the search time when using \( y \)) was proved by rejecting \( H_0 \) if \( p \leq .05 \) using \( n \) problems.
Standard Deviation Ranking

None of the standard methods for evaluating task planning algorithms (and by extension task plan repair by replanning) considers how predictable the search time is. The standard deviation ranking descriptive analysis aims to give an indication of this. It was calculated for a configuration by summing the configuration’s rank for each problem. Where the rank for a configuration and a problem with sample $X$ is 0 if $\infty \in X$ and otherwise it is the total number of configurations in the analysis minus the number of configurations in the analysis that has a lower standard deviation for the problem. If multiple configurations have the same standard deviation, then their ranks are averaged. The ranking gives an indication of which configuration that has the most predictable search time according to the standard deviation.

Standard Deviation Analysis

The comparison of standard deviation was done by summing a normal distribution as in the mean analysis. However, standard deviation ranking was used to select which configuration that is to be compared with the baseline. The comparison of configurations $x$ and $y$ were compared by calculating a $c$-value as follows ($\Psi$ is the set of all problems, $X_i$ is the sample for solving problem $i$ with configuration $x$ and $Y_i$ is the sample for solving problem $i$ with configuration $y$):

$$c = \sum_{i \in S} (\sqrt{V(X_i)}) - \sum_{i \in S} (\sqrt{V(Y_i)})$$

Where $S$ is defined as (the same as for the mean analysis):

$$S = \{ i | i \in \Psi \land \infty \notin X_i \land \infty \notin Y_i \}$$

The $c$-value shows how $x$ and $y$ compares to each other in terms of standard deviation when using $n = |S|$ problems as follows:

$c < 0$: $x$ has a lower standard deviation than $y$.
$c = 0$: $x$ has the same standard deviation as $y$.
$c > 0$: $x$ has a higher standard deviation than $y$.

7.7 Analysis

The research questions that the analysis methods in the previous section were used to answer were divided in two categories:

- Plan repair using different strategies for creating TMOs.
- Improving the results by using geometric information in $h_{cea}$.

Both of these categories were evaluated in the same way: The coverage, search time score and the mean analysis were used to evaluate the time it takes to solve the problems; and the standard deviation ranking and the standard deviation analysis was used to evaluate the predictability as measured by standard deviation of the time it takes to solve the problems. The difference between the categories lies in which configurations that were used. Table 7.2 shows the analysis methods and configurations that were used to answer research question 3 through 6. Moreover, table 7.3 shows the null hypothesis, the alternative hypothesis and the answers that could be gained from them for RQ3 and RQ5. If the results showed something that is not covered in the table, then no conclusions could be drawn. However, insights could still be gained.

---

4The standard deviation ranking is inspired by the ranking used in the Mann-Whitney U test [36].
Table 7.2: The analyses that were done in the evaluation to answer research question 3 through 6. Each line specifies: the analysis methods that were used to analyse the data for the research question; the configuration that was used as a baseline for the analysis; and the configurations that were treatments (i.e. those that were compared with the baseline).

<table>
<thead>
<tr>
<th>Research question</th>
<th>Analysis methods</th>
<th>Baseline configuration id</th>
<th>Treatment configuration id(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ3</td>
<td>Coverage, search time score &amp; mean analysis</td>
<td>0</td>
<td>1-12</td>
</tr>
<tr>
<td>RQ4</td>
<td>Standard deviation ranking &amp; Standard deviation analysis</td>
<td>0</td>
<td>1-12</td>
</tr>
<tr>
<td>RQ5</td>
<td>Coverage, search time score &amp; mean ranking</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>RQ6</td>
<td>Standard deviation ranking &amp; Standard deviation analysis</td>
<td>13</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7.3: Hypotheses for research questions RQ3 and RQ5 and answers that could be drawn from rejecting \( H_0 \).

<table>
<thead>
<tr>
<th>Hypotheses</th>
<th>On rejecting (any of the) ( H_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>For coverage:</strong></td>
<td></td>
</tr>
<tr>
<td>( H_0 ): The configuration with the highest coverage has a higher or equal mean search time than the baseline configuration.</td>
<td></td>
</tr>
<tr>
<td>( H_a ): The configuration with the highest coverage has a lower mean search time than the baseline configuration.</td>
<td></td>
</tr>
</tbody>
</table>

| **For search time score:** |
| \( H_0 \): The configuration with the highest search time score has a higher or equal mean search time than the baseline configuration. |
| \( H_a \): The configuration with the highest search time score has a lower mean search time than the baseline configuration. |
| Yes, it can decrease the search time. |
This chapter presents the results from the structured evaluation presented in chapter 7. Included in this, are the results from the analyses. Moreover, the most relevant means and standard deviations are presented in the chapter. The rest of them are presented in appendix E.

The layout of the chapter is organised after research question RQ3 through RQ6. First is section covering the results for RQ3 and RQ4 about plan repair and after that is a section covering the results for RQ5 and RQ6 about the heuristic extensions.

8.1 Plan Repair

The mean analysis and standard deviation analysis for the search time of the plan repair problems was conducted on configuration 6 and 1, respectively. Configuration 6 was selected for the mean analysis since it had the highest coverage and search time score. Moreover, configuration 1 was selected for the standard deviation analysis since it had the highest standard deviation ranking (see table 8.1). Unfortunately, the mean analysis did not give a significant result that configuration 6 solved the problems that both configurations managed to solve ($\mu = 2610.32, \sigma = 128.28$) faster than the baseline, configuration 0, solved it ($\mu = 2374.64, \sigma = 121.49$), $z = -1.39, p = .918, n = 47$. In fact, the means indicate that the baseline was faster. Figure 8.1 shows the normal distribution calculated to compare the configurations. Similar the results from the standard deviation analysis showed that the configuration 1 has a higher standard deviation ($\sigma = 122.84$) than baseline ($\sigma = 120.90$), $c = 1.94, n = 48$, though not by much.

Figure 8.2 gives an overview of the mean search time and standard deviation for the baseline and configurations 1 and 6 for all problems that they solved. Moreover, figure 8.3 shows the mean search time and standard deviation of configurations that was used in the mean analysis for all problems that both configurations solved. Similarly, figure 8.4 shows the mean search time and standard deviation of the configurations that was used in the standard deviation analysis for all problems that both configurations solved.

---

1The raw data is not presented in the thesis due to the significant amount of space it would require. However, the interested reader is welcome to contact the author for the data.
Table 8.1: The search time score, coverage and standard deviation ranking for the configurations used for the repair problem. The best values, not counting the baseline (configuration 0), are written with bold face.

<table>
<thead>
<tr>
<th>Configuration id</th>
<th>Search time score</th>
<th>Coverage</th>
<th>Standard deviation ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20.68</td>
<td>52</td>
<td>539</td>
</tr>
<tr>
<td>1</td>
<td>19.85</td>
<td>48</td>
<td><strong>502</strong></td>
</tr>
<tr>
<td>2</td>
<td>19.53</td>
<td>45</td>
<td>458</td>
</tr>
<tr>
<td>3</td>
<td>14.38</td>
<td>32</td>
<td>282</td>
</tr>
<tr>
<td>4</td>
<td>13.19</td>
<td>23</td>
<td>208</td>
</tr>
<tr>
<td>5</td>
<td>19.49</td>
<td>48</td>
<td>477</td>
</tr>
<tr>
<td><strong>6</strong></td>
<td><strong>20.18</strong></td>
<td><strong>49</strong></td>
<td>496</td>
</tr>
<tr>
<td>7</td>
<td>7.5</td>
<td>9</td>
<td>51</td>
</tr>
<tr>
<td>8</td>
<td>7.78</td>
<td>10</td>
<td>66</td>
</tr>
<tr>
<td>9</td>
<td>7.91</td>
<td>10</td>
<td>54</td>
</tr>
<tr>
<td>10</td>
<td>8.11</td>
<td>11</td>
<td>77</td>
</tr>
<tr>
<td>11</td>
<td>8.26</td>
<td>10</td>
<td>70</td>
</tr>
<tr>
<td>12</td>
<td>8.03</td>
<td>10</td>
<td>66</td>
</tr>
</tbody>
</table>

Figure 8.1: The normal distribution for the difference between the sum of the mean search times when using configuration 6 minus the sum of the mean search times when using the baseline (configuration 0) to solve the 47 problems that both configurations managed to solve within the time limit. The shaded area is the probability that the baseline solved the problems faster or equally fast as configuration 6.
Figure 8.2: The mean search time and standard deviation for the samples that does not contain a data point that exceeded the time limit for configurations 0, 1 and 6. Configuration 0 has 52 valid samples, configuration 1 has 48 valid samples and configuration 6 has 49 valid samples.
8.1. Plan Repair

Figure 8.3: The mean search time and standard deviation for the samples that was used in the mean analysis to compare configurations 0 and 6. In total 47 samples was used in the analysis.

Figure 8.4: The mean search time and standard deviation for the samples that was used in the standard deviation analysis to compare configurations 6 and 1. In total 48 samples was used in the analysis.
8.2 Heuristic Extension

The evaluation of the heuristic extension only consisted of two configurations, 0 and 13. Naturally, these two were the ones that was compared using the mean analysis and the standard deviation analysis. The mean analysis shows that the search time for solving the problems that both configurations managed to solve when using configuration 0 ($\mu = 54.86, \sigma = 6.54$) is significantly lower than when using configuration 13 ($\mu = 159.45, \sigma = 25.97$), $z = 4.45$, $p < .001$, $n = 10$. The normal distribution calculated when comparing configuration 0 and configuration 13 is shown in figure 8.5. Moreover, the standard deviation analysis showed that the standard deviation is lower when using configuration 0 ($\sigma = 6.54$) than when using configuration 13 ($\sigma = 25.97$), $c = -19.43$, $n = 10$. Both the mean analysis and the standard deviation analysis used 10 of the 66 problems.

A summary of the descriptive analyses is shown in table 8.2 and the data for the other analyses are shown in figures 8.6 and 8.7. The table shows the search time score, coverage and standard deviation ranking. Figure 8.6 shows the means (x-axis) plotted against the standard deviations (y-axis) for the samples that does not contain a data point that exceeded the time limit. Finally, figure 8.7 shows the samples that was used in the mean analysis and the standard deviation analyses.

<table>
<thead>
<tr>
<th>Configuration id</th>
<th>Search time score</th>
<th>Coverage</th>
<th>Standard deviation ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20.68</td>
<td>52</td>
<td>103</td>
</tr>
<tr>
<td>13</td>
<td>7.04</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>
Figure 8.5: The normal distribution for the difference between the sum of the mean search times when using configuration 0 minus the sum of the mean search times when using the baseline (configuration 13) to solve the 10 problems that both configurations managed to solve within the time limit. The shaded area is the probability, $p$, that the baseline solved the problems faster or equally fast as configuration 0. Note that this is the area below the curve from 0 to $\infty$ and it is not visible since $p < .001$. 
Figure 8.6: The mean search time and standard deviation for the samples that does not contain a data point that exceeded the time limit for configurations 0 and 13. Configuration 0 has 52 valid samples and Configuration 13 has 10. There are 66 samples of each in total.

Figure 8.7: The mean search time and standard deviation for the samples that was used in the mean analysis and the standard deviation analysis of configurations 0 and 13. 10 out of the 66 problems were used.
This chapter is a discussion about the theory extensions, design, evaluation methods and the results in the thesis. All these areas are discussed with the goal to answer the research questions. In addition, there is a part covering the ethical considerations when developing plan repair and planning techniques. The chapter starts with a section covering integration of motion planning in a temporal planner that does early commitment in time (section 9.1). After that is section 9.2 covering the types of TMOs that was developed, including how they are created and the results from evaluating them. Following that, is a discussion about the geometrical extension to $h_{cea}$ in section 9.3. The next section (9.4) covers the evaluation methods that was used to evaluate the system. Finally, the ethical aspects are discussed in section 9.5.

### 9.1 Integration of Motion Planning

There is one restriction that should be mentioned about the observations in this thesis regarding integration of motion planning in temporal planners that does early commitment: The observations are not verified. Hence, it is not possible to conclude that the identified restrictions on the integration designs are sound or complete without extending the work with a formal proof or empirical studies showing it. However, the observations can be used as a ground for further studies regarding the subject. Finally, motion planning has been integrated in TFD, which means that a proof of concept has been provided for one of the integration designs.

### 9.2 Temporal Macro Operator in Plan Repair

There are two types of research questions related to TMOs. The first type is about the possibility to express them using PDDL2.1 and the second is about the effects when they are used when repairing a plan with the repair as replan strategy. The discussion in the following sections covers both types of research questions.
Construction of Temporal Macro Operators

Constructing TMOs and expressing them in PDDL2.1 was shown to be possible in the theory extension chapter (chapter 5). In the chapter, the two construction methods and types of TMOs that are presented are a proof of concept. However, it should be noted that they are not the only possible way of constructing TMOs expressible in PDDL2.1. More importantly, they are not proved to be the best way to construct TMOs expressible in PDDL2.1. However, this does not affect the answer to research question RQ2.

Evaluation of Temporal Macro Operators

There are three main areas to discuss regarding the results from the evaluation of the TMOs: the performance of the STMO; the performance of the PTMO; and the methods that was used to select which TMOs to use. Overall, one can divide the configurations (all configurations are shown in table 7.1) that was evaluated into three groups based on their performance as measured by the coverage and the search time scores (as shown in table 8.1):

- **Best:** The configurations that only used STMOs and mutex evaluator when selecting which TMOs to use performed the best. These are the configurations 1, 2, 5 and 6 in tables 7.1 and 8.1. Moreover, configuration 0 that does not use any TMOs belongs to this category.

- **Middle:** The configurations that only used STMOs and did not use mutex evaluator when selecting which TMOs to use performed somewhere in the middle. These are the configurations 3 and 4 in tables 7.1 and 8.1.

- **Worst:** All configurations that used PTMOs performed the worst. These are the configurations 7, 8, 9, 10, 11 and 12 in tables 7.1 and 8.1.

It should be noted that none of the configurations that used any types of TMOs performed better than the baseline that did not use any TMOs. This was not an expected result because repair as replan with macro operators has earlier been used to successfully decrease the repair time for plan repair problems in a non-temporal context.

Before continuing with a detailed discussion about the different types of TMOs and the results regarding them, one common source of errors should be mentioned: Implementation details. All software can suffer from bad performance due to implementation details. Hence, it might be the case that implementation details in the software developed for the thesis have affected the results.

Sequential Temporal Macro Operators

The performance of the STMOs was in some cases close to that of the baseline. This means that the gain from using the STMOs was close to cost of introducing them. To provide a better insight into why this happened, one can divide the cost into two parts: Increased branching factor; and the overhead of introducing mutex-locks. The gain is, as stated earlier in the thesis, that the TMOs uses shortcuts in the search space.

Increasing the size of STMO, i.e. increase its number of component operators, does potentially increase the information gain of introducing the STMO. Unfortunately, increasing the size is also likely to increase the number of literals and resources that the mutex-lock protects. Hence, the cost of introducing the STMO also increases because the overhead of the mutex-lock increases. Fortunately, increasing the size of the STMO does not necessarily increase the branching factor since the new component operator might add more preconditions. This can causes the STMO to be applicable in fewer nodes and thereby yield a lower increase of the branching factor.
The arguments above and the fact that using the mutex-evaluator resulted in lower repair time compared to when not using it indicates that the mutex-locks are responsible for a significant part of the cost of introducing STMOs. A question that arise from this argument is if it is possible to weight the evaluators so that STMOs with a higher gain but with an equal or lower overhead due to the mutex-locks are selected. If this is possible, then using the resulting STMOs might give a better score than the baseline.

One other possibility that can improve the performance of the STMO is to change how mutex-locks are generated for STMOs. It is at least NP-hard problem to calculate an over all condition for a resource and the general case. However, it is possible to create over all conditions for a subset of the cases (e.g. linear conditions can be proved in polynomial time). Hence, it might be possible to increase the performance by removing some resources from those that are protected by a mutex-lock and introducing an over all condition for the resource that protects it instead.

PTMO

The performance of the configurations that uses PTMOs were even worse than those that uses STMOs. This is an unfortunate result, though not completely unexpected. The problem with a PTMO is that it tries to force the planner to add all component operator at the correct time points, relative to when the PTMO was added. Naturally, this means that all the component operators are separate operators that has to be checked if they can be added at each search step even if they cannot be applied in most states. Effectively increasing the cost of using PTMOs. Moreover, the extra component operators also increase the branching factor. Unfortunately, this increase scales with the size of the PTMO since larger PTMOs means PTMOs with more component operators.

The increase in the number of operators is not the only problem with introducing PTMOs. They have one more major drawback: The planner has to figure out how to apply the PTMOs. If the planner does not add the component operators at the correct time points, then it will visit nodes that always lead to a dead end. This can cause that a huge amount of nodes that lead to dead ends are visited. The planner can avoid visiting some of these nodes if it manages to identify them as nodes leading to dead ends. However, TFD did not do this, which meant that a lot of time in the search was spent on expanding nodes that lead to dead ends.

The large number of extra operators, increased branching factor and dead ends are a likely explanation to why the PTMOs performed the worst. Unfortunately, it is not possible to modify the PTMOs to completely get rid of these problems short of creating a completely new type of TMO.

Filters

The results from evaluating the TMOs showed two things for the filters that was used:

- The mutex evaluator seems to be quite useful when using STMOs. This can be seen from the fact that the configurations that used this evaluator performed better than the other configurations with STMOs according to the coverage and search time score measurements.
- The applicability and goal evaluator does not seem to add much important information when its weight is the same as the mutex evaluator. This can be seen from the fact that the configurations that only uses the mutex evaluator performs about as good as the one that uses the mutex evaluator and the applicability and goal evaluator.

---

1It should be mentioned that proving that a node is a dead end is in the worst case the same time complexity as the original problem. Hence, not all planners tries to find dead ends. Moreover, those that checks for dead ends usually does so with sound but not complete algorithms.
It should also be noted that the selectors do play an important role when using macro operators in repair. This is because they keep the branching factor down by only selecting the macro operators that are estimated to be the best. Naturally, a bad estimation of the usefulness of macro operators risk introducing TMOs that does not help the search but still cost in terms of search speed.

9.3 Heuristic extension

The data showed that using the heuristic extension (extending $h_{cea}$ with geometric information) had a positive effect on the mean search time and the predictability as measured by standard deviation (see section 8.2). Neither of these results were surprising because they show that adding high quality information that is cheap in terms of computation time is better than not adding any information. In addition, the data for the standard deviation indicates that the strategy for heuristic extension is not directly affecting the standard deviation. These two statements are elaborated on in the following sections.

Mean Search Time

The result that the mean search time is lower when using the path estimation strategy compared to when using the blind estimation strategy to solve the subset of problems that both configurations managed to solve (see figure 8.3) was expected. There are two contributing factors to this: the negligible extra time it takes to calculate the heuristic value for a node when using the path estimation strategy compared to the blind estimation strategy; and the relative number of nodes that are expanded before a goal node is encountered.

Using the path estimation strategy is unlikely to significantly increase the time it takes to calculate the heuristic value compared to when the blind estimation strategy is used. Firstly, it should be noted that the blind estimation strategy takes a constant amount of time to calculate (reading a constant value) when the heuristic function queries the duration of an action. Secondly, the path estimation strategy adds a constant amount of time when the heuristic function queries the duration of an action. This is because the strategy consists of calculating the Euclidean distance between two points in the $WS$ and divide it with the speed. Reading the variables take constant time and calculating the distance for a given $WS$ (i.e. the number of dimensions is constant) also takes constant time. Moreover, the difference in the constant time it takes for two configurations is most likely small since the only difference is a few mathematical operators. Because both strategies takes similar constant time, it is unlikely that there is a large difference in computation time.

It is likely that more nodes will be visited before a goal node is visited when using the blind estimation strategy compared to when using the path estimation strategy. In essence, the $A^*$-search that TFD does ensures that all nodes that are estimated to reach a goal cheaper than the actual cost of reaching a goal are visited before any goal node. However, one should remember that $h_{cea}$ is not an admissible heuristic. Hence, the strong underestimate that the blind estimation does likely result in that more nodes will be visited before a goal node is visited compared to when using the path estimate strategy that has an estimate that, in most cases, is closer to true cost. It should be noted that this only increases the likely-hood since the increase in the amount of nodes that are visited before a goal node depends on the difference $d−h$, where $d$ is the duration of the action with the found path and $h$ is the estimated duration, and not the estimation itself. This introduced a random factor to the amount of nodes that will be visited before a goal node because the duration of the action depends on a randomised path. However, an estimate that is closer to the real value, and is admissible, will still be more likely to yield a lower amount of visited nodes before the goal node. Hence, the path estimate strategy is more likely providing a better estimation and thereby results in that fewer nodes are visited compared to when using the blind estimation strategy.
From the reasoning in the previous paragraphs, it follows that a motion planner that finds paths that are closer to the heuristic value (disregarding of which heuristic function that is used, may it be Euclidean distance, a constant value or anything else) will decrease the number of nodes that are visited in the task planning part. This is an interesting result because it indicates that the search time of the combined planning problem can be improved by finding paths that are shorter because they actual distance is then closer to the estimated distance. This means that using a motion planner that finds paths that are shorter but takes a longer time to find the path might actually decrease the time it takes to solve the combined task and motion planning problem and result in that higher quality plans are found (because the paths are shorter). Unfortunately, no data in the thesis can prove it.

Standard Deviation

The explanation to why the standard deviation is smaller when using the path estimate strategy is a continuation of the explanation for the mean search time. Visiting fewer nodes makes it more likely that fewer paths are queried. Few queries means that fewer operations that has an execution time that depends on random factors are done. Hence, the standard deviation is likely to be lower considering that the path estimation strategy does not affect the distribution of the randomisation part of the algorithm and it is likely visiting fewer nodes.

Looking at the data presented in figures 8.6 and 8.7 in the last chapter shows that there seems to be a correlation between the mean of the search time and the standard deviation. Hence, it seems more likely that the strategy plays little role in affecting the standard deviation directly. However, it seems to be the case that the standard deviation is lower when using the path estimate strategy compared to the blind estimate strategy because the mean search time decrease. There is also indications for the correlation between search time and standard deviation from the evaluation of the TMOs (see figure 8.2, 8.3 and 8.4). Unfortunately, proving that there exists a correlation between the two requires further studies.

9.4 Evaluation Method

There are some topics that should be discussed about the evaluation method that was used to evaluate the system. These topics are the generation of problems and the analysis methods that was used. All the topics are covered in the following sections.

Problem Generation

The method for generating problems that was used have two advantages: The problems come from simulated executions; and the size of the problem set was reasonable. This means that the problem set is relevant for the scenario in this thesis and it was possible to gather the required data. However, it also means that the results are not as relevant for the general case. There are three other methods for generating problems that are more general and worth mentioning:

- Completely randomised plan repair problems is the option that, in theory, has the best possibility to draw conclusions for the general plan repair problem. However, the standard task plan problem is at least PSPACE-complete (i.e. all problems that are in P, NP and PSPACE can be modelled and solved as a planning problem). In addition, all task plan problems can be modelled as a plan repair problem (e.g. all task plan problems can be modelled using a task plan repair problem with an old plan that is empty). Therefore, all PSPACE problems can be modelled as a plan repair problem. As a result, this method gives a sample that could be used to draw a conclusion about all PSPACE
9.4. Evaluation Method

or easier problems. However, there are two drawbacks with it: Randomising problems from all P, NP and PSPACE problems is, to say the least, problematic; and the conclusion would in a sense be too general because a planner does not need to be fast on all possible P, NP and PSPACE problems (for example, a sorting algorithm that has a PSPACE complexity is not really interesting).

- To randomise plan repair problems from the scenario in this thesis has the advantage that it results in randomised problems from the relevant domain. This means that conclusions can be drawn about the scenario in this thesis and not only the typical scenarios. However, generating these problems are still problematic. There are, for example, no limit on how many targets there are in the scenario. Hence, it requires randomising integers without restrictions. This is in itself problematic since there are an infinite number of integers. Nevertheless, limiting all options in the scenario to be finite domains and randomise the plan repair problems using these restriction is still a good option. Doing this makes it possible to draw conclusions about the problems within the restrictions and it is possible to randomise the problems. The only drawback is that there is no guarantee that the problem is likely to be encountered in reality.

- The last option is to extract plan repair problems from randomised scenario using (simulated) execution traces. This is a combination of the problem generation method used in the evaluation and any of the above. Essentially, a scenario is randomised and then a simulation is executed to generate plan repair problems. This means that conclusions can be drawn about plan repair problems from simulations of the set of scenarios that the randomised scenarios were drawn from. Moreover, the simulation ensures that the repair problems are somewhat realistic.

The drawback with this generation method is that it can be time consuming. In addition, there is no guarantee that the number of plan repair problems generated from each scenario is limited. Moreover, generalising the results to repair problems from executions of the set of scenarios the randomised scenarios was drawn from requires that there are enough scenarios in the sample. For example, one could use 30, as commonly used as a rule of thumb for the CLT. This means that the evaluation would take roughly 6 times longer time than the method that was used to generate problems (5 typical scenarios was used in the evaluation). Unfortunately, this was not possible within the time frame of the thesis.

The three generation methods described above do all have benefits and drawbacks compared to the method for generating problems that was used. However, they were all rejected because realism was valued over generalisation of conclusions and because of the time frame.

Coverage

The adaptation of the coverage score should not be used to draw conclusions. This is because it is biased towards the configurations with a lower standard deviation. The motivation is the same as to why the \( \text{stdv} \)-value is biased (see section \ref{sec:stats}). Therefore, the adapted coverage score was only used to select which configurations to use for the mean analysis. Hence, using the coverage score does not introduce a risk of drawing a faulty conclusion. However, there is a risk that a possible conclusion is not drawn because the best configuration is not necessarily the configuration with the best coverage score.

An alternative adaptation of the coverage score is to use percentage. This has the benefit that it does not introduce a bias since all the data points in the samples are considered. However, it has the drawback that all data points in each sample has to be sampled. Hence, it would require longer sampling time than what is possible.

\footnote{If the task plan repair problem is at least PSPACE hard, then the combined task and motion plan repair problem is at least PSPACE hard.}
9.4. Evaluation Method

Search Time Score
The adaptation of the search time score has the same flaw as the adaptation of the coverage. That is, it is biased and should therefore not be used to draw conclusions. Hence, the search time score was only used to select a configuration that was to be tested with the mean analysis. Therefore, the only drawback is that a possible conclusion could be missed because the best configuration does not necessarily have the best search time score.

Getting rid of the unfairness in the adaptation of the search time score can easily be done by letting the search time score be calculated from each data point for a configuration instead of the mean for each sample for a configuration. Unfortunately, this requires that all samples are fully sampled. As stated earlier, this is not possible within the time frame of the thesis.

Mean Analysis
There are two points that are important to mention about the mean analysis that was done. One of them are about assumptions that are done with the analysis method. The other is about which problems that was analysed with the mean analysis.

The assumption that is done in the analysis is that the CLT holds when a sample consists of 30 data points. Assuming that CLT holds for 30 data points is quite standard in the area of statistics according to the literature. However, it should be noted that the requirement of a larger sample size has been proposed.

A more interesting part is the effect of the selection that was made when only using the subset of problems that both configurations in the comparison manages to solve, within the time limit, 30 times. This was required because a fully sampled sample was not possible. However, it means that the problems that are hard for at least one of the configurations are not considered. Unfortunately, the harder problems are often those that are most interesting. However, the coverage score can be used to get an indication for how the different configurations performs on the harder problem.

Standard Deviation Ranking
The standard deviation ranking is a non-validated measurement. Naturally, there are a multitude of problems that arise from using non-validated measurements and they should not be used to draw conclusions. This is not done in the thesis since the ranking is only used to select which configuration that should be compared when testing for improvements in the standard deviation. Moreover, the ranking selects the configuration that was the best on all the problems when only considering the ordering. Therefore, it is likely, but not guaranteed, that the ranking selects at least one of the best configurations in terms of standard deviation. Hence, some conclusions might be missed when using this to select the configuration for the standard deviation analysis.

Standard Deviation Analysis
The standard deviation analysis is a descriptive method for analysing which configuration that is the best in terms predictability as measured by standard deviation. This method for analysis has the same assumptions and restrictions to the problems that are analysed as the mean analysis since it is based on the same linear combination of normal distributions. Especially, the standard deviation analysis is based on the fact that the estimations of the populations standard deviation based on the samples are correct. However, as argued above, this assumption is a standard assumption within statistics so it should not be a hinder for drawing conclusions.
9.5 Ethical Aspects

All components in an autonomous system that make decisions or contribute to decisions play a role in ethical scenarios [38]. For planning and plan repair system these ethical decisions inherently becomes deliberate because planning and plan repair systems are deliberate systems. Naturally, it follows that developing these systems will have an impact on ethical scenarios. Planning and plan repair systems have potential to be used in many contexts and it can easily be argued that it is impossible to construct an ethical decision making component in the system itself [3]. Naturally, these areas ranges from the simple ethical decisions: Should an autonomous car park in the middle of the highway because no parking spots are available when there are no other contributing factors? To the harder: Should the autonomous car hit one or more humans on the pavement or someone crossing the street when no other option is available?

Moor argues that all systems that participates in ethical scenarios should at least have capabilities to handle explicit rules for ethical scenarios [38]. This means that the developer should be able configure how the system reacts in specific ethical scenarios. This is possible in all planners and plan repair systems by extending the planning problems or plan repair problems with an ethical model. For example, one could add a constraint to a “park” operator that the destination may not be a highway. Of course, this leaves it up to the user of the planning or plan repair system to include this in the problem specifications.

Finally, it should be noted that the work done in this thesis originates from a military scenario. Military, in general, is quite common targets for ethical discussions. However, it should also be noted that the technology in this thesis does not differ between this scenario and a rescue mission. Both scenarios have targets that they should find and do something with (for example, it could be delivering supplies or destroying hostile targets in a military scenario and delivering or identifying people in distress in a rescue mission). Moreover, both scenarios can have areas that the agents may not enter (for example houses and no-fly zones). Finally, there may be areas that an agent cannot fly within without risk of getting destroyed or damaged (hostile SAM units are a military example and areas that have flammable or explosive gas is a rescue mission example). Based on these similarities from the perspective of the technology, stating that the technology is good or bad can be compared to stating that the combustion engine is good or bad. Essentially, the usage area is too wide for any decisive conclusion.

\footnote{Remember that relatively simple cases of task planning is PSPACE-complete, meaning that all PSPACE problem \( p \) can be transformed to a planning problem \( p' \) in polynomial time and that the answer from solving \( p' \) can be transformed to the answer to \( p \) in polynomial time. Therefore, all ethical scenarios that any system that runs in PSPACE encounters can be encountered by a planner or a plan repair system.}
The answers to the research questions are presented in this chapter. Each section in the chapter covers one research question, which includes the question itself, the answer to the question and, in some cases, a short section with comments about the answer and indications that was found.

RQ1 - Integrating Motion Planning

How can motion planning be integrated in a temporal planner that does early commitment in time?

Answer: Motion planning can be integrated in a temporal task planner that does early commitment in time by querying a motion planner for a path whenever the temporal task planner adds an action that requires a path to the plan.

The observations in the thesis indicates that the other designs for integrating motion planning in task planning are not suitable for a temporal planner that does early commitment in time. This is because a temporal operator with a motion planning part might depend on the path to get the duration of a temporal action instantiated from the temporal operator. Hence, finding paths after a task plan has been constructed is likely to be quite ineffective for temporal planners that does early commitment in time.

RQ2 - Automatic Generation of Temporal Macro Operators

Can TMOs, expressed using PDDL2.1, be generated automatically?

Answer: Yes, TMOs, expressed using PDDL2.1, can be generated automatically.

RQ3 - Search Time Plan Repair

Can the repair time decrease when using TMOs when replanning as repair compared to when not using any TMOs?
Answer: The data does not show that using TMOs when replanning as repair decreases the repair time compared to when not using TMOs.

The data indicates that using the types of TMOs that was developed in the thesis increases the repair time. However, this does not mean that there are no types of TMOs that decreases it. Moreover, it should be stressed that the thesis only investigated TMOs that can be expressed in PDDL2.1.

RQ4 - Standard Deviation Plan Repair

Can the repair time be made more predictable, as measured by the standard deviation, while still being viable (finish within a time limit) by using TMOs when using replanning as repair?

Answer: The data does not show that the repair time can be made more predictable as measured by standard deviation when using TMOs when using replanning as repair compared to when not using TMOs.

The data indicates that using the TMO types that was used in the thesis increases the standard deviation. However, this cannot be generalised to an indication for all TMOs expressed using PDDL2.1. Moreover, the data indicates that the standard deviation depends on the search time and that using TMOs barely, if at all, affects the standard deviation directly.

RQ5 - Search Time Heuristic Extension

Can the search time decrease when using an extended version of TFD’s heuristic function that uses geometric information in the combined task and motion planning problem compared to the original heuristic function that does not use geometric information?

Answer: Yes, extending TFD’s heuristic function to use geometric information can decrease the search time compared to when geometric information is not used.

Note that not using geometric information for an action that has a duration that only depends on the path is equivalent to not having any information about the duration at all in the heuristic function.

RQ6 - Standard Deviation Heuristic Extension

Can the search time become more predictable, as measured by the standard deviation, when using an extended version of TFD’s heuristic function that uses geometric information in the combined task and motion planning problem compared to the original that does not use geometric information?

Answer: Yes, search time can be made more predictable as measured by standard deviation when using an extended version of TFD’s heuristic function that uses geometric information compared to when no geometric information is used.

Similar to the research question RQ4, the data gathered indicates that the standard deviation depends on the search time rather than if the geometric extension is used or not. Hence, it seems like the standard deviation decreases when using the geometric extension due to the decreased search time.
11 Future Work

There are much more to explore and investigate regarding integration of motion planning in temporal planning and TMOs than what is covered in this thesis. Naturally, many interesting areas have been identified when working on the thesis. A selection of them are presented in this chapter.

11.1 Impact of Motion Planners

The system that was implemented uses a very simple motion planner. Based on this, an interesting question is whether a motion planner that yields paths with higher quality would result in better search times. As argued earlier in the thesis (see section 9.3), this could quite possibly be the case for TFD since the number of nodes that are visited depends on, among other things, the difference in the estimated time it takes to travel between two points and the actual time it takes. Hence, a path that has a travel time that is closer to the estimated travel time will likely result in that fewer nodes are visited. Of course, the drawback is that finding better paths may take longer time.

11.2 Geometric Backtracking

Geometric backtracking is a technique that can be used to test new paths associated with a task plan when new paths are needed. It has been explained earlier in the thesis how geometric backtracking can be implemented in the system. However, it still remains to be done and evaluated. The main reason to why this is interesting is that geometric backtracking has been used successfully to improve the results when non-temporal task planning was used. Therefore, there are indications that this technique can be useful even in a temporal context.

11.3 Multiple Repair Strategies

The system, as it is designed at the moment, always tries to repair a problem by doing replanning with TMOs as repair. However, this means that unnecessary hard problems are solved sometimes. For example, if a path of the plan is blocked by a newly discovered threat, then the repair problem could potentially be solved by finding new paths and reschedule as
done by Rundqiwist, Grässel and Ruel. This is a simpler problem which means that the repair time is expected to be shorter. Naturally, the same principle applies to other categories of problems with the plan as well (see the categories listed in section 11.3 for more examples of categories). Of course, this leads to the question if using such categories and repair strategies actually decrease the search time.

11.4 Temporal Macro Operators in PDDL2.1

This thesis have far from closed the chapter on TMOs expressible in PDDL2.1. In fact, it can at most be described as scratching the surface. However, the indications in this thesis are that TMOs expressible in PDDL2.1 have too much overhead associated with them to be worth studying over studying other techniques for plan repair of temporal plans. Disregarding the negative indications, there is much to investigate about TMOs expressible in PDDL2.1 for the interested. Among those are other types of TMOs expressible in PDDL2.1, methods for selecting the temporal operators the TMOs should be created from and filter methods for selecting the best TMOs. Two other examples are to improve the mutex-lock generation when using STMOs by identifying some resources that can be protected by an over all condition and to prune some STMOs that are not internally consistent due to resource constraints.

11.5 Temporal Macro Operators

This thesis concerns TMOs expressible in PDDL2.1 when doing replanning as repair and it does therefore not include a study regarding TMOs that are not expressible in PDDL2.1. However, those are definitely of interest since they do not necessarily suffer from the same drawbacks as TMOs expressible in PDDL2.1. In fact, previous work on handwritten TMOs (or composite actions as they are sometimes called) have used TMOs with good results. Hence, a logical step is to investigate if TMOs can be automatically generated and used with a positive effect for planning systems that are aware of TMOs.

11.6 Preprocessing on What If Basis

The planning system does not compute anything during the execution of a plan. This means that there might be resources that are not utilised during the execution. These resources could be used for preprocessing based on “what if” assumptions. All the online macro generation techniques that have been studied in this thesis are online because there is no old plan until the current plan is rendered invalid. However, an old plan is simply a complete plan and a point in time that shows where the old plan is rendered invalid. Hence, one can use the current plan and a point in time to precompute which TMOs that could be used if the plan would be rendered invalid at that point in time. Moreover, this set of TMOs are not just valid for that specific point in time but from the previous event (i.e. the closest start or end of a temporal action before the point in time) to the next event because nothing interesting happens in the plan between those points in time. Hence, the number of possible sets of TMOs are limited to a function of the number of temporal actions in the plan. This makes it possible to precompute the set of TMOs for all possible time points in the plan in case it is rendered invalid. This preprocessing method can also be extended to include more informative filters that are computational demanding and that can be computed before the plan is rendered invalid. More importantly, if STMOs are used, then one can precompute the macro injection because this only requires a set of TMOs that the problem is to be extended from. Naturally, this has the potential to reduce the search time a lot since the mutex injection is a costly operation.
This appendix describes 5 typical scenarios that the solver must be able to handle.

A.1 Scenario 1

This scenario is one of the easier scenarios with a few sparse threats and targets (which are outside the range of the threats). Moreover, there are no malfunctions. The parameters to the scenario are shown in table A.1 and a conceptual sketch of the scenario is shown in figure A.1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent count</td>
<td>3</td>
</tr>
<tr>
<td>Area size</td>
<td>100km×100km</td>
</tr>
<tr>
<td>Base distance</td>
<td>50km</td>
</tr>
<tr>
<td>Risk policies</td>
<td>NO_LOSES</td>
</tr>
<tr>
<td>Threats</td>
<td>(10km, 10km), (50km, 80km), (90km, 10km)</td>
</tr>
<tr>
<td>Targets</td>
<td>(5km, 80km), (10km, 45km), (80km, 10km)</td>
</tr>
<tr>
<td>Malfunction events</td>
<td>-</td>
</tr>
</tbody>
</table>
A.1. Scenario 1

Figure A.1: Overview of scenario 1.
A.2 Scenario 2

Similar to scenario 1, the second scenario is a simple version where there are no threats but there are malfunctions in the sensors of the UAVs. The parameters to the scenario are shown in table A.2 and a conceptual sketch of the scenario is shown in figure A.2.

Table A.2: Parameters for scenario 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent count</td>
<td>3</td>
</tr>
<tr>
<td>Area size</td>
<td>100km×100km</td>
</tr>
<tr>
<td>Base distance</td>
<td>50km</td>
</tr>
<tr>
<td>Risk policies</td>
<td>NO_LOSES</td>
</tr>
<tr>
<td>Threats</td>
<td>-</td>
</tr>
<tr>
<td>Targets</td>
<td>(5km, 80km), (10km, 45km), (80km, 10km)</td>
</tr>
<tr>
<td>Malfunction events</td>
<td>(1h, ACTIVE_SAR_MALFUNCTION), (2h, ACTIVE_EO/IR_MALFUNCTION)</td>
</tr>
</tbody>
</table>

Figure A.2: Overview of scenario 2.
A.3 Scenario 3

The third scenario is a combination of scenario 1 and 2. The parameters to the scenario are shown in table A.3 and a conceptual sketch of the scenario is shown in figure A.3.

Table A.3: Parameters for scenario 3.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent count</td>
<td>3</td>
</tr>
<tr>
<td>Area size</td>
<td>100km × 100km</td>
</tr>
<tr>
<td>Base distance</td>
<td>50km</td>
</tr>
<tr>
<td>Risk policies</td>
<td>NO_LOSSES</td>
</tr>
<tr>
<td>Threats</td>
<td>(10km, 10km), (50km, 80km), (90km, 10km)</td>
</tr>
<tr>
<td>Targets</td>
<td>(5km, 80km), (10km, 45km), (80km, 10km)</td>
</tr>
<tr>
<td>Malfunction events</td>
<td>(1h, ACTIVE_SAR_MALFUNCTION), (2h, ACTIVE_EO/IR_MALFUNCTION)</td>
</tr>
</tbody>
</table>

Figure A.3: Overview of scenario 3.
A.4 Scenario 4

The fourth scenario contains many SAM units and a few targets, which are outside the weapon range and detection range. However, the targets may be within the radar detection range of the UAVs. The scenario is described in table A.4 and a sketch of the scenario is shown in figure A.4.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent count</td>
<td>3</td>
</tr>
<tr>
<td>Area size</td>
<td>100km×100km</td>
</tr>
<tr>
<td>Base distance</td>
<td>50km</td>
</tr>
<tr>
<td>Risk policies</td>
<td>NO_LOSSES</td>
</tr>
<tr>
<td>Threats</td>
<td>(10km, 10km), (20km, 30km), (30km, 10km), (40km, 40km), (55km, 25km), (55km, 15km), (55km, 5km), (60km, 70km), (70km, 50km), (75km, 10km), (80km, 30km), (15km, 55km), (25km, 75km), (85km, 85km), (45km, 85km)</td>
</tr>
<tr>
<td>Targets</td>
<td>(20km, 15km), (45km, 10km), (50km, 50km), (70km, 70km), (90km, 40km), (5km, 85km), (30km, 55km)</td>
</tr>
<tr>
<td>Malfunction events</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure A.4: Overview of scenario 4.
A.5 Scenario 5

The fifth scenario has less SAM units than the fourth scenario but requires that no UAVs are detected when flying at altitude 1.5 km. The scenario is described in table A.5 and a sketch of the scenario is shown in figure A.5.

Table A.5: Parameters for scenario 5.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent count</td>
<td>3</td>
</tr>
<tr>
<td>Area size</td>
<td>100km×100km</td>
</tr>
<tr>
<td>Base distance</td>
<td>50km</td>
</tr>
<tr>
<td>Risk policies</td>
<td>NO_ED/IR_DETECTION</td>
</tr>
<tr>
<td>Threats</td>
<td>(20km, 30km), (50km, 25km), (71km, 48km), (75km, 10km), (25km, 75km), (85km, 85km), (45km, 85km)</td>
</tr>
<tr>
<td>Targets</td>
<td>(20km, 5km), (45km, 5km), (50km, 50km), (70km, 70km), (90km, 40km), (5km, 85km), (30km, 55km)</td>
</tr>
<tr>
<td>Malfunction events</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure A.5: Overview of scenario 5.
This appendix consists of two examples using the definitions of the classical formalism extended with resources and of the temporal planning formalism in chapter 3. Both of the examples uses the same domain that consists of one agent moving cargoes between different locations in a dock. The examples are simple but covers the important aspects.

The following naming convention will be used in both examples:

**Constants:** These are lowercase italic strings (e.g. constant).

**Variables:** These are lowercase italic strings starting with a “?” (e.g. ?variable).

**Operator names:** These are lowercase strings (e.g. operator).

In addition, \text{op(?var}_1/\text{const}_1, ?var}_2/\text{const}_2, \ldots, ?var}_n/\text{const}_n) refers to the (temporal) action that is gained when replacing ?var}_1 with \text{const}_1, ?var}_2 with \text{const}_2 and so on in the (temporal) operator with the name “op”.

### B.1 Classical Planning with Resources

The example for the classical formalism with resource extension is explained in a bottom up manner, just like the definitions for it. This means that the example starts with the predicates, resources and constants. From there, the example continues with the operators, the actions and finally reaching the domain, problem and plans.

**Predicates, Resources and Constants**

**Constants:** \{\text{dock}_1, \text{dock}_2, \text{robot}_1, \text{cargo}_1, \text{cargo}_2\}.

**Predicates:** \{\text{at(?obj,?loc), is\text{-}location(?loc)}\}

**Resources:** \{\text{fuel(?agent)}\}
Operators and Actions

- (“pick-up”, \{\textit{at}(\textit{agent}, \textit{?loct}), \textit{at}(\textit{cargo}, \textit{?loc}), \textit{is\textendash}location(\textit{?loc})\}, \emptyset, \textit{at}(\textit{cargo}, \textit{?agent}), \neg \textit{at}(\textit{cargo}, \textit{?loc}))\}

- (“put-down”, \{\textit{at}(\textit{agent}, \textit{?loct}), \textit{at}(\textit{cargo}, \textit{?agent}), \textit{is\textendash}location(\textit{?loc})\}, \emptyset, \textit{at}(\textit{cargo}, \textit{?loc}), \neg \textit{at}(\textit{cargo}, \textit{?agent}))\}

- (“move”, \{\textit{at}(\textit{agent}, \textit{?from}), \textit{is\textendash}location(\textit{?from}), \textit{is\textendash}location(\textit{?to})\}, \{\geq (\textit{fuel}(\textit{?agent}), 1), \textit{at}(\textit{agent}, \textit{?to}), \neg \textit{at}(\textit{agent}, \textit{?from})\}, \{(\textit{fuel}(\textit{?agent}), \neg (\textit{fuel}(\textit{?agent}), 1))\})

The actions in the example are all possible grounded versions of the operators. For example, operator pick-up has a grounded form where \textit{?agent} = \textit{robot1}, \textit{?loct} = \textit{dock1} and \textit{?cargo} = \textit{cargo1}. Unfortunately, there is one action for every possible combination of replacements of variables with constants for each operator (this gives \(5^3 + 5^3 + 5^3 = 375\) distinct actions) so they cannot be listed here. Naturally, many of them are never applicable in most problems because some preconditions never holds in most problems. For example, \textit{is\textendash}location(\textit{robot1}) is seldom true in the initial state and no action makes it true. This means that an action that has replaced variables \textit{?loct}, \textit{?from} or \textit{?to} with \textit{robot1} is only applicable in the problems where \textit{robot1} is an object that is a location.

Planning Domain and Problem

The planning domain from this example is \(\Sigma(S, A, \gamma)\) where:

- \(S\) is an infinite set of states. Each state consists of a distinct set of grounded predicate and a set of resources. There are \(2^{30}\) different combinations of grounded predicates in the example domain. This huge number comes from that each of the 25 different grounded forms of the predicate \(\textit{at}(\textit{obj}, \textit{?loc})\) and 5 different grounded forms for the predicate \(\textit{is\textendash}location(\textit{?loc})\) can be true or false in a state. Moreover, there are five different resources (all possible grounded forms of resource \textit{fuel}(\textit{?agent})\) which map to a real number (i.e. there are an infinite number of possible mappings). For simplicity, only the relevant fuels (\textit{fuel}(\textit{robot1})) will be included in the states.

- \(A\) is the set of the 375 actions that are described above.

- \(\gamma\) is the state transition function that is defined in section 3.3 in chapter 3.

The planning problem of the example is \((\Sigma, s_0, g)\) where:

- \(s_0\) is the initial state:
  \(s_0 = ((\textit{is\textendash}location(\textit{dock1}), \textit{is\textendash}location(\textit{dock2}), \textit{at}(\textit{robot1}, \textit{dock1}), \textit{at}(\textit{cargo1}, \textit{dock1}), \textit{at}(\textit{cargo2}, \textit{dock2})), \{\textit{fuel}(\textit{robot1}) \Rightarrow 5\})\)

- \(g\) is the goal of the planning problem:
  \(g = ((\textit{at}(\textit{cargo1}, \textit{dock2}), \textit{at}(\textit{cargo2}, \textit{dock2}), \textit{at}(\textit{robot1}, \textit{dock1})), \{\geq (\textit{fuel}(\textit{robot1}), 2)\})\)
Classical Task Plan

The plan in the example is:

$$
\pi = \langle \text{pick-up}('agent\,robot1', 'loc\,dock1', 'cargo\,cargo1'), \\
\text{move}('agent\,robot1', 'from\,dock1', 'to\,dock2'), \\
\text{put-down}('agent\,robot1', 'loc\,dock2', 'cargo\,cargo1'), \\
\text{pick-up}('agent\,robot1', 'loc\,dock2', 'cargo\,cargo2'), \\
\text{move}('agent\,robot1', 'from\,dock2', 'to\,dock1'), \\
\text{put-down}('agent\,robot1', 'loc\,dock1', 'cargo\,cargo2') \rangle
$$

Applying the Plan

Applying the plan above results in the following (where $\pi[i]$ refers to action number $i$ in the plan, starting with $i = 1$):

1. $s_1 = \gamma(s_0, \pi[1]) = (\{\text{is-}\text{location}(\text{dock1}), \text{is-}\text{location}(\text{dock2}), \text{at}(\text{robot1}, \text{dock1}), \\
\text{at}(\text{cargo1}, \text{robot1}), \text{at}(\text{cargo2}, \text{dock2}), \\
\text{fuel(\text{robot1})} \mapsto 5\})$
2. $s_2 = \gamma(s_1, \pi[2]) = (\{\text{is-}\text{location}(\text{dock1}), \text{is-}\text{location}(\text{dock2}), \text{at}(\text{robot1}, \text{dock2}), \\
\text{at}(\text{cargo1}, \text{robot1}), \text{at}(\text{cargo2}, \text{dock2}), \\
\text{fuel(\text{robot1})} \mapsto 4\})$
3. $s_3 = \gamma(s_2, \pi[3]) = (\{\text{is-}\text{location}(\text{dock1}), \text{is-}\text{location}(\text{dock2}), \text{at}(\text{robot1}, \text{dock2}), \\
\text{at}(\text{cargo1}, \text{dock2}), \text{at}(\text{cargo2}, \text{dock2}), \\
\text{fuel(\text{robot1})} \mapsto 4\})$
4. $s_4 = \gamma(s_3, \pi[4]) = (\{\text{is-}\text{location}(\text{dock1}), \text{is-}\text{location}(\text{dock2}), \text{at}(\text{robot1}, \text{dock2}), \\
\text{at}(\text{cargo1}, \text{dock2}), \text{at}(\text{cargo2}, \text{robot1}), \\
\text{fuel(\text{robot1})} \mapsto 4\})$
5. $s_5 = \gamma(s_4, \pi[5]) = (\{\text{is-}\text{location}(\text{dock1}), \text{is-}\text{location}(\text{dock2}), \text{at}(\text{robot1}, \text{dock1}), \\
\text{at}(\text{cargo1}, \text{dock2}), \text{at}(\text{cargo2}, \text{robot1}), \\
\text{fuel(\text{robot1})} \mapsto 3\})$
6. $s_6 = \gamma(s_5, \pi[6]) = (\{\text{is-}\text{location}(\text{dock1}), \text{is-}\text{location}(\text{dock2}), \text{at}(\text{robot1}, \text{dock1}), \\
\text{at}(\text{cargo1}, \text{dock2}), \text{at}(\text{cargo2}, \text{dock1}), \\
\text{fuel(\text{robot1})} \mapsto 3\})$

One can see that the plan is applicable in the initial state. Moreover, $s_6 \models g$ because $\text{is-} \text{at}(\text{robot1}, \text{dock1}), \text{is-} \text{at}(\text{cargo2}, \text{dock1}), \text{is-} \text{at}(\text{cargo1}, \text{dock2})$ and $\text{fuel(\text{robot1})} \geq 2$ is true in state $s_6$. Hence, the plan ensures that the goal is reached when it is applied to $s_0$.

B.2 Temporal Planning

The temporal planning problem uses the same constants, predicates and resources as the classical planning problem. However, the operators, actions, domain, problem and plan are changed as explained in this section.
Temporal Operator and Temporal Actions

- (“pick-up”, \texttt{?start}, 1, \{at(?agent, ?loc), at(?cargo, ?loc), is\text{-}location(?loc), \varnothing, \{at(?agent, ?loc), \varnothing, \varnothing, \neg\text{at}(?cargo, ?loc), \varnothing, \{\text{at}(?cargo, ?agent)\}, \varnothing\})
- (“put-down”, \texttt{?start}, 1, \{at(?agent, ?loc), at(?cargo, ?agent), is\text{-}location(?loc), \varnothing, \{at(?agent, ?loc), \varnothing, \varnothing, \neg\text{at}(?cargo, ?agent), \varnothing, \{\text{at}(?cargo, ?loc)\}, \varnothing\})
- (“move”, \texttt{?start}, 4, \{at(?agent, ?from), is\text{-}location(?from), is\text{-}location(?to)\}, \{\geq (\text{fuel}(?agent), 1), \varnothing, \varnothing, \varnothing, \neg\text{at}(?agent, ?from), \varnothing, \{\text{at}(?agent, ?to)\}, \{(\text{fuel}(?agent), -(\text{fuel}(?agent), 1))\})

As stated in the definition, the temporal actions in the problem are all grounded versions of the temporal operators. In this case, the temporal operators are grounded in the same way as the operators in the example of the classical planning problem with one addition. The addition is that the variable \texttt{?start} is also replaced with a non-negative real number that indicates the start time of the temporal action. Naturally, there are an infinite number of temporal actions.

Planning Domain and Problem

The planning domain for the temporal planning problem is \(\Sigma^T = (S^T, A^T, \gamma^T)\) where:

- \(S^T\) is the set of all temporal states. Naturally, there is an infinite set of states (one only has to look at the time component of the state that is a non-negative real number to determine this).
- \(A^T\) is the set of all temporal actions. Again this set is infinite and consists of the actions explained above.
- \(\gamma^T\) is the state transition function for temporal planning as defined in section \ref{sec:temporal-planning} in chapter \ref{chap:temporal-planning}.

The planning problem in this example is \((\Sigma^T, s_0^T, g)\) where:

- \(\Sigma^T\) is the planning domain as described above.
- \(s_0^T\) is the initial temporal state of the planning problem:
  \[s_0^T = (0, ((\text{is\text{-}location}(\text{dock1})), \text{is\text{-}location}(\text{dock2})), \text{at}(\text{robot1}, \text{dock1}), \text{at}(\text{cargo1}, \text{dock1}), \text{at}(\text{cargo2}, \text{dock2}), \{(\text{fuel}(\text{robot1}) \Rightarrow 5)\}, \varnothing, \varnothing)\]
- \(g\) is the goal of the temporal planning problem, which is the same as in the example in the classical planning problem:
  \[g = ((\text{at}(\text{cargo1}, \text{dock2}), \text{at}(\text{cargo2}, \text{dock1}), \text{at}(\text{robot1}, \text{dock1})), \geq (\text{fuel}(\text{robot1}), 2))\]

Temporal Plan

One temporal plan in the planning domain in this example is as follows:

\[
\pi^T = \{\text{pick-up}(\texttt{?start}/0, \texttt{?agent/robot1}, \texttt{?loc/dock1}, \texttt{?cargo/cargo1}), \text{move}(\texttt{?start}/1, \texttt{?agent/robot1}, \texttt{?from/dock1}, \texttt{?to/dock1}), \text{put-down}(\texttt{?start}/5, \texttt{?agent/robot1}, \texttt{?loc/dock2}, \texttt{?cargo/cargo1}), \text{pick-up}(\texttt{?start}/6, \texttt{?agent/robot1}, \texttt{?loc/dock2}, \texttt{?cargo/cargo2}), \text{move}(\texttt{?start}/7, \texttt{?agent/robot1}, \texttt{?from/dock2}, \texttt{?to/dock1}), \text{put-down}(\texttt{?start}/11, \texttt{?agent/robot1}, \texttt{?loc/dock1}, \texttt{?cargo/cargo2})\}
\]
Applying the Temporal Plan

Applying the temporal plan is done with the state transition function defined in the domain, $\gamma^s$, and the progression function $\sigma$ (see section 3.1 in chapter 3). In the following, $\pi[i]$ is used to refer to temporal action number $i$ in $\pi$ using the order from the previous section, starting with $i = 1$.

1. $s_1 = \gamma^s(s_0, \pi[1]) = (0, (\{is-location(dock1), is-location(dock2), at(robot1, dock1),
\at(cargo2, dock2)\),
\{fuel(robot1) \Rightarrow 5\}),
\{\{at(robot1, dock1), \varnothing, 1)\}, \{(at(cargo1, robot1), \varnothing, 1)\}, \varnothing)\)

2. $s_2 = \sigma(s_1) = (1, (\{is-location(dock1), is-location(dock2), at(robot1, dock1),
\at(cargo1, robot1), at(cargo2, dock2)\),
\{fuel(robot1) \Rightarrow 5\}),
\varnothing, \varnothing, \varnothing)\)

3. $s_3 = \gamma^s(s_2, \pi[2]) = (1, (\{is-location(dock1), is-location(dock2), at(cargo1, robot1),
\at(cargo2, dock2)\),
\{fuel(robot1) \Rightarrow 5\}),
\varnothing, \varnothing, \varnothing)\)

4. $s_4 = \sigma(s_3) = (5, (\{is-location(dock1), is-location(dock2), at(robot1, dock2),
\at(cargo1, robot1), at(cargo2, dock2)\),
\{fuel(robot1) \Rightarrow 4\}),
\varnothing, \varnothing, \varnothing)\)

5. $s_5 = \gamma^s(s_4, \pi[3]) = (5, (\{is-location(dock1), is-location(dock2), at(robot1, dock2),
\at(cargo2, dock2)\),
\{fuel(robot1) \Rightarrow 4\}),
\{(at(robot1, dock2), \varnothing, 1)\}, \{(at(cargo1, dock2), \varnothing, 1)\}, \varnothing)\)

6. $s_6 = \sigma(s_5) = (6, (\{is-location(dock1), is-location(dock2), at(robot1, dock2),
\at(cargo1, dock2), at(cargo2, dock2)\),
\{fuel(robot1) \Rightarrow 4\}),
\varnothing, \varnothing, \varnothing)\)

7. $s_7 = \gamma^s(s_6, \pi[4]) = (6, (\{is-location(dock1), is-location(dock2), at(robot1, dock2),
\at(cargo1, dock2)\),
\{fuel(robot1) \Rightarrow 4\}),
\{(at(robot1, dock2), \varnothing, 1)\}, \{(at(cargo2, robot1), \varnothing, 1)\}, \varnothing)\)

8. $s_8 = \sigma(s_7) = (7, (\{is-location(dock1), is-location(dock2), at(robot1, dock2),
\at(cargo1, dock2), at(cargo2, robot1)\),
\{fuel(robot1) \Rightarrow 4\}),
\varnothing, \varnothing)\)
9. \( s_9 = \gamma^7(s_8, \pi[5]) = (7, (\{\text{is-\text{location}}(\text{dock1}), \text{is-\text{location}}(\text{dock2}), \at(\text{cargo1}, \text{dock2}), \at(\text{cargo2}, \text{robot1}) \}, \\{ \text{fuel}(\text{robot1}) \mapsto 4 \}), \\emptyset, ((\at(\text{robot1}, \text{dock1})), ((\text{fuel}(\text{robot1})), (\neg(\text{fuel}(\text{robot1})), 1)), 4), \\emptyset) \)

10. \( s_{10} = \sigma(s_6) = (11, (\{\text{is-\text{location}}(\text{dock1}), \text{is-\text{location}}(\text{dock2}), \at(\text{robot1}, \text{dock1}) \at(\text{cargo1}, \text{dock2}), \at(\text{cargo2}, \text{robot1}) \}, \\{ \text{fuel}(\text{robot1}) \mapsto 3 \}), \\emptyset, \emptyset, \emptyset) \)

11. \( s_{11} = \gamma(s_{10}, \pi[6]) = (11, (\{\text{is-\text{location}}(\text{dock1}), \text{is-\text{location}}(\text{dock2}), \at(\text{robot1}, \text{dock1}) \at(\text{cargo1}, \text{dock2}) \}, \\{ \text{fuel}(\text{robot1}) \mapsto 3 \}), \\emptyset, ((\at(\text{robot1}, \text{dock1})), \\emptyset, 1)), ((\at(\text{cargo2}, \text{dock1})), \\emptyset, 1), \\emptyset, \emptyset) \)

12. \( s_{12} = \sigma(s_{11}) = (12, (\{\text{is-\text{location}}(\text{dock1}), \text{is-\text{location}}(\text{dock2}), \at(\text{robot1}, \text{dock1}) \at(\text{cargo1}, \text{dock2}), \at(\text{cargo2}, \text{dock1}) \}, \\{ \text{fuel}(\text{robot1}) \mapsto 3 \}), \\emptyset, \emptyset) \)

Just like the classical plan, this plan is both applicable in the initial state and ensures that the goal is reached when the plan is applied the initial state. The applicability comes from that the preconditions holds for all temporal actions in the temporal state they are used in and because the active conditions holds in all temporal states. Moreover, it should be added that the plan ensures that the goal is reached because \( s_{12} \models g \) and \( s_{12} \) is consistent (\( \omega(s_{12}) \)).
This appendix contains a description of the syntax and the semantics of the motion planning constructs that PDDL2.1 was extended with. All the new constructs have a name starting with “path” to identify them as a motion planning construct. Each section covers one or more related constructs with a short description of the semantics, a grammar written in Backus normal form describing the syntax and an example. The only exception is the first section which covers a few grammatical rules that are common for multiple constructs.

C.1 Common Grammar

The grammar describing the constructs that are the motion planning extension to PDDL2.1 share some common grammatical rules of which some describe standard PDDL2.1 syntax. The following rules are those that are standard PDDL2.1 syntax:

- `<var>`: A PDDL2.1 variable (i.e. starting with “?” and consists of letters and “-”).
- `<type>`: A type specifier in PDDL2.1 (i.e. a name consisting of letters and “-”).
- `<number>`: A float or an integer.
- `<identifier>`: An identifier consisting of letters and “-”.

The following rules are the common grammatical rules used throughout the appendix:

<table>
<thead>
<tr>
<th>Grammar Rule</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;typed-list&gt;</code></td>
<td><code>( '&lt;typed-var&gt; &lt;typed-vars&gt; ' ) ' </code></td>
</tr>
<tr>
<td><code>&lt;typed-vars&gt;</code></td>
<td>`&lt;typed-var&gt; &lt;typed-vars&gt;</td>
</tr>
<tr>
<td><code>&lt;typed-var&gt;</code></td>
<td>`&lt;domain-var&gt; &lt;domain-vars&gt; ' - ' &lt;type&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;domain-vars&gt;</code></td>
<td>`&lt;domain-var&gt; &lt;domain-vars&gt;</td>
</tr>
<tr>
<td><code>&lt;domain-var&gt;</code></td>
<td><code>( '&lt;var&gt; &lt;vars&gt; ' ) ' </code></td>
</tr>
<tr>
<td><code>&lt;numbers&gt;</code></td>
<td><code>( '&lt;number&gt; &lt;numbers&gt; ' ) ' </code></td>
</tr>
<tr>
<td><code>&lt;vars&gt;</code></td>
<td>`&lt;var&gt; &lt;vars&gt;</td>
</tr>
<tr>
<td><code>&lt;identifiers&gt;</code></td>
<td>`&lt;identifier&gt; &lt;identifiers&gt;</td>
</tr>
</tbody>
</table>

106
C.2 Work Space and Configuration Space

The work and configuration space construct in the PDDL2.1 domain file is used to define named types with a limited domain. These types are tuples with a specified length and numeric domains for each index. The following grammatical rules describe the syntax for how to define a work or configuration space:

\[
\text{<path-domain>} ::= \text{(: path-domain ' <identifier> ': dimensions')}
\]

\[
\text{<dimensions>} ::= \text{('( ' <dimension> <more-dimensions> ')')}
\]

\[
\text{<more-dimensions>} ::= \text{<dimension> <more-dimensions> | ϵ}
\]

\[
\text{<dimension>} ::= \text{('( ' <number> <number> ')')}
\]

An example of using the construct is the following that defines a configuration space:

\[
\text{(: path-domain configuration :dimensions ((0 160000) (0 182000) (-3.1415 3.1415)))}
\]

C.3 Motion Planners

The motion planner construct is used to define, in the PDDL2.1 domain file, which motion planners that should be instantiated. This construct can be divided into two parts: One that is providing information about the motion planner; and one that is used to ensure that the PDDL2.1 domain file is correct. The former consists of an identifier for the planner, the type of motion planner that should be used (the system implements a RRT planner and a planner that always fails to find a plan) and the configuration space that is used to search for a path. The later consists of the parameters and return values and could be left out. However, they allow for the program to verify that the variables that stores the return values (the first <typed-list> rule in the <planner-call>) and the provided arguments (the second <typed-list>) in an operator are of the correct type. Note that each set of parameters and return values defines one possible call to the planner.

The grammar for defining the motion planners are as follows:

\[
\text{<motion-planner>} ::= \text{'((: path-planner ' <identifier> ': planner')}
\]

\[
\text{<planner-tail>} ::= \text{<empty-planner> | <rrt-planner>}
\]

\[
\text{<empty-planner>} ::= \text{'none'}
\]

\[
\text{<rrt-planner>} ::= \text{'rrt '': configuration-space ' <identifier>': parameters' <planner-call> <planner-calls>}
\]

\[
\text{<planner-calls>} ::= \text{<planner-call> <planner-calls> | ϵ}
\]

\[
\text{<planner-call>} ::= \text{'(=' <typed-list> <typed-list> ')}
\]
C.4 Motion Planning Resources and State Variables

There is one construct that models both the resources and state variables for the motion planning part. This is because a resource and a state variable are essentially the same but with different domains. Formally, a motion planning state variable is defined as

\[ s : t_0, t_1, \ldots, t_n \rightarrow D \]

where \( s \) is a state variable symbol, \( t_i \) is a typed term using the types from the standard PDDL2.1 definition (i.e. in this case, a constant or variable that can be replaced with a constant that is of a type that is not defined with the motion planning extension) and \( D \) is a domain of possible values the motion planning state variable can have. One important part about these motion planning state variables is that an initial value has to be provided for all grounded instances (as opposed to in PDDL2.1 where a resource is assumed to not be used when an initial value is not provided by some task planning systems).

There are three predefined domains and one domain for each configuration or work space that are defined. The predefined domains are: path-planner, the names of all the path planners that have been defined; functions, all real numbers; and bool, a Boolean value represented by 0 and 1. The configuration and work spaces make up the rest of the types and their domains are specified in the definition of the spaces.

The final part of the motion planning state variables in the PDDL2.1 domain file are the possibility to define default values for the motion planning state variables. These constructs can be used to decrease the size of the PDDL2.1 problem file when many motion planning state variables are initialised to the same value.

The following is an example of a subset of the motion planning state variables that were used in the typical scenarios and their default values:
C.5. Operator Extension

The last new constructs in the PDDL2.1 domain file are the extensions to the durative-action construct. These constructs are:

- The duration override and heuristics that specify how the duration is calculated and how an estimate of the duration is calculated using the motion planning state variables.

- The motion conditions that specify conditions in the same way as the standard PDDL2.1 conditions. However, these conditions use the motion planning state variables.

- The motion effects specify all effects the operator has on the motion planning state variables. This is done in the same way as the standard PDDL2.1 effects for resources.

In the following grammar, `<header>` refers to everything up to and including the parameters and `<tail>` refers to everything from and including the duration in the standard PDDL2.1 construct for a temporal operator:

```plaintext
(:path-state-var
 (planner ?u - uav) - path-planner
 (configuration ?u - uav) - configuration
 (turn-radius ?u - uav) - function
 (passive-in-air ?u - uav) - bool)
(:path-defaults
 (= (passive-in-air ?u - uav) 0)
 (= (planner ?u - uav) NONE))
```
An example of an operator where some conditions and effects have been removed (from both the motion planning part and the standard PDDL2.1 part) is the following:
C.6 Initial Values

The initial values construct is a new construct in the PDDL2.1 problem file that specify the values the motion planning state variables have in the initial state. Remember from the motion planning state variable description that all motion planning state variables must have an initial value. This means that all grounded motion planning state variables that do not have a default value must be present in this section of the PDDL2.1 problem file. The syntax for the initial values construct is as follows:

```
<motion-init> ::= '(:path-init ' <assignment> <assignments> ' )'
<assignments> ::= <assignment> <assignments> | ε
<assignment> ::= '=(' ('<identifier> <identifiers> ') '<init-val>' <init-val> ::= <identifier> | <number> | <domain>
```

An example of the initialisation (a subset of the initial values used in a planning problem from one of the typical scenarios) are the following:

```
(:path-init
 (= (planner uav1) SAR)
 (= (turn-radius uav1) 47.000000)
 (= (configuration uav1) (30000 1000 0)))
```
C.7 Collision Zones

Collision zones are the final part to the motion planning extension in the PDDL2.1 problem file. These are constructs that specify all zones that the motion planner should treat as objects that the agent may not pass within. The collision zones are defined for a specific planner, which means that different planners have different collision zones. Moreover, they also have an integer, coll-type value, associated with them (specified in the ":coll-type" part) that describes when the motion planner should respect the collision zone. Each query to a motion planner has an integer that means that all the planner’s collision zones that has a larger or equal coll-type value should be taken into consideration. The rest of the parameters to the collision zones make up a description of the geometrical object that is the collision zone.

The following is the syntax for defining collision zones:

```
<motion-collision> ::= '(: path-collision -zones' <zones> ')'
<zones> ::= <zone> <zones> | ε
<zone> ::= '(: zone' ': planner' <identifier> ': coll-type' <number> ': type' <geo-object> ')
<geo-object> ::= <circle> | <square>
<circle> ::= 'circle' ':x-coordinate' <number> ':y-coordinate' <number> ': radius' <number>
<square> ::= 'square' ':x-lower' <number> ':y-lower' <number> ':x-upper' <number> ':y-upper' <number>
```

Two examples of collision zones are as follows (the first is a SAM unit and the second is an unexplored area):

```
(:path-collision-zones
 (:zone
  :planner ED-IR
  :coll-type 5
  :type circle
  :x-coordinate 40000
  :y-coordinate 61000
  :radius 6800)
 (:zone
  :planner ED-IR
  :coll-type 0
  :type square
  :x-lower 30000
  :y-lower 51000
  :x-upper 130000
  :y-upper 71000))
```
Plan Visualisation

The plan visualiser that was implemented in the thesis had the main purpose to make it easier to find bugs in the motion planning part of the system. Hence, the visualiser is not designed to show the combined motion and task plans in an easy to understand manner. Neither does it contain all information in the plan. Some snapshots from the visualiser are shown here to give an insight in the information that can be gained from the visualiser.

There is one thing that should be kept in mind regarding the plans when looking at the snapshots: Plan quality is not a parameter in this thesis and is therefore never considered. That being said, each plan that is found by the system solves the given problem. However, the plan that is found might not be the most efficient or obvious solution.

A result from not considering the plan quality is that most of the plans include many unnecessary steps. Unfortunately, the visualisation for those plans are messy with paths going all over the snapshot and there is sometime a list of unnecessary actions at a location. This makes the visualisation more complex and harder to understand. Fortunately, the simpler plans that was found show the important parts of the visualisation. Hence, the examples in appendix are snapshots of the simple plans.

Snapshot 1: Current problem

Figure D.1 shows a visualisation of how the scenario looks after the third event in the scenario has occurred (as in, the third time new information was encountered). This snapshot does not provide any information about potential bugs but gives a clean overview of the problem. The entities in the snapshot are the following:

- The longitude, latitude and orientation in the longitude-latitude plane of each agent. This information is shown with the coloured arrows that are marked with the name of agents (uav1, uav2 and uav3).
- The base is shown as the blue box in the lower left part of the snapshot.
- Two SAM units that are located at the centre of the red circles. The range of their weapons (at altitude 1.5 km) is shown as the red filled circle and the range of their radar (also at altitude 1.5 km) is shown with the orange circle.
Simulation of scenario 1

Event 3

<table>
<thead>
<tr>
<th>UAV</th>
<th>SAR scanner</th>
<th>EO/IR scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>uav1</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>uav2</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>uav3</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Event Overview

Figure D.1: Snapshot of what the planning problem looks like when the third event occurs.

- The black area is the area that remains to be scanned using the SAR scanner.
- The yellow dot shows a target that has been found.
- The table shows the status of the UAVs’ scanners.

Snapshot 2: SAR Scanning

Figure D.2 shows a visualisation of the planned flight paths and actions at altitude 4.5 km of uav1. There are a couple of new entities in this snapshot that were not in the previous snapshot:

- The yellow circles with a number in them are the waypoints that the UAV doing the SAR scanning should pass between. A UAV has to fly in a straight line between each pair of waypoints \((i, i + 1)\) when \(i\) is odd.
- The white boxes with text in them are actions performed by the UAV at the location marked with a small orange dot in connection to the box.
Snapshots on this form are useful to identify two types bugs (it should be noted that no bugs are present in the shown snapshot): Flight paths within unexplored areas on altitude 4.5 km; and multiple UAVs doing SAR scanning. The first bug is that a flight path passes through the black area before that area has not been scanned with the SAR scanner. This should be avoided since there might be unknown threats within that area. The second bug is due to a modelling restriction, namely that only one UAV should do the scanning with the SAR scanner within one plan. The bug is found if non-empty subsets of the SAR scanning paths are drawn in different snapshot for the same event.
## Evaluation Data

The data that is presented in this appendix are the means and standard deviation for all the configurations and problems.

<table>
<thead>
<tr>
<th>scenario:problem</th>
<th>Configuration 0</th>
<th></th>
<th>Configuration 1</th>
<th></th>
<th>Configuration 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>s1:01</td>
<td>38.99</td>
<td>6.99</td>
<td>43.1</td>
<td>5.78</td>
<td>42.17</td>
<td>5.78</td>
</tr>
<tr>
<td>s1:02</td>
<td>55.76</td>
<td>16.28</td>
<td>44.22</td>
<td>15.75</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s1:03</td>
<td>85.45</td>
<td>26.69</td>
<td>78.52</td>
<td>24.13</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s1:04</td>
<td>63.86</td>
<td>29.5</td>
<td>79.35</td>
<td>37.39</td>
<td>94.72</td>
<td>46.37</td>
</tr>
<tr>
<td>s1:05</td>
<td>53.28</td>
<td>30.9</td>
<td>108.47</td>
<td>39.98</td>
<td>33.85</td>
<td>13.48</td>
</tr>
<tr>
<td>s1:06</td>
<td>25.88</td>
<td>12.69</td>
<td>27.26</td>
<td>7.25</td>
<td>24.74</td>
<td>12.04</td>
</tr>
<tr>
<td>s1:07</td>
<td>1.37</td>
<td>0.83</td>
<td>2.49</td>
<td>1.44</td>
<td>1.02</td>
<td>0.15</td>
</tr>
<tr>
<td>s1:08</td>
<td>0.32</td>
<td>$6.32 \cdot 10^{-2}$</td>
<td>0.32</td>
<td>$1.34 \cdot 10^{-2}$</td>
<td>0.31</td>
<td>$2.45 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>s2:01</td>
<td>57.09</td>
<td>14.19</td>
<td>48.04</td>
<td>10.85</td>
<td>47.74</td>
<td>28.07</td>
</tr>
<tr>
<td>s2:02</td>
<td>52.25</td>
<td>19.32</td>
<td>37.75</td>
<td>5.84</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s2:03</td>
<td>63.77</td>
<td>12.85</td>
<td>79.93</td>
<td>13.29</td>
<td>130.38</td>
<td>42.01</td>
</tr>
<tr>
<td>s2:04</td>
<td>46.7</td>
<td>12.98</td>
<td>57.36</td>
<td>13.8</td>
<td>45.86</td>
<td>15.93</td>
</tr>
<tr>
<td>s2:05</td>
<td>32.19</td>
<td>11.11</td>
<td>30.7</td>
<td>9.49</td>
<td>57.72</td>
<td>10.51</td>
</tr>
<tr>
<td>s2:06</td>
<td>5.9</td>
<td>0.32</td>
<td>6.36</td>
<td>0.19</td>
<td>0.82</td>
<td>$2.21 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>s2:07</td>
<td>0.26</td>
<td>$1.14 \cdot 10^{-2}$</td>
<td>0.26</td>
<td>$7.4 \cdot 10^{-3}$</td>
<td>0.27</td>
<td>$5.83 \cdot 10^{-3}$</td>
</tr>
<tr>
<td>s3:01</td>
<td>39.31</td>
<td>5.26</td>
<td>44.25</td>
<td>6.68</td>
<td>40.45</td>
<td>5.98</td>
</tr>
<tr>
<td>s3:02</td>
<td>44.04</td>
<td>14.81</td>
<td>42.86</td>
<td>13.05</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s3:03</td>
<td>32.92</td>
<td>7.47</td>
<td>44.16</td>
<td>13.38</td>
<td>57.76</td>
<td>27.03</td>
</tr>
<tr>
<td>s3:04</td>
<td>38.91</td>
<td>8.48</td>
<td>45.04</td>
<td>10.89</td>
<td>45.27</td>
<td>15.02</td>
</tr>
<tr>
<td>s3:05</td>
<td>56.18</td>
<td>14.47</td>
<td>70.53</td>
<td>13.54</td>
<td>66.78</td>
<td>14.84</td>
</tr>
<tr>
<td>s3:06</td>
<td>20.74</td>
<td>3.64</td>
<td>21.66</td>
<td>1.49</td>
<td>24.68</td>
<td>8.64</td>
</tr>
<tr>
<td>s3:07</td>
<td>29.08</td>
<td>11.98</td>
<td>33.87</td>
<td>13.54</td>
<td>35.87</td>
<td>18.3</td>
</tr>
<tr>
<td>s3:08</td>
<td>17.96</td>
<td>5.13</td>
<td>18.22</td>
<td>5.3</td>
<td>21.88</td>
<td>4.14</td>
</tr>
<tr>
<td>s3:09</td>
<td>14.88</td>
<td>6.26</td>
<td>30.72</td>
<td>9.42</td>
<td>25.54</td>
<td>2.27</td>
</tr>
<tr>
<td>s3:10</td>
<td>0.66</td>
<td>$2.01 \cdot 10^{-2}$</td>
<td>1.09</td>
<td>$2.35 \cdot 10^{-2}$</td>
<td>0.86</td>
<td>$1.35 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>s3:11</td>
<td>0.35</td>
<td>$6.02 \cdot 10^{-2}$</td>
<td>0.33</td>
<td>$1.05 \cdot 10^{-2}$</td>
<td>0.33</td>
<td>$1.1 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>scenario:problem</td>
<td>Configuration 0</td>
<td></td>
<td>Configuration 1</td>
<td></td>
<td>Configuration 2</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>s4:01</td>
<td>39.06</td>
<td>7.21</td>
<td>46.02</td>
<td>6.28</td>
<td>42.61</td>
<td>8.8</td>
</tr>
<tr>
<td>s4:02</td>
<td>81.74</td>
<td>38.48</td>
<td>45.43</td>
<td>14.4</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:03</td>
<td>41.86</td>
<td>11.64</td>
<td>44.5</td>
<td>11.73</td>
<td>44.93</td>
<td>17.54</td>
</tr>
<tr>
<td>s4:04</td>
<td>81.07</td>
<td>19.28</td>
<td>83.33</td>
<td>22.68</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:05</td>
<td>75.98</td>
<td>18.45</td>
<td>110.09</td>
<td>26.13</td>
<td>113.8</td>
<td>23.46</td>
</tr>
<tr>
<td>s4:06</td>
<td>70.14</td>
<td>16.21</td>
<td>115.41</td>
<td>26.39</td>
<td>91.63</td>
<td>20.67</td>
</tr>
<tr>
<td>s4:07</td>
<td>155.58</td>
<td>37.54</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>100.02</td>
<td>11.34</td>
</tr>
<tr>
<td>s4:08</td>
<td>68.91</td>
<td>14.26</td>
<td>73.49</td>
<td>15.29</td>
<td>76.66</td>
<td>13.11</td>
</tr>
<tr>
<td>s4:09</td>
<td>90.81</td>
<td>37.5</td>
<td>81.16</td>
<td>26.72</td>
<td>65.86</td>
<td>18.19</td>
</tr>
<tr>
<td>s4:10</td>
<td>88.75</td>
<td>22.36</td>
<td>109.85</td>
<td>28.01</td>
<td>94.31</td>
<td>21.62</td>
</tr>
<tr>
<td>s4:11</td>
<td>72.48</td>
<td>24.53</td>
<td>69.67</td>
<td>23.67</td>
<td>131.84</td>
<td>53.52</td>
</tr>
<tr>
<td>s4:12</td>
<td>42.29</td>
<td>11.36</td>
<td>62.02</td>
<td>21.13</td>
<td>52.21</td>
<td>16.63</td>
</tr>
<tr>
<td>s4:13</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:14</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:15</td>
<td>109.85</td>
<td>48.24</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>127.1</td>
<td>46.44</td>
</tr>
<tr>
<td>s4:16</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:17</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>98.62</td>
<td>43.19</td>
</tr>
<tr>
<td>s4:18</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:19</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:20</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:21</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:22</td>
<td>86.72</td>
<td>52.18</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:23</td>
<td>17.77</td>
<td>0.33</td>
<td>1.22</td>
<td>$7.49 \cdot 10^{-2}$</td>
<td>1.07</td>
<td>0.12</td>
</tr>
<tr>
<td>s4:24</td>
<td>0.28</td>
<td>$5.43 \cdot 10^{-2}$</td>
<td>0.27</td>
<td>$5 \cdot 10^{-2}$</td>
<td>0.28</td>
<td>$6.08 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>s5:01</td>
<td>82.62</td>
<td>18.16</td>
<td>66.85</td>
<td>7.75</td>
<td>65.17</td>
<td>8.32</td>
</tr>
<tr>
<td>s5:02</td>
<td>83.58</td>
<td>11.26</td>
<td>79.23</td>
<td>26.02</td>
<td>96.51</td>
<td>20.05</td>
</tr>
<tr>
<td>s5:03</td>
<td>73.63</td>
<td>9.69</td>
<td>82.8</td>
<td>9.96</td>
<td>105.67</td>
<td>30.66</td>
</tr>
<tr>
<td>s5:04</td>
<td>73.27</td>
<td>13.86</td>
<td>79.11</td>
<td>17.35</td>
<td>74.96</td>
<td>20.22</td>
</tr>
<tr>
<td>s5:05</td>
<td>116.77</td>
<td>16.2</td>
<td>138.48</td>
<td>32.51</td>
<td>108.3</td>
<td>26.93</td>
</tr>
<tr>
<td>s5:06</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:07</td>
<td>139.88</td>
<td>26.96</td>
<td>173.5</td>
<td>35.45</td>
<td>121.13</td>
<td>28.36</td>
</tr>
<tr>
<td>s5:08</td>
<td>135.3</td>
<td>58.25</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:09</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:10</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:11</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:12</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>103.34</td>
<td>55.75</td>
</tr>
<tr>
<td>s5:13</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:14</td>
<td>83.88</td>
<td>50.08</td>
<td>81.77</td>
<td>32.33</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:15</td>
<td>27.66</td>
<td>6.47</td>
<td>13.71</td>
<td>7.38</td>
<td>1.04</td>
<td>$3.55 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>s5:16</td>
<td>0.28</td>
<td>$4.48 \cdot 10^{-2}$</td>
<td>0.27</td>
<td>$1.21 \cdot 10^{-2}$</td>
<td>0.3</td>
<td>$8.16 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>scenario/problem</td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>s1:01</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>43.68</td>
</tr>
<tr>
<td>s1:02</td>
<td>45.98</td>
<td>12.68</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>45.6</td>
</tr>
<tr>
<td>s1:03</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>79.13</td>
</tr>
<tr>
<td>s1:04</td>
<td>95.78</td>
<td>41.62</td>
<td>126.49</td>
<td>41.88</td>
<td></td>
<td>86.78</td>
</tr>
<tr>
<td>s1:05</td>
<td>134.42</td>
<td>54.25</td>
<td>101.27</td>
<td>44.45</td>
<td></td>
<td>105.82</td>
</tr>
<tr>
<td>s1:06</td>
<td>35.41</td>
<td>13.76</td>
<td>35.01</td>
<td>12.25</td>
<td></td>
<td>29.39</td>
</tr>
<tr>
<td>s1:07</td>
<td>4.66</td>
<td>0.71</td>
<td>3.92</td>
<td>1.02</td>
<td></td>
<td>4.42</td>
</tr>
<tr>
<td>s1:08</td>
<td>0.31</td>
<td>$1.85 \cdot 10^{-2}$</td>
<td>0.33</td>
<td>$2.32 \cdot 10^{-2}$</td>
<td></td>
<td>0.31</td>
</tr>
<tr>
<td>s2:01</td>
<td>55.71</td>
<td>23.65</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>56.7</td>
</tr>
<tr>
<td>s2:02</td>
<td>204.09</td>
<td>4.56</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>37.46</td>
</tr>
<tr>
<td>s2:03</td>
<td>83.11</td>
<td>17.36</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>76.14</td>
</tr>
<tr>
<td>s2:04</td>
<td>234.65</td>
<td>27.86</td>
<td>78.41</td>
<td>11.76</td>
<td></td>
<td>62.36</td>
</tr>
<tr>
<td>s2:05</td>
<td>47.01</td>
<td>11.57</td>
<td>34.78</td>
<td>11.93</td>
<td></td>
<td>33.13</td>
</tr>
<tr>
<td>s2:06</td>
<td>44.76</td>
<td>1.38</td>
<td>0.94</td>
<td>0.14</td>
<td></td>
<td>6.36</td>
</tr>
<tr>
<td>s2:07</td>
<td>0.27</td>
<td>$3.47 \cdot 10^{-2}$</td>
<td>0.27</td>
<td>$6.64 \cdot 10^{-3}$</td>
<td></td>
<td>0.26</td>
</tr>
<tr>
<td>s3:01</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>44.3</td>
</tr>
<tr>
<td>s3:02</td>
<td>39.71</td>
<td>7.47</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>45.07</td>
</tr>
<tr>
<td>s3:03</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>40.68</td>
</tr>
<tr>
<td>s3:04</td>
<td>39.93</td>
<td>11.78</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>41.02</td>
</tr>
<tr>
<td>s3:05</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>67.95</td>
</tr>
<tr>
<td>s3:06</td>
<td>15.97</td>
<td>3.56</td>
<td>84.95</td>
<td>5.55</td>
<td></td>
<td>21.96</td>
</tr>
<tr>
<td>s3:07</td>
<td>32.85</td>
<td>11.41</td>
<td>60.53</td>
<td>14.53</td>
<td></td>
<td>35.71</td>
</tr>
<tr>
<td>s3:08</td>
<td>16.65</td>
<td>5.75</td>
<td>28.53</td>
<td>4.17</td>
<td></td>
<td>18.71</td>
</tr>
<tr>
<td>s3:09</td>
<td>29.64</td>
<td>5.81</td>
<td>19.89</td>
<td>9.15</td>
<td></td>
<td>29.33</td>
</tr>
<tr>
<td>s3:10</td>
<td>1.06</td>
<td>$2.26 \cdot 10^{-2}$</td>
<td>0.92</td>
<td>$1.7 \cdot 10^{-2}$</td>
<td></td>
<td>1.12</td>
</tr>
<tr>
<td>s3:11</td>
<td>0.33</td>
<td>$2.18 \cdot 10^{-2}$</td>
<td>0.35</td>
<td>$3.96 \cdot 10^{-2}$</td>
<td></td>
<td>0.35</td>
</tr>
<tr>
<td>s4:01</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>45.62</td>
</tr>
<tr>
<td>s4:02</td>
<td>52.41</td>
<td>22.64</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>45.68</td>
</tr>
<tr>
<td>s4:03</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>57.1</td>
</tr>
<tr>
<td>s4:04</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>87.17</td>
</tr>
<tr>
<td>s4:05</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>115.21</td>
</tr>
<tr>
<td>s4:06</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>109.15</td>
</tr>
<tr>
<td>s4:07</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>169.35</td>
</tr>
<tr>
<td>s4:08</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>205.09</td>
<td>21.33</td>
<td></td>
<td>76.13</td>
</tr>
<tr>
<td>s4:09</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>85.43</td>
</tr>
<tr>
<td>s4:10</td>
<td>123.44</td>
<td>35.59</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>111.36</td>
</tr>
<tr>
<td>s4:11</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>81.04</td>
</tr>
<tr>
<td>s4:12</td>
<td>164.01</td>
<td>17.96</td>
<td>84.66</td>
<td>15.47</td>
<td></td>
<td>61.98</td>
</tr>
<tr>
<td>s4:13</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:14</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:15</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:16</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:17</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>104.15</td>
<td>42.88</td>
<td></td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:18</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:19</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:20</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:21</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:22</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td>$\infty$</td>
</tr>
<tr>
<td>s4:23</td>
<td>1.15</td>
<td>$4.67 \cdot 10^{-2}$</td>
<td>1.16</td>
<td>0.14</td>
<td></td>
<td>1.22</td>
</tr>
<tr>
<td>scenario:problem</td>
<td>Configuration 3</td>
<td>Configuration 4</td>
<td>Configuration 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>s4:24</td>
<td>0.28</td>
<td>$6.39 \cdot 10^{-2}$</td>
<td>0.28</td>
<td>$4.74 \cdot 10^{-2}$</td>
<td>0.29</td>
<td>$5.46 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>s5:01</td>
<td>69.38</td>
<td>10.18</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>68.54</td>
<td>8.73</td>
</tr>
<tr>
<td>s5:02</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>72.05</td>
<td>26.33</td>
</tr>
<tr>
<td>s5:03</td>
<td>95.95</td>
<td>21.03</td>
<td>67.91</td>
<td>10.26</td>
<td>83.03</td>
<td>11.97</td>
</tr>
<tr>
<td>s5:04</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>78.28</td>
<td>17.84</td>
</tr>
<tr>
<td>s5:05</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>139.12</td>
<td>35.43</td>
</tr>
<tr>
<td>s5:06</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:07</td>
<td>188.7</td>
<td>48.72</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>173.42</td>
<td>39.06</td>
</tr>
<tr>
<td>s5:08</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:09</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:10</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:11</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:12</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:13</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:14</td>
<td>119.27</td>
<td>51.4</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:15</td>
<td>12.28</td>
<td>5.19</td>
<td>1.01</td>
<td>$6.98 \cdot 10^{-2}$</td>
<td>12.51</td>
<td>6.89</td>
</tr>
<tr>
<td>s5:16</td>
<td>0.28</td>
<td>$5.06 \cdot 10^{-2}$</td>
<td>0.27</td>
<td>$4.89 \cdot 10^{-2}$</td>
<td>0.26</td>
<td>$7.96 \cdot 10^{-3}$</td>
</tr>
<tr>
<td>scenario:problem</td>
<td>Configuration 6</td>
<td>Configuration 7</td>
<td>Configuration 8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:01</td>
<td>39.71</td>
<td>5.47</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:02</td>
<td>53.06</td>
<td>17.91</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:03</td>
<td>\infty</td>
<td>\infty</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:04</td>
<td>85.46</td>
<td>32.3</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:05</td>
<td>31.88</td>
<td>9.18</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:06</td>
<td>20.39</td>
<td>3.4</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:07</td>
<td>5.97</td>
<td>1.45</td>
<td>1.85</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:08</td>
<td>0.32</td>
<td>2.47 \cdot 10^{-2}</td>
<td>0.31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:01</td>
<td>49.32</td>
<td>14.41</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:02</td>
<td>\infty</td>
<td>\infty</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:03</td>
<td>172.35</td>
<td>32.14</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:04</td>
<td>87.16</td>
<td>46.28</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:05</td>
<td>68.35</td>
<td>8.95</td>
<td>7.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:06</td>
<td>0.8</td>
<td>3.21 \cdot 10^{-2}</td>
<td>0.26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:07</td>
<td>0.27</td>
<td>5.42 \cdot 10^{-2}</td>
<td>1.12 \cdot 10^{-2}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:01</td>
<td>42.94</td>
<td>7.16</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:02</td>
<td>61.54</td>
<td>47.54</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:03</td>
<td>58.96</td>
<td>18.83</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:04</td>
<td>40.4</td>
<td>8.84</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:05</td>
<td>71.94</td>
<td>14.9</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:06</td>
<td>23.2</td>
<td>1.56</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:07</td>
<td>45.33</td>
<td>17.07</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:08</td>
<td>20.5</td>
<td>4.57</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:09</td>
<td>46.92</td>
<td>14.6</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:10</td>
<td>0.87</td>
<td>1.54 \cdot 10^{-2}</td>
<td>2.35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:11</td>
<td>0.36</td>
<td>7.4 \cdot 10^{-2}</td>
<td>1.68 \cdot 10^{-2}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:01</td>
<td>41.62</td>
<td>5.32</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:02</td>
<td>56.21</td>
<td>20.3</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:03</td>
<td>46.86</td>
<td>18.28</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:04</td>
<td>102.46</td>
<td>17.45</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:05</td>
<td>86.15</td>
<td>36.42</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:06</td>
<td>86.48</td>
<td>16.7</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:07</td>
<td>104.78</td>
<td>15</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:08</td>
<td>71.97</td>
<td>13.87</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:09</td>
<td>58.27</td>
<td>10.54</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:10</td>
<td>96.6</td>
<td>24.75</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:11</td>
<td>75.91</td>
<td>21.76</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:12</td>
<td>51.41</td>
<td>22.17</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:13</td>
<td>\infty</td>
<td>\infty</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:14</td>
<td>\infty</td>
<td>\infty</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:15</td>
<td>118.07</td>
<td>30.6</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:16</td>
<td>\infty</td>
<td>\infty</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:17</td>
<td>78.22</td>
<td>18.92</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:18</td>
<td>146.28</td>
<td>53.31</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:19</td>
<td>\infty</td>
<td>\infty</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:20</td>
<td>\infty</td>
<td>\infty</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:21</td>
<td>\infty</td>
<td>\infty</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:22</td>
<td>\infty</td>
<td>\infty</td>
<td>\infty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:23</td>
<td>1.12</td>
<td>6.65 \cdot 10^{-2}</td>
<td>153.71</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>23.13</td>
<td>\infty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scenario:problem</td>
<td>Configuration 6</td>
<td>Configuration 7</td>
<td>Configuration 8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>s4:24</td>
<td>0.29 $\cdot 10^{-2}$</td>
<td>6.01 $\cdot 10^{-2}$</td>
<td>0.29 $\cdot 10^{-2}$</td>
<td>5.66 $\cdot 10^{-2}$</td>
<td>0.27 $\cdot 10^{-2}$</td>
<td></td>
</tr>
<tr>
<td>s5:01</td>
<td>64.94</td>
<td>12.84</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:02</td>
<td>98.24</td>
<td>15.5</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:03</td>
<td>90.78</td>
<td>15.44</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:04</td>
<td>77.91</td>
<td>16.93</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:05</td>
<td>129.12</td>
<td>24.96</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:06</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:07</td>
<td>121.86</td>
<td>26.93</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:08</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:09</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:10</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:11</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:12</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:13</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:14</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:15</td>
<td>1.02</td>
<td>1.1 $\cdot 10^{-2}$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>93.76</td>
<td>40.77</td>
</tr>
<tr>
<td>s5:16</td>
<td>0.26</td>
<td>1.04 $\cdot 10^{-2}$</td>
<td>0.29 $\cdot 10^{-2}$</td>
<td>6.4 $\cdot 10^{-2}$</td>
<td>0.27 $\cdot 10^{-2}$</td>
<td>2.52 $\cdot 10^{-2}$</td>
</tr>
<tr>
<td>scenario:problem</td>
<td>Configuration 9</td>
<td></td>
<td>Configuration 10</td>
<td></td>
<td>Configuration 11</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>s1:01</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s1:02</td>
<td>∞</td>
<td>∞</td>
<td>s1:02</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s1:03</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s1:04</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>s1:04</td>
<td>∞</td>
</tr>
<tr>
<td>s1:05</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s1:06</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s1:07</td>
<td>2.43</td>
<td>0.99</td>
<td>∞</td>
<td>11.7</td>
<td>2.14</td>
<td>0.93</td>
</tr>
<tr>
<td>s1:08</td>
<td>0.32</td>
<td>6.32 · 10^{-2}</td>
<td>0.32</td>
<td>1.04 · 10^{-2}</td>
<td>0.32</td>
<td>2.11 · 10^{-2}</td>
</tr>
<tr>
<td>s2:01</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s2:02</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s2:03</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s2:04</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s2:05</td>
<td>∞</td>
<td>28</td>
<td>∞</td>
<td>∞</td>
<td>2.91</td>
<td>∞</td>
</tr>
<tr>
<td>s2:06</td>
<td>13.77</td>
<td>9.02 · 10^{-2}</td>
<td>2.22</td>
<td>2.04 · 10^{-2}</td>
<td>10.58</td>
<td>0.5</td>
</tr>
<tr>
<td>s2:07</td>
<td>0.26</td>
<td>1.21 · 10^{-2}</td>
<td>0.27</td>
<td>4.93 · 10^{-2}</td>
<td>0.26</td>
<td>6.72 · 10^{-3}</td>
</tr>
<tr>
<td>s3:01</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s3:02</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s3:03</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s3:04</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s3:05</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s3:06</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s3:07</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s3:08</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s3:09</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s3:10</td>
<td>2.17</td>
<td>6.11 · 10^{-2}</td>
<td>1.98</td>
<td>6.81 · 10^{-2}</td>
<td>2.18</td>
<td>6.04 · 10^{-2}</td>
</tr>
<tr>
<td>s3:11</td>
<td>0.34</td>
<td>2.35 · 10^{-2}</td>
<td>0.33</td>
<td>2.19 · 10^{-2}</td>
<td>0.34</td>
<td>5.05 · 10^{-2}</td>
</tr>
<tr>
<td>s4:01</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:02</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:03</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:04</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:05</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:06</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:07</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:08</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:09</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:10</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:11</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:12</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:13</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:14</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:15</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:16</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:17</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:18</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:19</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:20</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:21</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>s4:22</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>scenario:problem</td>
<td>Configuration 9</td>
<td>Configuration 10</td>
<td>Configuration 11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>------------------</td>
<td>------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>s4:24</td>
<td>0.28</td>
<td>$7.24 \cdot 10^{-2}$</td>
<td>0.27</td>
<td>$7.97 \cdot 10^{-3}$</td>
<td>0.28</td>
<td>$5.26 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>s5:01</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:02</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:03</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:04</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:05</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:06</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:07</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:08</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:09</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:10</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:11</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:12</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:13</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:14</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>s5:15</td>
<td>11.83</td>
<td>2.15</td>
<td>53.03</td>
<td>11.43</td>
<td>13.08</td>
<td>2.89</td>
</tr>
<tr>
<td>s5:16</td>
<td>0.28</td>
<td>$5.96 \cdot 10^{-2}$</td>
<td>0.29</td>
<td>$6.07 \cdot 10^{-2}$</td>
<td>0.28</td>
<td>$4.1 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>scenario:problem</td>
<td>Configuration 12</td>
<td>Configuration 13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:01</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:02</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:03</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:04</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:05</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:06</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:07</td>
<td>$2.62$</td>
<td>$2.19$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1:08</td>
<td>$0.32$</td>
<td>$1.95 \cdot 10^{-2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:01</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:02</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:03</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:04</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:05</td>
<td>$41.84$</td>
<td>$7.98$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:06</td>
<td>$10.65$</td>
<td>$0.21$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2:07</td>
<td>$0.26$</td>
<td>$8.55 \cdot 10^{-3}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:01</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:02</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:03</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:04</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:05</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:06</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:07</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:08</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:09</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:10</td>
<td>$1.87$</td>
<td>$2.92 \cdot 10^{-2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3:11</td>
<td>$0.35$</td>
<td>$7.01 \cdot 10^{-2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:01</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:02</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:03</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:04</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:05</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:06</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:07</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:08</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:09</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:10</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:11</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:12</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:13</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:14</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:15</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:16</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:17</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:18</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:19</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:20</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:21</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:22</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:23</td>
<td>$\infty$</td>
<td>$27.03$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scenario:problem</td>
<td>Configuration 12</td>
<td>Configuration 13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td>$\mu$</td>
<td>$\sigma$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4:24</td>
<td>0.26</td>
<td>$7.06 \cdot 10^{-3}$</td>
<td>0.65</td>
<td>$7.53 \cdot 10^{-2}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:01</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:02</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:03</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:04</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:05</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:06</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:07</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:08</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:09</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:10</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:11</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:12</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:13</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:14</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:15</td>
<td>35.71</td>
<td>9.52</td>
<td>34.13</td>
<td>21.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5:16</td>
<td>0.28</td>
<td>$5.4 \cdot 10^{-2}$</td>
<td>0.54</td>
<td>0.12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bibliography


