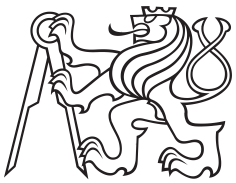
Bachelor Project



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

RRT-Based Solver for Classical Planning Problems

Marie Geislerová

Supervisor: Ing. Daniel Fišer, Ph.D.
August 2021

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Geislerová Marie** Personal ID number: **478042**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Branch of study: **Computer and Information Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

RRT-Based Solver for Classical Planning Problems

Bachelor's thesis title in Czech:

Plánovač pro klasické plánování postavený na RRT

Guidelines:

The goal of the thesis is to propose and implement a solver of classical planning problems by adapting some of the techniques used in the sampling-based methods for solving motion planning problems, primarly Rapidly-Exploring Random Trees.
The student should:
1) Study the literature related to classical planning and motion planning, in particular heuristic search methods and inference of state invariants for classical planning, and sampling-based methods for motion planning.
2) Propose how to use the RRT algorithm (or other similar sampling-based algorithm) to solve classical planning problems.
3) Implement the solution in C, experimentlly evalute it on the standard benchmark set, and compare the results to the state-of-the-art solvers.

Bibliography / sources:

[1] Steven M. Lavalle , James J. Kuffner , Jr. 2000. Rapidly-Exploring Random Trees: Progress and Prospects. In Proceedings of Algorithmic and Computational Robotics: New Directions, pp. 293-308, 2000
[2] Vidal Alcázar, Manuela M. Veloso, Daniel Borrajo. 2011. Adapting a Rapidly-Exploring Random Tree for Automated Planning. In Proc. SOCS'11
[3] Vidal Alcázar, Susana Fernández, Daniel Borrajo, Manuela M. Veloso. 2015. Using random sampling trees for automated planning. AI Commun. 28(4): 665-681, 2015
[4] Patrik Haslum. 2009. h^m(P) = h^1(P^m): Alternative Characterisations of the Generalisation From h^max To h^m. In Proc. AAAI'09

Name and workplace of bachelor's thesis supervisor:

Ing. Daniel Fišer, Department of Computer Science, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **10.01.2020** Deadline for bachelor thesis submission: **13.08.2021**

Assignment valid until: **30.09.2021**

Ing. Daniel Fišer	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

Acknowledgements

I would like to thank my supervisor Ing. Daniel Fišer, Ph.D. for all his invaluable help throughout the writing of this thesis. I would also like to thank my family and friends for their patience and support.

Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 13 August 2021

Abstract

Problems of classical planning are usually solved using the algorithms of forward search with heuristic. Although the search is usually able to achieve the desired results, in some cases the problem can have large plateaus where all states have the same heuristic value and it is difficult to choose the best direction. In motion planning similar problems can be resolved by introducing randomization.

This thesis deals with exploring the possibility of adapting and studying the Rapidly-exploring Random Trees (RRT) algorithm, which was designed for motion planning in continuous space, to classical planning.

Keywords: planning, RRT, Rapidly-exploring Random Tree

Supervisor: Ing. Daniel Fišer, Ph.D.

Abstrakt

Problémy klasického plánování se obvykle řeší pomocí algoritmů dopředného prohledávání (forward search) s heuristikou. Přestože obvykle dosahují požadovaných výsledků, v některých případech může problém obsahovat velké oblasti, kde všechny stavy mají stejnou hodnotu heuristiky a je složité zvolit nejlepší směr. Při plánování pohybu roborů mohou být podobné problémy řešeny randomizací.

Tato práce se zabývá zkoumáním možnosti adaptováním a studiem algoritmu Rapidly-exploring Random Trees (RRT), který byl navržen pro plánování ve spojitém prostoru, pro klasické prohledávání v diskrétním prostoru.

Klíčová slova: plánování, RRT, Rychle rostoucí náhodný strom

Překlad názvu: Plánovač pro klasické plánování postavený na RRT

Contents

Figures

Tables

Chapter 1

Introduction

The usual way of solving problems of classical planning is using forward search accompanied by heuristic function, which should help the planner reach the goal more quickly. Moreover, when using admissible heuristic, algorithms such as A* are guaranteed to find the optimal path.

Nevertheless, greedy searches have issues choosing the best path when dealing with problems with large plateaus which consist of states with the same heuristic value.

The reason to explore the possibility of using RRT in classical planning is because RRTs are one of the most successful state-of-the-art techniques in motion planning. The possible use of the algorithm in classical planning is being explored because of its useful properties - mainly for the randomization used in sampling of the state space, which can help to prevent getting stuck in a plateau, as opposed to sampling using greedy search.

On the other hand, uniform sampling over all possible states is not as straightforward as in motion planning, since random sampling could return states that might be unreachable either from the initial state or towards a goal. This issue could be overcome by sampling only over the subset of goal states, but that would devalue the idea of uniform sampling.

The goal of the thesis is to propose and implement a solver of classical planning problems based on the techniques used in the sampling-based methods for solving motion planning problems. We will be focusing mainly on the RRT

algorithm. In this thesis we used the algorithm outline proposed in [AVB11] and suggested ways to adapt and implement the individual subprocedures to classical planning and then examined the performance of the implementation.

Chapter 2

Background

2.1 Classical planning

Classical planning is a special case of restricted automated planning. A classical planning domain (or a state-transition system) is defined in [GNT16] as a tuple $\sum = (S, A, \gamma, cost)$, where

- S is a finite set of states.
- A is a finite set of actions.
- $\gamma : S \times \mathrm{A} \rightarrow S$ is a state-transition function. If $\gamma(s, a)$ is defined, then a is applicable in s, with $\gamma(s, a)$ being the predicted outcome. Otherwise a is inapplicable in s.
- cost: $S \times \mathrm{A} \rightarrow [0, \infty)$ is a cost function which assignes a value to each action. In case the cost function isn't specified, then $\mathrm{cost}(s, a) = 1$ whenever $\gamma(s, a)$ is defined.

In addition to that, classical planning domain is limited by a set of restrictive classical planning assumptions:

- Finite, static environment. Changes happen only as a response to actions.

- No explicit time, no concurrency. We are working with a discrete sequence of states and actions and we do not consider time at all.
- Determinism, no uncertainty. We are able to predict the result of action a in a state s.

2.1.1 Finite domain representation

A finite domain representation [Hel09] (FDR), also known as multi-valued planning task or SAS+, can be described as a tuple $P = \langle V, O, s_{init}, s_{goal}, c \rangle$, where:

- V is a finite set of state variables. Each variable $v \in V$ has a finite domain D_v.

  A partial state s is a partial variable assignment over V. A state is a partial state assigned over all variables $v \in V$.
- O is a set of operators. Operator $o \in O$ is a pair $o = \langle \text{pre}(o), \text{eff}(o) \rangle$, where precondition $\text{pre}(o)$ and effect $\text{eff}(o)$ are both partial states. We require that $V = v$ cannot be both a precondition and an effect.
- s_{init} is the initial state of the task.
- s_{goal} is a partial states which describes its goal.
- c is a cost function mapping each operator to a non-negative real number.

A partial state s is consistent with a partial state s' if each variable defined in s' was assigned the same value as the corresponding variable in s.

An operator o is applicable in a state s if all the values in $pre(o)$ are equal to values assigned to variables in s. The resulting state of applying an applicable operator o in the state s is the state $s' = \text{res}(o, s)$. If a variable is defined in $\text{eff}(o)$, then the corresponding variable in s' is assigned the same values as the variable in $\text{eff}(o)$. Otherwise, the variable is assigned a value from s.

A sequence of operators $\pi = \langle o_1, ..., o_n \rangle$ is applicable in a state s_0 if there are states $s_1, ..., s_n$ where every operator o_i is applicable in s_{i-1} and $s_i = \text{res}(o_i, s_{i-1})$ for $i \in [1, n]$. The result of this action is $\text{res}(\pi, s_0) = s_n$.

A sequence of operators π is called a plan if $s = \text{res}(\pi, s_{init})$ is consistent with s_{goal}. The cost of a plan is $c(\pi) = \sum_{o \in \pi} c(o)$. Optimal plan is a plan with the minimal cost over all plans.

2.2 Relaxed heuristics

The relaxed heuristic functions [BG01] is obtained by considering a relaxed problem in STRIPS [FN71] representation where all delete effects are ignored. Even though we are not focusing on STRIPS in this thesis, we can modify FDR and instead of having a variable and its value, we combine them into one fact which is either true or false. We can obtain the relaxed plan by assuming that once a fact is true, it will always be true.

The optimal cost for solving the relaxed problem is a lower bound of the optimal cost of the original problem and can be used estimation of the cost of achieving the goal state.

The cost of achieving a fact p from the state s (denoted as $g_s(p)$) is defined as

$$g_s(p) = \begin{cases} 0, & \text{if } p \in s, \\ \min_{o \in O(p)}[1 + g_s(pre(o))], & \text{otherwise.} \end{cases} \tag{2.1}$$

where $O(p)$ stands for the operators o that add p, $g_s(pre(o))$ is estimated cost of the set of facts given by the preconditions of operator o.

The cost of sets of atoms can be defined as the weighted sum of costs of individual atoms is the additive heuristic h_{add}. The cost acquired by combining the cost of atoms by the max operation is the max heuristic h_{max}. However, the h_{FF} heuristics takes different approach. First, it finds a state which is consistent with the goal partial state in the relaxed plan, then it uses supporters (actions which cause a fluent to be true) to compute its value from the goal to the initial state.

2.3 Forward search

Forward search [GNT16] is an algorithm that represents a large number of algorithms which start their search from the initial state and head towards the goal state.

The search is described in Algorithm 1, where *Frontier* is set of nodes waiting to be visited and *Expanded* is a set of already visited nodes. A node is a pair $\nu = (\pi, s)$, where π is a plan and $s = \gamma(s_0, \pi)$. The initial node is $(\langle\rangle, s_0)$.

At first the initial node $(\langle\rangle, s_0)$ is inserted into *Frontier* and *Expanded* is set to be an empty set. In each loop the algorithm selects a node $\nu = (\pi, s)$, removes it from *Frontier* and inserts it into *Expanded*, generates its *Children*, prunes unpromising nodes and inserts *Children* into the *Frontier*.

Algorithm 1: Forward search from [GNT16]

Data: $\sum$, s_0, g
$Frontier \leftarrow \{(\langle\rangle, s_0)\}$;
$Expanded \leftarrow \emptyset$;
while $Frontier \neq \emptyset$ **do**
  select a node $\nu = (\pi, s) \in Frontier$;
  remove ν from *Frontier* and add to *Expanded*;
  if s *satisfies* g **then**
    return π;
  end
  $Children \leftarrow \{(\pi, a, \gamma(s, a)) \mid s \text{ satisfies pre}(a)\}$;
  prune 0 or more nodes from *Children*, *Frontier* and *Expanded*;
  $Frontier \leftarrow Frontier \cup Children$;
end
return failure;

2.3.1 Greedy Best-First Search

Greedy best-first search [GNT16] is the most frequently chosen algorithm for classical planning problems which do not require optimal solution.

It is a forward search, where a node selection is specified as the selection

of a node from the $Frontier$ with the minimal heuristic value. Pruning work as follows: For each node $\nu = (\pi, s) \in Children$, if there are more than one plan that goes to s, keep the one with minimal cost and remove the others.

2.4 Mutex

Mutex, mutex groups and fact-alternating mutex groups are defined in [FK18] as follows:

Mutex $M \subseteq F$ is a set of facts such that for every reachable state $s \in R$ it holds that $M \nsubseteq s$.

A mutex group $M \subseteq F$ is a set of facts such that for every reachable state $s \in R$ it holds that $\mid M \cap s \mid \leq 1$.

A fact-alternating mutex group (fam-group) $M \subseteq F$ is a set of facts such that $\mid M \cap s_{init} \mid \leq 1$ and $\mid M \cap \text{add}(o) \mid \leq \mid M \cap \text{pre}(o) \cap \text{del}(o) \mid$ for every operator $o \in O$.

As mentioned in [AT15], one of the methods to obtain mutexes is using the h^m heuristic [BG01], where h^m performs a reachability analysis in P^m [Has09]. P^m is a semi-relaxed version of the original problem in which atoms are sets of m fluents. If the value of h^{max} of an atom in P^m is infinite, then the atom is a mutex of size m.

Algorithm 2 shows us a way to infer fam-groups using integer linear program

[FK18].

Algorithm 2: Inference of fact-alternating mutex groups using ILP (from [FK18])

Input: Planning task $\Pi = \langle F, O, s_{init}, s_{goal} \rangle$
Output: A set of fam-groups M
Initialize ILP with constraints according to Equation (2.2) and Equation (2.3);
Set objective function of ILP to maximize $\sum_{f_i \in F} x_i$;
Solve ILP and save the resulting fam-group into M;
while | M |≥ 1 **do**
  Add M to the output set M;
  Add constraint according to Equation (2.4) using M;
  M $\leftarrow \emptyset$;
  Solve ILP and if a solution was found, save the resulting fam-group into M;
end

$$\sum_{f_i \in s_{init}} x_i \leq 1. \tag{2.2}$$

$$\forall o \in O : \sum_{f_i \in \text{add}(o)} x_i \leq \sum_{f_i \in \text{del}(o) \cap \text{pre}(o)} x_i. \tag{2.3}$$

$$\sum_{f_i \notin s_{init}} x_i \geq 1. \tag{2.4}$$

The most frequently used method to find invariants is h^2. Apart from this method, in this thesis we will use the forward and backward computation of h^2 and also h^3 and fam-groups.

2.5 Motion planning

Motion planning [LaV06] is a type of planning that deals with motions of a robot in a configuration space with obstacles. A plan in motion planning determines the configuration of the robot so that it reaches a goal state and avoids colliding with obstacles.

2.6 Rapidly Exploring Random Trees

The rapidly exploring random tree (RRT) [L+98] is an algorithm successfully used for exploring the continuous space. It is popular for its preference to expand towards unsearched parts of the search space and yet still being simple to implement.

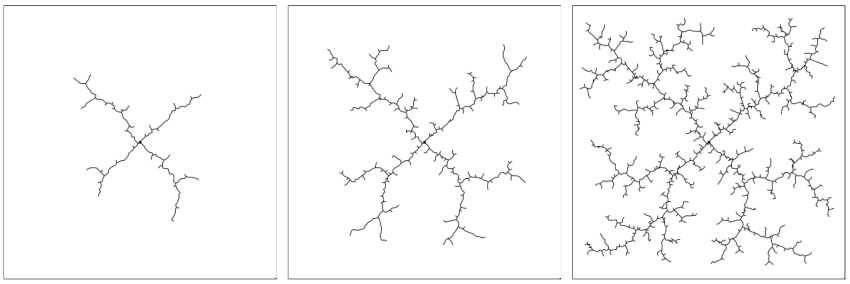


Figure 2.1: Rapidly-Exploring Random Tree (picture from [L+98])

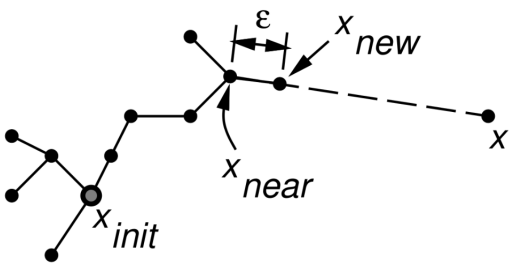


Figure 2.2: The extend phase of RRT algorithm (picture from [LKD+01])

At first, the tree is initialized and a random state is selected from the domain. Sampled state is assigned its nearest neighbor, which is a state already present in the tree. Then an input is selected in a way that minimizes the distance from the sampled state to the nearest neighbour and avoids colliding with obstacles. The algorithm attempts to join these states while following the path of input. If the distance to sampled state is within given limit, the state is added to the tree. If not, the algorithm adds a new state, that follows the same direction from the nearest neighbour and its distance is

equal to the limit.

Algorithm 3: Generate RRT (from [L+98])

Data: $x_{init}, K, \Delta t$
T.init(x_{init});
for $k = 1$ *to* K **do**
  $x_{rand} \leftarrow$ RANDOM_STATE();
  $x_{near} \leftarrow$ NEAREST_NEIGHBOUR(x_{rand}, T);
  $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
  $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
  T.add_vertex(x_{new});
  T.add_edge(x_{near}, x_{new}, u);
end
return T

Chapter 3

Related Work

This section introduces us to several papers on a similar topic and shows us how the authors dealt with issues of adapting algorithms used in motion planning to classical planning.

3.1 Sampling-Based Planning for Discrete Spaces

In this paper [MB04] we are introduced to discrete space search algorithms based on motion-planning techniques such as Rapidly-exploring Random Trees and Probabilistic Roadmaps [KSLO96] to discrete space. We will be focusing on the RRT algorithms.

3.1.1 Discrete RRTs

The way the discrete algorithm determines the nearest state is replacing the distance metric with heuristic estimate of the cost-to-go that is used in general informed search methods.

The algorithm starts with an initial state q_{start}. At each step selects a random state q_{rand} which is not present in the tree and find a nearest state q_{near} based on a heuristic estimate of the cost-to-go from each state to q_{rand}.

Every operator is applied on q_{near} and the state which is closest to q_{rand} and is not present in the tree becomes q_{new}. The state q_{new} is added to the tree connect to q_{near} with an edge.

The paper also mentions a variation of the algorithm using Rapidly-Exploring Random Leafy Tree, which keeps an open list of all states reachable in one step from the tree. Therefore, instead of considering successors of only one state, we are able to use the successors of the whole tree.

Algorithm 4: GrowRRT (from [MB04])

```
Data: q_start
T.init(q_start);
for n = 1 to N do
    q_rand = randomUnexploredState();
    extendRRT(q_rand, T);
end
```

Function `extendRRT`(q_{rand}, T)**:**

```
q_near = T.nearestTreeNode(q_rand) ;
if q_near.hasUnseenSucessors() then
    q_new = nearestSucessor(q_rand, q_near);
    T.addChildNode(q_new, q_near);
end
```

Function `extendRRLT`(q_{rand}, T)**:**

```
q_new = T.nearestTreeLeaf(q_rand) ;
T.changeLeafToNode(q_new);
T.addNewLeaves(q_new);
```

3.1.2 RRTs with Local Planners

RRTs with local planner uses the Algorithm 4 from the previous section as a global planner, but uses a different planner for local planning. Instead of picking successors closest to q_{rand}, a local planner limited by depth, size or time is used from q_{near} to q_{rand}. When the local search is finished, the node closest to q_{rand} and nodes along its path are added to the tree.

3.2 RRT-Plan: a Randomized Algorithm for STRIPS Planning

The authors of this paper [BPD06] proposed a randomized STRIPS planning algorithm inspired by Rapidly exploring Random Trees.

The algorithm follows the outline of the RRT-Connect [KL00] algorithm. RRT-Connect grows the tree from the inital state, and after every expansion it attempts to connect to the goal.

To select a random state, the RRT-Plan algorithm generates possible q_{rand} states by taking random subsets from the goal. After obtaining a q_{rand} state, a nearest neighbor must be found. To estimate distances RRT-Plan uses the HSP [BG01] technique h^{+}_{add}. Then a planner is invoked to connect q_{near} to q_{rand}. The authors have chosen a limited version of FF [Hof01] as a local planner. A new node q_{new} is added to the tree. After that, the algorithm attempts to connect q_{new} to goal. RRT-Plan also uses other techniques such as goal subset locking and adapting search paramterers to improve its performance.

Algorithm 5: One iteration of the RRT-Plan algorithm [BPD06]

Function `iteration()`**:**
- select random goal subset RGS;
- find nearest neighbor q_{near} to RGS;
- invoke planner to connect q_{near} to RGS;
- **if** q_{new} is found that satisfies RGS **then**
  - add q_{new} to tree as child of q_{near};
  - calculate atom costs for q_{new};
  - attempt to connect q_{new} to final goal;
- **end**

3.3 Adapting a Rapidly-Exploring Random Tree for Automated Planning

In this paper [AVB11] a new use of RRTs in automated planning is proposed. Since we are using the outline of the Algorithm 6 in our implementation, we will focus on the description more in the next chapter.

The planner implemted in this paper uses Fast Downward as its local planner configured to greedy best-first search with lazy evaluation. The chosen heuristic the relaxed plan heuristic used by FF. Sampling uses mutexes computed by the invariant analysis in Fasts Downward.

Algorithm 6: RRT for discrete space (from [AVB11])

Data: Search space S, limit ϵ, initial state q_{new}, goal q_{goal}
Result: Plan *solution*
$tree \leftarrow q_{init}$;
while $\neg goalReached()$ **do**
  if $p < random()$ **then**
    $q_{rand} \leftarrow sampleSpace(S)$;
    $q_{near} \leftarrow findNearest(tree, q_{rand}, S)$;
    $q_{new} \leftarrow join(q_{near}, q_{rand}, \epsilon, S)$;
    $addNode(tree, q_{near}, q_{new})$;
    $q_{near_{goal}} \leftarrow q_{new}$;
  else
    $q_{near_{goal}} \leftarrow findNearest(tree, q_{goal}, S)$;
  end
  $q_{new_{goal}} \leftarrow join(tree, q_{goal}, S)$;
  $addNode(tree, q_{near_{goal}}, q_{new_{goal}})$;
end
$solution \leftarrow traceBack(tree, q_{goal})$;
return *solution*;

Chapter 4

Description of the Algorithm

This chapter describes the Algorithm 6, which was proposed in [AVB11] and further examined in [AFBV15] and chosen to be studied in this thesis.

The goal is to plan a solution to a problem in FDR. The algorithm starts with search space, limit for local planner, initial state and partial goal state. The tree is initialized with the initial state.

With probability $1 - p$ it samples the space, finds its nearest neighboring state stored in the tree, and tries to reach the sampled state from the nearest neighbor. If the sampled state is reached within the limited number of steps, the state is added to the tree. In case the limit of steps is reached first, a newly found state with the lowest heuristic value is added to the tree.

With probability p, instead of sampling the space, we search for a state in the tree, which has the smallest distance towards the partial goal state.

The algorithm then attempts to join the state used in the previous step with the partial goal state. The loop repeats until a solution is found.

4.1 Sampling

If we decided to sample the state by choosing random number of facts and also random facts from a search space of a problem represented in STRIPS, we would most likely generate an enormous number of unreachable states. The idea behind random sampling in this thesis is taking advatage of using the FDR representation and having each state consist of a given number of variables. That helps us prune large number of the possible unreachable states, since we are picking from a domain corresponding to a variable and not from the whole state space.

1. First sampling option will select random variable and then a random value from its domain without considering any other mutexes.
2. Second sampling option (also described in [AVB11]) will start by choosing random order in which variables will be assigned. In that given order, we again randomly select a value for each variable, this time however anytime a new value is assigned, we check whether it is not mutex with any other already selected values in the state. In case we are not able to add a new value and the state is not completed yet, we start the whole process again.

We will be comparing several configurations:

- **Random.** Apart from variables in FDR no additional mutex groups added. Uses first sampling option.
- **Lifted.** Mutex groups obtained during preprocessing and translation. Uses second sampling option.
- $\boldsymbol{h_2}$**.** Additionally computed h_2 mutexes. Uses second sampling option.
- $\boldsymbol{h_2}$ **forward backward**. Additionally computed h_2 mutexes forward and backward. Uses second sampling option.
- $\boldsymbol{h_3}$**.** Additionally computed h_3 mutexes. Uses second sampling option.
- **Fam.** Additionally computed fact-alternationg groups. Uses second sampling option.

As a last step of sampling, we verify that the state is reachable from the initial state and towards the goal by using h_max. In case the state is not reachable, we discard it and sample a new state.

4.2 Search for a nearest state

A state s' is considered the nearest state, when it is stored in the tree and its heuristic value is the lowest towards the sampled state s. For this project we have chosen to use h_{FF} to help estimate the distance.

4.3 Join

The join stage tries to connect the sampled state s with the nearest state s' in a limited number of steps using a local planner. We have chosen greedy search with lazy evaluation, where h_{FF} is the heuristic function. If s is reached within the limit, the output is the state s. If not, it returns the state, which was encountered during the search and has the lowest heuristic value.

Chapter 5

Experiments

In this chapter we examine the results of the implemented solver.

In order to test the planner we used a dataset of 45 domains from the satisficing track at the IPC. Each domain contains 20 problems. The computations were executed using MetaCentrum resources with the time limit being set to 1800 second and the memory being restricted to 8GB.

The planner was built on top of cpddl library `https://gitlab.com/danfis/cpddl-dev`. Greedy algorithm with lazy evaluation from the library was chosen as the local planner with h_{FF} as its heuristic function.

Three configurations of the solver were chosen based on number of steps the local planner was allowed to take. Since the proposed algorithm relies on the element of randomization, every configuration will run three times. The tables contain result of a planner that uses either a different method of sampling or a different mutex groups in sampling. The displayed results are the mean and standard deviation of the three iterations for each configuration. Every time a planner samples a new state, its reachability from the initial state and towards the goal is verified using h_{max}.

5.1 Results

To put performance of our newly implemented planner into context, we compare the results with a commonly used greedy algorithm with lazy evaluation implemented in the cpddl library `https://gitlab.com/danfis/cpddl-dev`. Table 5.1 shows the numbers of successfully found plans of said algorithm. It also informs us about how the situation would look if we were to run only the local planner.

It is very clear, that no matter which version of sampling was used, the planners with the largest limit in local search have the best results. The most successful configuration appears to be the planner using fact-alternating mutex groups with the limit of steps in local planner set to 100,000.

None of the solvers (including the local greedy search) was able to complete any problem from the domains elevators11, ged14 and visitall14. The planners struggle and work on long searches with large state space and result in timeout. Other than that, visitall is a domain known to perform better with different heuristic function than h_{FF}. Planner using h_3 mutex groups exits in several problems (e.g., parking14) while trying to obtain additional mutex pairs.

Results in domain mystery98 might look mediocre, but this domain contains problems that were deemed unsolvable during translation and pruning of FDR.

Some of the domains contain problems where the solvers struggle with sampling a reachable state (e.g., tetris14, spider18). That causes the planner to discard large number of states and sample again, causing it to get stuck in a loop. This problem is amplified for planners using shorter local search, as it is required from them to sample more often. For that reason, solvers were allowed 100,000 attempts to sample a reachable state, and in case even that number was surpassed, we allow an unreachable state to be returned.

To further examine whether allowing using unreachable states in sampling could be benefitial, we added another set of tests featuring the most succesful version of the planner and this time, we do not check the reachability of the newly sampled state neither from the initial state nor towards the goal. Table 5.8 shows clear improvement in configuration limited to 1,000 steps in local planner.

5.2 Tables

Domain		greedy
agricola18	(20)	9
barman11	(20)	3
barman14	(20)	4
blocks00	(20)	20
caldera18	(20)	15
cavediving14	(20)	7
childsnack14	(20)	0
data-network18	(20)	3
depot02	(20)	16
driverlog02	(20)	18
elevators11	(20)	0
floortile11	(20)	7
floortile14	(20)	2
freecell00	(20)	19
ged14	(20)	0
gripper98	(20)	20
hiking14	(20)	20
logistics00	(20)	20
logistics98	(20)	16
maintenance14	(20)	6
mprime98	(20)	17
mystery98	(20)	11
nomystery11	(20)	8
openstacks06	(20)	20
parking11	(20)	20
parking14	(20)	17
pegsol11	(20)	20
pipesworld-notankage04	(20)	19
rovers06	(20)	18
satellite02	(20)	19
scanalyzer11	(20)	18
snake18	(20)	6
sokoban11	(20)	18
spider18	(20)	11
storage06	(20)	18
termes18	(20)	16
tetris14	(20)	11
thoughtful14	(20)	12
tidybot11	(20)	17
tpp06	(20)	19
trucks06	(20)	15
visitall11	(20)	4
visitall14	(20)	0
woodworking11	(20)	17
zenotravel02	(20)	20
SUM	(900)	576

Table 5.1: Results of the greedy algorithm with lazy evaluation.

Domain		1000	10000	100000
agricola18	(20)	5.67+-0.58	6.33+-0.58	6.33+-0.58
barman11	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
barman14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
blocks00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
caldera18	(20)	12.33+-0.58	14.00+-0.00	15.00+-0.00
cavediving14	(20)	0.00+-0.00	7.00+-0.00	7.00+-0.00
childsnack14	(20)	0.00+-0.00	0.00+-0.00	0.67+-0.58
data-network18	(20)	1.67+-0.58	2.67+-0.58	4.67+-0.58
depot02	(20)	12.33+-1.15	14.67+-0.58	16.33+-1.53
driverlog02	(20)	15.67+-0.58	18.33+-1.53	19.00+-0.00
elevators11	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
floortile11	(20)	5.00+-1.00	5.67+-0.58	6.00+-0.00
floortile14	(20)	2.00+-0.00	2.00+-0.00	2.33+-0.58
freecell00	(20)	19.67+-0.58	20.00+-0.00	20.00+-0.00
ged14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
gripper98	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
hiking14	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics98	(20)	10.33+-0.58	15.00+-1.00	19.67+-0.58
maintenance14	(20)	11.33+-1.53	15.00+-0.00	13.67+-0.58
mprime98	(20)	19.33+-1.15	20.00+-0.00	18.67+-0.58
mystery98	(20)	13.00+-0.00	13.00+-0.00	13.00+-0.00
nomystery11	(20)	6.67+-0.58	7.67+-1.53	7.67+-0.58
openstacks06	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
parking11	(20)	0.67+-0.58	7.33+-0.58	19.00+-0.00
parking14	(20)	0.00+-0.00	0.67+-0.58	15.00+-1.00
pegsol11	(20)	19.00+-0.00	19.33+-0.58	19.67+-0.58
pipesworld-notankage04	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
rovers06	(20)	18.33+-0.58	20.00+-0.00	20.00+-0.00
satellite02	(20)	17.67+-0.58	20.00+-0.00	19.67+-0.58
scanalyzer11	(20)	14.33+-0.58	16.67+-0.58	18.67+-0.58
snake18	(20)	4.33+-0.58	6.33+-0.58	7.00+-1.00
sokoban11	(20)	8.00+-0.00	12.00+-1.00	15.67+-1.53
spider18	(20)	2.33+-1.53	6.33+-0.58	10.33+-0.58
storage06	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
termes18	(20)	5.33+-0.58	14.67+-1.15	18.00+-1.00
tetris14	(20)	3.00+-0.00	10.00+-1.00	14.00+-0.00
thoughtful14	(20)	17.33+-1.53	16.33+-0.58	15.67+-0.58
tidybot11	(20)	18.00+-0.00	18.00+-0.00	19.00+-0.00
tpp06	(20)	13.00+-0.00	15.00+-0.00	17.67+-1.53
trucks06	(20)	16.33+-0.58	17.67+-0.58	18.67+-1.15
visitall11	(20)	0.33+-0.58	3.33+-0.58	4.00+-0.00
visitall14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
woodworking11	(20)	9.00+-1.00	17.33+-0.58	17.33+-0.58
zenotravel02	(20)	19.00+-0.00	20.00+-0.00	20.00+-0.00
SUM	(900)	461.00+-4.00	542.33+-3.06	599.33+-2.52

Table 5.2: Results of random RRT without additional mutex groups.

Domain		1000	10000	100000
agricola18	(20)	6.00+-1.00	6.33+-0.58	7.00+-0.00
barman11	(20)	0.00+-0.00	0.00+-0.00	2.00+-1.73
barman14	(20)	0.00+-0.00	0.00+-0.00	0.33+-0.58
blocks00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
caldera18	(20)	12.33+-0.58	14.00+-0.00	15.00+-0.00
cavediving14	(20)	0.33+-0.58	7.00+-0.00	7.00+-0.00
childsnack14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
data-network18	(20)	1.67+-0.58	3.33+-0.58	5.00+-1.00
depot02	(20)	12.67+-0.58	14.33+-0.58	16.33+-0.58
driverlog02	(20)	15.67+-0.58	19.00+-1.00	19.33+-0.58
elevators11	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
floortile11	(20)	6.00+-0.00	6.67+-0.58	6.67+-0.58
floortile14	(20)	2.00+-0.00	2.33+-0.58	2.00+-0.00
freecell00	(20)	19.67+-0.58	20.00+-0.00	20.00+-0.00
ged14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
gripper98	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
hiking14	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics98	(20)	9.33+-0.58	14.67+-1.53	20.00+-0.00
maintenance14	(20)	11.00+-1.00	14.33+-0.58	13.67+-1.53
mprime98	(20)	18.67+-0.58	18.67+-0.58	17.67+-1.15
mystery98	(20)	13.00+-0.00	13.00+-0.00	13.00+-0.00
nomystery11	(20)	6.00+-1.00	6.33+-0.58	8.00+-0.00
openstacks06	(20)	20.00+-0.00	20.00+-0.00	19.67+-0.58
parking11	(20)	15.33+-0.58	20.00+-0.00	20.00+-0.00
parking14	(20)	4.33+-0.58	15.67+-1.53	17.33+-1.15
pegsol11	(20)	19.00+-0.00	19.00+-1.00	19.67+-0.58
pipesworld-notankage04	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
rovers06	(20)	19.00+-0.00	20.00+-0.00	20.00+-0.00
satellite02	(20)	17.33+-1.15	19.33+-0.58	18.67+-1.15
scanalyzer11	(20)	17.00+-0.00	18.67+-1.15	20.00+-0.00
snake18	(20)	5.33+-0.58	6.33+-0.58	6.67+-1.15
sokoban11	(20)	7.33+-1.15	13.33+-0.58	16.67+-0.58
spider18	(20)	2.00+-0.00	8.00+-1.00	10.33+-0.58
storage06	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
termes18	(20)	6.00+-1.73	15.00+-0.00	17.33+-0.58
tetris14	(20)	3.00+-1.00	10.33+-4.16	13.67+-2.08
thoughtful14	(20)	16.33+-0.58	17.67+-0.58	16.33+-0.58
tidybot11	(20)	17.67+-0.58	18.33+-0.58	19.00+-1.00
tpp06	(20)	14.33+-0.58	16.00+-0.00	18.33+-1.15
trucks06	(20)	17.00+-1.73	19.00+-0.00	19.33+-0.58
visitall11	(20)	1.00+-1.00	2.33+-0.58	4.33+-0.58
visitall14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
woodworking11	(20)	10.00+-1.73	18.33+-0.58	18.67+-0.58
zenotravel02	(20)	19.67+-0.58	20.00+-0.00	20.00+-0.00
SUM	(900)	486.00+-5.29	577.33+-5.03	609.00+-6.24

Table 5.3: Results of RRT with lifted mutex groups.

Domain		1000	10000	100000
agricola18	(20)	5.67+-0.58	6.33+-0.58	7.00+-0.00
barman11	(20)	0.00+-0.00	0.33+-0.58	3.33+-0.58
barman14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
blocks00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
caldera18	(20)	12.67+-0.58	13.33+-1.15	14.00+-1.73
cavediving14	(20)	0.67+-1.15	6.67+-0.58	7.00+-0.00
childsnack14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
data-network18	(20)	1.67+-0.58	3.00+-1.00	4.33+-0.58
depot02	(20)	13.33+-1.15	14.33+-1.15	16.00+-0.00
driverlog02	(20)	15.67+-0.58	18.33+-0.58	19.00+-1.00
elevators11	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
floortile11	(20)	6.00+-0.00	6.00+-0.00	7.00+-1.00
floortile14	(20)	2.00+-0.00	2.00+-0.00	2.33+-0.58
freecell00	(20)	19.67+-0.58	20.00+-0.00	20.00+-0.00
ged14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
gripper98	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
hiking14	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics98	(20)	9.00+-0.00	15.33+-0.58	18.67+-0.58
maintenance14	(20)	10.33+-0.58	15.00+-0.00	13.00+-1.00
mprime98	(20)	19.67+-0.58	19.33+-0.58	18.67+-0.58
mystery98	(20)	13.00+-0.00	13.00+-0.00	13.00+-0.00
nomystery11	(20)	6.00+-0.00	8.00+-0.00	7.67+-1.15
openstacks06	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
parking11	(20)	13.33+-2.52	20.00+-0.00	20.00+-0.00
parking14	(20)	3.00+-1.00	17.00+-1.00	18.33+-0.58
pegsol11	(20)	19.33+-0.58	19.33+-0.58	19.00+-1.00
pipesworld-notankage04	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
rovers06	(20)	18.00+-0.00	20.00+-0.00	20.00+-0.00
satellite02	(20)	17.33+-1.53	20.00+-0.00	19.33+-0.58
scanalyzer11	(20)	17.33+-0.58	19.33+-0.58	19.67+-0.58
snake18	(20)	4.33+-1.15	6.67+-1.15	7.67+-1.15
sokoban11	(20)	8.00+-1.00	13.00+-1.73	15.00+-1.00
spider18	(20)	3.67+-0.58	8.33+-0.58	10.00+-1.00
storage06	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
termes18	(20)	6.67+-1.53	15.00+-0.00	17.67+-0.58
tetris14	(20)	4.33+-1.15	10.00+-1.73	14.67+-0.58
thoughtful14	(20)	16.67+-0.58	16.67+-1.15	15.67+-0.58
tidybot11	(20)	17.67+-0.58	18.00+-0.00	19.33+-0.58
tpp06	(20)	13.33+-0.58	15.67+-0.58	18.33+-0.58
trucks06	(20)	16.00+-1.00	18.00+-0.00	19.00+-0.00
visitall11	(20)	0.00+-0.00	2.67+-0.58	4.33+-1.15
visitall14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
woodworking11	(20)	12.00+-1.00	18.00+-1.00	17.33+-0.58
zenotravel02	(20)	18.67+-1.15	20.00+-0.00	20.00+-0.00
SUM	(900)	485.00+-4.36	578.67+-7.09	606.33+-5.13

Table 5.4: Results of RRT with h_2 mutexes.

Domain		1000	10000	100000
agricola18	(20)	6.33+-0.58	7.00+-0.00	6.67+-0.58
barman11	(20)	0.00+-0.00	0.67+-0.58	2.00+-1.00
barman14	(20)	0.00+-0.00	0.00+-0.00	0.33+-0.58
blocks00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
caldera18	(20)	10.67+-1.15	14.00+-0.00	15.00+-0.00
cavediving14	(20)	7.00+-0.00	7.00+-0.00	7.00+-0.00
childsnack14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
data-network18	(20)	1.67+-0.58	3.33+-0.58	4.67+-1.53
depot02	(20)	11.67+-0.58	13.67+-1.15	15.67+-0.58
driverlog02	(20)	16.00+-0.00	19.00+-1.00	18.67+-0.58
elevators11	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
floortile11	(20)	6.67+-0.58	6.33+-0.58	7.00+-0.00
floortile14	(20)	2.00+-0.00	2.00+-0.00	2.33+-0.58
freecell00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
ged14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
gripper98	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
hiking14	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics98	(20)	9.67+-0.58	15.33+-1.15	19.33+-0.58
maintenance14	(20)	12.00+-1.00	14.67+-0.58	14.33+-0.58
mprime98	(20)	19.33+-0.58	19.67+-0.58	19.00+-1.00
mystery98	(20)	13.00+-0.00	13.00+-0.00	13.00+-0.00
nomystery11	(20)	6.00+-1.00	7.33+-0.58	8.67+-0.58
openstacks06	(20)	19.67+-0.58	20.00+-0.00	20.00+-0.00
parking11	(20)	15.33+-3.79	20.00+-0.00	19.67+-0.58
parking14	(20)	2.00+-1.00	15.33+-1.15	18.33+-1.15
pegsol11	(20)	19.00+-0.00	19.33+-0.58	20.00+-0.00
pipesworld-notankage04	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
rovers06	(20)	19.00+-0.00	20.00+-0.00	20.00+-0.00
satellite02	(20)	17.00+-1.00	19.67+-0.58	18.33+-0.58
scanalyzer11	(20)	18.67+-0.58	18.67+-1.15	19.67+-0.58
snake18	(20)	4.67+-0.58	6.00+-1.00	6.33+-0.58
sokoban11	(20)	8.00+-0.00	11.67+-0.58	15.67+-0.58
spider18	(20)	3.00+-1.00	8.67+-0.58	10.33+-1.15
storage06	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
termes18	(20)	5.67+-0.58	15.00+-0.00	17.33+-0.58
tetris14	(20)	2.33+-1.15	10.00+-2.65	14.33+-0.58
thoughtful14	(20)	15.00+-1.00	16.33+-0.58	15.67+-0.58
tidybot11	(20)	18.00+-0.00	18.67+-0.58	19.67+-0.58
tpp06	(20)	12.67+-0.58	15.67+-1.15	17.67+-0.58
trucks06	(20)	15.67+-1.53	18.33+-1.53	19.00+-1.00
visitall11	(20)	1.33+-0.58	1.67+-0.58	4.33+-0.58
visitall14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
woodworking11	(20)	10.67+-1.53	17.00+-1.00	17.67+-0.58
zenotravel02	(20)	19.33+-0.58	20.00+-0.00	20.00+-0.00
SUM	(900)	489.00+-6.08	575.00+-4.00	607.67+-3.06

Table 5.5: Results of RRT with h_2 forward backward mutexes.

Domain		1000	10000	100000
agricola18	(20)	6.33+-0.58	7.00+-0.00	8.00+-0.00
barman11	(20)	0.00+-0.00	0.00+-0.00	1.33+-0.58
barman14	(20)	0.00+-0.00	0.00+-0.00	0.33+-0.58
blocks00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
caldera18	(20)	8.33+-0.58	10.00+-0.00	9.67+-0.58
cavediving14	(20)	0.33+-0.58	6.67+-0.58	7.00+-0.00
childsnack14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
data-network18	(20)	1.33+-0.58	3.67+-1.53	4.33+-1.15
depot02	(20)	13.33+-0.58	14.00+-1.00	15.67+-0.58
driverlog02	(20)	15.33+-0.58	17.67+-1.15	18.67+-0.58
elevators11	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
floortile11	(20)	6.33+-0.58	6.33+-0.58	7.33+-0.58
floortile14	(20)	2.00+-0.00	2.00+-0.00	2.33+-0.58
freecell00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
ged14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
gripper98	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
hiking14	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics98	(20)	9.00+-0.00	13.33+-0.58	15.00+-0.00
maintenance14	(20)	10.67+-1.15	15.00+-0.00	14.33+-1.15
mprime98	(20)	15.00+-1.00	15.00+-0.00	14.67+-0.58
mystery98	(20)	9.33+-1.15	10.00+-0.00	10.00+-0.00
nomystery11	(20)	6.33+-1.15	7.33+-0.58	7.67+-1.53
openstacks06	(20)	20.00+-0.00	20.00+-0.00	19.67+-0.58
parking11	(20)	2.00+-0.00	2.00+-0.00	2.00+-0.00
parking14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
pegsol11	(20)	19.33+-0.58	19.67+-0.58	19.67+-0.58
pipesworld-notankage04	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
rovers06	(20)	18.33+-0.58	20.00+-0.00	20.00+-0.00
satellite02	(20)	16.33+-0.58	20.00+-0.00	19.33+-1.15
scanalyzer11	(20)	18.00+-1.00	17.00+-0.00	19.00+-1.00
snake18	(20)	4.00+-1.00	6.33+-0.58	6.00+-0.00
sokoban11	(20)	8.67+-0.58	12.67+-1.53	15.33+-1.15
spider18	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
storage06	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
termes18	(20)	7.00+-1.73	15.00+-0.00	17.33+-0.58
tetris14	(20)	3.33+-0.58	9.33+-1.53	15.00+-1.00
thoughtful14	(20)	15.00+-1.00	14.00+-2.65	12.33+-4.62
tidybot11	(20)	17.67+-0.58	18.00+-0.00	18.00+-0.00
tpp06	(20)	13.00+-0.00	15.67+-0.58	18.33+-0.58
trucks06	(20)	16.67+-0.58	18.67+-0.58	18.00+-1.00
visitall11	(20)	0.33+-0.58	2.67+-0.58	4.00+-0.00
visitall14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
woodworking11	(20)	9.00+-2.00	18.00+-0.00	18.33+-0.58
zenotravel02	(20)	18.33+-0.58	19.00+-1.00	18.67+-0.58
SUM	(900)	450.67+-3.51	516.00+-4.36	537.33+-3.06

Table 5.6: Results of RRT with h_3 mutexes.

Domain		1000	10000	100000
agricola18	(20)	5.67+-0.58	7.33+-1.15	7.33+-1.15
barman11	(20)	0.00+-0.00	0.00+-0.00	2.00+-1.00
barman14	(20)	0.00+-0.00	0.00+-0.00	0.33+-0.58
blocks00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
caldera18	(20)	12.67+-1.53	13.67+-1.15	13.67+-1.15
cavediving14	(20)	4.33+-1.53	7.00+-0.00	7.00+-0.00
childsnack14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
data-network18	(20)	1.33+-0.58	2.00+-0.00	5.00+-1.00
depot02	(20)	13.33+-0.58	15.67+-0.58	17.33+-1.15
driverlog02	(20)	15.00+-0.00	18.67+-0.58	19.00+-0.00
elevators11	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
floortile11	(20)	6.00+-0.00	6.00+-0.00	7.33+-0.58
floortile14	(20)	2.00+-0.00	2.33+-0.58	3.33+-0.58
freecell00	(20)	17.33+-0.58	18.00+-0.00	20.00+-0.00
ged14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
gripper98	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
hiking14	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics98	(20)	10.33+-0.58	15.33+-1.53	18.67+-0.58
maintenance14	(20)	12.00+-1.00	15.00+-0.00	14.00+-1.00
mprime98	(20)	19.33+-0.58	19.33+-1.15	18.33+-1.53
mystery98	(20)	13.00+-0.00	13.00+-0.00	13.00+-0.00
nomystery11	(20)	6.33+-1.15	7.67+-1.15	9.67+-0.58
openstacks06	(20)	20.00+-0.00	19.67+-0.58	20.00+-0.00
parking11	(20)	14.33+-0.58	20.00+-0.00	20.00+-0.00
parking14	(20)	3.67+-1.53	14.33+-2.89	18.67+-0.58
pegsol11	(20)	18.33+-0.58	19.67+-0.58	19.67+-0.58
pipesworld-notankage04	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
rovers06	(20)	18.67+-0.58	20.00+-0.00	20.00+-0.00
satellite02	(20)	17.67+-1.15	20.00+-0.00	19.67+-0.58
scanalyzer11	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
snake18	(20)	2.67+-0.58	3.00+-0.00	5.33+-0.58
sokoban11	(20)	9.00+-1.00	13.00+-1.00	16.33+-0.58
spider18	(20)	2.33+-0.58	9.33+-2.52	11.00+-0.00
storage06	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
termes18	(20)	7.67+-1.15	15.33+-0.58	17.67+-0.58
tetris14	(20)	7.67+-2.08	14.33+-2.08	17.67+-0.58
thoughtful14	(20)	18.33+-0.58	19.00+-0.00	17.33+-0.58
tidybot11	(20)	18.00+-0.00	18.67+-0.58	19.00+-0.00
tpp06	(20)	13.00+-0.00	15.67+-0.58	18.33+-0.58
trucks06	(20)	16.00+-1.00	18.00+-1.00	18.33+-0.58
visitall11	(20)	0.67+-0.58	2.67+-0.58	4.33+-0.58
visitall14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
woodworking11	(20)	10.00+-1.00	17.33+-0.58	17.67+-1.15
zenotravel02	(20)	19.33+-0.58	20.00+-0.00	20.00+-0.00
SUM	(900)	496.00+-3.46	581.00+-5.00	617.00+-1.00

Table 5.7: Results of RRT with fam groups.

Domain		1000	10000	100000
agricola18	(20)	6.00+-1.00	6.33+-0.58	7.00+-0.00
barman11	(20)	0.00+-0.00	0.00+-0.00	0.33+-0.58
barman14	(20)	0.00+-0.00	0.00+-0.00	0.33+-0.58
blocks00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
caldera18	(20)	12.00+-1.73	13.33+-0.58	14.00+-1.73
cavediving14	(20)	2.67+-1.53	7.00+-0.00	7.00+-0.00
childsnack14	(20)	0.00+-0.00	0.00+-0.00	0.33+-0.58
data-network18	(20)	1.67+-0.58	3.67+-0.58	5.00+-0.00
depot02	(20)	13.33+-1.15	17.00+-1.00	17.33+-0.58
driverlog02	(20)	15.67+-0.58	17.67+-1.15	19.33+-0.58
elevators11	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
floortile11	(20)	4.67+-1.53	5.33+-0.58	4.33+-2.52
floortile14	(20)	2.00+-0.00	2.33+-0.58	1.00+-0.00
freecell00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
ged14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
gripper98	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
hiking14	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics00	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
logistics98	(20)	10.67+-1.15	14.67+-0.58	19.33+-0.58
maintenance14	(20)	10.67+-0.58	14.00+-0.00	13.00+-0.00
mprime98	(20)	20.00+-0.00	18.67+-1.15	18.33+-0.58
mystery98	(20)	13.00+-0.00	13.00+-0.00	13.00+-0.00
nomystery11	(20)	6.33+-0.58	8.33+-0.58	9.00+-1.00
openstacks06	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
parking11	(20)	15.00+-0.00	20.00+-0.00	20.00+-0.00
parking14	(20)	1.33+-0.58	15.67+-1.53	19.00+-0.00
pegsol11	(20)	19.67+-0.58	19.00+-0.00	19.00+-1.00
pipesworld-notankage04	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
rovers06	(20)	18.33+-0.58	20.00+-0.00	20.00+-0.00
satellite02	(20)	17.00+-0.00	19.67+-0.58	18.67+-0.58
scanalyzer11	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
snake18	(20)	4.00+-0.00	5.33+-0.58	6.33+-0.58
sokoban11	(20)	10.33+-0.58	14.00+-1.00	16.00+-0.00
spider18	(20)	6.67+-1.15	10.33+-2.31	11.67+-0.58
storage06	(20)	20.00+-0.00	20.00+-0.00	20.00+-0.00
termes18	(20)	5.00+-2.00	14.67+-0.58	17.67+-0.58
tetris14	(20)	12.33+-2.52	18.33+-0.58	18.67+-0.58
thoughtful14	(20)	19.67+-0.58	19.00+-0.00	19.00+-0.00
tidybot11	(20)	18.00+-0.00	18.00+-0.00	17.33+-0.58
tpp06	(20)	13.00+-0.00	15.33+-0.58	17.67+-1.15
trucks06	(20)	14.00+-1.73	17.67+-0.58	16.67+-0.58
visitall11	(20)	1.00+-0.00	2.00+-0.00	4.00+-0.00
visitall14	(20)	0.00+-0.00	0.00+-0.00	0.00+-0.00
woodworking11	(20)	12.33+-0.58	20.00+-0.00	19.00+-1.00
zenotravel02	(20)	19.67+-0.58	20.00+-0.00	20.00+-0.00
SUM	(900)	506.00+-1.00	590.33+-4.16	609.33+-3.06

Table 5.8: Results of RRT with fam groups without discarding unreachable sampled states.

Chapter 6

Conclusions

The goal of the thesis was to propose a way to adapt the RRT algorithm, which is commonly used in motion planning, for a solver of classical planning problems. Then implement a planner and evaluate its performance on a dataset.

We implemented the RRT-based planner in C and observed the influence of used mutex groups:

- Random. Apart from variables in FDR no additional mutex groups added.
- Lifted. Mutex groups obtained during preprocessing and translation.
- h_2. Additionally computed h_2 mutexes.
- h_2 forward backward. Additionally computed h_2 mutexes forward and backward.
- h_3. Additionally computed h_3 mutexes.
- Fam. Additionally computed fact-alternationg groups.

For each variation of the planner we also studied how changing the limit of steps the local planner is allowed to take affects the results. We then compared our planner with a planner using greedy algorithm. Based on the chosen configuration, our solver can yield better results than just the greedy

algorithm. For a future evaluation another possible variation could show us a different outcome - modifying the value of probability.

Appendix A

Content of the Attached Disc

- cpddl-dev - directory containing the planner build on top of cpddl library
- README - file describing the way how to run the planner

Appendix B

Bibliography

[AFBV15] Vidal Alcázar, Susana Fernández, Daniel Borrajo, and Manuela Veloso, *Using random sampling trees for automated planning*, Ai Communications **28** (2015), no. 4, 665–681.

[AT15] Vidal Alcázar and Alvaro Torralba, *A reminder about the importance of computing and exploiting invariants in planning*, Twenty-Fifth International Conference on Automated Planning and Scheduling, 2015.

[AVB11] Vidal Alcázar, Manuela Veloso, and Daniel Borrajo, *Adapting a rapidly-exploring random tree for automated planning*, Fourth Annual Symposium on Combinatorial Search, 2011.

[BG01] Blai Bonet and Héctor Geffner, *Planning as heuristic search*, Artificial Intelligence **129** (2001), no. 1-2, 5–33.

[BPD06] Daniel Burfoot, Joelle Pineau, and Gregory Dudek, *RRT-Plan: A Randomized Algorithm for STRIPS Planning.*, ICAPS, 2006, pp. 362–365.

[FK18] Daniel Fišer and Antonín Komenda, *Fact-alternating mutex groups for classical planning*, Journal of Artificial Intelligence Research **61** (2018), 475–521.

[FN71] Richard E Fikes and Nils J Nilsson, *Strips: A new approach to the application of theorem proving to problem solving*, Artificial intelligence **2** (1971), no. 3-4, 189–208.

[GNT16] Malik Ghallab, Dana Nau, and Paolo Traverso, *Automated planning and acting*, Cambridge University Press, 2016.

[Has09] Patrik Haslum, *hm (p)= h 1 (p m): Alternative characterisations of the generalisation from h max to hm*, Nineteenth International Conference on Automated Planning and Scheduling, 2009.

[Hel09] Malte Helmert, *Concise finite-domain representations for PDDL planning tasks*, Artificial Intelligence **173** (2009), no. 5-6, 503–535.

[Hof01] Jörg Hoffmann, *Ff: The fast-forward planning system*, AI magazine **22** (2001), no. 3, 57–57.

[KL00] James J Kuffner and Steven M LaValle, *Rrt-connect: An efficient approach to single-query path planning*, Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), vol. 2, IEEE, 2000, pp. 995–1001.

[KSLO96] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars, *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*, IEEE transactions on Robotics and Automation **12** (1996), no. 4, 566–580.

[L+98] Steven M LaValle et al., *Rapidly-exploring random trees: A new tool for path planning*, The annual research report (1998).

[LaV06] Steven M LaValle, *Planning algorithms*, Cambridge university press, 2006.

[LKD+01] Steven M LaValle, James J Kuffner, BR Donald, et al., *Rapidly-exploring random trees: Progress and prospects*, Algorithmic and computational robotics: new directions **5** (2001), 293–308.

[MB04] Stuart Morgan and Michael S Branicky, *Sampling-based planning for discrete spaces*, 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 2, IEEE, 2004, pp. 1938–1945.