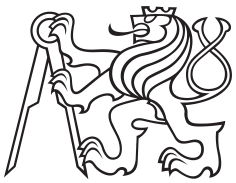
Bachelor Project



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.
January 2022

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: **04.01.2022**

Assignment valid until: **13.02.2022**

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, 4 January 2022

Abstract

Problems of classical planning are usually solved by 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 case a similar problem occurs in motion planning, such problems can be resolved by algorithms using randomization. It could be benefitial to see their potential in classical planning.

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

Keywords: classical 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. Když nastane podobná situace u plánování pohybu robotů, mohou být takové problémy řešeny algoritmy, které používají randomizaci. Mohlo by být vhodné vidět jejich potenciál v klasickém plánování.

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

Klíčová slova: klasické 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 a heuristic function. Algorithms using heuristic functions can 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, in some cases greedy searches have issues and get stuck when dealing with problems containing large plateaus, which consist of states with the same heuristic value. A planner can also get easily sidetracked if it encounters a local minimum.

In motion planning, solving a problem means finding a path which leads towards goal without colliding with obstacles. Sampling-based motion planning algorithms are very popular and are considered state-of-the-art techniques used to solve motion planning problems.

One of the most successful and effective sampling-based algorithms is Rapidly-exploring Random Trees (RRT) [L+98]. It has many positive qualities, such as being relatively simple to implement, the expansion of an RRT is biased towards yet unexplored space, the tree data structure is always connected, etc.

Because of these useful properties, the possible use of the algorithm in classical planning is being explored. Unfortunately, adapting an algorithm from motion planning to classical planning is not trivial.

The main issue is the different state space. While motion planning deals with continuous spaces, the classical planning uses discrete space.

During the search, it is desirable to produce uniform random samples, which is very simple to do in configuration space formed from Cartesian products. Uniform sampling over all possible states is not as straightforward in classical 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, for example, sampling only over the subset of goal states, but that would devalue the idea of uniform sampling.

It is required to measure the distance between a sampled state and a node present in the tree. In motion planning, the distance can be directly computed by using a chosen metric (e.g., the Euclidean metric, the Manhattan metric). Planners in classical planning need to find alternative methods, such as using heuristic functions to estimate the distance.

The goal of the thesis is to propose a way to solve these issues 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 problem can be described as a state model defined in [BG01] as a tuple $\sum = (S, s_0, S_G, A, \gamma, cost)$, where:

- S is a finite and non-empty set of states s.
- $s_0 \in S$ is the initial state.
- $S_G \subseteq S$ is a non-empty set of goal states.
- A is a finite set of actions. $A(s) \subseteq A$ denotes the actions applicable in each state $s \in S$
- $\gamma(a, s)$ is a state transition function for all $s \in S$ and $a \in A(s)$.
- $cost(a, s)$ is a cost of performing action a in state s.

A solution or a plan of a state model is a sequence of actions $\pi = a_0, a_1, ..., a_n$ generating a sequence $s_0, s_1 = \gamma(s_0), ..., s_{n+1} = \gamma(a_n, s_n)$, where each action a_i is applicable in s_i and s_{n+1} is a goal state. A state s_r, for which exist a sequence $s_0, s_1 = \gamma(s_0), ..., s_{r+1} = \gamma(a_r, s_r)$ is called reachable. The plan is called optimal when the total cost $\sum_{i=0}^{n} cost(a_i, s_i)$.

Most of the problems of classical planning problems are specified in standardized planning language called Planning Domain Definition Language (PDDL) [AHK+98]. However, planners usually work with different representation, mainly STRIPS [FN71] and finite domain representation [Hel09], and therefore the problems need to be translated.

2.1.1 STRIPS

With the help of definition of a state model $\sum = (S, s_0, S_G, A, \gamma, cost)$, A STRIPS representation is defined in [BG01] as a tuple $P = \langle \mathcal{A}, \mathcal{O}, I, G, c \rangle$, where:

- $\mathcal{A}$ is a set of atoms. States $s \in S$ are collection of atoms α from $\mathcal{A}$.
- $\mathcal{O}$ is a set of operators $op \in \mathcal{O}$ corresponding to $a \in A(s)$. Each operator has a:
  - precondition list $Prec(op)$ - set of atoms which need to be true in the state s before the operator executes his action. An operator op is called applicable in s if $Prec(op) \subseteq s$.
  - add list $Add(op)$ - set of atoms added to the state after the action was executed.
  - delete list $Del(op)$ - set of atoms deleted from the state after the action was executed.

  To apply an operator op in a state s, op must be applicable in state s. The new state $s' = (s \setminus Del(op)) \cup Add(op)$
- $I \subseteq \mathcal{A}$ represents the initial state s_0.
- $G \subseteq \mathcal{A}$ describes the goal situations. For a state s to be considered goal, it must stand that $G \subseteq s$.
- c is a cost function mapping each operator to a non-negative real number.

A sequence of operators $\pi = \langle op_1, ..., op_n \rangle$ is applicable in a state s_0 if there are states $s_1, ..., s_n$, where every operator op_i is applicable in s_{i-1} and $s_i = res(op_i, s_{i-1})$ for $i \in [1, n]$. The result of this action is $res(\pi, s_0) = s_n$.

A sequence of operators π is called a plan if $s = res(\pi, I)$ and $G \subseteq s$. The cost of a plan is $c(\pi) = \sum_{op \in \pi} c(op)$.

2.1.2 Finite domain representation

A finite domain representation (FDR) can be described as a tuple $\Pi = \langle V, O, s_{init}, s_{goal}, c \rangle$, where:

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

  A fact is a pair $\langle v, d \rangle$, where $v \in V$ and $d \in D_v$. A partial variable assignment is a set of facts, where each fact has a different variable. A partial state s is a partial variable assignment over V. A state is a variable assignment over all variables $v \in V$.

- O is a set of operators. Operator $o \in O$ is a pair $o = \langle pre(o), eff(o) \rangle$, where precondition $pre(o)$ and effect $eff(o)$ are both partial states.

- s_{init} is the initial state of the task.

- s_{goal} is a partial state which describes the 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 $s' = res(o, s)$. If a variable is defined in $eff(o)$, then the corresponding variable in s' is assigned the same values as the variable in $eff(o)$. 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 $s_i = res(o_i, s_{i-1})$ for $i \in [1, n]$. The result of this action is $res(\pi, s_0) = s_n$.

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

2.2 Relaxed heuristics

A heuristic function [GNT16] is a function h which returns the estimate $h(s)$ of the minimum cost $h^*(s)$ of reaching a goal state from a state s. Heuristic function is admissible if $0 \leq h(s) \leq h^*(s)$ for every s. If $h(s) = 0$, s is a goal.

Heuristic functions are produced by relaxation. Generally, that would mean weakening some constraints present in the task. For a STRIPS planning task $P = \langle \mathcal{A}, \mathcal{O}, I, G, c \rangle$, that would be $P^+ = \langle \mathcal{A}, \mathcal{O}^+, I, G, c \rangle$. In STRIPS representation relaxation would mean ignoring the delete list. Therefore once a fact is true, it will always be true. Even though the STRIPS is not our main focus in this thesis, we can modify FDR and to every fact $\langle v, d \rangle$ assign either true or false.

One of the most known heuristic functions is h^{max}. It is an admissible heuristic function, but it's values don't offer as much information as other alternatives. The function $h^{max}(s)$ returns the estimate of the distance from s to a goal as

$$h^{max}(s) = \max_{\alpha \in G} \Delta(s, \alpha), \ \alpha \in \mathcal{A}, \tag{2.1}$$

where

$$\Delta(s, op) = \max_{\alpha \in Prec(op)} \Delta(s, \alpha), \ \forall op \in \mathcal{O}, \tag{2.2}$$

and

$$\Delta(s, \alpha) = \begin{cases} 0, & \text{if } \alpha \in s, \\ \infty, & \text{if } \forall op \in \mathcal{O} : f \notin Add(op), \\ \min\{c(op) + \Delta(s, op) \mid op \in \mathcal{O}, \alpha \in Add(op)\}, & \text{otherwise.} \end{cases} \tag{2.3}$$

h^{FF} takes different approach. First, it finds a goal in the relaxed plan and then it uses supporters (actions which caused an atom to be true) to compute its value from the goal back to the initial state.

As stated in [Has09], it is possible to generalize the h^{max} into h^m, allowing as to utilize other heuristics, mostly h^2 and h^3. Heuristic h^m is very useful for inferring mutexes and translating problems into FDR.

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.

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

At first, the initial node $(\langle\rangle, s_0)$ is inserted into *Frontier* and *Expanded* is set to be an empty set. In each while 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, S_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* S_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 works as follows: For each node $\nu = (\pi, s) \in Children$, if there is more than one possible sequence of actions which can reach s, keep the one with minimal cost and remove the others.

2.4 Mutex

Mutex, mutex groups and fact-alternating mutex groups [FK18] can be defined in STRIPS representation $P = \langle \mathcal{A}, \mathcal{O}, I, G, c \rangle$ as:

Mutex $M \subseteq \mathcal{A}$ is a set of atoms, such that for every reachable state $s \in R$ it holds that $M \not\subseteq s$, where R is a set of all reachable states.

A mutex group $M \subseteq \mathcal{A}$ is a set of atoms, such that for every reachable state $s \in R$ it holds that $\mid M \cap s \mid \leq 1$, where R is a set of all reachable states.

A fact-alternating mutex group (fam-group) $M \subseteq \mathcal{A}$ is a set of atoms, such that $\mid M \cap I \mid \leq 1$ and $\mid M \cap Add(op) \mid \leq \mid M \cap Prec(op) \cap Del(op) \mid$ for every operator $op \in \mathcal{O}$.

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 its atoms are sets of m atoms from the problem P. If the value of h^{max} of an atom in P^m is infinite, then the atom is a mutex of size m.

The most frequently used method to help us find invariants is h^2. Since the cost of computation rapidly increases, it is not very common to encounter h^m with m more than 3. We can also consider the possibility of reversing the process and starting the search from a goal and move towards the initial state.

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: STRIPS planning task $P = \langle \mathcal{A}, \mathcal{O}, I, G, c \rangle$
Output: A set of fam-groups M
Initialize ILP with constraints according to Equation (2.4) and Equation (2.5);
Set objective function of ILP to maximize $\sum_{\alpha_i \in \mathcal{A}} x_i$;
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 (2.6) using M;
  M $\leftarrow \emptyset$;
  Solve ILP and if a solution was found, save the resulting fam-group into M;
end

$$\sum_{\alpha_i \in I} x_i \leq 1. \tag{2.4}$$

$$\forall op \in \mathcal{O} : \sum_{\alpha_i \in \text{Add}(op)} x_i \leq \sum_{\alpha_i \in \text{Del}(op) \cap \text{Prec}(op)} x_i. \tag{2.5}$$

$$\sum_{\alpha_i \notin I} x_i \geq 1. \tag{2.6}$$

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.

Motion planning problem can be defined by:

- $\mathcal{W}$ is either a 2D world, $\mathcal{W} = \mathbb{R}^2$ or 3D world, $\mathcal{W} = \mathbb{R}^3$
- $\mathcal{O}$ is a obstacle region, $\mathcal{O} \subseteq \mathcal{W}$. Obstacles remain fixed in the world $\mathcal{W}$.

- $\mathcal{A}$ is either a robot $\mathcal{A} \subseteq \mathbb{R}^2$ or $\mathbb{R}^3$, or set of attached bodies $\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_m$. A robot is required to move in the world $\mathcal{W}$.

- $\mathcal{C}$ is defined as a configuration space, which is a special case of state space $\mathcal{X}$. It is determined by every transformation applicable to the robot $\mathcal{A}$. From $\mathcal{C}$ we derive $\mathcal{C}_{obs}$ and $\mathcal{C}_{free}$. $q \in \mathcal{C}$ is called a configuration and $x \in \mathcal{X}$ is called a state.

- $q_{init} \in \mathcal{C}_{free}$ is the initial configuration.

- $q_{goal} \in \mathcal{C}_{free}$ is the goal configuration.

- $\tau : [0, 1] \rightarrow \mathcal{C}_{free}$ is a continuous path, such that $\tau(0) = q_{init}$ and $\tau(1) = q_{goal}$. Otherwise it reports, that path does not exist.

2.6 Rapidly Exploring Random Trees

The rapidly exploring random tree (RRT) algorithm [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.

The data structure Rapidly-Exploring Random Tree (RRT) is constructed from vertices (nodes), where all vertices are states $x \in \mathcal{X}_{free}$, and edges, where each each edge is a path, which lies in $\mathcal{X}_{free}$.

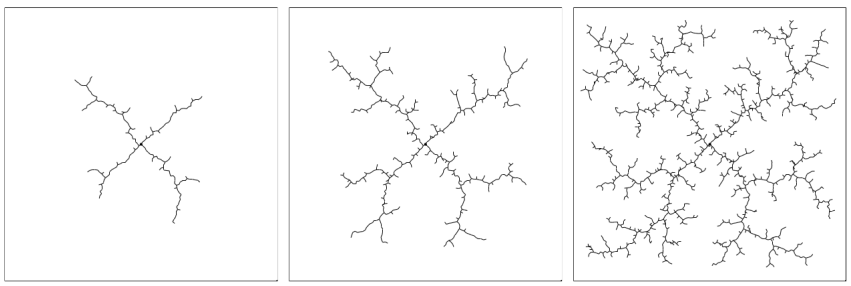


Figure 2.1: Rapidly-Exploring Random Tree (picture from [L+98])

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

Following the Algorithm 3, we begin generating the data structure Rapidly-Exploring Random Tree (RRT) T with the initial state $x_{init} \in \mathcal{X}_{free}$, K vertices and time interval Δt.
The algorithm enters a for loop. In the loop, a random state x_{rand} is selected from state space $\mathcal{X}$. Next, NEAREST_NEIGHBOUR finds the vertex x_{near} from T that is closest to x_{rand}. SELECT_INPUT returns a vector u called input, which minimizes the distance from x_{near} towards x_{rand} while ensuring, that the state remains in $\mathcal{X}_{free}$. NEW_STATE applies u on x_{near} and returns a new state x_{new}. x_{new} and the edge from x_{near} to x_{new} alongside with the input u are added to T.
The loop is terminated when the number of vertices in T reaches K.

As mentioned in [LaV06], the algorithm can be simply modified to be used for solving planning problems. The tree would be initialized with the initial state x_{init} of the problem. With a probability p, RANDOM_STATE would still sample a random state x_{rand}. With a probability $1-p$, RANDOM_STATE would be replaced with $x_{rand} \leftarrow x_{goal}$.

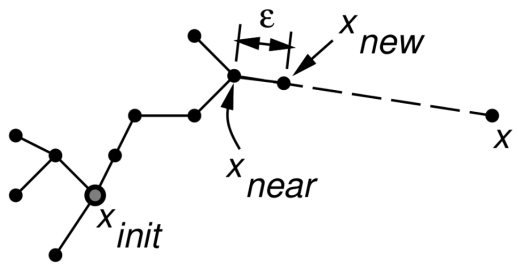


Figure 2.2: The extend phase of RRT algorithm (picture from [LKD+01])

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 by replacing the distance metric with heuristic estimate of the cost-to-go, that is used in general informed search methods.

The algorithm described in Algorithm 4 starts with an initial state s_0. A tree T is initialized with s_0. A for loop set to run N times is entered. At each step, a random state s_{rand}, which is not present in the tree is selected.

Then we use extendRRT function. We find a nearest state s_{near} based on a heuristic estimate of the cost-to-go from each state to s_{rand}. Every operator is applied on s_{near}. The resulting state, which is closest to s_{rand} and is not present in the tree, becomes s_{new}. The state s_{new} is added to the tree and is connected to s_{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. The modification can be seen in a function extendRRLT in Algorithm 4.

Algorithm 4: GrowRRT (from [MB04])

```
Data: s_0
T.init(s_0);
for n = 1 to N do
    s_rand = randomUnexploredState();
    extendRRT(s_rand, T);
end

Function extendRRT(s_rand, T):
    s_near = T.nearestTreeNode(s_rand) ;
    if s_near.hasUnseenSucessors() then
        s_new = nearestSucessor(s_rand, s_near);
        T.addChildNode(s_new, s_near);
    end

Function extendRRLT(s_rand, T):
    s_new = T.nearestTreeLeaf(s_rand) ;
    T.changeLeafToNode(s_new);
    T.addNewLeaves(s_new);
```

3.1.2 RRTs with Local Planners

RRTs with local planners use the Algorithm 4 from the previous section as a global planner, but use a different planner for local planning. Instead of simply picking successors closest to s_{rand}, a local planner limited by depth, size or time is used from s_{near} to s_{rand}. When the local search is finished, the node closest to s_{rand} and the whole sequence of nodes from s_{near} to s_{rand} 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 based on Rapidly exploring Random Trees concepts.

To select a random state, the RRT-Plan algorithm generates possible s_{rand} states by taking random subsets from the goal G, as if the problem can be solved, it provides us with reachable states. It also helps to direct the search towards a goal.

After obtaining a s_{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 s_{near} to s_{rand}. The authors have chosen the Enforced Hill Climbing phase of FF [Hof01] as a local planner, and decided to set a node expansion limit ϵ. After reaching the limit, the search is terminated.

When the planner achieves s_{rand} from s_{near}, s_{rand} becomes s_{new} and is added to the tree as a child of s_{near}.

In the next step, the algorithm attempts to connect s_{new} to the goal G using the same planner that was used during the attempt to connect s_{near} and s_{rand}. This time, in case of failure to reach the state, the search would not be abandoned completely, but the state with the best value will be added to the tree as a new node.

RRT-Plan also uses other techniques such as goal subset locking and adapting search paramterers to improve its performance.

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 same outline of the algorithm in our implementation,

we will focus on the description more in the next chapter.

The authors chose the SAS+ [BN95] representation of the problem and use its properties in sampling.

To estimate the distance, the planner uses heuristic function, which computes the cost of cached best supporters, where best supporters can be defined in STRIPS as operators, which first achieve an atom.

The planner implemented 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 5: RRT for discrete space in FDR (from [AVB11])

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

Chapter 4

Description of the Algorithm

This chapter describes the Algorithm 5, which was proposed in [AVB11], 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 S, limit for local planner ϵ, initial state s_{init}, partial goal state s_{goal} and cost function c. The tree T is initialized with the initial state. A while loop is entered and based on the random value, one of the following happens:

- With probability $1 - p$, the algorithm samples the space and returns sampled random state s_{rand}. For s_{rand}, we find a nearest state s_{near} in the tree T. After that, we attempt to join s_{rand} and s_{near} using a local planner with the limit of expansion ϵ. If s_{rand} is reached within the limit ϵ, it is returned from the join function as s_{new}. Otherwise, a state with the best heuristic value encountered during the local search is returned as s_{new}. The state s_{new} is added into the tree as a child of s_{near}. A new state $s_{near_{goal}}$ is set to s_{new}.
- With probability p, for a partial state s_{goal}, we find a nearest state $s_{near_{goal}}$ in the tree T.

In the next step, we attempt to join the state $s_{near_{goal}}$ with s_{goal}. The state $s_{new_{goal}}$ returned by the function is then added to the tree T as a child of $s_{near_{goal}}$.

The loop repeats until a solution is found.

4.1 Sampling

Algorithm 6: Sample space

Data: $V, O, s_{init}, s_{goal}, n, M, c$
$reachableInitToState \leftarrow false$;
$reachableStateToGoal \leftarrow false$;
while $\neg reachableInitToState\ OR\ \neg reachableStateToGoal$ **do**
  $s \leftarrow samplingMethod(V, n, (M))$;
  $reachableInitToState \leftarrow isReachable(V, O, c, s_{init}, s)$;
  $reachableStateToGoal \leftarrow isReachable(V, O, c, s, s_{goal})$;
end
return s

According to Algorithm 6, we need set of variables V, set of operators O, initial state s_{init}, partial goal state s_{goal}, number of variables n and set of mutexes M. We set variables reachableInitToState and reachableStateToGoal to false and enter the while loop. We choose a sampling method, either random sampling or mutex sampling. The sampling method returns state s. 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.

In our implementation, we observed how many times the algorithm sampled an unreachable state and if the number rose above 10,000, we let the sampling return an unreachable state.

4.1.1 Random sampling

If we decided to sample the space by choosing random number of atoms, and then also random atoms 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 advantage 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.

Algorithm 7: Random sampling

Data: V, n
for $i = 1$ *to* n **do**
  $v_i \leftarrow returnVariable(V, i)$;
  $val_i \leftarrow chooseRandomValue(D_{v_i})$;
  $s.addFact(v_i, val_i)$;
end
return s

For random sampling in FDR, we need a set of variables V, n is a number of variables in V. Each variable $v_i \in V$ is assign a random value from the domain D_{v_i}. The fact $\langle v_i, D_{v_i} \rangle$ is then assigned to s until every variable has value. The state s is returned.

4.1.2 Mutex sampling

To make to the previous method more strict and hopefully more effective in regards to sampling more reachable states, we add additional mutexes:

- **lifted.** Mutex groups obtained during preprocessing and translation of FDR.
- **h_2.** Additionally computed h_2 mutexes.
- **h_2 forward backward** ($h_2 fwbw$) Additionally computed h_2 mutexes both forwards and backwards.
- **h_3.** Additionally computed h_3 mutexes.
- **fam-groups.** Additionally computed fact-alternating mutex groups.

Algorithm 8: Mutex sampling

```
Data: V, n, M
order ← makeOrder(n);
s.setEmptyValues;
for i = 1 to n do
    r ← getOrder(order, i);
    v_r ← returnVariable(V, r);
    while isMutex(s, M) OR s.variableNotAssignedValue(v_r) do
        val_r ← chooseRandomValue(D_{v_r});
        s.addFact(v_r, val_r) ;
        if noPossibleSolution() then
            order ← makeOrder(n);
            s.setEmptyValues();
            i ← −1;
            continue;
        end
    end
end
return s
```

We have the set of variables V, number of variables n, and set of mutexes M. Mutex sampling method (influenced by [AVB11]) starts by choosing random order for n variables and making sure that no values are assigned to variables in s. We enter a for loop.

A variable v_r is selected in that given order. Since no value is assigned in s for v_r, we enter the while loop. A value is randomly selected for variable v_r and added into s. If we added a value, which is mutex with the rest of (partial) state s, we try again. If there is no possible solution and the algorithm is stuck, we start the process again with a new order. If all variables received a correctly assign value, the algorithm returns the state s.

4.2 Search for a nearest state

Search for a nearest state relies on the ability to determine distance. In our implementation, we use a heuristic function h_{FF} to estimate the distance between two states. The state with the lowest heuristic value is the nearest state.

4.3 Join

The join stage tries to connect the sampled state s_{rand} with the nearest state s_{near} in a limited number of steps using a local planner. 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. In case several states have the same heuristic value, the most recent state, which is not present in the tree size, is chosen. We have chosen greedy search with lazy evaluation as the local planner, where h_{FF} is the heuristic function.

Algorithm 9: Join

```
Data: V, O, s_1, s_2, c, ε, T
minValue ← ∞;
minState ← s_1;
planner ← newLocalPlanner(V, O, s_1, s_2, c);
currentState ← localPlannerInitStep(planner);
for step = 1 to ε do
    currentState ← localPlannerStep(planner);
    heuristicValue ← heuristicEstimate(V, O, currentState, s_2, c);
    if isGoal(currentState, s_2) then
        minState ← currentState;
        break;
    end
    if heuristicValue < minValue then
        minState ← currentState;
        minValue ← heuristicValue;
    end
    if heuristicValue == minValue AND
     notInTree(T, currentState) then
        minState ← currentState;
    end
end
return minState
```

The algorithm as described starts with set of variables V, set of operators O, initial state s_1, (partial) state goal s_2, cost function c, limit ϵ and tree T.

At first, we initialize variables $minValue$ to hold infinity, $minState$ to be s_1 and then we initialize the local planner. In the next step, we enter the for loop restricted by the limit ϵ. Local planner performs a step and returns a current state $currentState$, for which we estimate its value $heuristicValue$ by using a heuristic function.

If $currentState$ is s_2, our search found the required solution and returns $currentState$.
If $heuristicValue$ is lower then the minimal value $minValue$, $currentState$ is set to be the new minimal state $minState$ and its value $heuristicValue$ becomes $minValue$.
If $heuristicValue$ is equal to the minimal value $minValue$ and $currentState$ is not in the tree T, then $currentState$ is the new minimal state $minState$ and its value $heuristicValue$ becomes $minValue$.

If the search terminated and the goal was not reached, the algorithm returns the state $minState$ with minimal 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 chosen from the satisficing track at the IPC (International planning competition). 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.

The distance in search for a nearest state was estimated by h^{FF}. Every time a planner samples a new state, its reachability from the initial state and towards the goal is verified using h^{max}. The probability p was set to 0.5.

To properly compare properties of the algorithm, we prepared several configurations. Every configuration has two variables:

- A sampling method - either random sampling, or mutex sampling with lifted, h_2, h_2fwbw, h_3, fam mutexes.
- The maximal number of steps for the local planner ϵ - 1,000, 10,000 and 100,000.

In the end, we are left with 18 configurations of RRT algorithm. Since the proposed algorithm relies on the element of randomization, every configuration will run 30 times.

5.1 Results

To put performance of our newly implemented planner into context, we compare the results of RRT planner with a commonly used greedy algorithm with lazy evaluation implemented in the cpddl library `https://gitlab.com/danfis/cpddl-dev`. Apart from the number of found plans, we examine other various properties of the search. We are able to observe the length and runtime of plans generated by the greedy planner as well as the planner based on RRT. Our planner was not intended to be optimal, yet it is still interesting to see the lengths of plans found by different planners side by side. In addition to that, the RRT solver allows us to study the size of the tree used to discover the plan, and a number of sampling attempts, which were discarded during the search.

5.1.1 Number of succesfully found plans

The table C.1 shows number of successfully found plans by the planner using the greedy algorithm. It also informs us about how the situation would look like, if we were to run only the local planner. For the RRT configurations, the tables C.2 - C.7 display the mean and standard deviation of number of found plans over the 30 iterations.

The configuration using fact-alternating mutex groups with the limit of steps in local planner being 100,000 was the most successful planner with 612.80 as its average number of plans. It is very clear, that no matter which sampling method was used, the planners with limit set to 100,000 steps in local search produced better results. Apart from configuration of sampling method h_3, every configuration with limit set to 100,000 performed better in regards to found number of plans.

None of the solvers (including the greedy solver) was able to complete any problem from the domains elevators11, ged14 and visitall14. The RRT planners struggle on long searches with large state space and result in timeout.

Furthermore, visitall is a domain known to perform better with different heuristic function than h^{FF}.

Planner configuration using h^3 mutex groups finishes unsuccessfully in several problems (e.g., parking14) while trying to obtain additional mutexes without even starting the search.

Some of the domains (e.g., mystery98) contain problems that were deemed unsolvable during translation and pruning of FDR. Resulting lower number of found plans may negatively affect the impression of the planners, even though some configurations were able to discover a plan for every solvable problem.

5.1.2 Length, time, attempts, tree size

Since it is more appropriate to analyze the length, time, discarded sampling attempts and tree size separately for every problem, this section mentions only selected domains.

The complete results containing the mean and standard deviation of length, time, attempts and tree size for every domain as well as the number of successfully discovered plans can be found in the attached directory "results.zip".

Length

Length of a plan is a number of steps needed from the initial state to achieve a goal. The average length and its standard deviation for the gripper98 domain can be found in the table C.8. Every planner used in this thesis was able to find a plan for every problem in every iteration.

All the results seem to be very similar for most of the planners, with the exception of configurations with the limit in local search set to 1,000. For larger problems of the domain, these configurations are slightly higher and their standard deviation is relatively elevated as well. This is no surprise, since they introduce random values more often than other.

Gripper98	greedy	random			lifted		
Problems		1000	10000	100000	1000	10000	100000
prob01	13	12.93±0.81	12.53±0.85	12.73±0.85	12.67±0.75	12.67±1.04	12.93±0.96
prob02	21	20.53±1.23	20.47±0.88	20.27±1.09	20.27±0.96	19.80±0.98	20.33±0.94
prob03	29	28.20±0.98	27.60±1.56	27.93±1.44	28.13±0.99	27.67±0.94	28.07±1.00
prob04	37	36.00±1.24	35.13±1.86	35.73±1.59	35.80±0.98	35.60±0.92	35.93±1.00
prob05	45	44.13±1.12	43.47±1.77	43.73±2.03	44.00±1.00	43.53±0.88	44.00±1.00
prob06	53	52.20±0.98	51.80±1.68	52.00±1.91	51.80±0.98	51.40±0.80	52.00±1.00
prob07	61	60.40±1.05	60.00±1.44	60.40±1.38	59.80±0.98	59.40±0.80	60.00±1.00
prob08	69	68.47±1.02	68.40±1.05	68.47±1.36	67.80±0.98	67.60±0.92	68.07±1.00
prob09	77	76.67±0.91	76.40±1.17	76.87±0.72	75.80±0.98	75.60±0.92	75.93±1.00
prob10	85	84.53±0.85	84.53±0.99	84.73±1.00	84.40±1.28	83.80±0.98	84.33±0.94
prob11	93	92.67±0.75	92.67±1.27	92.93±0.36	92.20±1.22	91.80±0.98	92.53±0.85
prob12	101	115.27±30.03	100.87±0.72	101.00±0.00	113.67±27.25	100.60±1.40	100.60±0.80
prob13	109	116.93±19.61	108.93±0.36	108.93±0.36	111.13±12.30	108.67±0.75	108.67±0.75
prob14	117	116.60±0.95	117.07±0.36	117.00±0.00	118.27±8.06	116.80±0.60	116.60±0.80
prob15	125	130.67±18.64	124.87±0.50	125.00±0.00	126.53±9.78	124.93±0.63	125.13±0.50
prob16	133	136.33±10.50	132.80±0.79	132.67±1.27	133.53±4.79	132.93±0.96	132.93±0.36
prob17	141	145.07±16.47	140.80±0.79	141.00±0.00	141.20±8.98	141.07±0.36	141.07±0.81
prob18	149	158.60±23.61	148.93±0.36	148.87±0.72	159.27±25.02	149.00±0.00	149.27±1.12
prob19	157	161.47±11.49	156.73±1.00	156.87±0.50	163.93±16.66	157.00±0.00	157.20±1.08
prob20	165	172.13±19.10	164.73±0.85	164.93±0.63	178.93±28.73	165.13±0.72	165.00±0.00

Figure 5.1: A section of the table C.8 showing average length and standard deviation.

Time

The table C.9 displays the average time needed to find a plan and its standard deviation for the zenotravel02 domain. The average number of plans in this domain for every planner is above 17 plans.

Zenotravel02	greedy	random			lifted		
Problems		1000	10000	100000	1000	10000	100000
pfile10	0.14	0.19±0.13	0.24±0.12	0.23±0.15	0.19±0.16	0.28±0.24	0.19±0.12
pfile11	0.05	0.15±0.07	0.10±0.03	0.14±0.04	0.18±0.14	0.13±0.06	0.14±0.09
pfile12	0.04	0.11±0.05	0.10±0.04	0.13±0.05	0.12±0.09	0.13±0.11	0.13±0.07
pfile13	0.03	0.11±0.07	0.10±0.04	0.17±0.25	0.10±0.06	0.13±0.12	0.12±0.05
pfile14	1.64	3.96±3.29	3.28±1.70	7.36±11.24	6.50±5.04	2.83±1.62	2.90±1.52
pfile15	24.74	3.99±2.68	9.95±8.31	34.59±46.49	3.51±2.20	7.93±6.24	51.22±54.46
pfile16	1.11	19.68±30.15	7.45±8.23	12.92±31.71	23.36±29.85	9.22±7.84	5.95±5.05
pfile17	4.49	177.49±443.90	17.55±8.80	18.30±13.58	46.93±87.83	19.62±18.35	15.48±4.96
pfile18	15.16	305.88±447.66	28.59±23.02	53.81±37.22	106.23±155.81	41.71±34.30	39.63±14.53
pfile19	5.99	157.58±172.34	47.51±35.43	36.38±28.62	559.00±632.02	59.77±69.54	26.05±13.87

Figure 5.2: A section of the table C.9 showing average time(s) and standard deviation.

The solver using the greedy algorithm the best. Its maximal time spent on a problem is 24.74 seconds, while maximal values for RRT planners are over 50s. Configurations using h_3 sampling method need over 1300 seconds for their worst case. One interesting problem is the pfile15 problem. In this case, most of the configurations limited to 1,000 and even 10,000 steps finish faster than the greedy planner. But in other cases, the greedy planner vastly outperforms planners based on RRT.

Attempts

The average of discarded attempts during sampling for the domain parking11 and its standard deviation can be found in the table C.10. In this domain, configurations with h_3 manage to start the search only in the first two problems. On average most of the planners with the limit set to 100,000 solve at least 19 problems, while planners limited to 1,000 steps perform worse.

Parking11	random			lifted		
Problems	1000	10000	100000	1000	10000	100000
pfile08-031	52959.47±27414.83	7000.70±8226.80	3713.50±4692.64	237.60±278.57	7.80±8.17	5.63±7.39
pfile08-032	26969.39±27263.81	6000.60±6633.91	4519.00±4851.89	70.73±84.67	6.50±10.70	3.47±5.12
pfile09-033	60006.00±0.00	14334.77±11161.56	5411.63±4923.38	390.17±340.90	10.57±11.19	4.50±7.24
pfile09-034	no results	18870.27±14225.91	5125.17±4920.26	388.21±316.30	13.90±18.38	3.10±5.69
pfile09-035	38411.20±30078.62	14385.43±10353.70	5410.93±4923.82	285.73±278.04	12.63±14.54	4.70±7.59
pfile09-036	47782.56±22001.08	11044.73±7838.43	4344.00±4947.20	231.90±209.74	8.63±12.39	2.77±4.63
pfile10-037	17231.10±17224.16	11035.59±12134.05	5081.33±4938.23	386.58±377.37	9.13±12.05	3.57±5.63
pfile10-038	25002.50±5000.50	16061.73±10344.68	4333.77±4955.85	395.84±265.00	20.73±23.04	6.97±11.60

Figure 5.3: A section of the table C.10 showing average number of states discarded during samplin and standard deviation.

In this domain it is more common to sample a unreachable state and the results between the configurations with 1,000, 10,000 and 100,000 are very noticeable. Those, that are required to sample more often discard more states. Configurations with random sampling method don't restrict their generated states with mutexes and often return unreachable state resulting in very large number of discarded states.

Tree size

The table C.11 shows the average tree size and its standard deviation for the tetris14 domain. Planners with the limit of 1,000 steps find around 10 plans, other configurations around 17 plans.

Tetris14	h2fwbw		
Problems	1000	10000	100000
p020	32.83±29.04	17.03±13.17	10.40±6.95
p021	102.80±68.78	43.10±45.11	26.25±19.19
p022	92.27±70.23	30.53±32.44	11.30±8.86
p023	72.57±70.66	49.69±39.54	16.63±14.15
p024	107.12±48.98	52.50±53.94	7.47±4.22
p025	71.29±62.79	42.10±42.34	9.10±11.15
p026	160.54±102.75	51.38±50.41	17.00±12.00
p027	133.79±80.66	45.59±42.02	11.93±10.28
p028	112.59±68.25	14.87±15.83	6.37±6.95
p029	57.75±25.59	47.80±47.17	24.21±17.16

Figure 5.4: A section of the table C.11 showing average tree size and standard deviation.

Tree size values mainly show, that configurations with lower limit of steps have bigger trees. Since a node is not added into the tree during the local search, planners with longer local search don't add new nodes as often.

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:

- sampling method:
  - Random sampling. Apart from variables in FDR no additional mutex groups added.
  - Mutex sampling.
    - Lifted. Mutex groups obtained during preprocessing and translation.
    - h_2. Additionally computed h_2 mutexes.
    - $h_2 fwbw$. Additionally computed h_2 mutexes forward and backward.
    - h_3. Additionally computed h_3 mutexes.
    - Fam. Additionally computed fact-alternating groups.
- limited number of steps in the local search:
  - 1,000
  - 10,000

- 100,000

For each configuration of the planner with chosen sampling method and limit of steps the local planner was allowed to take, we studied the results. We examined number of found plans and other properties such as length of the plan or runtime. We then compared our planner with a planner using greedy algorithm.

Based on the chosen configuration, our solver can yield better results than 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.zip** - directory containing the planner built on top of cpddl library
- **dataset.zip** - dataset used in this thesis
- **README.zip** - file describing how to run the planner
- **results.zip** - directory containing tables with results in .ods and .xlsx format

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.

[AHK+98] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony Barrett, Dave Christianson, et al., *Pddl| the planning domain definition language*, Tech. report, Technical report, 1998.

[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.

[BN95] Christer Bäckström and Bernhard Nebel, *Complexity results for sas+ planning*, Computational Intelligence **11** (1995), no. 4, 625–655.

[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.

[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.

Appendix C

Tables

Note: The pipesworld-notankage04 domain will be reffered to as pipesworld in all of the tables.*

C.1 Average number of successfully completed plans

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*	(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 C.1: Number of plans achieved by the greedy algorithm.

Domains		1000	10000	100000
agricola18	(20)	6.33±0.65	6.77±0.62	7.07±1.00
barman11	(20)	0.00±0.00	0.00±0.00	0.13±0.34
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)	12.80±0.60	14.00±0.00	14.60±0.80
cavediving14	(20)	0.00±0.00	7.00±0.00	7.00±0.00
childsnack14	(20)	0.00±0.00	0.00±0.00	0.03±0.18
data-network18	(20)	1.63±0.48	2.97±0.75	4.47±1.02
depot02	(20)	12.97±0.71	15.27±0.85	16.07±0.77
driverlog02	(20)	15.53±0.50	18.27±0.73	18.83±0.52
elevators11	(20)	0.00±0.00	0.00±0.00	0.00±0.00
floortile11	(20)	5.83±0.37	6.10±0.30	7.07±0.44
floortile14	(20)	2.00±0.00	2.13±0.34	2.30±0.53
freecell00	(20)	19.30±0.64	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)	10.10±0.91	14.63±0.71	18.37±0.66
maintenance14	(20)	11.03±1.20	14.93±0.36	13.63±0.55
mprime98	(20)	19.30±0.69	18.60±0.71	17.97±0.71
mystery98	(20)	12.97±0.18	13.00±0.00	12.97±0.18
nomystery11	(20)	10.07±0.85	10.30±0.74	10.40±0.76
openstacks06	(20)	20.00±0.00	20.00±0.00	20.00±0.00
parking11	(20)	2.17±1.07	15.97±1.54	19.70±0.64
parking14	(20)	0.07±0.25	3.67±1.40	15.23±1.15
pegsol11	(20)	18.23±0.92	18.40±0.76	18.83±0.64
pipesworld*	(20)	20.00±0.00	20.00±0.00	20.00±0.00
rovers06	(20)	18.27±0.51	20.00±0.00	20.00±0.00
satellite02	(20)	17.13±0.99	19.63±0.48	19.23±0.72
scanalyzer11	(20)	16.77±0.92	17.63±1.25	19.27±0.73
snake18	(20)	3.90±0.65	6.20±0.75	6.43±0.84
sokoban11	(20)	6.33±0.83	10.20±1.19	15.07±1.03
spider18	(20)	5.77±1.20	9.87±0.43	10.93±0.25
storage06	(20)	20.00±0.00	20.00±0.00	20.00±0.00
termes18	(20)	6.13±1.28	14.77±0.62	16.67±0.70
tetris14	(20)	10.30±1.44	17.07±1.26	18.43±0.99
thoughtful14	(20)	17.53±0.81	18.13±0.62	17.03±0.84
tidybot11	(20)	17.60±0.66	18.17±0.64	18.60±0.84
tpp06	(20)	13.03±0.48	15.60±0.61	17.57±0.67
trucks06	(20)	16.87±0.92	18.53±0.72	18.57±0.50
visitall11	(20)	1.20±0.48	4.03±0.71	6.27±0.68
visitall14	(20)	0.00±0.00	0.00±0.00	0.00±0.00
woodworking11	(20)	13.97±1.11	19.50±0.72	18.53±0.85
zenotravel02	(20)	19.13±0.76	20.00±0.00	20.00±0.00
SUM	(900)	484.27±4.43	571.33±4.23	607.27±3.82

Table C.2: Average number of plans achieved by using the RRT algorithm with a random sampling method.

Domains		1000	10000	100000
agricola18	(20)	6.33±0.54	6.57±0.62	7.23±0.99
barman11	(20)	0.00±0.00	0.23±0.42	1.97±1.17
barman14	(20)	0.00±0.00	0.00±0.00	0.27±0.44
blocks00	(20)	20.00±0.00	20.00±0.00	20.00±0.00
caldera18	(20)	12.10±1.11	13.80±0.48	14.63±0.84
cavediving14	(20)	0.00±0.00	6.97±0.18	7.00±0.00
childsnack14	(20)	0.00±0.00	0.00±0.00	0.07±0.25
data-network18	(20)	1.60±0.49	2.50±0.62	4.00±0.89
depot02	(20)	12.87±0.96	14.80±0.87	16.03±0.84
driverlog02	(20)	15.47±0.56	18.03±0.84	18.70±0.59
elevators11	(20)	0.00±0.00	0.00±0.00	0.00±0.00
floortile11	(20)	6.03±0.18	6.27±0.44	7.23±0.42
floortile14	(20)	2.03±0.18	2.10±0.30	2.73±0.73
freecell00	(20)	19.37±0.60	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)	10.13±0.92	14.67±0.83	18.60±0.84
maintenance14	(20)	10.87±1.20	15.03±0.18	13.73±0.93
mprime98	(20)	19.27±0.81	18.60±0.76	17.53±0.96
mystery98	(20)	12.93±0.25	13.00±0.00	12.77±0.42
nomystery11	(20)	9.90±0.91	10.10±0.47	10.37±0.66
openstacks06	(20)	20.00±0.00	20.00±0.00	20.00±0.00
parking11	(20)	12.67±1.96	19.63±0.66	19.87±0.34
parking14	(20)	1.83±1.10	11.00±2.25	15.67±2.17
pegsol11	(20)	17.70±1.35	18.23±1.20	19.13±0.67
pipesworld*	(20)	20.00±0.00	20.00±0.00	20.00±0.00
rovers06	(20)	18.13±0.34	20.00±0.00	20.00±0.00
satellite02	(20)	17.13±0.88	19.60±0.55	18.90±0.75
scanalyzer11	(20)	17.77±0.72	18.23±0.72	19.17±0.82
snake18	(20)	3.83±0.86	6.13±0.50	6.67±1.01
sokoban11	(20)	6.63±0.87	10.23±0.96	15.10±0.98
spider18	(20)	5.80±1.28	9.63±0.60	10.93±0.25
storage06	(20)	19.97±0.18	20.00±0.00	20.00±0.00
termes18	(20)	6.57±1.26	14.77±0.56	16.73±0.63
tetris14	(20)	9.87±1.91	17.90±1.19	18.77±0.96
thoughtful14	(20)	18.03±0.71	18.47±0.62	17.17±0.97
tidybot11	(20)	17.70±0.53	18.30±0.69	18.43±0.80
tpp06	(20)	13.13±0.56	15.53±0.50	17.57±0.88
trucks06	(20)	16.77±0.80	18.53±0.62	18.60±0.71
visitall11	(20)	1.17±0.45	4.23±0.76	5.97±0.80
visitall14	(20)	0.00±0.00	0.00±0.00	0.00±0.00
woodworking11	(20)	13.80±1.19	19.40±0.66	18.63±0.75
zenotravel02	(20)	18.93±0.77	20.00±0.00	20.00±0.00
SUM	(900)	496.33±5.50	582.50±4.46	610.17±4.36

Table C.3: Average number of plans achieved by the RRT algorithm with a sampling method using lifted mutexes.

Domains		1000	10000	100000
agricola18	(20)	6.40±0.61	6.80±0.60	7.47±0.76
barman11	(20)	0.00±0.00	0.13±0.34	1.83±0.78
barman14	(20)	0.00±0.00	0.00±0.00	0.47±0.56
blocks00	(20)	20.00±0.00	20.00±0.00	20.00±0.00
caldera18	(20)	11.07±0.93	13.93±0.36	13.43±1.33
cavediving14	(20)	0.00±0.00	7.00±0.00	7.00±0.00
childsnack14	(20)	0.00±0.00	0.00±0.00	0.10±0.30
data-network18	(20)	1.70±0.46	2.73±0.68	3.80±1.01
depot02	(20)	12.90±0.98	14.77±0.88	16.33±0.75
driverlog02	(20)	15.63±0.48	18.03±0.98	18.70±0.64
elevators11	(20)	0.00±0.00	0.00±0.00	0.00±0.00
floortile11	(20)	6.07±0.44	6.27±0.51	6.97±0.71
floortile14	(20)	2.00±0.00	2.27±0.51	2.50±0.67
freecell00	(20)	19.47±0.50	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)	10.20±1.19	14.57±0.96	18.70±0.69
maintenance14	(20)	10.83±1.13	15.00±0.26	14.00±0.82
mprime98	(20)	19.03±0.66	18.73±0.93	17.90±1.04
mystery98	(20)	12.80±0.40	13.00±0.00	12.90±0.30
nomystery11	(20)	10.00±0.93	10.37±0.87	10.43±0.80
openstacks06	(20)	20.00±0.00	20.00±0.00	20.00±0.00
parking11	(20)	12.73±1.61	19.70±0.53	19.93±0.25
parking14	(20)	1.93±1.15	13.07±2.46	15.80±1.40
pegsol11	(20)	17.93±0.93	18.57±0.67	18.97±0.80
pipesworld*	(20)	20.00±0.00	20.00±0.00	20.00±0.00
rovers06	(20)	18.43±0.50	20.00±0.00	20.00±0.00
satellite02	(20)	17.23±0.56	19.73±0.51	18.83±0.86
scanalyzer11	(20)	17.80±0.79	18.37±0.71	19.50±0.72
snake18	(20)	4.27±0.85	6.20±0.48	6.40±0.66
sokoban11	(20)	6.07±0.96	10.50±1.23	13.70±1.19
spider18	(20)	6.17±1.00	9.63±0.66	10.87±0.34
storage06	(20)	20.00±0.00	20.00±0.00	20.00±0.00
termes18	(20)	6.00±1.21	14.60±0.61	17.03±0.66
tetris14	(20)	9.97±2.04	17.00±1.95	18.37±0.98
thoughtful14	(20)	18.00±0.68	18.33±0.47	17.03±0.66
tidybot11	(20)	17.57±0.56	18.00±0.52	18.23±0.76
tpp06	(20)	13.00±0.68	15.57±0.62	17.47±0.88
trucks06	(20)	16.70±0.69	18.83±0.90	18.73±0.57
visitall11	(20)	1.17±0.58	3.90±0.75	6.20±0.83
visitall14	(20)	0.00±0.00	0.00±0.00	0.00±0.00
woodworking11	(20)	14.20±0.83	19.57±0.62	18.77±0.76
zenotravel02	(20)	19.00±0.68	20.00±0.00	20.00±0.00
SUM	(900)	496.27±4.51	585.17±4.36	608.37±3.59

Table C.4: Average number of plans achieved by the RRT algorithm with a sampling method using h_2 mutexes.

Domains		1000	10000	100000
agricola18	(20)	6.37±0.66	6.67±0.54	7.60±1.05
barman11	(20)	0.00±0.00	0.27±0.44	1.93±1.03
barman14	(20)	0.00±0.00	0.00±0.00	0.67±0.65
blocks00	(20)	20.00±0.00	20.00±0.00	20.00±0.00
caldera18	(20)	11.20±0.98	13.73±0.77	14.73±0.68
cavediving14	(20)	7.00±0.00	7.00±0.00	7.00±0.00
childsnack14	(20)	0.00±0.00	0.00±0.00	0.10±0.30
data-network18	(20)	1.60±0.49	2.77±0.72	4.00±1.10
depot02	(20)	12.90±1.04	14.40±0.80	16.03±0.66
driverlog02	(20)	15.57±0.56	17.97±1.02	18.67±0.60
elevators11	(20)	0.00±0.00	0.00±0.00	0.00±0.00
floortile11	(20)	6.00±0.26	6.20±0.48	7.27±0.51
floortile14	(20)	2.13±0.34	2.20±0.48	2.83±0.90
freecell00	(20)	19.40±0.71	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)	10.07±0.96	14.63±1.02	18.83±0.93
maintenance14	(20)	10.83±1.04	15.03±0.41	13.93±0.77
mprime98	(20)	19.03±0.66	18.83±0.73	17.90±0.83
mystery98	(20)	12.90±0.30	13.00±0.00	12.87±0.34
nomystery11	(20)	10.03±0.80	10.43±0.67	10.23±0.76
openstacks06	(20)	20.00±0.00	20.00±0.00	20.00±0.00
parking11	(20)	13.00±1.46	19.77±0.50	19.93±0.25
parking14	(20)	1.97±1.14	11.50±1.91	15.33±1.30
pegsol11	(20)	17.60±1.05	18.33±1.01	18.53±1.06
pipesworld*	(20)	20.00±0.00	20.00±0.00	20.00±0.00
rovers06	(20)	18.37±0.66	20.00±0.00	20.00±0.00
satellite02	(20)	17.00±0.97	19.63±0.55	18.90±0.91
scanalyzer11	(20)	17.87±0.81	18.10±0.91	18.83±0.78
snake18	(20)	3.83±0.73	5.90±0.75	6.43±0.84
sokoban11	(20)	6.13±1.18	10.60±1.02	13.87±0.99
spider18	(20)	6.43±0.96	9.83±0.37	10.87±0.34
storage06	(20)	20.00±0.00	20.00±0.00	20.00±0.00
termes18	(20)	6.53±1.02	14.77±0.67	16.93±0.57
tetris14	(20)	9.47±1.73	16.83±1.24	18.70±0.94
thoughtful14	(20)	18.03±0.75	18.20±0.65	17.13±0.76
tidybot11	(20)	17.77±0.42	18.17±0.64	18.43±0.99
tpp06	(20)	12.97±0.48	15.77±0.72	17.57±0.76
trucks06	(20)	16.67±0.75	18.57±0.72	18.53±0.72
visitall11	(20)	1.17±0.45	4.23±0.80	6.17±0.93
visitall14	(20)	0.00±0.00	0.00±0.00	0.00±0.00
woodworking11	(20)	13.43±1.12	19.33±0.83	18.40±0.95
zenotravel02	(20)	19.37±0.75	20.00±0.00	20.00±0.00
SUM	(900)	502.63±4.68	582.67±4.53	609.17±5.39

Table C.5: Average number of plans achieved by the RRT algorithm with a sampling method using $h2$ forward backward mutexes.

Domains		1000	10000	100000
agricola18	(20)	6.40±0.66	6.50±0.50	7.40±0.76
barman11	(20)	0.00±0.00	0.40±0.49	2.23±0.96
barman14	(20)	0.00±0.00	0.00±0.00	0.33±0.70
blocks00	(20)	20.00±0.00	20.00±0.00	20.00±0.00
caldera18	(20)	7.80±0.60	9.27±0.68	9.70±0.46
cavediving14	(20)	0.03±0.18	7.00±0.00	7.00±0.00
childsnack14	(20)	0.00±0.00	0.00±0.00	0.03±0.18
data-network18	(20)	1.73±0.44	2.87±0.85	3.80±1.01
depot02	(20)	12.90±0.91	14.77±0.96	15.73±0.85
driverlog02	(20)	15.43±0.56	17.50±0.92	17.80±0.75
elevators11	(20)	0.00±0.00	0.00±0.00	0.00±0.00
floortile11	(20)	6.00±0.45	6.03±0.18	7.07±0.36
floortile14	(20)	2.23±0.42	2.07±0.25	2.57±0.56
freecell00	(20)	19.23±0.67	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)	10.00±1.06	13.17±0.52	14.93±0.25
maintenance14	(20)	11.03±0.91	15.07±0.36	13.73±0.93
mprime98	(20)	14.67±1.19	14.03±1.11	13.97±0.80
mystery98	(20)	9.93±0.36	9.87±0.43	9.77±0.56
nomystery11	(20)	10.30±0.86	10.03±0.60	10.27±0.77
openstacks06	(20)	20.00±0.00	20.00±0.00	20.00±0.00
parking11	(20)	1.90±0.30	1.90±0.40	1.87±0.50
parking14	(20)	0.00±0.00	0.00±0.00	0.00±0.00
pegsol11	(20)	17.87±0.99	18.23±1.02	18.23±1.12
pipesworld*	(20)	20.00±0.00	20.00±0.00	20.00±0.00
rovers06	(20)	18.37±0.48	20.00±0.00	20.00±0.00
satellite02	(20)	17.13±0.88	19.50±0.56	19.03±0.98
scanalyzer11	(20)	17.63±0.60	18.30±0.90	19.40±0.66
snake18	(20)	3.73±0.68	5.73±0.57	6.03±0.55
sokoban11	(20)	6.20±1.11	10.33±1.07	13.90±1.19
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)	6.20±1.05	14.67±0.94	16.90±0.70
tetris14	(20)	9.40±1.98	15.97±1.91	18.07±0.73
thoughtful14	(20)	11.03±3.62	15.97±1.85	11.80±2.90
tidybot11	(20)	17.43±0.76	17.70±0.53	17.53±0.67
tpp06	(20)	13.10±0.40	15.50±0.67	17.23±0.72
trucks06	(20)	16.70±0.64	18.50±0.76	18.70±0.64
visitall11	(20)	1.13±0.43	4.27±0.68	5.63±0.66
visitall14	(20)	0.00±0.00	0.00±0.00	0.00±0.00
woodworking11	(20)	14.27±1.09	19.33±0.60	18.53±1.02
zenotravel02	(20)	17.93±0.77	18.93±0.36	18.90±0.40
SUM	(900)	457.73±5.30	523.40±4.92	538.10±4.77

Table C.6: Average number of plans achieved by the RRT algorithm with a sampling method using h_3 mutexes.

Domains		1000	10000	100000
agricola18	(20)	6.50±0.50	6.63±0.66	7.07±0.81
barman11	(20)	0.00±0.00	0.17±0.37	2.07±1.09
barman14	(20)	0.00±0.00	0.00±0.00	0.63±0.71
blocks00	(20)	20.00±0.00	20.00±0.00	20.00±0.00
caldera18	(20)	11.87±1.02	14.00±0.89	13.43±0.80
cavediving14	(20)	1.73±1.24	7.00±0.00	7.00±0.00
childsnack14	(20)	0.00±0.00	0.00±0.00	0.10±0.30
data-network18	(20)	1.57±0.50	2.90±0.75	3.93±0.93
depot02	(20)	13.17±0.82	14.90±0.94	16.43±1.05
driverlog02	(20)	15.47±0.62	18.23±0.72	18.67±0.54
elevators11	(20)	0.00±0.00	0.00±0.00	0.00±0.00
floortile11	(20)	6.00±0.00	6.20±0.40	7.03±0.55
floortile14	(20)	2.03±0.18	2.13±0.34	2.53±0.76
freecell00	(20)	17.87±0.81	19.83±0.37	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)	9.73±1.21	14.67±0.70	18.53±0.96
maintenance14	(20)	10.73±1.21	15.00±0.26	13.97±0.91
mprime98	(20)	19.27±0.77	18.93±0.51	17.93±0.85
mystery98	(20)	12.97±0.18	13.00±0.00	12.93±0.25
nomystery11	(20)	9.77±0.96	10.73±0.89	9.93±0.89
openstacks06	(20)	20.00±0.00	20.00±0.00	20.00±0.00
parking11	(20)	12.80±1.49	19.70±0.69	20.00±0.00
parking14	(20)	2.10±1.22	11.67±1.85	15.93±1.59
pegsol11	(20)	18.23±0.99	18.67±1.11	19.13±0.72
pipesworld*	(20)	20.00±0.00	20.00±0.00	20.00±0.00
rovers06	(20)	18.30±0.46	20.00±0.00	20.00±0.00
satellite02	(20)	17.37±0.80	19.60±0.61	19.07±0.77
scanalyzer11	(20)	19.97±0.18	19.87±0.34	19.97±0.18
snake18	(20)	3.03±0.55	4.83±0.82	5.60±0.55
sokoban11	(20)	6.60±0.92	10.70±1.04	14.83±0.97
spider18	(20)	5.50±1.15	10.00±1.00	12.53±0.62
storage06	(20)	20.00±0.00	20.00±0.00	20.00±0.00
termes18	(20)	6.40±1.17	14.80±0.75	16.97±0.75
tetris14	(20)	7.13±1.69	14.63±1.30	17.53±0.85
thoughtful14	(20)	18.30±0.59	18.97±0.41	18.57±0.72
tidybot11	(20)	17.73±0.51	18.17±0.69	18.80±0.87
tpp06	(20)	13.00±0.52	15.50±0.67	17.73±0.93
trucks06	(20)	17.10±0.91	18.53±0.72	18.80±0.60
visitall11	(20)	1.10±0.47	4.07±0.68	6.23±0.80
visitall14	(20)	0.00±0.00	0.00±0.00	0.00±0.00
woodworking11	(20)	13.87±0.99	19.57±0.50	18.90±0.94
zenotravel02	(20)	19.03±0.75	20.00±0.00	20.00±0.00
SUM	(900)	496.23±5.92	583.60±4.41	612.80±4.25

Table C.7: Average number of plans achieved by the RRT algorithm with a sampling method using fact-alternating mutexes.

C.2 Average length, time, sampling attempts and tree size

Gripper98	greedy	random			lifted			h2		
Problems		1000	10000	100000	1000	10000	100000	1000	10000	100000
prob01	13	12.93±0.81	12.53±0.85	12.73±0.85	12.67±0.75	12.67±1.04	12.93±0.96	13.00±0.73	12.67±0.75	12.93±0.36
prob02	21	20.53±1.23	20.47±0.88	20.27±1.09	20.27±0.96	19.80±0.98	20.33±0.94	20.47±0.88	20.13±1.12	20.67±0.75
prob03	29	28.20±0.98	27.60±1.56	27.93±1.44	28.13±0.99	27.67±0.94	28.07±1.00	28.13±0.99	27.93±1.00	28.33±0.94
prob04	37	36.00±1.24	35.13±1.86	35.73±1.59	35.80±0.98	35.60±0.92	35.93±1.00	36.13±0.99	35.87±0.99	36.20±0.98
prob05	45	44.13±1.12	43.47±1.77	43.73±2.03	44.00±1.00	43.53±0.88	44.00±1.00	44.07±1.00	43.73±0.96	44.20±0.98
prob06	53	52.20±0.98	51.80±1.68	52.00±1.91	51.80±0.98	51.40±0.80	52.00±1.00	52.00±1.00	51.93±1.00	52.27±0.96
prob07	61	60.40±1.05	60.00±1.44	60.40±1.38	59.80±0.98	59.40±0.80	60.00±1.00	59.93±1.00	60.00±1.00	60.27±0.96
prob08	69	68.47±1.02	68.40±1.05	68.47±1.36	67.80±0.98	67.60±0.92	68.07±1.00	67.93±1.00	67.93±1.00	68.33±0.94
prob09	77	76.67±0.91	76.40±1.17	76.87±0.72	75.80±0.98	75.60±0.92	75.93±1.00	75.93±1.00	76.13±0.99	76.27±0.96
prob10	85	84.53±0.85	84.53±0.99	84.73±1.00	84.40±1.28	83.80±0.98	84.33±0.94	84.13±0.99	84.27±0.96	84.40±0.92
prob11	93	92.67±0.75	92.67±1.27	92.93±0.36	92.20±1.22	91.80±0.98	92.53±0.85	92.33±0.94	92.73±1.77	92.40±0.92
prob12	101	115.27±30.03	100.87±0.72	101.00±0.00	113.67±27.25	100.60±1.40	100.60±0.80	108.40±21.06	100.93±1.67	100.60±0.80
prob13	109	116.93±19.61	108.93±0.36	108.93±0.36	111.13±12.30	108.67±0.75	108.67±0.75	117.07±20.08	108.80±0.60	108.93±0.81
prob14	117	116.60±0.95	117.07±0.36	117.00±0.00	118.27±8.06	116.80±0.60	116.60±0.80	121.93±17.27	116.93±0.63	117.07±0.81
prob15	125	130.67±18.64	124.87±0.50	125.00±0.00	126.53±9.78	124.93±0.63	125.13±0.50	137.80±25.38	124.93±0.36	125.00±0.00
prob16	133	136.33±10.50	132.80±0.79	132.67±1.27	133.53±4.79	132.93±0.96	132.93±0.36	140.20±19.06	132.93±0.36	133.00±0.00
prob17	141	145.07±16.47	140.80±0.79	141.00±0.00	141.20±8.98	141.07±0.36	141.07±0.81	146.20±22.44	141.00±0.52	140.87±0.50
prob18	149	158.60±23.61	148.93±0.36	148.87±0.72	159.27±25.02	149.00±0.00	149.27±1.12	161.60±23.48	149.00±0.52	148.93±0.36
prob19	157	161.47±11.49	156.73±1.00	156.87±0.50	163.93±16.66	157.00±0.00	157.20±1.08	172.73±29.07	157.00±0.89	156.93±0.36
prob20	165	172.13±19.10	164.73±0.85	164.93±0.63	178.93±28.73	165.13±0.72	165.00±0.00	178.13±25.10	164.80±0.60	164.93±0.36

Gripper98	h2fwbw			h3			fam		
Problems	1000	10000	100000	1000	10000	100000	1000	10000	100000
prob01	12.93±0.81	12.87±0.50	12.73±0.85	12.73±0.68	12.80±0.60	12.80±0.79	13.13±0.50	12.93±0.36	12.93±0.81
prob02	20.33±1.07	20.60±0.95	20.07±1.00	20.33±0.94	20.73±0.68	20.07±1.00	20.40±0.92	20.47±0.88	20.40±1.17
prob03	28.07±1.00	28.47±0.88	28.00±1.00	28.40±0.92	28.33±0.94	28.13±0.99	28.33±0.94	28.40±0.92	28.13±0.99
prob04	36.00±1.00	36.27±0.96	35.87±0.99	36.00±1.00	36.40±0.92	35.80±0.98	36.07±1.00	36.27±0.96	36.20±0.98
prob05	43.80±0.98	44.27±0.96	43.87±0.99	43.87±0.99	44.40±0.92	44.00±1.00	43.93±1.00	44.33±0.94	44.00±1.00
prob06	51.67±0.94	52.27±0.96	51.73±0.96	51.93±1.00	52.27±0.96	51.87±0.99	52.00±1.00	52.40±0.92	52.00±1.00
prob07	59.67±0.94	60.27±0.96	59.73±0.96	59.87±0.99	60.33±0.94	60.00±1.00	60.07±1.00	60.40±0.92	59.87±0.99
prob08	67.67±0.94	68.27±0.96	67.73±0.96	67.87±0.99	68.27±0.96	67.93±1.00	68.07±1.00	68.33±0.94	67.93±1.00
prob09	75.93±1.12	76.20±0.98	75.87±0.99	76.00±1.00	76.20±0.98	76.00±1.00	76.00±1.00	76.47±0.88	76.13±0.99
prob10	83.80±0.98	84.33±0.94	84.00±1.00	84.20±1.22	84.33±0.94	84.20±0.98	84.20±1.11	84.40±0.92	84.40±0.92
prob11	92.07±1.24	92.33±0.94	92.53±1.52	92.60±1.31	92.53±0.85	92.07±1.00	92.47±1.02	92.60±0.80	92.67±0.75
prob12	105.60±17.47	101.07±1.31	100.20±0.98	103.40±14.56	100.80±1.08	100.13±0.99	106.33±18.12	100.60±0.80	100.80±0.60
prob13	108.67±2.48	108.87±1.02	108.67±1.16	117.80±22.08	108.80±1.08	109.07±1.41	114.00±17.73	109.13±1.78	108.60±0.80
prob14	122.87±20.03	117.20±1.19	116.80±1.08	120.27±12.61	116.93±0.96	116.60±1.40	118.87±5.80	117.00±0.89	117.13±1.15
prob15	131.53±18.18	124.87±0.50	124.80±0.60	131.00±15.65	124.93±1.09	125.00±1.79	131.27±17.09	124.67±0.75	124.93±0.63
prob16	143.33±23.30	132.93±0.63	133.07±0.63	141.47±18.72	133.47±1.33	133.13±1.71	151.53±29.13	133.00±1.26	132.80±0.60
prob17	144.07±15.24	140.80±0.60	141.00±0.89	150.87±29.29	141.47±1.69	141.47±1.84	146.07±15.98	140.80±0.60	141.07±0.81
prob18	170.67±33.65	148.80±0.60	148.80±0.60	173.93±31.86	149.40±1.31	149.47±1.91	163.00±23.31	148.87±0.50	149.00±0.52
prob19	168.73±21.43	156.87±1.02	156.80±0.60	182.07±29.13	157.00±1.26	157.00±1.55	169.93±26.27	157.33±1.87	157.20±1.08
prob20	176.80±28.73	165.00±0.89	164.80±1.08	177.87±34.63	164.93±1.75	165.00±1.26	178.80±31.58	164.40±0.92	165.13±1.15

Table C.8: Average length of plans for the gripper98 domain.

Zenotravel02	h2fwbw			h3			fam		
Problems	1000	10000	100000	1000	10000	100000	1000	10000	100000
pfile10	0.19±0.12	0.23±0.11	0.27±0.16	0.54±0.21	0.60±0.28	0.56±0.21	0.19±0.09	0.25±0.11	0.20±0.11
pfile11	0.16±0.06	0.14±0.06	0.19±0.16	0.81±0.31	0.74±0.24	0.71±0.18	0.15±0.06	0.12±0.04	0.14±0.07
pfile12	0.11±0.03	0.11±0.03	0.12±0.04	0.99±0.45	0.93±0.35	0.87±0.23	0.09±0.03	0.10±0.03	0.10±0.03
pfile13	0.11±0.05	0.11±0.05	0.13±0.05	1.13±0.47	1.04±0.35	1.00±0.27	0.09±0.04	0.12±0.05	0.10±0.04
pfile14	5.35±5.63	3.08±1.94	4.68±3.43	19.79±9.15	18.07±5.72	18.79±5.62	3.59±3.53	3.56±2.66	3.88±4.36
pfile15	4.12±2.22	14.87±9.61	63.78±105.55	59.90±23.83	60.49±18.57	89.73±56.33	2.44±1.28	7.69±4.97	55.73±54.39
pfile16	27.37±39.69	7.89±8.09	14.12±22.09	147.52±85.90	110.68±34.96	113.76±28.78	45.41±73.04	9.42±13.00	5.33±5.50
pfile17	140.26±314.16	21.14±18.62	26.86±23.93	431.60±272.31	309.53±101.17	309.93±98.71	114.36±310.56	21.74±14.29	19.52±25.47
pfile18	179.66±295.74	44.14±27.39	77.24±118.69	610.10±239.53	557.53±180.72	578.69±189.81	110.16±166.50	27.89±14.55	38.36±17.62
pfile19	185.66±374.13	49.12±31.74	48.56±36.21	1303.66±224.62	1415.29±244.64	1403.99±241.90	58.75±70.85	64.87±72.58	29.53±21.96
pfile1	0.01±0.00	0.01±0.00	0.01±0.00	0.01±0.01	0.01±0.00	0.01±0.01	0.01±0.00	0.01±0.00	0.01±0.00
pfile20	330.73±436.26	97.70±94.33	88.49±70.17	no results	1676.45±0.00	1510.43±0.00	218.03±272.46	86.37±45.89	88.46±79.17
pfile2	0.01±0.00	0.01±0.00	0.01±0.00	0.01±0.01	0.01±0.01	0.01±0.00	0.01±0.00	0.02±0.00	0.01±0.00
pfile3	0.02±0.01	0.02±0.01	0.02±0.01	0.03±0.01	0.03±0.01	0.03±0.01	0.02±0.01	0.02±0.00	0.02±0.01
pfile4	0.02±0.01	0.02±0.01	0.02±0.01	0.04±0.01	0.03±0.01	0.03±0.01	0.02±0.01	0.02±0.01	0.02±0.01
pfile5	0.02±0.01	0.02±0.01	0.03±0.01	0.06±0.02	0.05±0.01	0.05±0.02	0.02±0.01	0.03±0.01	0.03±0.01
pfile6	0.03±0.01	0.02±0.01	0.03±0.01	0.08±0.03	0.07±0.02	0.07±0.02	0.03±0.01	0.03±0.01	0.03±0.01
pfile7	0.03±0.01	0.03±0.01	0.04±0.02	0.08±0.03	0.08±0.02	0.07±0.02	0.03±0.01	0.03±0.01	0.03±0.01
pfile8	0.07±0.02	0.06±0.02	0.09±0.06	0.40±0.15	0.38±0.13	0.37±0.09	0.06±0.02	0.07±0.03	0.07±0.02
pfile9	0.08±0.03	0.08±0.02	0.09±0.03	0.48±0.20	0.45±0.14	0.44±0.12	0.08±0.03	0.07±0.01	0.09±0.04

Zenotravel02	greedy	random			lifted			h2		
Problems		1000	10000	100000	1000	10000	100000	1000	10000	100000
pfile10	0.14	0.19±0.13	0.24±0.12	0.23±0.15	0.19±0.16	0.28±0.24	0.19±0.12	0.15±0.07	0.25±0.12	0.22±0.12
pfile11	0.05	0.15±0.07	0.10±0.03	0.14±0.04	0.18±0.14	0.13±0.06	0.14±0.09	0.12±0.04	0.13±0.04	0.13±0.05
pfile12	0.04	0.11±0.05	0.10±0.04	0.13±0.05	0.12±0.09	0.13±0.11	0.13±0.07	0.10±0.05	0.11±0.09	0.11±0.07
pfile13	0.03	0.11±0.07	0.10±0.04	0.17±0.25	0.10±0.06	0.13±0.12	0.12±0.05	0.12±0.08	0.09±0.04	0.11±0.07
pfile14	1.64	3.96±3.29	3.28±1.70	7.36±11.24	6.50±5.04	2.83±1.62	2.90±1.52	10.11±16.69	5.24±5.06	6.19±6.32
pfile15	24.74	3.99±2.68	9.95±8.31	34.59±46.49	3.51±2.20	7.93±6.24	51.22±54.46	3.30±1.86	10.34±10.24	40.13±44.05
pfile16	1.11	19.68±30.15	7.45±8.23	12.92±31.71	23.36±29.85	9.22±7.84	5.95±5.05	29.96±34.17	8.12±8.34	8.39±10.68
pfile17	4.49	177.49±443.90	17.55±8.80	18.30±13.58	46.93±87.83	19.62±18.35	15.48±4.96	44.22±96.63	21.98±12.19	17.74±10.96
pfile18	15.16	305.88±447.66	28.59±23.02	53.81±37.22	106.23±155.81	41.71±34.30	39.63±14.53	115.06±173.11	34.42±19.81	35.27±18.50
pfile19	5.99	157.58±172.34	47.51±35.43	36.38±28.62	559.00±632.02	59.77±69.54	26.05±13.87	97.49±222.92	80.62±79.44	35.33±26.63
pfile1	0.00	0.01±0.00	0.01±0.00	0.01±0.00	0.01±0.00	0.01±0.00	0.01±0.00	0.01±0.01	0.01±0.00	0.01±0.00
pfile20	14.84	257.86±362.81	87.70±72.80	70.80±31.60	310.36±384.58	87.75±76.24	96.10±89.92	317.07±404.27	64.51±47.24	68.55±41.90
pfile2	0.00	0.01±0.00	0.01±0.00	0.01±0.00	0.01±0.00	0.01±0.00	0.01±0.00	0.01±0.01	0.01±0.00	0.01±0.00
pfile3	0.00	0.02±0.01	0.02±0.01	0.02±0.01	0.02±0.01	0.02±0.01	0.02±0.01	0.02±0.01	0.02±0.00	0.02±0.00
pfile4	0.01	0.02±0.00	0.02±0.01	0.02±0.01	0.02±0.01	0.02±0.01	0.02±0.01	0.02±0.01	0.02±0.01	0.02±0.01
pfile5	0.01	0.02±0.01	0.02±0.01	0.02±0.01	0.02±0.01	0.03±0.01	0.02±0.01	0.02±0.01	0.02±0.01	0.02±0.01
pfile6	0.00	0.02±0.01	0.02±0.01	0.02±0.01	0.03±0.01	0.03±0.01	0.02±0.01	0.03±0.01	0.02±0.01	0.02±0.01
pfile7	0.01	0.03±0.01	0.03±0.01	0.03±0.01	0.03±0.01	0.04±0.02	0.03±0.01	0.03±0.01	0.03±0.01	0.03±0.01
pfile8	0.02	0.06±0.02	0.07±0.04	0.06±0.02	0.06±0.02	0.09±0.06	0.06±0.02	0.06±0.02	0.06±0.03	0.06±0.02
pfile9	0.03	0.07±0.02	0.07±0.02	0.08±0.02	0.08±0.02	0.08±0.03	0.07±0.02	0.08±0.03	0.07±0.02	0.08±0.03

Table C.9: Average time(s) for the zenotravel02 domain.

Parking11	random			lifted			h2		
Problems	1000	10000	100000	1000	10000	100000	1000	10000	100000
pfile08-031	52959.47±27414.83	7000.70±8226.80	3713.50±4692.64	237.60±278.57	7.80±8.17	5.63±7.39	213.38±235.77	11.03±12.43	5.77±7.15
pfile08-032	26969.39±27263.81	6000.60±6633.91	4519.00±4851.89	70.73±84.67	6.50±10.70	3.47±5.12	72.80±91.89	9.50±12.57	5.60±10.27
pfile09-033	60006.00±0.00	14334.77±11161.56	5411.63±4923.38	390.17±340.90	10.57±11.19	4.50±7.24	164.55±109.87	9.63±11.44	5.00±8.81
pfile09-034	no results	18870.27±14225.91	5125.17±4920.26	388.21±316.30	13.90±18.38	3.10±5.69	388.41±242.59	11.03±16.10	5.57±9.04
pfile09-035	38411.20±30078.62	14385.43±10353.70	5410.93±4923.82	285.73±278.04	12.63±14.54	4.70±7.59	277.21±256.81	16.33±18.46	5.47±8.97
pfile09-036	47782.56±22001.08	11044.73±7838.43	4344.00±4947.20	231.90±209.74	8.63±12.39	2.77±4.63	192.80±264.69	14.77±18.82	4.57±5.74
pfile10-037	17231.10±17224.16	11035.59±12134.05	5081.33±4938.23	386.58±377.37	9.13±12.05	3.57±5.63	327.15±424.51	10.83±20.76	5.00±7.08
pfile10-038	25002.50±5000.50	16061.73±10344.68	4333.77±4955.85	395.84±265.00	20.73±23.04	6.97±11.60	425.48±359.71	10.67±11.64	4.13±6.51
pfile10-039	no results	18453.70±9938.83	6923.10±4551.17	441.28±399.36	21.60±20.76	6.37±8.77	494.74±323.12	20.20±17.86	6.37±11.92
pfile10-040	30003.00±0.00	19788.75±13141.18	7038.57±4529.41	492.08±439.71	18.53±20.33	4.43±7.84	403.04±326.06	14.77±14.28	6.53±11.34
pfile11-041	no results	16801.68±13483.19	4667.13±4989.38	385.14±238.49	18.87±21.91	3.67±5.96	429.43±272.02	25.60±20.36	12.97±21.64
pfile11-042	40004.00±0.00	24547.91±13049.21	3245.86±4592.46	303.88±200.38	36.23±38.25	6.97±8.40	549.10±417.36	30.60±29.99	4.00±7.50
pfile11-043	no results	26252.62±12184.71	5333.87±4989.38	432.11±304.91	24.43±18.97	6.34±9.21	544.62±335.74	34.90±26.52	5.53±10.20
pfile11-044	30003.00±0.00	13215.61±12264.29	5333.87±4989.38	556.20±318.64	23.67±25.43	2.87±4.56	472.18±276.85	14.53±17.34	6.17±10.85
pfile12-045	no results	13479.61±10472.00	5667.23±4955.85	519.75±269.90	26.21±35.81	4.40±8.97	563.00±164.50	31.63±40.71	2.00±3.52
pfile12-046	no results	8622.72±8475.60	5333.87±5617.99	384.00±107.92	41.37±38.29	5.14±9.99	474.17±206.15	31.17±32.54	7.63±10.68
pfile12-047	no results	25002.50±9820.79	5172.93±5645.64	521.44±420.47	63.71±35.99	7.63±13.66	465.11±215.55	47.04±36.98	8.23±13.68
pfile12-048	no results	23079.23±7216.75	4643.32±4987.73	441.22±355.05	64.57±55.25	6.20±12.27	448.75±247.82	48.48±25.28	3.23±5.74
pfile13-049	no results	12501.25±4330.56	2500.25±4330.56	408.50±224.05	48.46±40.85	9.90±13.52	780.00±140.00	41.74±32.53	5.90±9.34
pfile13-050	no results	7222.94±4479.48	5185.70±4997.07	306.00±185.94	34.25±26.87	9.21±14.80	417.00±356.66	33.34±27.42	4.24±8.43

Parking11	h2fwbw			h3			fam		
Problems	1000	10000	100000	1000	10000	100000	1000	10000	100000
pfile08-031	196.67±188.28	9.43±11.08	3.77±7.18	140.31±120.94	8.07±10.23	6.61±10.49	156.60±201.75	12.80±12.53	5.70±6.09
pfile08-032	57.90±118.07	4.17±8.20	8.40±9.78	100.32±109.47	5.24±8.90	3.46±6.55	48.47±62.14	4.57±9.89	6.80±9.60
pfile09-033	262.31±259.86	12.17±15.49	3.03±5.94	no results	no results	no results	309.69±406.28	6.87±10.87	3.83±9.54
pfile09-034	380.45±328.58	11.57±16.47	5.77±9.57	no results	no results	no results	391.47±323.43	13.37±19.49	4.07±6.12
pfile09-035	260.21±220.14	12.70±16.01	6.80±16.93	no results	no results	no results	313.96±308.45	15.03±20.46	5.93±11.51
pfile09-036	277.52±323.00	12.00±14.53	4.80±7.83	no results	no results	no results	194.17±206.00	6.73±10.18	4.53±7.50
pfile10-037	273.31±391.90	11.73±16.93	6.97±11.00	no results	no results	no results	345.70±371.04	8.40±15.64	7.27±10.11
pfile10-038	309.11±243.73	15.50±18.97	6.50±9.48	no results	no results	no results	435.86±342.94	11.23±12.50	4.03±6.65
pfile10-039	421.75±290.60	21.50±16.85	2.80±5.91	no results	no results	no results	430.54±341.60	19.83±19.74	7.60±11.13
pfile10-040	420.90±336.50	18.70±21.73	4.77±7.78	no results	no results	no results	414.69±284.28	19.00±18.85	7.20±9.34
pfile11-041	658.76±330.65	28.57±28.61	4.40±9.58	no results	no results	no results	535.29±267.42	23.73±18.06	3.57±6.39
pfile11-042	471.39±308.35	30.47±25.87	5.27±6.58	no results	no results	no results	440.18±365.36	35.13±36.15	6.83±10.60
pfile11-043	451.00±330.85	44.90±32.54	3.97±6.55	no results	no results	no results	527.83±228.37	34.13±24.10	9.10±15.31
pfile11-044	578.06±442.38	13.10±17.76	3.45±6.16	no results	no results	no results	395.74±262.29	17.57±25.57	8.47±15.60
pfile12-045	201.17±122.79	30.80±43.36	7.07±8.74	no results	no results	no results	360.75±256.43	36.47±37.71	5.47±9.75
pfile12-046	472.00±226.88	32.00±40.77	6.13±10.75	no results	no results	no results	375.50±264.51	39.43±41.70	8.07±12.87
pfile12-047	358.22±366.31	54.52±52.00	6.70±10.26	no results	no results	no results	512.22±186.01	37.20±31.00	1.23±4.14
pfile12-048	842.00±410.28	62.23±48.29	6.77±9.91	no results	no results	no results	527.20±232.37	67.96±58.55	5.17±9.04
pfile13-049	527.00±356.52	55.93±45.43	9.87±12.34	no results	no results	no results	481.60±275.05	34.81±33.33	7.87±14.56
pfile13-050	371.17±275.93	45.83±44.49	7.48±12.56	no results	no results	no results	719.50±101.50	28.43±27.06	4.50±8.34

Table C.10: Average number of discarded sampling attempts for the parking11 domain.

Tetris14	h2fwbw			h3			fam		
Problems	1000	10000	100000	1000	10000	100000	1000	10000	100000
p020	32.83±29.04	17.03±13.17	10.40±6.95	33.20±33.68	12.23±10.46	8.07±7.22	26.57±28.21	20.20±17.30	16.37±10.40
p021	102.80±68.78	43.10±45.11	26.25±19.19	79.75±55.44	44.38±43.89	20.71±14.96	243.36±174.11	157.78±145.27	52.00±28.09
p022	92.27±70.23	30.53±32.44	11.30±8.86	92.83±54.19	25.10±18.64	7.13±4.01	197.93±147.64	38.03±27.23	13.87±12.21
p023	72.57±70.66	49.69±39.54	16.63±14.15	77.31±56.33	28.82±27.66	10.72±7.99	338.44±224.51	110.56±98.07	26.93±24.84
p024	107.12±48.98	52.50±53.94	7.47±4.22	105.09±80.17	33.68±26.99	7.57±6.67	419.27±257.15	117.62±122.60	8.40±7.92
p025	71.29±62.79	42.10±42.34	9.10±11.15	64.35±59.79	31.62±20.96	10.50±12.18	122.31±104.82	77.79±64.75	9.67±11.91
p026	160.54±102.75	51.38±50.41	17.00±12.00	147.07±112.87	66.14±50.16	15.67±12.59	405.13±293.49	57.10±61.82	13.37±10.96
p027	133.79±80.66	45.59±42.02	11.93±10.28	180.33±93.69	64.88±52.44	13.13±10.85	392.56±381.42	99.18±99.48	11.13±5.65
p028	112.59±68.25	14.87±15.83	6.37±6.95	115.62±87.11	23.43±30.29	8.13±10.29	344.00±321.44	42.43±71.40	10.17±13.28
p029	57.75±25.59	47.80±47.17	24.21±17.16	84.00±49.98	57.20±30.10	17.79±10.26	367.00±259.00	169.17±152.32	39.95±29.79
p031	84.93±58.90	42.69±45.75	4.70±3.18	63.50±40.75	27.89±30.38	6.87±4.67	110.80±43.63	70.33±73.15	8.40±7.78
p032	91.57±54.20	40.45±33.28	18.79±19.41	48.00±23.57	44.58±30.08	15.17±13.82	70.50±59.83	72.12±49.34	20.88±15.65
p033	72.19±38.55	49.52±36.69	18.83±12.51	80.00±43.54	30.18±23.89	14.64±12.01	209.00±0.00	81.73±65.61	28.88±21.45
p034	83.83±46.79	53.43±46.00	17.27±15.12	75.17±41.44	55.41±41.46	27.46±21.73	88.75±67.49	75.44±57.98	21.86±18.89
p035	37.21±40.73	7.27±4.29	2.70±0.64	23.39±23.05	8.43±6.01	2.27±0.44	74.33±58.12	8.97±4.80	2.47±0.81
p036	65.20±64.69	5.77±2.69	2.37±0.48	51.46±65.13	8.77±6.86	2.40±0.49	79.00±74.33	7.93±3.58	2.53±0.50
p037	56.00±32.69	60.20±36.83	16.55±9.16	no results	no results	no results	57.00±0.00	68.00±44.95	27.67±8.58
p038	61.83±31.42	52.04±38.73	12.43±8.77	76.83±49.18	53.85±40.86	15.67±15.99	120.00±91.00	53.00±54.12	28.36±20.13
p039	82.89±46.70	38.69±35.16	6.28±4.59	60.67±57.11	36.86±30.24	5.53±3.59	83.50±72.49	40.92±56.31	8.17±13.03
p040	85.67±34.99	68.21±36.87	21.64±11.16	no results	23.50±9.84	8.69±5.16	52.00±0.00	73.75±89.24	29.29±16.28

Tetris14	random			lifted			h2		
Problems	1000	10000	100000	1000	10000	100000	1000	10000	100000
p020	21.20±11.66	15.60±16.48	11.37±12.13	23.97±20.41	15.57±14.15	8.67±5.45	32.77±29.49	16.43±14.32	9.60±4.92
p021	70.04±59.80	51.21±41.64	19.23±19.00	93.22±70.85	37.53±38.15	19.20±17.67	90.38±90.07	41.47±41.13	16.52±12.67
p022	97.70±79.83	26.10±28.29	8.73±6.27	99.33±77.73	31.03±41.80	12.17±10.99	115.79±76.75	30.63±41.97	8.03±4.62
p023	74.00±62.45	55.29±37.04	16.00±14.04	76.75±51.18	49.47±42.56	13.73±13.10	92.36±78.35	38.44±36.54	18.83±13.85
p024	96.71±78.93	46.17±44.74	5.57±2.68	106.42±76.12	39.20±39.36	6.03±2.60	130.18±83.49	41.36±30.97	6.77±4.51
p025	78.00±48.34	43.66±45.54	6.73±8.68	91.16±70.45	42.83±36.72	6.90±7.08	96.40±60.83	46.86±53.14	5.47±4.94
p026	99.75±54.70	79.17±70.10	13.03±8.54	137.71±117.88	59.48±57.15	17.37±21.84	110.23±120.90	73.62±65.92	12.67±9.45
p027	108.92±77.03	41.23±36.90	9.40±5.57	122.33±87.98	50.82±52.15	9.60±8.14	139.93±94.17	66.31±59.90	12.07±9.71
p028	111.05±89.28	26.63±36.72	5.23±6.49	86.76±73.54	15.07±14.08	5.17±6.85	116.19±94.42	16.30±24.31	4.90±5.70
p029	no results	59.47±46.29	20.88±13.42	99.00±41.72	67.12±41.93	19.65±12.86	83.00±49.19	74.11±56.30	21.25±15.72
p031	99.64±49.00	43.34±35.57	7.33±6.66	73.71±54.22	38.41±34.94	7.83±9.72	105.71±54.26	24.90±21.55	7.13±9.62
p032	48.43±47.39	64.17±46.18	17.12±12.21	100.50±74.61	50.17±44.40	15.14±10.84	97.11±63.48	56.50±47.58	19.63±17.14
p033	70.07±35.54	45.15±38.27	14.55±13.92	74.22±53.07	31.10±24.48	21.67±16.65	75.24±41.60	29.30±24.17	14.88±10.54
p034	75.27±55.91	65.04±46.90	16.66±12.67	43.67±28.45	54.00±46.53	19.76±15.44	63.20±27.06	44.48±39.49	22.93±21.76
p035	34.63±26.16	8.97±5.97	2.33±0.47	48.34±46.35	6.27±3.74	2.53±0.56	29.10±19.87	8.00±4.37	2.73±0.77
p036	61.07±59.02	5.90±3.38	2.53±0.50	58.40±49.65	6.93±3.53	2.40±0.49	52.67±46.97	7.87±6.90	2.43±0.50
p037	73.00±0.00	42.60±18.00	27.70±12.37	75.00±61.00	47.40±25.75	17.00±11.41	no results	39.00±30.11	26.83±13.33
p038	111.91±42.24	47.32±44.84	15.67±14.97	71.00±38.20	41.89±30.29	15.34±12.55	105.30±57.09	31.16±20.42	13.23±12.19
p039	64.60±37.24	36.00±32.84	7.90±7.16	75.65±55.14	24.43±32.10	6.00±4.40	63.16±55.65	29.50±28.77	6.27±5.95
p040	73.25±55.19	55.36±35.42	15.89±9.61	131.00±0.00	48.52±24.42	22.78±10.21	109.00±35.00	44.20±33.56	22.00±10.31

Table C.11: Average tree sizes for the tetris14 domain.