University of Hamburg
Department of Informatics
Group Technical Aspects of Multi-modal Systems

Master Thesis

Hierarchical Plan Generation and Selection for Shortest Plans based on Experienced Execution Duration

Using Parallel Plan Execution

Prof. Dr. Jianwei Zhang
Sebastian Rockel

Lasse Einig
eMail: einig@informatik.uni-hamburg.de
Matriculation-No. 5918387

Synopsis

This thesis presents the Plan Evaluator (PlEv), a method to find the optimal plan, executed in parallel, from a set of sequential Hierarchical Task Network (HTN) generated plans for reaching a goal state from an initial state. Plan decisions made by HTN planners are either based on counting the amount of steps within a plan or by statically assigned weights to the plan steps. The PlEv uses dynamically generated task execution durations, based on experience from experiments or simulations, to calculate the shortest plan considering the total plan execution duration. In addition, the plan is arranged in a parallel executable order and the benefit from this parallel execution is incorporated to the decision made by the PlEv. The proposed method is evaluated with two scenarios and the results are verified with additional scenarios. The results show that for any scenario a configuration exists at which the PlEv will find a plan which is executed faster or as fast as the original decision made by the HTN planner.

Zusammenfassung

In dieser Arbeit wird der Plan Evaluator (PlEv) vorgestellt, eine Methode, den optimalen, parallel ausführbaren Plan aus einer Menge von sequentiellen, Hierarchical Task Network (HTN) generierten Plänen zum Erreichen einer Zielkonfiguration aus einer Startkonfiguration, zu finden. Die Entscheidung von HTN Planern basiert auf der Anzahl der Planschritte in einem Plan oder statisch zugewiesenen Gewichten dieser Planschritte. Der PlEv benutzt dynamisch generierte Ausführungszeiten, welche in Experimenten und Simulationen ermittelt werden, um den kürzesten Plan anhand der gesamten Ausführungsdauer des Planes zu berechnen. Zusätzlich wird der Plan in eine parallel ausführbare Ordnung gebracht. Die Zeitersparnis durch die parallele Ausführung fließt in die Entscheidungsfindung des PlEv ein. Die vorgestellte Methode wird an zwei Szenarien ausgewertet und die Ergebnisse mit weiteren Szenarien verifiziert. Die Resultate zeigen, dass für jedes Szenario eine Konfiguration existiert, ab der der PlEv einen schneller oder mindestends gleich schnell ausführbaren Plan findet als die ursprüngliche Entscheidung, die von dem HTN Planer getroffen wird.

Contents

1 Introduction

1.1 Motivation

Improving the overall execution of robotic tasks within human environments is one of the biggest research fields for robotics. This includes autonomous and robust task execution. Also, the execution of tasks is an important factor for real-world applications. Operating efficiency and thus profitability is not only required for everyday deployment, but also during development cycles and test phases. Reducing the required execution duration of tasks motivated the author to optimize the plan execution.

Parallelization of sequential Hierarchical Task Network (HTN) plans offers great benefits concerning execution time on mobile service robots. Previous work showed the possibility to decrease the execution time to less than 75 %, by finding a parallel order of the tasks in the sequential plan [Einig et al., 2013]. This was accomplished by introducing a three-layer architecture. The existing HTN planner generates the plan, which is interpreted and adapted by the parallelizer and then executed by the execution manager. This architecture also allows to introduce changes to each layer without affecting the overall system. Following the promising results of the previous work, the question arises, whether it may not only be possible to parallelize the shortest sequential plan found by the HTN planner, but instead find the shortest parallel plan for all generated sequential plans and whether this plan will differ from the one chosen by the HTN planner. HTN planners, especially SHOP2, are capable of returning all possible plans that will reach the goal from the given state. The usual approach is to take the shortest plan, which, in case the planning domain does not specify the costs for an action, is the amount of actions in the plan. Thus a plan containing three actions, which require ten time units execution time per action is considered shorter than a plan with five actions, which takes one time unit per action. In this case, even the sequential execution of the longer plan is faster than the execution of the shorter plan. Although HTN planners such as SHOP2 are capable of calculating the total cost for a plan from the cost provided in the planning domain, this is not always applicable. The cost for each task has to be entered into the planning domain by hand and has to be known by the time the domain is written. During the development for the previous work, the author had to measure the execution time for actions in order to be able to compare sequential and parallel execution. A similar tool can be used to record the execution duration for each action during operation and receive information on average task execution duration. These collected durations can be used for the robot to learn the temporal cost of its actions.

1.2 Objective

The functionality of the parallelization layer has been shown in Einig [2012] and thus is not part of this work. The parallelization layer is utilized in this work in order to fulfill the objective. The execution layer which was presented in Einig [2012] has been refined within the further research and the current modified version will be used for this work.

The objective is to evaluate, whether a plan, whose cost in sequential order is higher than the cost of the shortest sequential plan, may be executed in a parallel order, which has less cost than the parallel order of the shortest sequential plan, based on gained knowledge. A very basic example would involve filling a large glass of water with two bottles next to it. It is possible to grab a bottle with one hand and pour water into the glass. Although it requires more actions and more skill, it is quicker to grab a bottle with each hand and pour the water from both bottles at the same time.

A more general example assumes the cost for each action is exactly one. There might be a sequential plan with five actions and a sequential plan with six actions to reach the goal. If the plan with six actions is executable in parallel, such that three actions are executed at the same time, the parallel plan may only take four time units, whereas the five action plan has no parallelization potential and takes five time units. In order to find such a shorter parallel plan, not only the best plan returned by the sequential HTN planner, but all possible sequential plans within a reasonable range must be considered. Additionally, the actions of the plan may not require exactly one time unit. Concerning the mobile service robot, the task of driving to a position may take more or less time depending on the distance, the obstacle density of the area and other factors. These factors include increased time for obstacle detection and path planning for certain configurations and driving at reduced speed as a result of external influences. If these durations for the actions are used to calculate the cost of the sequential plan, more improvements are possible beyond regarding the amount of actions in a plan.

In order to be able to use the actual execution durations for the tasks, especially for driving tasks, these durations have to be gathered. Where the handwritten knowledge in the planning domain is not capable of differentiating between a driving task from location A to B and a driving task from A to C, the gained knowledge will enable to resolve any driving task, which has been performed already before, to an execution duration.

This will be evaluated using two scenarios, closely related to the EU-funded project RACE[1] [Hertzberg et al., 2014]. These scenarios will be described in sections 3.1 and 3.2. Some adaptations to the RACE planner and execution manager will be made.

This work will therefore focus on solving the following problems:

1. Adapt the RACE planning to the evaluation scenarios,
2. Evaluate the generated plans using the parallelization layer,
3. Execute the best parallelized plan and compare it to the execution of the best sequential plan,
4. Gather knowledge about task execution durations during daily operation.

Tasks 1-3 correspond to the three-layer architecture presented in Einig [2012]. Task 4 will be integrated into the existing architecture of the RACE project, which focuses on gathering experiences and learning from these.

1.3 Outline

This work presents the Plan Evaluator (PlEv), a method to find the optimal plan from a set of generated plans. In chapter 2, *State of the Art*, previous attempts to parallel plan execution are presented, including the proposed previous attempt of the author, which is the foundation to this work. An overview on cost-based

[1]Project RACE, http://www.project-race.eu, Last checked 24.11.2014

planning is given, which mostly focuses on path planning approaches. The project setting, including the overall architecture, global scenario of the RACE project and the robot platform with modifications, which is used for evaluation and experiments, is described.

In chapter 3, *Scenarios*, the two scenarios, which will be used for the evaluation and verification of the PlEv and its components, are described. The *attend table* scenario drives the robot from a position in the restaurant to positions at different distances to show the impact of the length of the variable driving task in relation to the fixed elements of the plan. The *door* scenario shows the effect of door properties and the constraints for the passing of these doors to the plan execution duration.

In chapter 4, *Temporal Experience Extraction*, a part of the PlEv responsible for extracting temporal knowledge while executing tasks within the RACE environment is proposed. The State MACHine (SMACH) execution manager as foundation for the temporal experience extraction is introduced. The implementation of the extractor is described and the results from running the extractor within the RACE environment are shown as well as the necessary data preprocessing.

The main part of this work, the PlEv, is introduced in chapter 5, *Plan Evaluator*. The foundation regarding the RACE architecture and the three-layer architecture described in the authors previous work, as well as the improvements to the architecture by transforming the three-layer architecture to a closed system and decoupling it from the RACE Blackboard is presented. Adaptations to the planning domain in order to be able to decompose multiple plans for the scenarios (chapter 3) are suggested. The definition of the PlEv for finding the shortest feasible plan regarding execution duration is given and the implementation of this definition is explained. The metrics for the evaluation are also introduced in this chapter.

In chapter 6, *Results*, the results from the two scenarios (*attend table* scenario and *door* scenario as described in chapter 3) are presented and summarized.

The results are evaluated in chapter 7, *Evaluation*. For both scenarios, the switch points are calculated. An additional scenario as well as an extension to this scenario, are presented to demonstrate the correctness of the PlEv for equivalent plans. The results from chapter 6 will be used to compute results for the additional scenarios.

The difficulties and drawbacks are discussed in chapter 8, *Discussion*, and the work is concluded in chapter 9, *Conclusion and Outlook*. In the last chapter, suggestions for possible improvements and applications of the proposed PlEv are made.

2 State of the Art

In chapter 1 the motivation for this work was explained and the target objective and the outline of this work were presented. In this chapter an overview on approaches to parallel plan execution, including the previous work of the author are given. After introducing cost-based planning and its impact on path planning, the project setting with the RACE architecture and the experimental platform is presented.

2.1 Parallel Plan Execution

Multiple different approaches to executing plans in parallel have been described by Einig [2012]. The approaches may be categorized into generating parallel plans by enhancing existing planners or writing new planners and parallelizing previously generated sequential plans using additional layers. Most of these approaches focus on overall project scheduling or crisis management. In robotics, the plan optimization usually relates to specializing in certain domains, developing multi-core enabled plan generation [Devaurs et al., 2011; Jacobs et al., 2012] or parallelizing planning and execution [Miura and Shirai, 1998]. Castillo et al. [2006] presented the generation of parallel plans for forest fire fighting. It is one of the few publications regarding parallel planning in general. The planner presented is based on SHOP and enhanced with qualitative ordering and temporal constraints. Aside the field of robotics, Lingard and Richards [1998]; Luh and Lin [1985] and others presented different approaches on scheduling parallel tasks.

Within the field of robotics, the work on parallel planning or rather executing plans in parallel is limited to the implementation of a middle layer between planner and executor. Gat [1998] introduced the three-layer architecture to robotic scheduling and execution. This architecture can also be used to run software on multiple different robots by exchanging the execution layer [Taipalus and Halme, 2009]. A very sophisticated three-layer architecture is presented by Asfour et al. [2006], which also enables the ARMAR-III to execute tasks in parallel using the coordination layer [Asfour et al., 2004].

2.2 Previous Work

Einig [2012] proposed a three-layer architecture for parallel plan execution of sequential plans on mobile service robots. The sequential plan is generated using a SHOP2 planning system, which is part of the RACE architecture (figure 2.1). The middle-layer is a resource-based scheduling algorithm for parallel task execution. The SMACH executor, capable of executing tasks in parallel, is used to dispatch the tasks on the robot. SMACH was developed by Bohren and Cousins [2010].

Einig [2012] shows how parallel execution can reduce the execution time by more than 25 % for certain application areas. Even if only basic tasks can be parallelized, the benefit is still significant. In Einig et al. [2013] an enhancement to the three-layer architecture is proposed, closing the loop by introducing a replanning layer, which connects the execution and planning layer.

This work will reuse the scheduling layer, in order to show the effect of incorporating semantic costs for sequential and parallel plans.

2.3 Cost-Based Planning

Cost-based planning is a well-known approach for path planning. Usually the path planning algorithm is provided with a cost map, resulting from obstacles or challenges such as elevation or connectivity problems [Ohki et al., 2013; Suh and Oh, 2012; Yong and Meiling, 2012]. Although SHOP2 is capable of incorporating task costs in the planning domain and evaluating the resulting plans based on these costs, no major works regarding this topic have been published in recent years. The partial-order planning capabilities of SHOP2 even allow for disregarding the cost of tasks which are planned to be executed in parallel [Nau et al., 2003]. The reason for the lack of works on cost-based high-level planning with SHOP2 may be, that the information on task cost, as well as partial-order planning and thus reduced total plan cost have to be inserted into the domain before planning. Tasks without relevant cost for the total plan cost, as they are planned to be executed in parallel, have to be marked in the domain as well as the information on which tasks may interleave each other. This does not only increase the complexity of the planning domain, it is also uncomfortable to maintain. When adding cost for a task in the domain it is not possible to differentiate between the same task with different parameters. For example, driving to different positions will have the same cost when the same operator in the planning domain is used. The possibilities for using learned knowledge for cost-based planning with HTN planners such as SHOP2 are thus very limited. Off [2012] proposed an uncertainty planner based on SHOP2 principles, which puts cost and uncertainty of the generated plans into relation.

Rather than doing cost-based high-level planning, previous works tried to enhance the HTN planner to include temporal knowledge, focusing on wait-for relations, deadlines and starting points [Bhowmick et al., 2012; Tang, 2013]. Other attempts to enhance HTN planning are geometric and spatial planning. This is already part of the RACE project and the generated information by the spatial reasoner is used by the SHOP2 planner [Hertzberg et al., 2014; Rockel et al., 2013; Rost et al., 2012].

Belker et al. [2004] presented an approach for the prediction of the expected execution duration for a navigation plan within the APPEAL architecture [Belker et al., 2003]. They use the term Plan Projection, which has been used by McDermott [1992] and Beetz [2000]. The projection focuses on a plan containing planar motion tasks. The platform used for APPEAL is the PIONEER II robot with an attached laser scanner. It is not capable of performing complex manipulation tasks, thus the work by Belker et al. [2004] focuses on predicting navigation execution duration based on learned models of the environment, such as the passage width properties.

Although temporal enhancements exist and SHOP2 features basic capabilities for implementing knowledge for parallel planning into the planning domain, up to the authors knowledge, no attempts for generating optimal parallel plans from learned knowledge exist.

2.4 Project Setting

This work and the previous work have been realized at group TAMS[1], University of Hamburg, as part of the EU-funded project RACE. The architecture proposed in

[1]http://tams.informatik.uni-hamburg.de/, Last checked: 15.12.2014

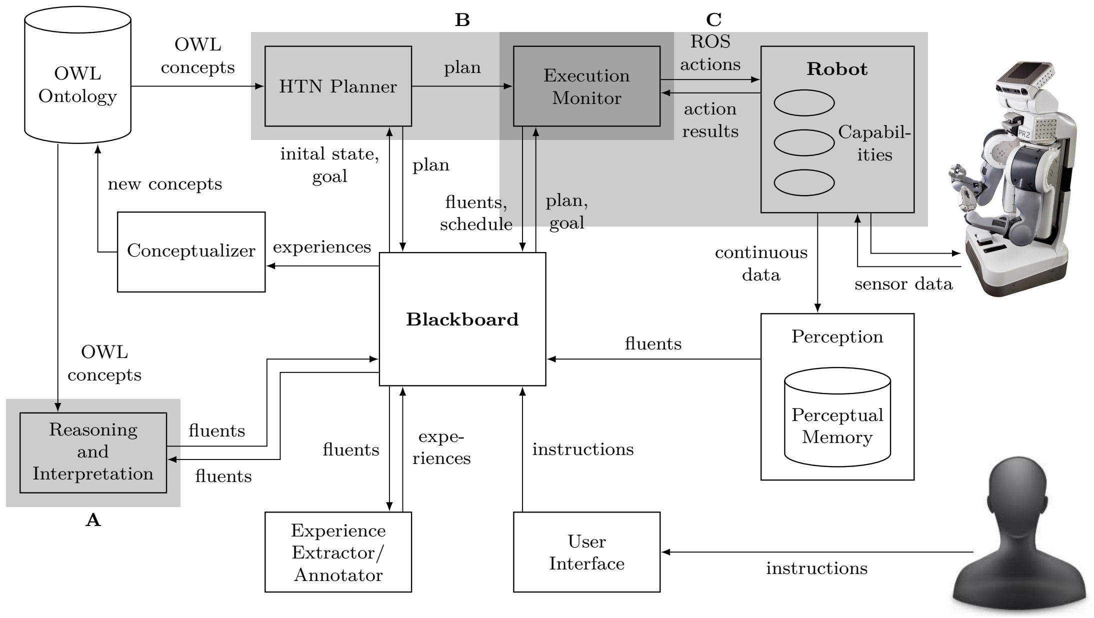
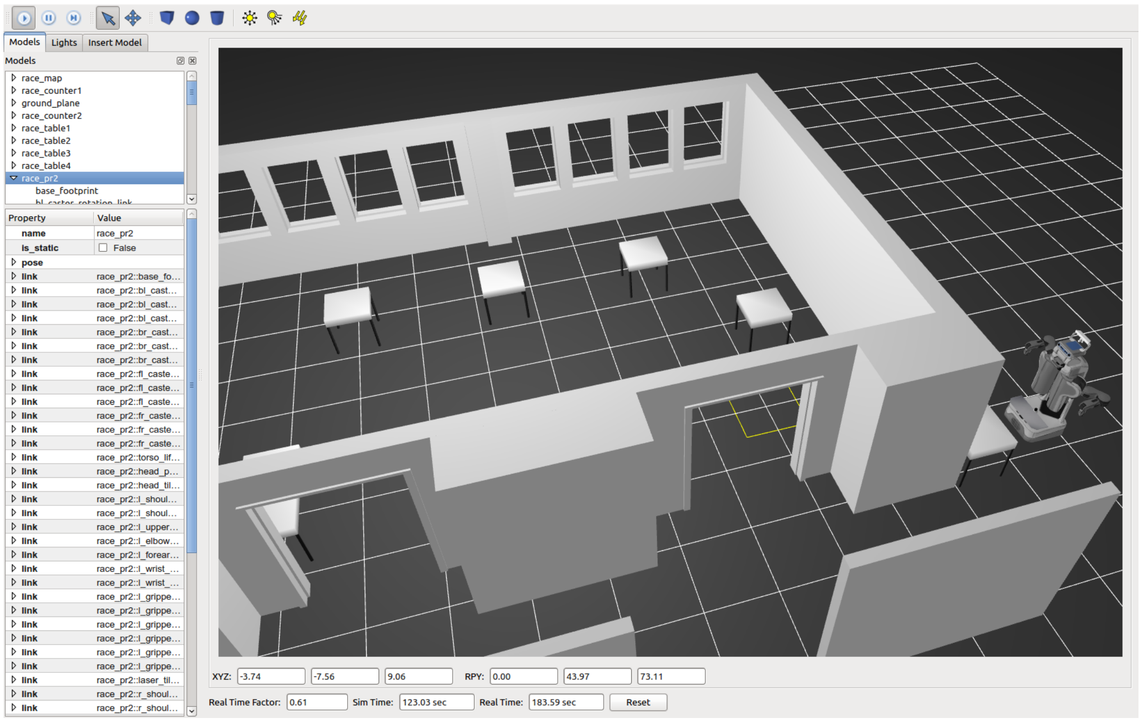


Figure 2.1: Global architecture of the RACE project. Block **A** highlights the reasoners, block **B** and **C** represent the three-layer architecture presented in Einig [2012]. The architecture is based on Hertzberg et al. [2014].

Rockel et al. [2013] is used as framework and for execution of the scenarios. The RACE system is capable of collecting experiences from the execution of tasks and storing them for future use. Another feature of the project is the imagination-based reasoner, allowing for the simulation of the plan execution before the plan is executed physically, in order to detect problems and failures [Rockel et al., 2014]. This system is capable of gathering experiences, such as execution durations, without physically conducting experiments. Einig [2012] introduced a three-layer architecture for planning and executing. This work wraps the planning and scheduling layer into a more complex component with the ability to reason on semantic cost resulting from the gathered knowledge during simulated and physical experiments. Figure 2.1 shows a sketch of the RACE architecture, as seen in Hertzberg et al. [2014]. Block **A** contains a set of reasoners, which push occurrences to the Blackboard. These occurrences are made publicly available by the Blackboard. Block **B** and **C** contain the three-layer architecture, which is used for this work. Part of the three-layer architecture is the HTN planner JSHOP2 in block **C**, which has been changed to SHOP2 recently as JSHOP2 was missing necessary features. Block **B** holds the scheduling and the execution layer. An interface to the robot has been introduced to the RACE architecture to simplify calling the robot capabilities from the executor.

The process of generating and executing plans within the race architecture is explained by Stock et al. [2014]. They described domain modeling and decomposition for the first year scenario of RACE: Serving a cup of coffee. The described executor is derived from the execution layer which is also the basis of this work [Einig, 2012]. The executor utilizes an execution middle-layer, which provides a ROS service for the executable capabilities of the Personal Robot 2 (PR2). Exchanging this middle-layer allows for running the executor on different robots. The described planning domain has been extended to increase the abilities of the robot.



Models

- race_map
- race_counter1
- ground_plane
- race_counter2
- race_table1
- race_table2
- race_table3
- race_table4
- race_pr2
  - base_footprint

Property	Value
name	race_pr2
is_static	☐ False
pose	
link	race_pr2::base_fo…
link	race_pr2::bl_cast…
link	race_pr2::bl_cast…
link	race_pr2::bl_cast…
link	race_pr2::br_cast…
link	race_pr2::br_cast…
link	race_pr2::br_cast…
link	race_pr2::fl_caste…
link	race_pr2::fl_caste…
link	race_pr2::fl_caste…
link	race_pr2::fr_caste…
link	race_pr2::fr_caste…
link	race_pr2::fr_caste…
link	race_pr2::torso_lif…
link	race_pr2::head_p…
link	race_pr2::head_til…
link	race_pr2::l_shoul…
link	race_pr2::l_shoul…
link	race_pr2::l_upper…
link	race_pr2::l_elbow…
link	race_pr2::l_forear…
link	race_pr2::l_wrist_…
link	race_pr2::l_wrist_…
link	race_pr2::l_grippe…
link	race_pr2::l_grippe…
link	race_pr2::l_grippe…
link	race_pr2::l_grippe…
link	race_pr2::l_grippe…
link	race_pr2::l_grippe…
link	race_pr2::laser_til…
link	race_pr2::r_shoul…
link	race_pr2::r_shoul…

XYZ: -3.74 -7.56 9.06 RPY: 0.00 43.97 73.11

Real Time Factor: 0.61 Sim Time: 123.03 sec Real Time: 183.59 sec

Figure 2.2: Gazebo simulator GUI. The modelled world is equivalent to the TAMS environment and the blueprints in chapter 3. The simulator is used to perform testing and evaluation.

The global scenario of the RACE project is given by a waiter in a restaurant. The robot is, amongst other features, supposed to be able to set the table, serve the guests and clean the table. While gathering experiences during the operation, these experiences are used to increase the robustness of the execution and competence of the robot. While the RACE scenarios consist of detecting objects and manipulating them within the restaurant environment, this work will omit the manipulation and focus on moving the robot to the desired location. The RACE project is thoroughly described in Hertzberg et al. [2014]. Additionally to the physical robot available at TAMS, the Gazebo simulator is used. The simulator with the modeled environment is depicted in figure 2.2.

The robot, which will be used to demonstrate the results of this work, is the PR2 developed by Willow Garage. The PR2 is a mobile service robot with a four-wheeled omni-directional base. The base has a laser scanner attached for obstacle detection and localization. A height-adjustable torso connects the base with the head and the two seven-Degrees of Freedom (DOF) manipulators. Each manipulator has a one-DOF gripper attached and an additional camera in the forearm. Mounted between the arms is a tilting laser scanner, which can be used to detect objects in the manipulation space. The head of the robot contains the remaining sensors. It can be tilted and turned to focus on an area for object detection. The head includes a wide-angle stereo camera and a narrow-angle stereo camera as well as an active infrared depth camera. The group TAMS attached an additional ASUS Xtion Pro Live sensor to the forehead. The robot is powered by two network connected Quad-Core i7 Xeon machines and runs on Robotic Operating System (ROS)[2]. In

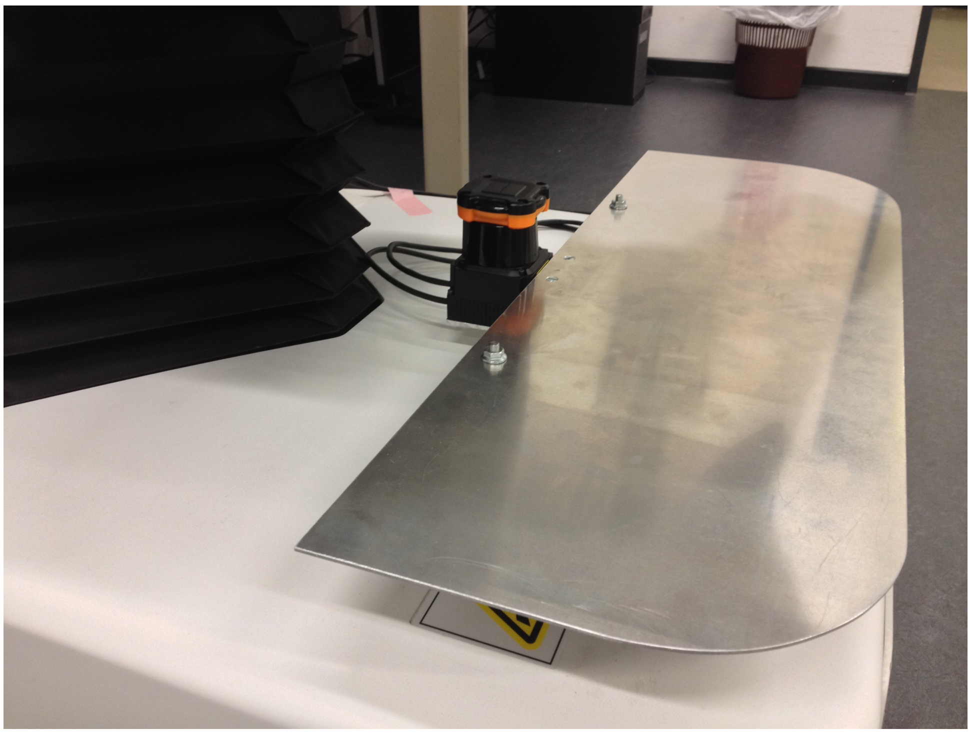


Figure 2.3: Tray mounted to the PR2 service robot. The tray is observed by a laser scanner to track objects on the tray and allows the robot for placing objects on in order to carry more than two objects at the same time. Image by courtesy of Sebastian Rockel.

the course of the RACE project a tray has been added to the base with a dedicated laser scanner to detect objects on the tray. This tray allows for performing multiple manipulations without moving by the robot. Without the tray, the robot could only carry two objects from one location to the other. By adding the tray, the robot may now place multiple objects on the tray and then drive with more than two objects. When driving with objects on the tray, the robot must consider the steadiness of the object. A tall object with a high center of mass is likely to fall over when the

[2]http://www.willowgarage.com/pages/pr2/overview, Last checked: 15.12.2014

robot turns, accelerates or decelerates.

In this chapter an overview on approaches to parallel plan execution and cost-based planning were given. The RACE architecture and the experimental platform for this work were introduced. In the next chapter two scenarios for evaluation and verification of the PlEv will be proposed.

3 Scenarios

In the previous chapter the environment for this work including the RACE architecture and the mobile service robot PR2 was explained. The foundation of the PlEv, the plan parallelization method by Einig et al. [2013] and other approaches to parallel plan execution were introduced. In this chapter the scenarios which will be used to evaluate the proposed planning architecture are described. These scenarios are closely related to the scenarios of the RACE project. In order to demonstrate the improvements of the plan execution, the layout of the restaurant has been adapted. The original restaurant only features two tables and one counter. The tables are arranged in a central position in the restaurant and have a north-south and a east-west orientation. Another two tables have been added and the tables have been rearranged to provide an increasing distance from the counter within the restaurant as well as a table which is not in between the doors. The original layout is limited to a single room, which is labeled *Restaurant*. The modified blueprint in figure 3.1 shows the adjacent hallway with the two connecting doors. For demonstration purposes, one of the doors is artificially narrowed and a counter is added to the hallway. This counter is positioned so the robot has to drive a U path in order to reach table 4. Therefore, the path length is directly proportional to the distance d between the center of each door and the counter in the hallway.

According to the spatial reasoner described in Hertzberg et al. [2014]; Mansouri and Pecora [2013], each piece of inventory is assigned corresponding premanipulation and manipulation areas as well as placing areas on top of the inventory. In order to be able to manipulate objects on a table or a counter, the robot must be within the manipulation area. In this area, the robot may not perform arm actions below the edge of the table, for example tucking the arms or assuming an arms posture enabling the robot to carry objects while driving. This arm posture is referenced to as *carryarm*. The manipulation area is reachable from the premanipulation area by driving without obstacle detection. Additionally, each door is assigned three areas in order to be able to traverse the door.

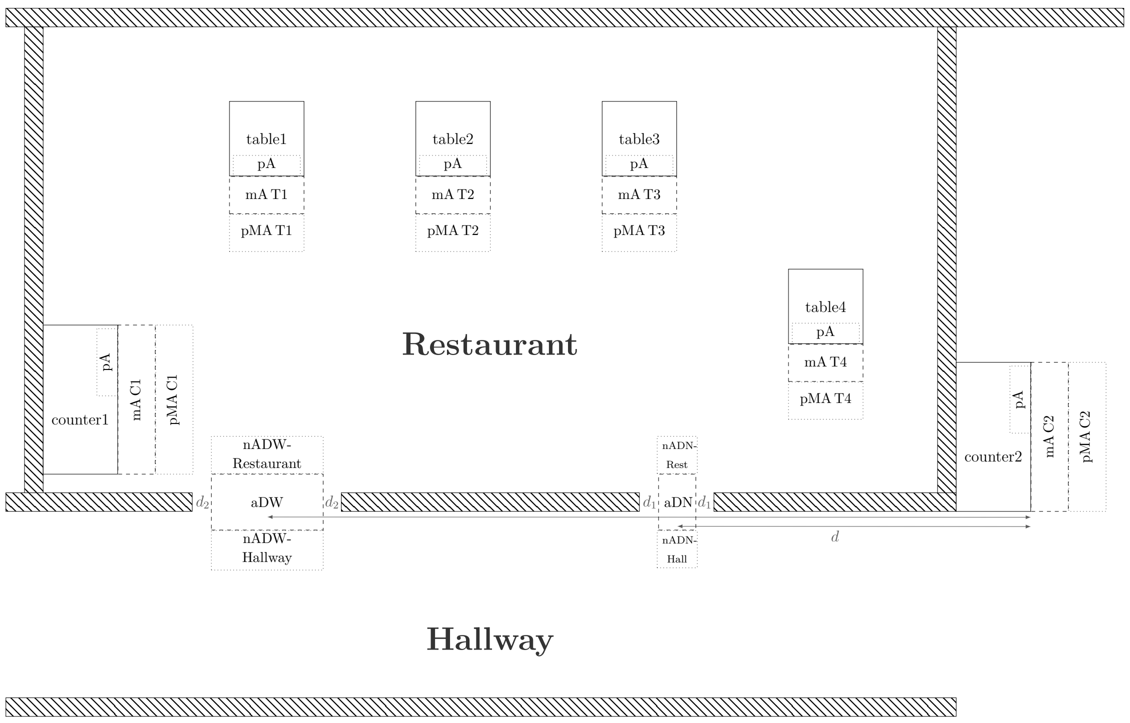


Figure 3.1: Blueprint of the modified restaurant environment, featuring two rooms, two counters to pick up objects from, four tables to place objects on and two traversable doors. The doors differ in their width property.

The domain for the planner had to be modified. A new top-level task had to be added, which is required to decide, if the robot has to traverse a door (listing A.1). This task checks whether the robot is in the same room as the target. If this is true, the original driving method is chosen. If this is false, the robot will use the method for traversing doors. This method only works for adjacent rooms, minor changes and a recursive call will enable this method to find a path through multiple rooms to the target. The door traversing methods are shown in listing A.2. The method for driving the robot through a door checks whether there is a door from the current room to the target room and decomposes to driving to the door, traversing the door and then driving to the target position. The traversing method also checks the door width property and enforces the arms to be tucked and the torso lowered, if the door has the property narrow.

The original driving methods are shown in listing A.3. In the original domain, the planner always assumes a driving posture with torso and arms, by lowering the torso and tucking the arms. The modifications in listing A.4 enable the planner to decompose plans into driving slowly without assuming a torso driving posture or assuming a general driving posture in order to be able to drive fast. This will still allow the robot to decompose to executing the next step by moving slowly, giving the planner more freedom in its choices. The available *!move_base_param*

?to ?speed operator is used and extended with the current location of the robot. This is required to find the expected duration for the driving task in the next step. The modified moving operators are shown in listing A.8. Giving the planner a choice for decomposing the driving task results in a set of plans which differ, among other things, in the point of time within the plan, where the robot will assume a driving posture, if the torso driving posture is assumed at all. It is expected, that either the plan with the earliest step or without assuming the torso driving posture will be the optimal plan. The parameterized movement operator also checks if there is an object on the tray which forces the robot to drive at a slower speed than the maximum velocity. In order to be able to remove objects from the tray, the *clear_tray* method shown in listing A.5 is available. Listings A.6 and A.7 show the corresponding operator for picking and placing an object in an area on the tray. These methods allow the robot to decompose more complex tasks into efficient plans. For instance, the robot is able to carry up to five objects at the same time by placing three objects on the tray and carrying two objects in the grippers with the arms in a carry posture.

3.1 Attend Table

The *attend table* scenario puts the mobile service robot PR2 in front of the restaurant counter. The base-mounted tray is not holding any objects. The scenario is run by passing a *drive_to table#* command, where # may be 1-4. This will make the planner decompose a plan for driving from the current location to the desired table. This scenario omits the manipulation of objects, as this did not show parallelization potential in Einig et al. [2013]. The torso of the robot is in an upper position and both arms are in an untucked state. The original planner decomposes this problem to the following plan.

1. *!MOVE_BASE_BLIND PREMANIPULATIONAREACOUNTER1*
2. *!MOVE_TORSO TORSODOWNPOSTURE*
3. *!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE*
4. *!MOVE_BASE TABLE#*

A sketch of the scenario and possible paths for moving to the tables are depicted in figure 3.2.

First of all, the robot has to get out of the manipulation area of the counter, to be able to perform any further actions. As there is no object on the tray and the

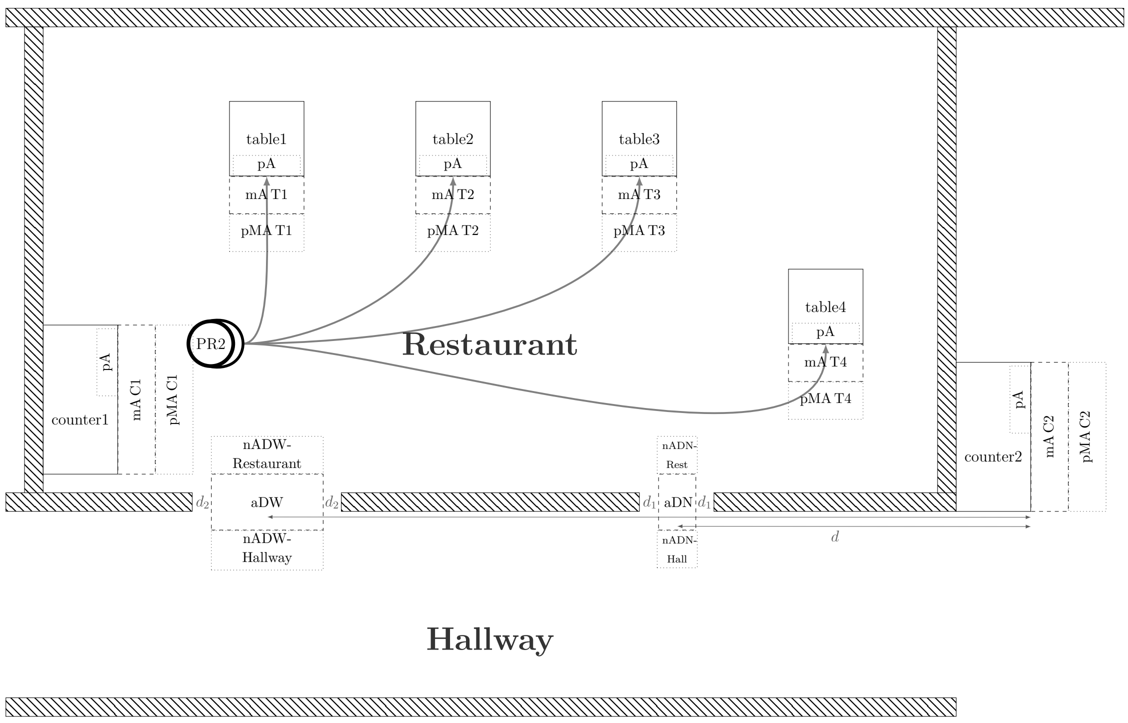


Figure 3.2: Sketch of the *attend table* scenario. The robot has a no objects on its tray, the torso is in an upper position and both arms are in an untucked state. The arrows show the possible paths for attending the four tables within the room.

robot is not holding any objects, the planner decides to move the torso into a lower position and then tuck the arms to get into a driving posture. The robot will then move to the desired table at regular speed.

The target for this scenario is to find a plan which is shorter in execution time. Depending on the distance to the target table, the evaluation layer will find a plan which is faster than or as fast as the original planner. A possible decomposition will be to skip moving the torso to a lower position and drive at a slow speed. Depending on the difference between fast and slow speed, combined with the distance to the target table, skipping this step will be the shorter plan. If given the decomposition choices presented in the adaptations to the planning domain previously in this chapter, the original decision will always be the three-stepped plan without moving the torso. The evaluator will be able to find a shorter plan for both the original domain as well as the modified domain when comparing the minimum steps decision with the temporal evaluation. Additionally, adaptations to this scenario will be discussed, including manipulation of objects and the integration of the tray.

3.2 Door

This second scenario is more complex than the *attend table* scenario, as it involves moving from one room into another. With the current planning domain, it is assumed, that the two rooms share a door, making them adjacent rooms. As described in section 3, it is also possible to find plans for non-adjacent rooms. It is also assumed, that the doors which connect the rooms are open. An approach on how to deal with uncertain knowledge for example about the state of doors is presented by Off [2012] and is not relevant for this work.

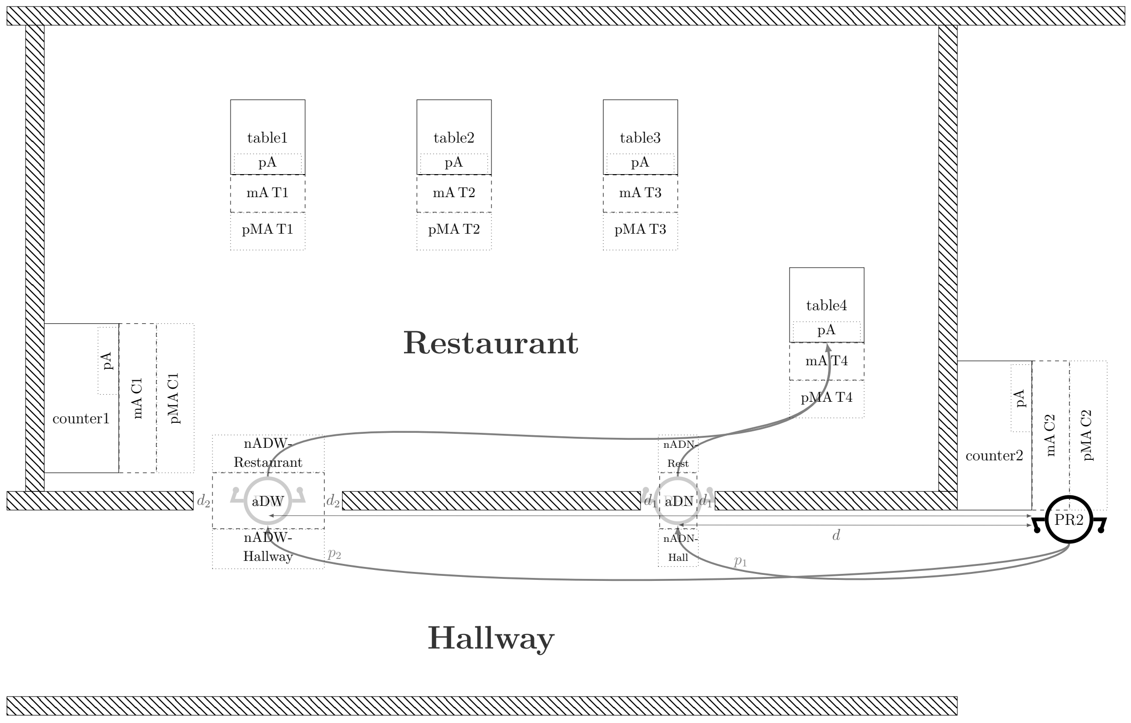


Figure 3.3: Sketch of the *door* scenario. The robot has a no objects on its tray, the torso is in an upper position and both arms are in an untucked state. The arrows show the possible paths for traversing the doors in order to arrive at table 4.

The *door* scenario puts the PR2 in front of the hallway counter. The tray is not holding any objects, the torso is in an upper position and both arms are in an untucked state. The command issued to the planner is *drive_to table4*. Table 4 is located in the restaurant room, which has two connecting doors to the hallway. For evaluation purposes, the distance d between the two doors is relevant for the results of this scenario and shifting the door labeled d_2 to adjust the distance will be discussed in the evaluation. It is also possible to change the target table to receive more significant results. The two doors d_1 and d_2 are open and the robot can pass them. Door d_1 is a narrow door, d_2 is a wide door. Narrow doors can only be passed by the robot, if the robot is in a driving posture. Both arms must be tucked and the torso must not be in an upper position. This reduces the footprint of the robot, thus allowing the robot to pass through narrow spaces. The wide door may be passed without any further restrictions, as it is wide enough for the robot to fit with untucked arms and a torso in an upper position

A method to traverse doors is missing in the original planner and thus it is not capable of decomposing this problem. After adding this method to the planning domain, without further changes, the planner decomposes the problem to one of the following plan.

(A) Plan for narrow door
   1. *!MOVE_BASE_BLIND PREMANIPULATIONAREACOUNTER2*
   2. *!MOVE_TORSO TORSODOWNPOSTURE*
   3. *!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE*
   4. *!MOVE_BASE NEARAREADOORNARROWHALLWAY*
   5. *!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT SLOW*
   6. *!MOVE_BASE TABLE4*

(B) Plan for wide door
   a) *!MOVE_BASE_BLIND PREMANIPULATIONAREACOUNTER2*
   b) *!MOVE_TORSO TORSODOWNPOSTURE*
   c) *!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE*
   d) *!MOVE_BASE NEARAREADOORWIDEHALLWAY*
   e) *!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT SLOW*
   f) *!MOVE_BASE TABLE4*

A sketch of the scenario and possible paths for moving to the tables are depicted in figure 3.3.

The first step for the robot is to get out of the manipulation area of the counter. The planner tucks the arms and moves the torso into a driving posture, which is a lower position. This enables the planner to take either p_1 or p_2 as the prerequisites for both paths are met and both plans have the same amount of steps for completion. In order to traverse a door, the robot moves into the corresponding *nearAreaDoor* within the room it is currently in. From this area, the robot is capable of traversing the door by moving to the *nearAreaDoor* in the target room and then continuing to the target table.

The target of this scenario is to demonstrate, that even though the original planner finds two plans with the same amount of steps for traversing both doors, there is an optimal plan depending on the distance d between the two doors. Additionally, by allowing the planner to find plans without assuming a driving posture ahead of moving, a shorter plan can be found by choosing the wide door and omitting the torso adjustment. It is expected, that for long distances d, the optimal plan will be plan (A), whereas for short distances or a target table 1, the plan (B) or a plan without assuming the driving posture and driving slowly is the shortest plan. If the original planner is given the opportunity not to assume the driving posture and drive through d_2, it will always choose this plan due to the reduced amount of steps. Clearly, this will not always be the shortest plan, as the robot has to drive slowly if it is not in a driving posture.

In this chapter two scenarios for evaluating and verifying the PlEv and the current plan decomposition by the RACE HTN planner were introduced. In the next chapter the Temporal Experience Extractor (TXX) is presented, which collects the required temporal data for evaluating the plans based on their total execution duration.

4 Temporal Experience Extraction

In the previous chapter two scenarios to evaluate the proposed plan optimization layer and the required domain modifications were presented. In this chapter, the Temporal Experience Extractor (TXX) for extracting execution durations is presented. The knowledge about execution duration is required to evaluate the semantic cost for a generated sequential plan. The TXX will utilize the RACE architecture, to find begin and finish times for tasks as well as to store information about execution durations for the tasks. In chapter 3 modifications to the domain which add the starting area for movement tasks, such that the planner is able to find the corresponding duration were already presented.

4.1 Foundation

The TXX maps execution time durations to plan operators. Einig et al. [2013] presented the SMACH plan executor. How to create an executable state machine from a plan using SMACH is briefly described in Einig [2012]. Each SMACH state directly corresponds to a plan operator. Therefore, the execution duration for each state is also equal to the execution duration of the plan operator. The total execution time for a plan operator is not only the visible active time, e.g. while the robot is moving, but also the required computation time for path planning. The TXX may track the time duration for each state of the state machine to collect the required temporal knowledge. This gathered knowledge must be stored and preprocessed accessible for the plan optimization layer.

4.2 Implementation

The measured time duration for each SMACH state is passed to the execution time duration interface. This interface stores the 50 newest time duration values for each plan operator. A plan operator in this context is the action and all arguments which are to be executed. As the PR2 always tucks the right arm below the left arm, the *!ARM_TO_SIDE* operator has different execution durations depending on the arm which is passed as argument. Extracted durations are stored in a local data file.

The time duration interface passes preprocessed time durations to the plan optimization layer. At first, outliers are filtered from the measured durations by calculating the standard deviation and removing all entries which exceed this limit. The average of the remaining durations is passed to the plan optimization layer. The time duration interface as well as the plan optimization layer already hold an additional field for the expected deviation of each operator. This may be used for calculating minimum, maximum and average expected plan durations in the future and thus allows further improvements. Another possible improvement is the introduction of a weighted average in order to apply a higher weight to recently measured durations. This will improve response time to environmental changes. As for the evaluation a static environment is assumed, the previously mentioned improvements have no priority for this work. Paths or actions which are not yet in the database are initialized with a duration of zero in order to discover new paths. For each new environment, initial durations should be collected using simulation to prevent the robot from unnecessarily running long paths and plans.

Algorithm 1: Time duration preprocessing

```
1  def getOperatorCost()
2      if (OperatorName, OperatorArguments) in loadDurations() then
3          average = average(durations)
4          deviation = std_dev(durations)
5          foreach duration in durations do
6              durations = filter(removeOutliers, durations, average, deviation)
7          return average(durations), std_dev(durations)
8      else
9          return 0,0
10 def removeOutliers(duration, average, deviation)
11     if absolute(duration- average) <= deviation then
12         return duration
```

4.3 Evaluation

In order to verify and evaluate the measured durations for this work, multiple simulations have be conducted, generating at least 22 durations for each operator and path. In total 1810 time duration values have been collected. Figure 4.1 shows the unfiltered deviation from the mean. The thin line shows the occurrences of the percentile deviation in one percent intervals, the thick line shows the bezier smoothed distribution. For each operator, the mean is calculated and the measured time durations are converted to percentile deviation from this mean. As expected, the percentile deviations for all operators combined result in a normal distribution.

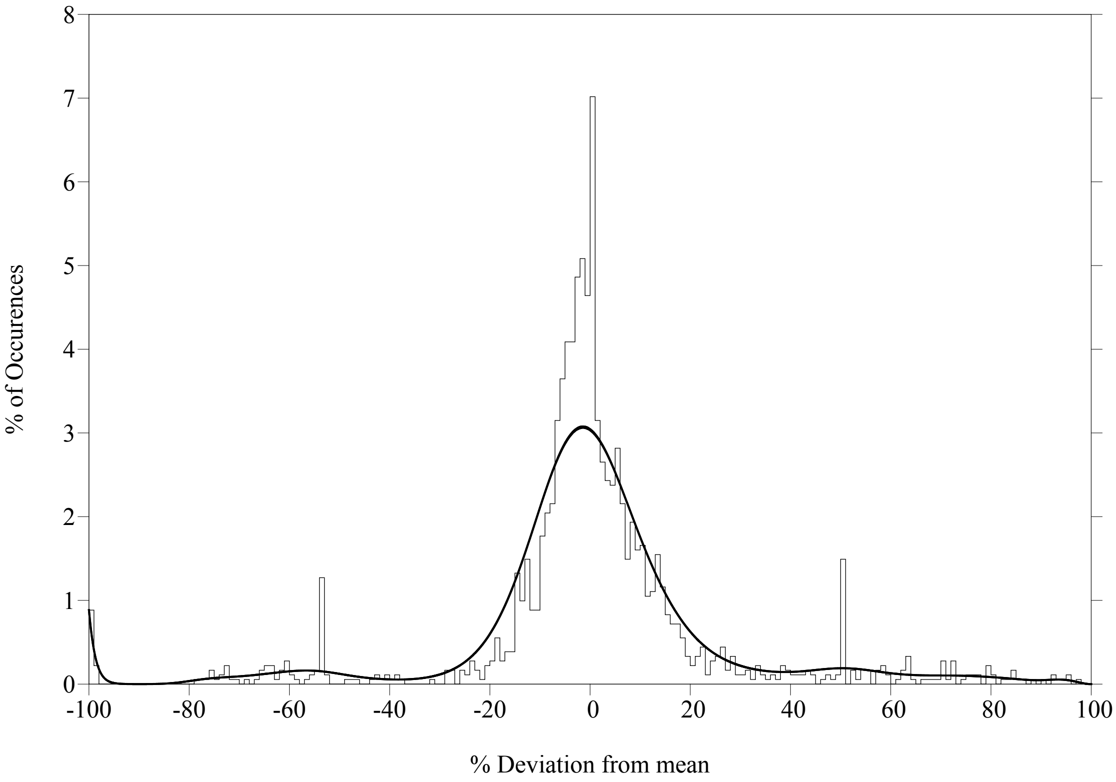


Figure 4.1: Deviation from the mean of unfiltered time duration data. The thin line represents actual data consolidated to whole percent steps. The thick line represents bezier smoothed data. The x-axis shows the deviation from the mean in percent, the y-axis shows the frequency of occurrences.

Figure 4.1 shows three peaks aside the normal distribution. The peak at -100 % results from operators which had execution times near zero seconds. This occurs, e.g., if the robot attempts to tuck its arms when the arms are already in a tucked posture or tries to move to a location, where it already is. In order to ensure, that the arms are tucked for certain actions, the planner sometimes introduces unnecessary operators to the plan. The other peaks at approximately ±50 % result from the

!MOVE_TORSO and *!MOVE_ARMS* actions. These actions provide no information on the previous position. Therefore, moving the torso from an upper position to a lower position is assigned to the same time duration list, as moving the torso from a middle position to a lower position. The arms actions also depend on the previous posture of the arms. Moving the arms to the side is faster coming from an untucked posture than coming from a tucked posture.

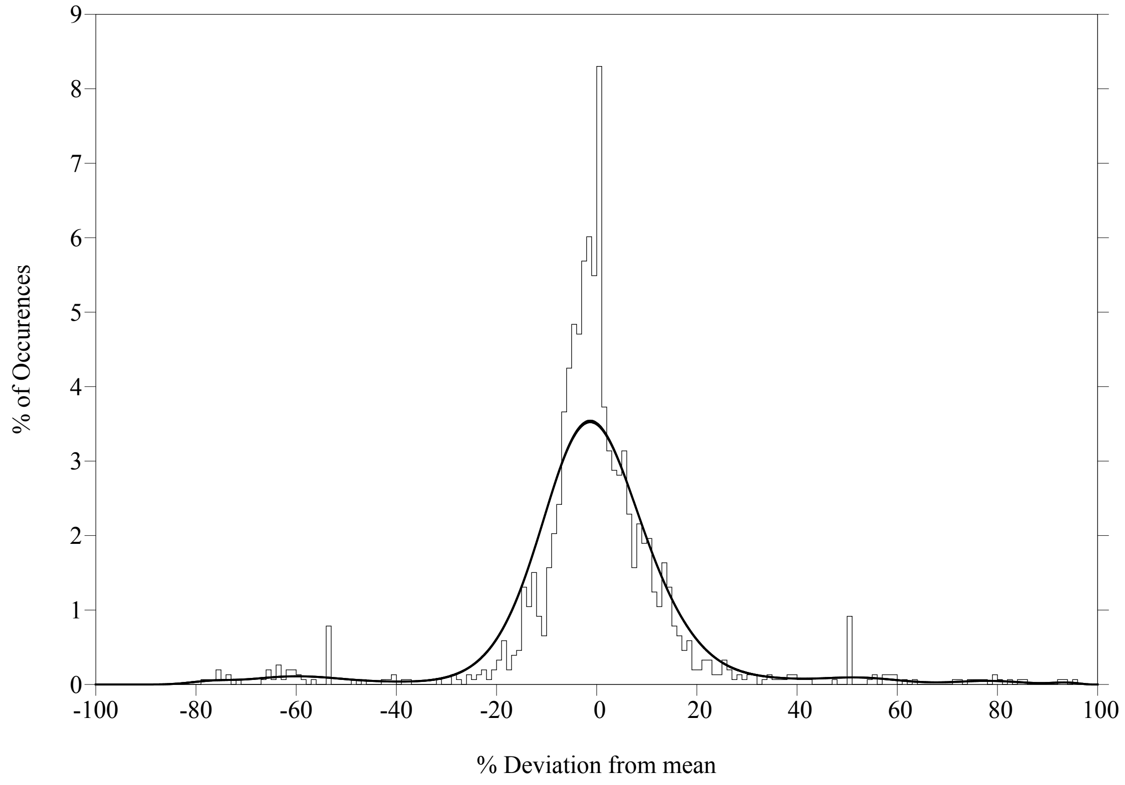


Figure 4.2: Deviation from the mean of filtered time duration data. Outliers have been removed.

Figure 4.2 shows the same data, but as described in Algorithm 1 outliers have been removed. The peak at -100 % is removed and also a few data-points far beyond +100 %, which are not plotted in figure 4.1. These occurrences are less than 0.7 % of the total data points and result from failed simulation runs. The problem with the torso and arm actions described above persists after removing the outliers. Figure 4.3 shows the time durations with removed outliers. Additionally, the operators with large deviation have been omitted. These operators are:

- *!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE*,
- *!MOVE_ARMS_TO_CARRYPOSTURE*,
- *!MOVE_TORSO TORSOMIDDLEPOSTURE*,
- *!MOVE_ARM_TO_SIDE LEFTARM1*,
- *!MOVE_ARM_TO_SIDE RIGHTARM1*.

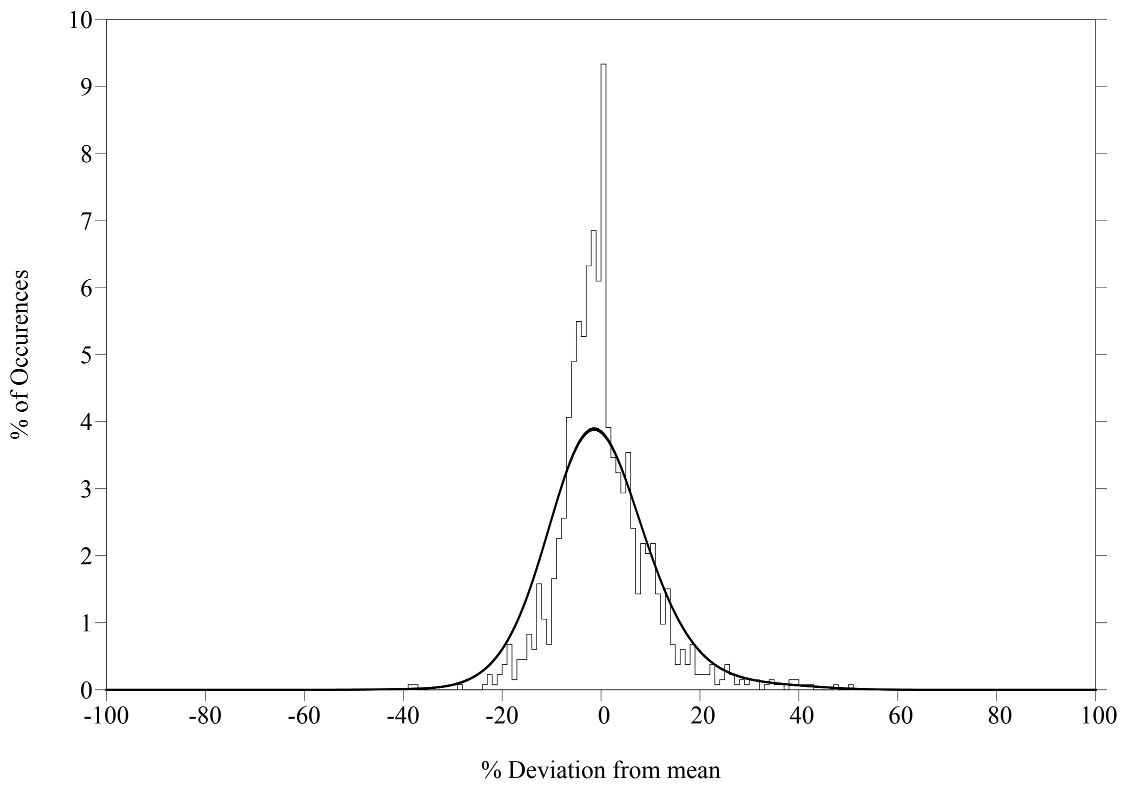


Figure 4.3: Deviation from the mean of filtered time duration data. Outliers and operators with large deviations have been removed.

By adapting the planning domain and the RACE architecture shown in figure 2.1, it is possible to assign the previous arm and torso state to these operators and thus increase the precision of the TXX. In general, a more precise and small-stepped planning domain results in more precise temporal data and thus increases the precision and reliability of the plan optimization layer. Yet, the collected data are sufficient to demonstrate the capabilities and improvements of the plan optimization layer presented in chapter 5.

In this chapter the TXX which collects the required temporal data for evaluating plans based on their total execution duration was presented. The normal deviation shows the feasibility of the collected data.

5 Plan Evaluator

The knowledge gained by the TXX proposed in chapter 4 provides the input for the plan optimization layer Plan Evaluator (PlEv) which evaluates the scenarios described in chapter 3. In this chapter an introduction to the foundations of the PlEv is given and planning based on semantic cost, as well as the implementation of the PlEv is presented. In chapter 6, the results from simulating the scenarios will be displayed.

5.1 Foundation

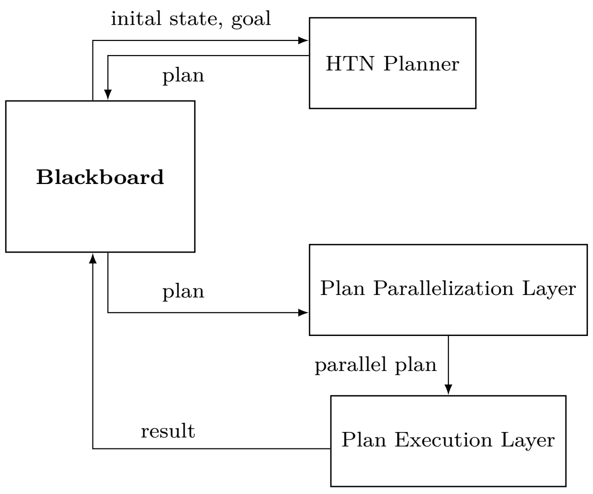


Figure 5.1: Original architecture of the plan parallelizer. The three-layer architecture is connected via the Blackboard. The planning layer pushes plans to the Blackboard, the parallelization layer pulls the plans and initiates the execution. The result is pushed to the Blackboard.

The relevant components of the plan parallelizer presented in Einig [2012] are pictured in figure 5.1. Although it is presented as a three-layer architecture, the planning layer and the parallelization layer do not communicate directly, but rather utilize the Blackboard presented in Hertzberg et al. [2014] and Rockel et al. [2013]. The plan generated by the HTN planner is pushed into the Blackboard. The plan parallelization layer listens to the Blackboard waiting for a plan. If a plan is received, the plan is converted to a parallel plan and forwarded to the execution layer. The execution result of the plan and each step is passed to the Blackboard for further reasoning.

For this architecture it is not possible to pass multiple plans generated by the HTN planner to the parallelization layer or the PlEv. Additionally, this architecture would not work without using the Blackboard. The PlEv is supposed to rely on as few external components as possible. With the architecture presented in figure 5.2, it is possible to use the PlEv without the Blackboard, although some components have to be replaced. For example, the execution layer relies on information from the Blackboard for coordinates of locations in the plan. In order to work with the RACE system, all required information is passed to the Blackboard, including the optimal plan and the result.

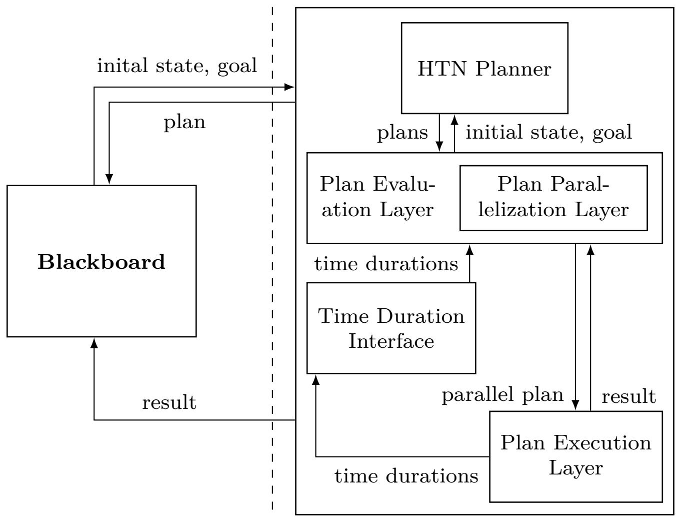


Figure 5.2: Enhanced architecture of the Plan Evaluator. The components are encapsulated. The Blackboard listens to the plan and receives the result. Planning, optimization and execution are directly coupled.

The HTN planner is called by the plan evaluation layer and returns all possible plans to reach the goal from the initial state. The plan evaluation layer fetches the temporal cost for each plan step from the time duration interface and evaluates the plans after finding a parallel order for each plan. The parallel plan with the least expected temporal cost is then passed to the execution layer. During the execution, the execution layer updates the time duration interface with the recently collected temporal data.

The control flow of the three-layer architecture presented in Einig [2012] was top-down, starting with the HTN planner, via the parallelization layer to the execution layer and, regarding the enhancement presented in Einig et al. [2013], the replanning layer. The updated architecture changes the control flow, while remaining a three-layer architecture. Planning and execution is controlled by the plan evaluation layer, which includes the parallelization sub-layer. It is still possible to use the replanning layer as a feedback from the execution layer to the evaluation layer. The replanning layer may also be included into the evaluation layer, as the feedback from the execution layer is already received. The closed-loop architecture in Einig et al. [2013] is no longer required.

5.2 Planning Based on Semantic Cost

In order to be able to evaluate plans based on semantic cost, the planning domain has to be adapted to allow for a more freely decomposition of tasks. The previous planning domain enforces assuming a driving posture ahead of every driving task. The two methods for decomposition of the driving tasks are shown in listing A.3. The methods differ in the previous location. The first method is used for arbitrary positions other than manipulation positions. The second method first leaves the manipulation position, then assumes the driving posture. This is required for the robot to be able to drive close to tables, which is prevented by the obstacle detection otherwise. Listing A.4 shows the adaptations to the driving methods. For both, coming from a manipulation position or arbitrary position, a method is added, which, in case there is an object on the tray, picks this object from the tray (listing A.4, lines 11-27; 58-76). The methods for driving without assuming a driving posture require the robot to move at a slower speed (listing A.4, lines 1-10; 45-57). These methods can be applied for decomposition if there is an object on the tray, as well as if there is no object on the tray. This gives the robot at least two choices for each driving task. If there are multiple driving tasks, the amount of possible different decompositions increases exponentially, in addition to other possible options the planner has.

The HTN planner as utilized for RACE decides to use the shortest plan by the plan length. As the domain holds no information on the plan costs, the plan length is equal to the amount of steps. Even for the adapted planning domain with the new driving methods, the shortest plan will be distinct, as the driving methods, which the planner may choose between, differ by at least one plan step.

Besides the chosen plan evaluation strategy, the HTN planner itself is able to evaluate plans by choosing the shortest parallelized plan. The planner can generate parallel plans if the domain is adapted accordingly. These plans can be evaluated either by using the number of plan steps or by manually adding the estimated cost to each operator. Another possibility is to generate a domain from a database of methods, operators and expected time durations. There are multiple drawbacks to these approaches. The largest drawback is the parallelization feature of SHOP2. It is required to give the planning domain hand-written information on the operators, which can be executed in parallel. This has to be done for each method, instead of attaching the information to the operator. For a complex domain with a small amount of operators (<10) and a large amount of methods (>25), the required effort exceeds the benefit and compromises the consistency of the execution order due to human mistakes. Additionally, for every change in the planning domain, the parallel execution information has to be rechecked. By manually adding the estimated cost for each plan operator to the planning domain the precision and the currentness of the temporal data pose a problem. For a newly created domain, operator cost have to be either guessed, or discovered by conducting simulations or experiments ahead of creating the domain. During daily operation, operator cost may change and thus the planning domain has to be adapted by hand. Hand-written operator cost requires constant human supervision. This is not required, if the domain with the respective operator cost is created using a specialized compiler. A possible option is to write the methods for the domain by hand and attach the compiled operators with the expected cost. The effort for creating a compiler for this purpose, regarding the parallelization drawbacks mentioned above, may be too large. A general tool for compiling the planning domains which is not only capable of adding the expected cost to the operators, but also includes the parallelization information for the methods, seems reasonable. This tool could support the process of writing a domain in general but the complexity exceeds a master thesis.

5.3 Definition

Definition. *Assuming, that the planner finds all feasible plans to reach the desired goal state from an initial state within a given environment and previously computed time durations for each step in every feasible plan, the presented PlEv will always find the plan with the shortest expected time duration required for execution.*

The set S of all feasible plans holds all plans, which are safely executable, without violating the execution constraints for certain tasks, such as assuming an arms driving posture ahead of driving. Each plan p within S consists of n_p plan steps s_i with an expected required execution duration $d(s_i^p) = t_i^p$ for each step. For the sequential execution order, a plan is a simple list of plan steps and thus

$$min_t \left(\left\{ \sum_{i=0}^{n_p} d(s_i^p) \middle| \text{ for each } p \in S \right\} \right) \tag{5.1}$$

returns the shortest plan. For parallel executable plans, a plan is a convoluted list of lists. With s_i being a single step within any plan and L_{seq} a list of sequential steps and L_{par} a list of a parallel steps, then

$$\begin{aligned} d(L_{seq}) &= \sum_{i=0}^{n} d(s_i) \\ , d(L_{par}) &= max_t(\{d(s_i)\}). \end{aligned} \tag{5.2}$$

Extending the possible elements of such a list of sequential or parallel steps from single steps to including lists of sequential or parallel steps $c_i = \{s_i | L_{par} | L_{seq}\}$, then

$$\begin{aligned} d(L_{seq}) &= \sum_{i=0}^{n} d(c_i) \\ d(L_{par}) &= max_t(\{d(c_i)\}). \end{aligned} \tag{5.3}$$

Resulting from (5.1) and (5.3),

$$min_t \left(\left\{ \sum_{i=0}^{n_p} d(c_i^p) \middle| \text{ for each } p \in S \right\} \right) \tag{5.4}$$

returns the shortest feasible plan by expected execution duration time.

5.4 Implementation

The evaluation process is depicted in Algorithm 2. In the first step, the SHOP2 planner is called. As described in chapter 3, all possible plans for reaching the goal from the initial state are returned. For each action of each plan, the cost interface presented in chapter 4 attaches the cost calculated from the previously gained knowledge. These plans are passed to the parallelization algorithm presented in Einig [2012]. In order to be able to compare the different approaches to find the

Algorithm 2: Plan Evaluation

```
def evaluatePlans()
    planList = callSHOP2()
    foreach plan in planList do
        foreach operator in plan do
            getOperatorCost(operator)
    planListParallel = planToParallel(planList)
    foreach plan in planList do
        stepList = length(plan)
        sequentialList = getPlanDuration(plan)
    foreach plan in planListParallel do
        parallelList = getPlanDuration(plan)
    return minimum(parallelList),minimum(sequentialList),minimum(stepList)
```

shortest plan, the total plan length is calculated thrice for each plan:

- The length of the plan corresponds to the previously described approach of finding the shortest plan, by selecting the plan with the minimum amount of steps.
- For the sequential plans, the single execution time duration values are summed up to receive the total execution time (Algorithm 3).
- For the parallelized plans, the execution durations are summed up. For parallel executable tasks or sublists, the task or sublist with the maximum required execution duration represents the total execution duration. This recursively applies for nested parallel task lists (Algorithm 3).

The parallelized plan with the minimum total execution duration is passed to the execution layer. The computation of the sequential duration execution and the step length of the plans is omitted for application purpose. For this work, these three values are used to evaluate the advantages of each approach and to discuss possible scenarios besides the two evaluation scenarios in chapter 3 and 6.

While the program logic for evaluating the plans returned by the HTN planner is rather simple, uncoupling the SHOP2 planner from the RACE architecture and including the planner into the PlEv architecture in figure 5.2 required more effort. Within the RACE architecture the SHOP2 planner is wrapped in a ROS node listening to a goal topic. Depending on the received goal, the planning process is started and the resulting plan is returned on a ROS topic. This architecture does not allow passing multiple plans to the evaluator. Therefore, the planning node is removed and a new wrapper is introduced which allows direct access to the Lisp-

Algorithm 3: Total Plan Duration

```
def getPlanDuration(list)
    foreach item in list do
        if item is Operator then
            duration,durationDev += item.cost, item.deviation
        if item is PList then
            duration,durationDev += getParallelPlanDuration(item)
        if item is SList then
            duration,durationDev += getPlanDuration(item)
    return duration,durationDev
def getParallelPlanDuration(list)
    foreach item in list do
        if item is Operator or SList then
            duration,durationDev = getPlanDuration(item)
        if item is PList then
            duration,durationDev += getParallelPlanDuration(item)
        maxDuration,maxDurationDev =
        maximum(maxDuration,duration),maximum(maxDurationDev,durationDev)
    return maxDuration,maxDurationDev
```

written SHOP2 planner from within the Python-written PlEv architecture. This new wrapper is capable of returning all possible plans from the planner.

In this chapter the plan evaluation layer, which is capable of finding the optimal plan regarding total execution duration from a set of sequential HTN generated plans was introduced. In the next chapter the results of applying the PlEv to the scenarios described in chapter 3 are presented.

6 Results

In the previous chapter the plan evaluation layer as part of the PlEv architecture was presented. This architecture includes a SHOP2 planner with two evaluation scenarios (chapter 3) and the TXX (chapter 4). In this chapter the results from applying the evaluation option of the PlEv to these scenarios are described. The results will be evaluated in chapter 7 and possible modifications to the layout of the existing scenarios or other scenarios will be discussed in chapter 8.

6.1 Attend Table Scenario

The resulting plans for the *attend table* scenario regarding tables 1, 2 and 3 (figure 3.1) only differ in the total execution time. The shortest plans regarding the three metrics are equivalent. Therefore, only table 1 will be considered to represent the result as well as the evaluation afterwards. In general, there are three different possible decompositions to reach the goal state, which is arriving in front of the desired table.

The first decomposition contains three actions and is referred to as plan A. The planner makes the robot drive from the manipulation position of counter 1 to the premanipulation position of counter 1, so further actions may be performed. The next step is tucking the arms, which is always necessary to fit the robots footprint to the model for collision detection. Lowering the torso is omitted, as it is not required. As the torso is not lowered, the only possible further decomposition is moving slowly from the premanipulation position of counter 1 to the premanipulation position of table 1, respectively tables 2, 3 and 4.

The other two decompositions are very similar. They are referred to as plan B, respectively plan C and only differ by the choice of speed the planner makes, as described in chapter 3. As for plan A, the planner starts with decomposition of moving to the premanipulation area of counter 1 and then tucking the arms. This plan also contains a step for moving the torso to a lower position in order to assume a torso driving posture. As the planner is given the choice, the next step is moving to the premanipulation area of table 1, respectively tables 2, 3 and 4, at slow or fast speed. This choice is possible as the assumed driving posture allows for both speeds. Naturally, plan C will always be slower as plan B. It is just listed for integrity. All possible resulting plans for tables 1 to 4 are shown in appendices B.1-B.4. The plans are listed with their respective cost and sorted by their parallel execution duration. The first number is the cost of the parallel execution, the second number in brackets is the cost of the sequential execution and the last number is the amount of steps in the plan. These are also the metrics presented in section 5.4, which will be used to evaluate the shortest plan, regarding the following evaluation strategies:

- shortest plan by amount of plan steps,
- shortest plan by sequential execution duration and
- shortest plan by parallel execution duration.

For all tables, the shortest plan by the amount of steps is plan A, as it omits lowering the torso. For tables 1, 2 and 3, this is also the shortest plan by sequential execution duration and parallel execution duration as well as for table 4 by sequential execution duration. For table 4 by parallel execution duration plan B is the shortest plan.

The difference between table 1 and table 4 is the distance, which has to be traveled and thus the required time. The time required for moving the robot depends on the distance and the speed at which the robot moves.

Table 1

For plan A (listing B.1), there is no choice of the speed, thus the robot has to travel at slow speed, which requires 9.8 seconds from counter 1 to table 1. Plan B (listing B.2) allows for driving at a fast speed. Traveling from counter 1 to table 4 at fast speed takes 7.9 seconds, which is a difference of 1.9 seconds. The additional torso movement takes 19.0 seconds for a sequential execution order. As the torso movement may be executed in parallel with tucking the arms, the drawback by executing the torso movement is only 8.2 seconds as tucking the arms takes 10.8 seconds. The parallel execution of plan B requires 8.2 seconds more for the additional torso movement and 1.9 seconds less due to the available faster moving speed enabled by the torso movement. Thus, in total, plan B requires 6.3 seconds more than plan A, even when plan A is executed sequentially. The resulting shortest plan for table

1 is shown in listing 6.1.

Listing 6.1: Plan with shortest parallel duration for *attend table* 1 scenario

```
Plan with 32.2 (32.2) cost and 3 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE1 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1- 9.8302608696
)
```

Table 4

For plan A (listing B.20), traveling at slow speed requires 26.9 seconds from counter 1 to table 4. Moving at fast speed in plan B requires 17.5 seconds, which is a difference of 9.4 seconds. The other steps require the same execution duration as for table 1, as they do not depend on the target area. The additional torso movement of plan B (listing B.19) therefore requires 8.2 seconds in parallel and 19.0 seconds sequentially.

Listing 6.2: Plan with shortest parallel duration for *attend table* 4 scenario

```
Plan with 48.2 (58.9) cost and 4 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
    (
        [
          -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
          -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
              10.7535128205
        ]
      -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
          PREMANIPULATIONAREAEASTCOUNTER1- 17.5291590909
    )
)
```

For the parallel execution, the benefit from driving at fast speed is larger than the additional time required for moving the torso. The additional time required for sequential execution is still larger than the benefit of moving faster. The resulting shortest plan for table 1 is shown in listing 6.2. Figure 6.1 shows the execution schedule for both plan A and plan B for attending table 4. Although the *!MOVE_TORSO TORSODOWNPOSTURE* task increases the execution duration even when executed in parallel with the *!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE*, the significantly decreased execution duration for moving at a fast speed also decreases the total execution time of plan B.

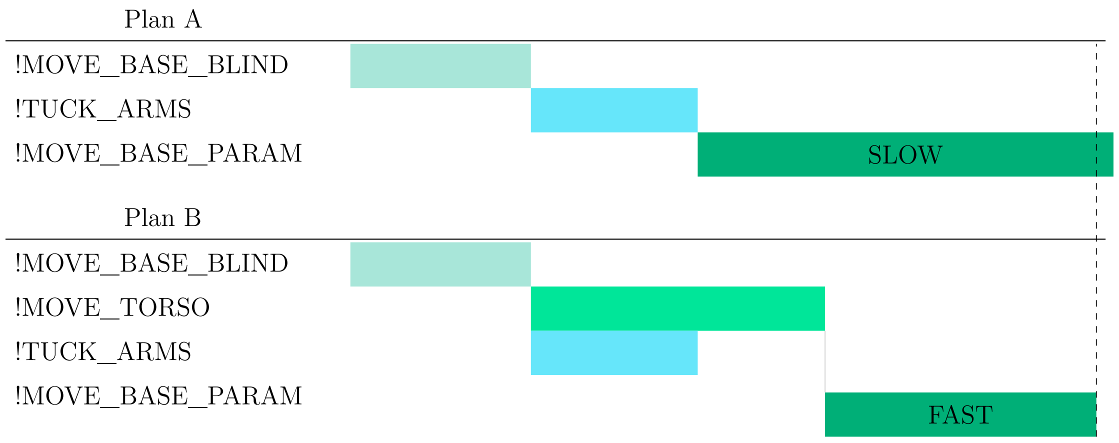


Figure 6.1: Gantt-chart of *attend table* table 4. Plan A corresponds to the execution schedule of the plan shown in listing B.20. Plan B corresponds to listing B.19. In plan B, an additional action for lowering the torso is executed in parallel to arms tucking action. The moving action in plan B may be executed at fast speed, thus reducing the required execution duration and speeding up the total plan execution duration.

6.2 Door Scenario

The *door* scenario gives the planner more options than the *attend table* scenario. Decomposing the *drive_to table4* task results in a total of 27 different plans, which may be categorized by the general execution order. The plans are shown in appendix B.5. The plans for the *attend table* scenario are sorted by the parallel execution duration and the three metrics are applied. The influence of the distance to the door will be evaluated in chapter 7.

Category 1

Category 1 consists of 16 plans. All plans in this category first move back from the manipulation position in front of counter 2 to the premanipulation position, where the arms are tucked and the torso is moved to a lower position. Now the robot is in a torso driving posture as well as an arms driving posture. The next task is to traverse the door in order to get in front of table 4. Traversing the door and reaching the target requires three movement actions:

- driving to the door,
- driving through the door and
- driving to the target.

As the driving posture is assumed, the robot may drive at fast or slow speed for these driving tasks. And also, as the criteria for driving through the narrow door are met, the planner may decide to traverse the narrow door or the wide door. This is the only category with parallel execution potential.

Listing 6.3: Plan with shortest parallel duration for *door* scenario

```
Plan with 63.8 (74.5) cost and 6 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
       MANIPULATIONAREAEASTCOUNTER2- 10.9211875
    (
        [
          -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
          -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
               10.7535128205
        ]
      -!MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY FAST
          PREMANIPULATIONAREAEASTCOUNTER2- 18.3392
      -!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT FAST
          NEARAREADOORNARROWHALLWAY- 9.4315
      -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
          NEARAREADOORNARROWRESTAURANT- 6.0564
    )
)
```

Category 1.1 holds the eight plans of Category 1 for traversing the narrow door. The plans differ by the decomposition of the three driving tasks. As there are three driving tasks and the planner may decompose the tasks to driving fast or slow, there are 2^3 resulting plans. Driving fast for all driving tasks is the fastest. Regarding total execution duration, this is also the fastest category, thus the overall fastest plan for the *door* scenario, shown in listing 6.3, is also a plan of this category. This plan is also the fastest plan when executed in sequential order. The eight plans in this category are plans A to H (listings B.25-B.32).

Category 1.2 holds the remaining eight plans of Category 1 for traversing the wide door, similar to category 1.1. Due to the distance between the narrow and the wide door, for this scenario, it is the third fastest category. The eight plans in this category are plans J, L, O to R, T and V (listings B.34, B.36, B.39-B.42, B.44 and B.46).

Category 2

This category consists of ten plans. All plans move back from the manipulation position to the premanipulation position and tuck the arms, which is required for any moving task. The plans in this category again differ by the door which they are traversing and also by the point of time, when the torso is being lowered.

Category 2.1 holds four plans for traversing the narrow door. As the narrow door requires the torso to be in a lower posture, all four plans lower the torso after arrival in the near door area. Thus the planner is left to decompose the remaining two driving tasks to fast or slow driving, leaving 2^2 combinations. Again, driving fast for both remaining tasks is the fastest decomposition and is even faster than the fastest parallel plan in category 1.2. As tucking the arms and moving the torso to a lower posture is separated by a moving task, the parallel execution potential is removed. Still, this category is the second fastest category for parallel execution. The plans in this category are plans I, K, M and N (listings B.33, B.35, B.37 and B.38).

Category 2.2 holds six plans for traversing the wide door. The wide door does not require the torso to be in a lower posture, thus the planner may decompose the torso movement before or after traversing the wide door.

Category 2.2.1 includes four plans which decompose the torso movement before traversing the door, similar to category 2.1. This also leaves 2^2 combinations for the remaining two driving tasks. Plans U, X to Z (listings B.45, B.48-B.50) represent this category, which is the second slowest category.

Category 2.2.2 contains only two plans, as the torso movement is decomposed after traversing the door, before moving to table 4. This decomposition is only possible for the wide door, as the narrow door requires the torso to be lowered before traversing. As the robot is not in a driving posture, the first two driving tasks must be executed slowly. The only driving task left may be executed fast or slowly leaving only 2^1 options. This is also the slowest category and consists of plans V and AA (listings B.46 and B.51).

Category 3

There is only one plan in this category, which is plan S (listing B.43). This plan does not lower the torso at all. Therefore, all driving tasks must be executed slowly and the only door which may be traversed without lowering the torso is the wide door. For the sequential execution order, this is still 8.3 seconds faster than plan V (listing B.46) and 19.0 seconds faster than the sequential execution order. All plans in categories 1 and 2 require a total of six plan steps (moving back from the counter, tucking the arms, lowering the torso and three driving tasks) in different permutations. Plan S requires only five steps, why it is chosen when evaluating by the minimum amount of plan steps, as seen in listing 6.4.

Listing 6.4: Plan with minimum amount of steps for *door* scenario

```
Plan with 102.2 (102.2) cost and 5 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2- 10.9211875
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  -!MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
      PREMANIPULATIONAREAEASTCOUNTER2- 44.06175
  -!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT SLOW
      NEARAREADOORWIDEHALLWAY- 10.1771052632
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
      NEARAREADOORWIDERESTAURANT- 26.3236842105
)
```

In this chapter the results from applying the PlEv method to the scenarios described in chapter 3 are presented. The results have been preprocessed and categorized in order to evaluate them in chapter 7.

7 Evaluation

In the previous chapter the results of applying the existing plan decision and the new plan evaluation based on temporal execution duration to the *attend table* and *door* scenarios described in chapter 3 were presented. In that chapter the tray manipulation operators and methods were described, which will be used to evaluate a third, theoretical scenario named *pepper mill*, introducing object manipulation and carrying. This scenario will complement the impairment of the *attend table* and *door* scenarios. The evaluation will be discussed in chapter 8.

The basic assumption for the evaluation is the time linearity of moving tasks. A movement task will require a time duration proportional to the distance, which has to be traveled, besides a small offset for finding a path. This scales fairly well and does not affect the linearity. The variation of the degree of clutter in the room or moving obstacles which affect the required duration for driving is not considered for this evaluation. Prediction or imagination may cover this problem.

An overall problem is the comparability of the scenarios. In order to receive multiple different plans for evaluating the scenarios, the constraints of the planner had to be reduced. The original planning domain for the RACE project was constrained, so the variations of the plan found by the planner are as few as possible. The decomposition of most of the high-level tasks is straight forward in order to find the shortest plan for the current scenario. For instance, this decomposition to the shortest plan forced the planner to move the torso to a driving posture every time a driving task has to be performed. The adaption to the planning domain will prove that this is not always the optimal plan regarding the execution time, both for sequential and for parallel execution order. The domain was adapted to allow a maximum freedom during the decomposition with only security and feasibility relevant constraints left.

The evaluation will be conducted using the three metrics presented in section 5.4.

M1: shortest plan by amount of steps (HTN planner decision)

M2: shortest plan by execution duration for sequential execution

M3: shortest plan by execution duration for parallel execution

7.1 Attend Table Scenario

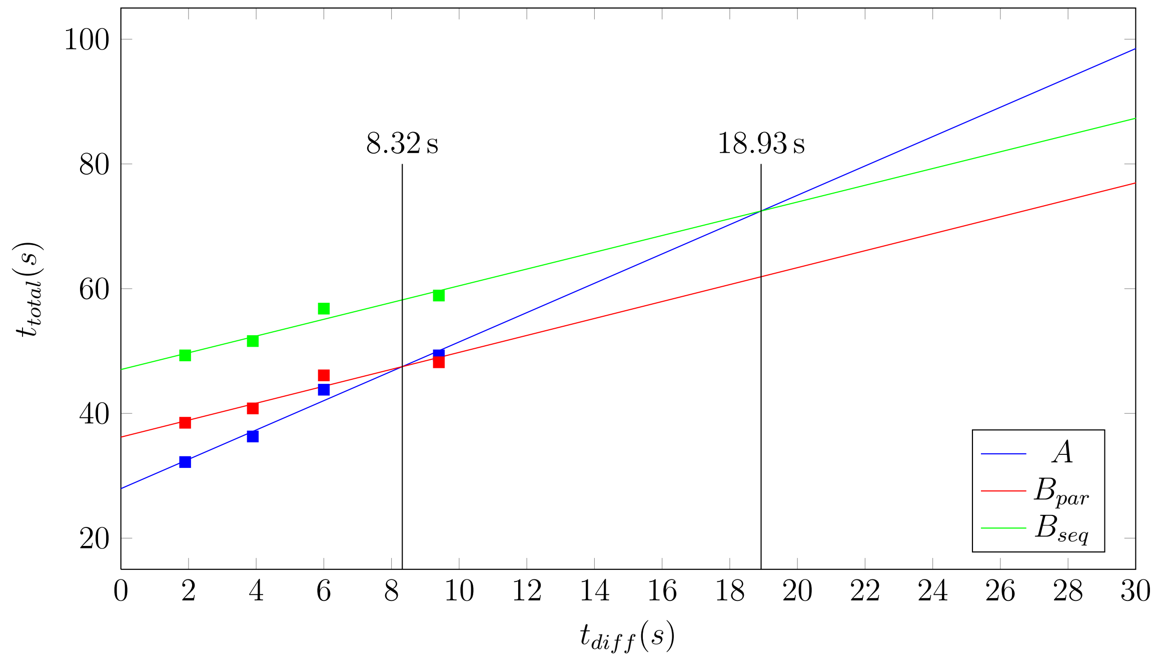


Figure 7.1: Extrapolation of *attend table* scenario results. The x-axis shows the time duration difference for moving from the premanipulation area of counter 1 to the correspondent table at slow or fast speed. The y-axis shows the total execution duration. The marks from left to right represent the time durations for table 1 to 4. *A* corresponds to the plan chosen by shortest plan length regarding steps, corresponding to plan A in listings B.1-B.20. B_{par} and B_{seq} represent the parallel, respectively sequential execution duration of plan B in listings B.2-B.19. Extrapolation was conducted using linear regression.

In section 6.1 the results of the *attend table* scenario were presented. All plans discovered by the SHOP2 planner are listed in appendices B.1-B.4. For all three metrics, the evaluator returned the same plan for tables 1, 2 and 3. This plan is shown in listing 6.1 and is referred to as plan *A* within this section. The optimal plan returned by the evaluator with metric *M1* and metric *M2* for table 4 remains plan *A* but for metric *M3* the optimal plan for table 4 is the plan depicted in listing 6.2, which will be referred to as plan *B* in this section. The results already show that for the limited space in the restaurant environment, the decision made by the HTN planner (metric *M1*) is not the optimal decision. The optimal plan returned by the original RACE planning domain would always be plan *B*, which is not optimal regarding tables 1, 2 and 3. Chapter 5 defined, that the plan returned by metric *M2* for sequential execution, respectively metric *M3* for parallel execution is always the optimal plan regarding execution duration time.

Figure 7.1 shows the required total duration for executing plan A, plan B in parallel execution order (B_{par}) and plan B in sequential execution order (B_{seq}). The x-axis indicates the time difference which results from driving to the desired table at slow or fast speed. The y-axis indicates the total duration required for executing the plan. The marks represent the data points from decomposing the plans for tables 1-4, where table 1 is the leftmost data point and data points are in ascending order, as the distance between counter 1 and the tables increase with greater table numbers. Extrapolation using linear regression results in the plotted lines. Linear regression could be applied due to the assumption earlier in this chapter, where clutter may be ignored and thus the required time duration for driving only depends on the linear distance between counter and tables.

Metric *M1* will always result in plan A represented by the blue line. Metric *M2* results in the minimum of B_{seq} represented by the green line and A. For moving duration differences less than 18.93 seconds this is A and B_{seq} for greater differences. Metric *M3* results in the minimum of B_{par}, represented by the red line and A. This is A for moving time differences less than 8.32 seconds and B_{par} for greater differences. The figure shows that starting at 8.32 seconds, for larger driving time duration differences, the deviation between the optimal parallel plan found by the evaluator and the optimal plan found by the HTN planner increases proportional. The advantage of the PlEv over the original RACE planner is the ability to decide to use plan A for tables which are close enough to the counter. For different driving speeds, the effect will increase or decrease.

7.2 Door Scenario

Section 6.2 presented the results for the *door* scenario. All plans discovered by the SHOP2 planner are listed in appendix B.5. The optimal plan for this scenario with metric *M1* is shown in listing 6.3, which is the same for metric *M2* and is referred to as plan A in this section. The HTN planner chooses the plan with one step less, presented in 6.4. This is referred to as plan S in this scenario. For the setup as depicted in figure 3.3, the distance between the wide and the narrow door is too large to be compensated by not lowering the torso. Additionally, plan S cannot

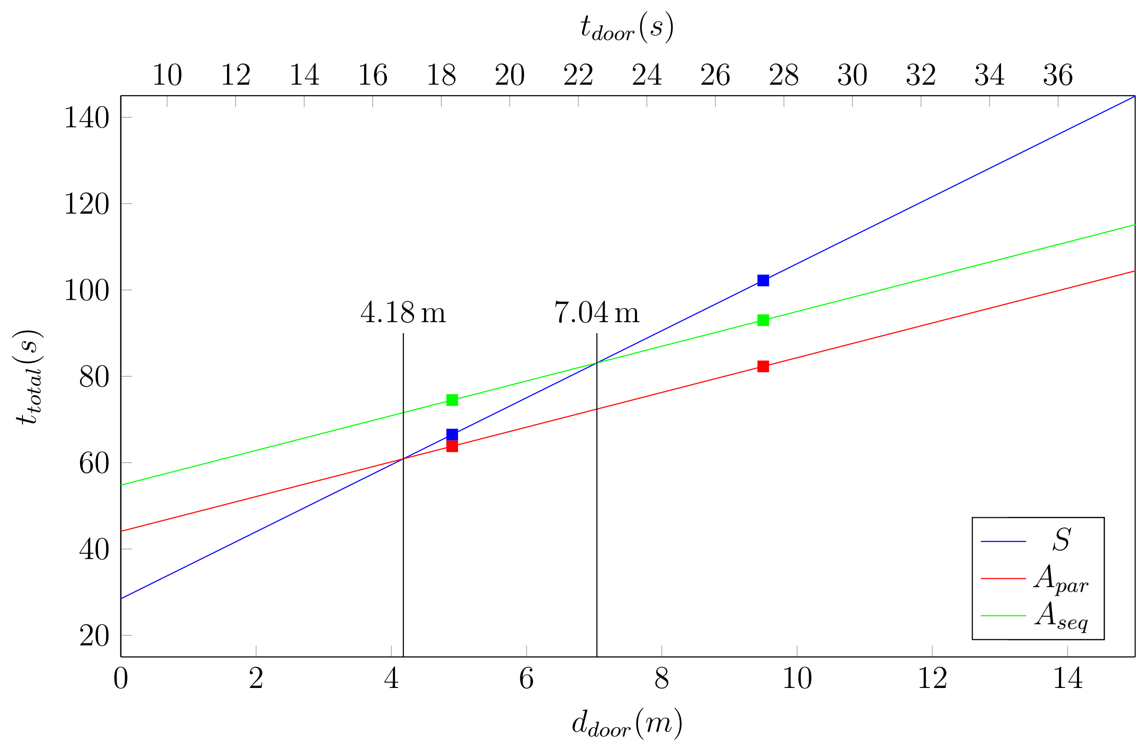


Figure 7.2: Extrapolation of the *door* scenario results. The lower x-axis shows the distance d as depicted in figure 3.3 between counter 2 and the door. The upper x-axis shows the time duration required. The y-axis shows the total execution duration. The left marks represent the result for the narrow door, the right marks for the wide door. S corresponds to the five-stepped plan shown in listing B.43, which is the shortest plan regarding steps. A_{par} and A_{seq} represent the parallel, respectively sequential execution duration of plan A in listing B.25. Extrapolation was conducted by constructing linear equation from two points.

pass through the closer door, as the requirements are not met.

Figure 7.2 shows the required total duration for executing plan S, plan A in parallel execution order (A_{par}) or plan A in sequential execution order (B_{seq}). The lower x-axis plots the distance between counter 2 and the door, which will be traversed. In the scenario, the narrow door is 4.6 meter (9.5 meter for the wide door) aways from the counter, measured in a straight line, as shown in figure 3.3. The upper x-axis shows the corresponding equivalent in moving duration. The marks represent the resulting plans for each door. In order to find the required duration for plan S traversing the narrow door d_1, this door attribute has been temporarily removed. Constructing a linear equation from both points was sufficient for finding the slopes.

For this scenario, the switching points between the optimal plan is slightly different from the *attend table* scenario. Regarding the blueprint in figure 3.3, the doors may be shifted left or right. The constraints for shifting the doors are:

- door d_1 must always be closer to the counter, than door d_2,
- door d_1 must always have the property narrow,
- door d_2 must always have the property wide and
- if the doors exceed the boundaries of hallway or restaurant, the relevant room must be extended as well.

Following these constraints, one may shift door d_2 left of the 7.04 meter mark and door d_1 must stay within a triangle t constructed by:

a the vertical line at $x = distance(d_1)$,
b the horizontal line intersecting the intersection of S and a and
c A_{seq}, respectively A_{par}.

Figure 7.3 illustrates the construction for A_{seq} and d_2 at a distance of 4.5 meter. For this case, d_1 must be at least 2.14 meter away from counter 2. Similar, this may be applied to A_{par}. If the above is true, plan S is chosen by the PlEv, as the duration for driving slowly through door d_2 without moving the torso in a lower position is faster than lowering the torso in order to drive through door d_1.

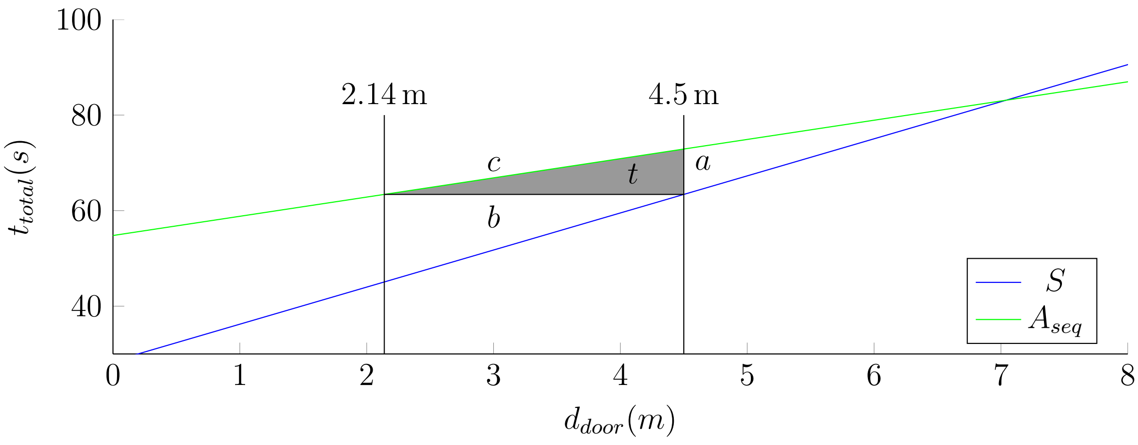


Figure 7.3: Illustration of shifting the door distance. Door combinations within the marked triangle will result in a shorter total execution duration of plan S, if they match the constraints described in section 7.2.

Figure 7.2 also shows, that for distances larger than 7.04 meter (sequential execution) and 4.18 meter (parallel execution) between counter 2 and the doors, plan S resulting from metric *M1* will always be slower than metric *M2* and metric *M3* if the constraints are met. For this scenario, it is not possible to compare the results to the original RACE planning domain, as it does not include traversing a door, but similiar results as for scenario *attend table* are expected.

7.3 Pepper Mill Scenario

Both scenarios, *attend table* and *door*, leave the robot in a different state after finishing the execution of the plan resulting from metric *M1* and metric *M2*, respectively metric *M3*. When the robot arrives at the table, in one case the torso is in a lower position, in the other case the torso is in an upper position. Embedding this scenario as part of a bigger scenario may falsify or confirm the results presented in section 7.1 and 7.2, depending on the required torso posture for further operation. If the robot needs to manipulate at the table, the torso requires to be in an upper position. The additional time required for raising the torso will diminish the advantages, if the respective table is table 4. For tables 1-3, the same applies if the robot only needs to attend the table and recognize an input from the guest. In this case, the torso must not be in an upper position. Vice versa, the benefit of the partial plan may increase.

The scenario is constructed to show that a more complex plan will provide the same results. For this scenario, both plans will leave the robot in the same state at the table. The initial state is the same as for scenario *attend table* except that a pepper mill is located on the right tray area of the robot. The target is arriving at the desired table with both hands free for manipulation. The decomposition of this problem allows the planner for finding the same plan as for the *attend table* scenario, moving back to the premanipulation area, tucking the arms and driving slowly to the table, shown in listing B.55, B.57, B.59 and B.61. This plan will be referred to as plan *A* for this section. In order to achieve a driving posture, the torso needs to be in a lower posture and the tray needs to be clear of objects which may topple, such as the pepper mill. After moving back from the counter, the robot picks the pepper mill from the tray, moves the arms into a carrying posture and lowers the torso. It may then move fast to the desired table, where the arm with the pepper mill is moved to the side to prepare for manipulation and the torso is moved to an upper posture. Next, the pepper mill is placed on the tray and, in order to achieve the same state as resulting from plan *A*, the arms are being tucked. This plan will be referred to as plan *B* and is shown in listing B.56, B.58, B.60 and B.62. The redundant plans for moving slowly and picking with left arm are omitted. Moving slowly will not decrease the execution time in any case and due to the right arm being tucked above the left arm, moving the right arm to the side is faster than

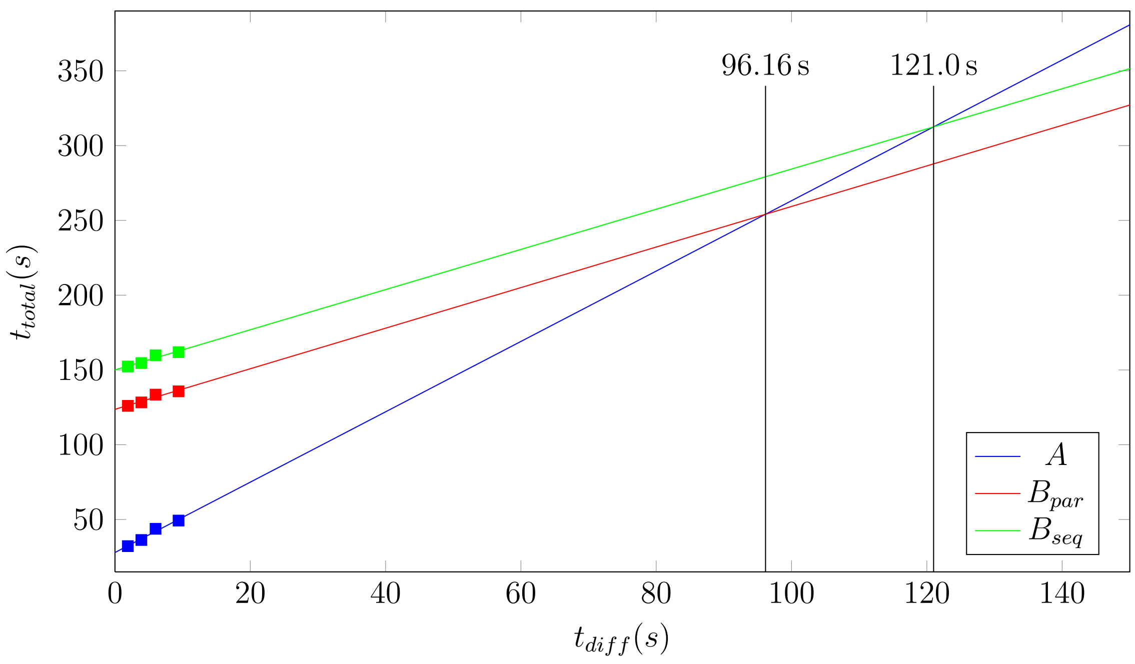


Figure 7.4: Extrapolation of *pepper mill* scenario results. The x-axis shows the time duration difference for moving from the premanipulation area of counter 1 to the correspondent table at slow or fast speed. The y-axis shows the total execution duration. The marks from left to right represent the time durations for table 1-4. *A* corresponds to the plan chosen by shortest plan length regarding steps, corresponding to plan A in listing B.55-B.61. B_{par} and B_{seq} represent the parallel, respectively sequential execution duration of plan B in listing B.56-B.62. Extrapolation was conducted using linear regression.

moving the left arm to the side.

For this scenario, metric *M1* will prefer plan *A*, as it requires only three steps instead of 9 for plan *B*. Also, for all tables within the restaurant environment, metric *M2* and metric *M3* will choose plan *A*, as the additional fixed time duration for picking from the tray, moving the arms to the carry posture, lowering the torso, moving the arm to the side, raising the torso, placing on the tray and tucking the arms (requiring 118.1 seconds in parallel, 144.4 seconds sequentially) exceeds the benefit from driving fast to the table (2.0 seconds for table 1, 9.4 seconds for table 4) by far. Nevertheless, figure 7.4 shows, eventually the PlEv will find a plan which requires less execution duration than the plan found by applying metric *M1* or the RACE planner.

Similar to scenario *attend table* and figure 7.1, figure 7.4 shows the required total execution duration with respect to the time difference for driving to the target at slow or fast speed. Extrapolation using linear regression shows the breakpoint for parallel execution at 96.16 seconds and 121.0 seconds for sequential execution. As this scenario was constructed primarily to show that the PlEv will find an improved plan for almost any scenario at some point in time or at least a plan which is equal to the plan found by HTN planners such as the one used in project RACE, the switch point is beyond a regular scenario in a restaurant environment. Nevertheless, this scenario still is only a partial scenario regarding restaurant applications or other areas of application. Deploying this partial plan to a scenario with forced manipulation after reaching the target table, will reduce fixed duration offset between plan A and B. This is due to the final arm tucking not being required for plan B and for plan A. The arms will have to be untucked in order to reach the required position for manipulating. This change will reduce the offset by 17.0 seconds. Requiring the robot to serve the pepper mill to the target table will have even more effect, shifting the breakpoint into regions close to the distance between counter 1 and table 4.

Listing 7.1: Plan A_s for *serving the peppermill* scenario to table 4

```
Plan with 84.0 (84.0) cost and 5 steps:
(
   -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
        MANIPULATIONAREAEASTCOUNTER1- 11.6365
   -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
   -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
        PREMANIPULATIONAREAEASTCOUNTER1- 26.9018636364
   -!MOVE_ARM_TO_SIDE RIGHTARM1- 12.7538043478
   -!PICK_FROM_TRAY PEPPERMILL RIGHTARM1- 22
)
```

Listing 7.1 shows the described plan for *serving the peppermill* scenario to table 4, which is related to plan A and is referenced to as plan A_s. This plan requires 34.8 seconds more than plan A without preparing to serve the pepper mill, increasing the fixed duration from 22.4 seconds to 47.1 seconds. Listing 7.2 shows the adapted plan B_s, related to plan B without preparing to serve the pepper mill. Plan B_s requires 43.8 seconds less, reducing the fixed duration from 118.1 seconds to 74.4 seconds for the parallel execution. The fixed portions of the plans differ by 17.2 seconds and the difference in driving at fast or slow speed to table 4 is 9.4 seconds. Therefore, the PlEv will not find a faster plan for serving the pepper mill to table 4, but for a table at twice the distance. Regarding the limited restaurant environment used for the experiments, this is conceivable.

The evaluation demonstrated how the PlEv optimizes the plan regarding execution duration for the proposed scenarios in chapter 3. Additionally, a theoretical scenario and its extension proved that the optimization does not follow from omitting plan steps which led to a deviating final state, but the optimization is also possible for more complex scenarios with an equivalent final state.

Listing 7.2: Plan B_s for *serving the peppermill* scenario to table 4

```
Plan with 91.9 (118.2) cost and 7 steps:
(
  –!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
       MANIPULATIONAREAEASTCOUNTER1– 11.6365
  –!PICK_FROM_TRAY PEPPERMILL RIGHTARM1– 22
    (
        [
          –!MOVE_ARMS_TO_CARRYPOSTURE– 13.5361914894
          –!MOVE_TORSO TORSODOWNPOSTURE– 19.0144878049
        ]
      –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
           PREMANIPULATIONAREAEASTCOUNTER1– 17.5291590909
        (
            [
              –!MOVE_ARM_TO_SIDE RIGHTARM1– 12.7538043478
              –!MOVE_TORSO TORSOUPPOSTURE– 21.7206086957
            ]
        )
    )
)
```

In this chapter the results presented in chapter 6 were evaluated. The evaluation showed the feasibility and functionality of the PlEv by analyzing the results from the scenarios described in chapter 3 and verifying the results with an additional scenario. The results are discussed in the next chapter.

8 Discussion

The foundations for the PlEv are presented in chapter 2 of this thesis. The scenarios for the experiments and evaluation are described in chapter 3. The required component for predicting task execution duration from previous execution or simulation is introduced in chapter 4. The PlEv itself is presented in chapter 5. The results and their evaluation prove the successful operation in chapter 6 and 7.

One of the central ideas for the PlEv, similar to the parallelization layer, is the usability [Einig et al., 2013]. The adapted architecture (figure 5.2) and the execution middle-layer (section 2.4) ensure an easy way to run the PlEv on various robots. This provides the possibility to publish and migrate the proposed PlEv as a configurable ROS Package. In order to be able to publish the package, improvements are required, such as (1) adding an interface for plan generation initiation, (2) adding an interface for monitoring the plan execution to allow access to the execution information for reasoning and learning, (3) improving the TXX, (4) removing RACE project constraints and adding a configuration interface for the robots capabilities as well as some of the improvements discussed in Einig [2012].

As this work is bound to the RACE architecture, experiments, results and evaluation are limited to the environment depicted in figure 3.1. The scenarios for evaluation were chosen in order to be executable within this environment and demonstrate the capabilities of the robot. The *attend table* and *door* scenarios demonstrate the possibility to find an optimal plan which differs from the shortest plan chosen by the RACE planner and SHOP2 planner in general even within the limited TAMS restaurant environment. The resulting state after a successful plan execution differs for both scenarios when a different plan is chosen. Although this does not necessarily limit the benefit from the shorter plan, as a more complex scenario may take advantage from the different finishing state, another scenario is presented in section 7.3, which ensures the same finishing state for all plans. This scenario exceeds the limits of the TAMS environment in order to find an optimal plan which differs from the original decision. The PlEv is not yet capable of dealing with execution time duration deviation. Figure 4.3 shows the deviation of up to 20 % from the expected mean. A plan which requires less total execution duration but has a large expected deviation may require much longer than a plan with a very small deviation. While this does not pose a problem for the results of this work, where the overall statistical benefit is given even for large deviations, it may be a problem when introducing further constraints such as handling a warm dish. A constraint may be to serve this warm dish within a given time to a target table. A 45 seconds plan with 5 seconds expected deviation is more feasible, than a plan requiring 40 seconds in average having a 25 seconds deviation. In general, planning with uncertainty is not part of this work, but should not be ignored regarding optimal plan execution. This includes open/closed doors, physical location of objects and moving obstacles.

The evaluation in chapter 7 also shows, that although the PlEv will eventually find an optimal plan for a scenario, which is different from the original decision met by the HTN planner, this switch point may not reside within feasible limits. For instance the *pepper mill* scenario would require a restaurant environment, which is ten times the size of the TAMS restaurant. This is the case for a large-scale restaurant but the application in sight does not deploy the robot in such a large uncertain environment, but rather small-scaled environments. Different application areas though, may provide the necessary environment, which is required to exploit the improvements proposed in this thesis. One of the most prominent fields for robotic research is elderly care. Even small nursing homes for elderly care are quite spacious with wide corridors and doors. Deploying the PlEv in such a scenario may provide a good evaluation scenario for demonstration of the improved plan decision.

Besides the possibility to collect temporal data using the TXX (chapter 4), it may be possible to deduce new temporal data from previously experienced temporal data, instead of requiring the task to be run physically or virtually previously. Including the spatial knowledge about positions into the cost interface (section 4.2) may allow to deduce the required execution duration of driving from counter 1 to table 2 (figure 3.1) from the durations for driving from counter 1 to table 1 respectively table 3 as well as the relative positions of all three tables to the counter 1. This would reduce the required simulation runs and allow an estimation of execution durations for new targets or actions.

The results in chapter 6 prove the overall functionality and usability of the PlEv. The limitations resulting from environment, architecture and time prevent this thesis from providing results on applicability for real world deployment. The analysis of the *pepper mill* and *serve peppermill* scenarios give an impression of possible plan optimization of slightly more complex scenarios. Yet, results and evaluation for real world complex scenarios are required to show applicability and grade of optimization of plan execution.

As the HTN planner decomposes all possible plans even when returning a single plan, there is no drawback in returning all possible plans. The required time for parallelization and evaluation of the set of plans using the PlEv is negligible compared to the execution duration systems and thus the PlEv may be even used in an environment which provides little to no opportunities finding an improved plan. Environments which do not provide these opportunities may result from various reasons. It may be an environment which is very small and thus offers few possibilities for finding deviating plans. Imagining the TAMS environment in figure 3.1 without the attached hallway. The restaurant area reduced to the part left of table 2 would also provide such an environment. Another reason may be an over-constrained environment. The original RACE planning domain would be over-constrained regarding the scenarios described in chapter 3, as it does not offer decompositions without lowering the torso into a driving posture. An over-constrained scenario within the described TAMS environment would be the assignment given to the robot to transport three to five toppling objects at the same time (due to temporal constraints) from a counter to a table. This will force the robot to place the objects on the tray and thus remove the option to drive fast. Although, removing the temporal constraint for serving all dishes at once, the planner may decide to serve two objects at once and repeat serving the other objects. This scenario will depend on the amount of objects which have to be carried. Given three and four objects, the robot has to repeat the serving twice. Five objects require the robot to repeat the serving without the tray three times. In a serving scenario with six objects, the robot will have to drive twice when using the tray and even three times without the tray.

9 Conclusion and Outlook

This thesis presents the PlEv, a method for finding an optimal plan, executed in parallel order, from a set of sequential HTN generated plans for reaching a goal state from an initial state. In chapter 2 some approaches to execute plans in parallel are presented, cost-based planning, the RACE setting and the experimental platform are introduced. The scenarios for the experiments were presented in chapter 3 and the TXX with the SMACH foundation. The evaluation of the experienced temporal information was presented in chapter 4. The PlEv was proposed in chapter 5. The foundation for the PlEv, the RACE architecture [Hertzberg et al., 2014] and the Parallel Plan Executor [Einig et al., 2013], the RACE planning domain and the metrics for evaluating the results were also introduced. Chapter 6 prepared the results of the two scenarios for evaluation. These results were evaluated in chapter 7 and an additional scenario was introduced to verify the results. The challenges and possibilities were discussed in chapter 8.

This thesis showed the feasibility and the improvement to plan decision making of the PlEv within the RACE restaurant environment and furthermore for all application fields as it finds a plan at least equal to the decision of the HTN planner, regarding execution duration, if no faster plan exists. If, for any planning problem, multiple plans with different task sets exist, the PlEv will, at some point, find a plan which is faster than the plan which is chosen by the HTN planner. Although this point may exceed the physical limits of the environment the robot is deployed in, there is no disadvantage in using the PlEv for deciding on the optimal plan as the required additional time for evaluating the set of plans is negligible.

The PlEv may pose a valuable piece of software for developing robotic control within ROS. As already addressed in chapter 8, various improvements are possible or required ahead of publishing the joint Evaluator and Executor. The task duration deviation may be included to improve reliability of the total plan execution duration and consider the total plan deviation for plan evaluation. An interface for including other rating scores, for instance uncertainty score, execution robustness score or resource requirements score may be implemented. The access to the planning functionality of the PlEv must be granted in order for users or other nodes to be able to call the planning functionality and start the evaluation process. Also, access to the plan, which has been chosen by the evaluator and the information of the execution of tasks should be published using ROS topics.

Deduction by precomputing new temporal data from experienced task execution durations will improve the speed of acclimatization to new environments by providing task execution durations for actions, which have not been executed before and especially paths, which have not been traveled before. For this purpose, additional information such as passage properties will be useful [Belker et al., 2004].

Deploying the PlEv in various complex environments will give the opportunity to collect data on the total saving of execution duration provided by executing the evaluated plans in parallel. Especially operation in environments and scenarios beside the TAMS restaurant environment, for instance in nursing homes for elderly care, may provide valuable results for evaluating the overall improvements over static HTN planner decisions.

Bibliography

[Asfour et al. 2006] Asfour, T. ; Regenstein, K. ; Azad, P. ; Schroder, J. ; Bierbaum, A. ; Vahrenkamp, N. ; Dillmann, R. : ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control. In: *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, 2006, pp. 169 – 175 2.1

[Asfour et al. 2004] Asfour, T. ; Ly, D. N. ; Regenstein, K. ; Dillmann, R. : Coordinated Task Execution for Humanoid Robots. In: Jr., M. H. A. (Editor) ; Khatib, O. (Editor): *ISER* Bd. 21, Springer, 2004 (Springer Tracts in Advanced Robotics). – ISBN 978–3–540–28816–9, pp. 259 – 267 2.1

[Beetz 2000] Beetz, M. : *Lecture Notes in Computer Science*. Bd. 1772: *Concurrent Reactive Plans, Anticipation and Forestalling Execution Failures*. Springer, 2000. `http://dx.doi.org/10.1007/3-540-46436-0`. `http://dx.doi.org/10.1007/3-540-46436-0`. – ISBN 3–540–67241–9 2.3

[Belker et al. 2003] Belker, T. ; Hammel, M. ; Hertzberg, J. : Learning to Optimize Mobile Robot Navigation Based on HTN Plans. In: *Proceedings of the International Conference on Robotics and Automation*, 2003 2.3

[Belker et al. 2004] Belker, T. ; Hammel, M. ; Hertzberg, J. : Plan projection under the APPEAL robot control architecture. In: *Intelligent autonomous systems 8*, 2004, pp. 809–817 2.3, 9

[Bhowmick et al. 2012] Bhowmick, P. ; Dey, S. ; Samantaray, A. ; Mukherjee, D. ; Misra, P. : A temporal constraint based planning approach for city tour and travel plan generation. In: *Intelligent Human Computer Interaction (IHCI), 2012 4th International Conference on*, 2012, pp. 1–6 2.3

[Bohren and Cousins 2010] Bohren, J. ; Cousins, S. : The SMACH High-Level

Executive [ROS News]. In: *Robotics Automation Magazine, IEEE* 17 (2010), dec., no. 4, pp. 18 – 20 2.2

[Castillo et al. 2006] Castillo, L. ; Fdez-Olivares, J. ; García-Pérez, O. ; Palao, F. : Temporal enhancements of an HTN planner. In: *Proceedings of the 11th Spanish association conference on Current Topics in Artificial Intelligence.* Berlin, Heidelberg : Springer-Verlag, 2006 (CAEPIA'05), pp. 429 – 438 2.1

[Devaurs et al. 2011] Devaurs, D. ; Simeon, T. ; Cortes, J. : Parallelizing RRT on distributed-memory architectures. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011. – ISSN 1050–4729, pp. 2261–2266 2.1

[Einig 2012] Einig, L. : *Parallel plan execution on a mobile robot with an HTN planning system: a resource based approach.* Hamburg, Germany, University of Hamburg, Departement Informatics, Bachelorthesis, 2012 1.2, 1.2, 2.1, 2.2, 2.1, 2.4, 2.4, 4.1, 5.1, 5.1, 17, 8

[Einig et al. 2013] Einig, L. ; Klimentjew, D. ; Rockel, S. ; Zhang, L. ; Zhang, J. : Parallel plan execution and re-planning on a mobile robot using state machines with HTN planning systems. In: *ROBIO'13*, 2013, pp. 151–157 1.1, 2.2, 3, 3.1, 4.1, 5.1, 8, 9

[Gat 1998] Gat, E. : On Three-Layer Architectures. In: *Artificial Intelligence and Mobile Robots*, MIT Press, 1998 2.1

[Hertzberg et al. 2014] Hertzberg, J. ; Zhang, J. ; Zhang, L. ; Rockel, S. ; Neumann, B. ; Lehmann, J. ; Dubba, K. S. R. ; Cohn, A. G. ; Saffiotti, A. ; Pecora, F. ; Mansouri, M. ; Konečný, Š. ; Günther, M. ; Stock, S. ; Lopes, L. S. ; Oliveira, M. ; Lim, G. H. ; Kasaei, H. ; Mokhtari, V. ; Hotz, L. ; Bohlken, W. : The RACE Project. In: *KI - Künstliche Intelligenz* 28 (2014), no. 4, 297-304. `http://dx.doi.org/10.1007/s13218-014-0327-y`. – DOI 10.1007/s13218–014–0327–y. – ISSN 0933–1875 1.2, 2.3, 2.1, 2.4, 2.4, 3, 5.1, 9

[Jacobs et al. 2012] Jacobs, S. A. ; Manavi, K. ; Burgos, J. ; Denny, J. ; Thomas, S. ; Amato, N. M.: A scalable method for parallelizing sampling-based motion planning algorithms. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012. – ISSN 1050–4729, pp. 2529 – 2536 2.1

[Lingard and Richards 1998] Lingard, A. R. ; Richards, E. B.: Planning parallel actions. In: *Artif. Intell.* 99 (1998), Mar., no. 2, 261 – 324. http://dx.doi.org/10.1016/S0004-3702(97)00080-5. – DOI 10.1016/S0004–3702(97)00080–5. – ISSN 0004–3702 2.1

[Luh and Lin 1985] Luh, J. ; Lin, C. : Scheduling parallel operations in automation for minimum execution time based on pert. In: *Computers & Industrial Engineering* 9 (1985), no. 2, 149 – 164. http://dx.doi.org/10.1016/0360-8352(85)90014-2. – DOI 10.1016/0360–8352(85)90014–2. – ISSN 0360–8352 2.1

[Mansouri and Pecora 2013] Mansouri, M. ; Pecora, F. : A representation for spatial reasoning in robotic planning. In: *:*, 2013 3

[McDermott 1992] McDermott, D. : Transformational planning of reactive behavior / DTIC Document. 1992. – Forschungsbericht 2.3

[Miura and Shirai 1998] Miura, J. ; Shirai, Y. : Scheduling parallel execution of planning and action for a mobile robot considering planning cost and vision uncertainty. In: *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on* Bd. 1, 1998, pp. 545–550 vol.1 2.1

[Nau et al. 2003] Nau, D. ; Ilghami, O. ; Kuter, U. ; Murdock, J. W. ; Wu, D. ; Yaman, F. : SHOP2: An HTN planning system. In: *Journal of Artificial Intelligence Research* 20 (2003), pp. 379–404 2.3

[Off 2012] Off, D. M.: *Hierarchical Plan-based Robot Control in Open-Ended Environments*. Von-Melle-Park 3, 20146 Hamburg, Universität Hamburg, Diss., 2012. http://ediss.sub.uni-hamburg.de/volltexte/2013/6189 2.3, 3.2

[Ohki et al. 2013] Ohki, T. ; Nagatani, K. ; Yoshida, K. : Path planning for mobile robot on rough terrain based on sparse transition cost propagation in extended elevation maps. In: *Mechatronics and Automation (ICMA), 2013 IEEE International Conference on*, 2013, pp. 494–499 2.3

[Rockel et al. 2014] Rockel, S. ; Klimentjew, D. ; Zhang, L. ; Zhang, J. : An Hyperreality Imagination based Reasoning and Evaluation System (HIRES). In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 5705–5711 2.4

[Rockel et al. 2013] Rockel, S. ; Neumann, B. ; Zhang, J. ; Dubba, S. K. R. ; Cohn, A. G. ; Konecny, S. ; Mansouri, M. ; Pecora, F. ; Saffiotti, A. ; Günther, M. et al.: An Ontology-based Multi-level Robot Architecture for

Learning from Experiences. In: *AAAI Spring Symposium: Designing Intelligent Robots*, 2013 2.3, 2.4, 5.1

[Rost et al. 2012] Rost, P. ; Hotz, L. ; Riegen, S. von: Supporting Mobile Robot‘s Tasks through Qualitative Spatial Reasoning. In: *9th International Conference on Informatics in Control, Automation and Robotics*. Rome, Italy, 2012 2.3

[Stock et al. 2014] Stock, S. ; Günther, M. ; Hertzberg, J. : Generating and Executing Hierarchical Mobile Manipulation Plans. In: *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of* VDE, 2014, pp. 1–6 2.4

[Suh and Oh 2012] Suh, J. ; Oh, S. : A cost-aware path planning algorithm for mobile robots. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012. – ISSN 2153–0858, pp. 4724–4729 2.3

[Taipalus and Halme 2009] Taipalus, T. ; Halme, A. : An action pool architecture for multi-tasking service robots with interdependent resources. In: *Proceedings of the 8th IEEE international conference on Computational intelligence in robotics and automation*. Piscataway, NJ, USA : IEEE Press, 2009 (CIRA’09). – ISBN 978–1–4244–4808–1, 228 – 233 2.1

[Tang 2013] Tang, P. : A temporal HTN planning paradigm and its’ application in flood controlling. In: *Conference Anthology, IEEE*, 2013, pp. 1–4 2.3

[Yong and Meiling 2012] Yong, Y. ; Meiling, W. : The Path Planning Algorithm Research Based on Cost Field for Autonomous Vehicles. In: *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference on* Bd. 2, 2012, pp. 38–41 2.3

Eidesstattliche Erklärung

Ich, Lasse Einig, Matrikel-Nr. 5918387, versichere meine Masterarbeit mit dem Thema

> *Hierarchical Plan Generation and Selection for Shortest Plans based on Experienced Execution Duration – Using Parallel Plan Execution*

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Mir ist bekannt, dass ich meine Masterarbeit zusammen mit dieser Erklärung fristgemäß nach Vergabe des Themas in dreifacher Ausfertigung und gebunden im Prüfungsamt der Universität Hamburg, Departement Informatik abzugeben oder spätestens mit dem Poststempel des Tages, an dem die Frist abläuft, zu senden habe.

Ich bin mit der Veröffentlichung in der Bibiliothek des Departement Informatik an der Universität Hamburg einverstanden.

Hamburg, den 4. März 2015

Lasse Einig

List of Figures

Listings

Glossary

aDN areaDoorNarrow.

aDW areaDoorWide.

APPEAL Architecture for Projection-based Planning, Execution and Learning.

ARMAR-III A service robot developed by the KIT which is supposed to learn and execute everyday tasks within a kitchen environment autonomously, `http://his.anthropomatik.kit.edu/241.php`.

Blackboard A Blackboard architecture is a distributed problem solving architecture, which is capable of incorporating multiple problem solving agents. The Blackboard itself a central working memory and is part of the Blackboard architecture.

C1 Counter1.

C2 Counter2.

DOF Degrees of Freedom.

Gazebo The Gazebo simulator is a robotics toolbox for simulating multiple different robots in a world model, `http://gazebosim.org/`.

GUI Graphical User Interface.

HTN Hierarchical Task Network.

JSHOP2 Java Implementation of Simple Hierachical Ordered Planner 2, an HTN planning system written in Java.

KIT Karlsruher Institut für Technologie (Karlsruhe Institute for Technology).

mA manipulationArea.

nADN nearAreaDoorNarrow.

nADW nearAreaDoorWide.

pA placingArea.

PIONEER II Differential-drive mobile robot platform for indoor and outdoor environments.

PlEv Plan Evaluator.

pMA premanipulationArea.

PR2 Personal Robot 2.

RACE Project RACE: Robustness by Autonomous Competence Enhancement, a EU-funded research project for mobile service robots.

ROS Robotic Operating System.

ROS node In ROS, a node is a computation component similar to a process within an operating system using communication protocols like topics. Due to this component architecture, nodes can be replaced or reused easily, `http://wiki.ros.org/Nodes`.

ROS Package In ROS, a package is a reusable piece of software which can be made publicly available to other ROS users. A ROS package may contain ROS nodes, libraries datasets or other useful software, `http://wiki.ros.org/Services`.

ROS service In ROS, a service is a RPC-like communication. One ROS node offers a service, which may be requested by another ROS node. The providing ROS node will reply with the result, `http://wiki.ros.org/Services`.

ROS topic In ROS, topics are used for inter-node communication. Each topic is named and nodes can publish or subscribe to a topic, `http://wiki.ros.org/Topics`.

RPC Remote Procedure Call.

SHOP Simple Hierarchical Ordered Planner, a total-order HTN planning system written in Lisp.

SHOP2 Simple Hierarchical Ordered Planner 2, a partial-order HTN planning system written in Lisp.

SMACH State MACHine.

T1 Table1.

T2 Table2.

T3 Table3.

T4 Table4.

TAMS Group Technical Aspects of Multi-modal Systems, Department Informatics, University of Hamburg.

TXX Temporal Experience Extractor.

Willow Garage Robotics research lab, developing the PR2 and other robotic systems, as well as maintaining ROS software, `https://www.willowgarage.com`.

Appendices

A Modified SHOP2 planning domain

A.1 Traversing Doors

Listing A.1: Top-level task for traversing doors

```
(:method (drive_to ?to)
    (not (robot_in_same_room ?to))
    ((drive_robot_through_door ?to))
    ((robot_in_same_room ?to))
    ((drive_robot ?to))
    )
(:-
    (robot_in_same_room ?to)
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?robotArea)
     (HasRoom ?robotArea ?robotRoom)
     (HasRoom ?to ?robotRoom)
     )
    )
```

Listing A.2: Methods for traversing doors

```
(:method (drive_robot_through_door ?to)
     ((Instance RobotAt ?robotat)
      (HasRobot ?robotat ?trixi1)
      (HasArea ?robotat ?robotArea)
      (HasRoom ?robotArea ?fromRoom)
      (HasRoom ?to ?toRoom)
      (Instance Door ?door)
      (HasArea ?door ?preDoorAreaTo)
      (HasArea ?door ?preDoorAreaFrom)
      (Instance PreDoorArea ?preDoorAreaTo)
      (Instance PreDoorArea ?preDoorAreaFrom)
      (HasRoom ?preDoorAreaTo ?toRoom)
      (HasRoom ?preDoorAreaFrom ?fromRoom)
     )
     ((drive_robot ?preDoorAreaFrom)
      (traverse_door ?preDoorAreaFrom ?door ?preDoorAreaTo)
      (drive_robot ?to)
      )
     )
(:method (traverse_door ?from ?door ?to)
     (HasDoorProperty ?door wide)
     (drive_robot ?to)
     (HasDoorProperty ?door narrow)
     ((torso_assume_driving_pose)
      (:immediate arms_assume_driving_pose)
      (drive_robot ?to)
      )
     )
```

A.2 Driving the Robot

Listing A.3: Original methods for driving the robot

```
(:method (drive_robot ?to)
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?robotArea)
     (different ?robotArea ?to)
     (not (Instance ManipulationArea ?robotArea))
     )
    ((torso_assume_driving_pose)
     (:immediate arms_assume_driving_pose)
     (:immediate !move_base ?to)
     )
    )
(:method (drive_robot ?to)
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?robotArea)
     (different ?robotArea ?to)
     (Instance ManipulationArea ?robotArea)
     (HasPreManipulationArea ?robotArea ?preArea)
     )
    ((!move_base_blind ?preArea)
     (:immediate torso_assume_driving_pose)
     (:immediate arms_assume_driving_pose)
     (:immediate !move_base ?to)
     )
    )
```

Listing A.4: Modified methods for driving the robot

```
(:method (drive_robot ?to)
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?robotArea)
     (different ?robotArea ?to)
     (not (Instance ManipulationArea ?robotArea))
     )
    ((arms_assume_driving_pose)
     (:immediate !move_base_param ?to slow ?robotArea))
    )
(:method (drive_robot ?to)
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?robotArea)
     (different ?robotArea ?to)
     (not (Instance ManipulationArea ?robotArea))
     (Instance On ?on)
     (HasArea ?on ?trayArea)
     (HasPhysicalEntity ?on ?object)
     (Instance TrayArea ?trayArea)
     )
    ((torso_assume_driving_pose)
     (:immediate clear_tray)
     (:immediate arms_assume_driving_pose)
     (:immediate !move_base_param ?to fast ?robotArea)
     )
    )
(:method (drive_robot ?to)
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?robotArea)
     (different ?robotArea ?to)
     (not (Instance ManipulationArea ?robotArea))
     (Instance TrayArea ?trayArea)
     (not ((Instance On ?trayOn)
           (HasArea ?trayOn ?trayArea)
           (HasPhysicalEntity ?trayOn ?otherObject))
           )
     )
    ((torso_assume_driving_pose)
     (:immediate arms_assume_driving_pose)
     (:immediate !move_base_param ?to fast ?robotArea)
     )
    )
(:method (drive_robot ?to)
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?robotArea)
     (different ?robotArea ?to)
     (Instance ManipulationArea ?robotArea)
     (HasPreManipulationArea ?robotArea ?preArea)
     )
    ((!move_base_blind ?preArea ?robotArea)
     (:immediate arms_assume_driving_pose)
     (:immediate !move_base_param ?to slow ?preArea)
     )
    )
(:method (drive_robot ?to)
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?robotArea)
     (different ?robotArea ?to)
     (Instance ManipulationArea ?robotArea)
     (HasPreManipulationArea ?robotArea ?preArea)
     (Instance On ?on)
     (HasArea ?on ?trayArea)
     (HasPhysicalEntity ?on ?object)
     (Instance TrayArea ?trayArea)
     )
    ((!move_base_blind ?preArea ?robotArea)
     (:immediate clear_tray)
     (:immediate torso_assume_driving_pose)
     (:immediate arms_assume_driving_pose)
     (:immediate !move_base_param ?to fast ?preArea)
     )
    )
(:method (drive_robot ?to)
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?robotArea)
     (different ?robotArea ?to)
```

```
      (Instance ManipulationArea ?robotArea)
      (HasPreManipulationArea ?robotArea ?preArea)
      (Instance TrayArea ?trayArea)
      (not ((Instance On ?trayOn)
            (HasArea ?trayOn ?trayArea)
            (HasPhysicalEntity ?trayOn ?otherObject))
            )
      )
     ((!move_base_blind ?preArea ?robotArea)
      (:immediate torso_assume_driving_pose)
      (:immediate arms_assume_driving_pose)
      (:immediate !move_base_param ?to fast ?preArea)
      )
     )
```

A.3 Clearing the Tray

Listing A.5: Modified method for clearing the tray

```
(:method (clear_tray)
    ((Instance On ?on)
     (HasArea ?on ?trayArea)
     (HasPhysicalEntity ?on ?object)
     (Instance TrayArea ?trayArea)
     (Instance Arm ?arm)
     (HasGripper ?arm ?gripper)
     (not ((Instance Holding ?holding)
           (HasGripper ?holding ?gripper)
           (hasPassiveObject ?holding ?otherObject))
     )
     (HasArmPosture ?arm ?armposture)
     (not (Instance ArmToSidePosture ?armPosture))
     (not (Instance TorsoUpPosture ?torsoposture))
     (Instance ?objtype ?object)
     (not (hasAffordanceException ?objtype pick_from_tray))
     )
    ((!move_torso TorsoUpPosture)
     (!move_arm_to_side ?arm)
     (!pick_from_tray ?object ?arm)
     )
    ((Instance On ?on)
     (HasArea ?on ?trayArea)
     (HasPhysicalEntity ?on ?object)
     (Instance TrayArea ?trayArea)
     (Instance Arm ?arm)
     (HasGripper ?arm ?gripper)
     (not ((Instance Holding ?holding)
           (HasGripper ?holding ?gripper)
           (hasPassiveObject ?holding ?otherObject))
     )
     (HasArmPosture ?arm ?armposture)
     (not (Instance ArmToSidePosture ?armPosture))
     (Instance TorsoUpPosture ?torsoposture)
     (Instance ?objtype ?object)
     (not (hasAffordanceException ?objtype pick_from_tray))
     )
    ((!move_arm_to_side ?arm)
     (!pick_from_tray ?object ?arm)
     )
    ((Instance On ?on)
     (HasArea ?on ?trayArea)
     (HasPhysicalEntity ?on ?object)
     (Instance TrayArea ?trayArea)
     (Instance Arm ?arm)
     (HasGripper ?arm ?gripper)
     (not ((Instance Holding ?holding)
           (HasGripper ?holding ?gripper)
           (hasPassiveObject ?holding ?otherObject))
     )
     (HasArmPosture ?arm ?armposture)
     (Instance ArmToSidePosture ?armPosture)
     (not (Instance TorsoUpPosture ?torsoposture))
     (Instance ?objtype ?object)
     (not (hasAffordanceException ?objtype pick_from_tray))
     )
    ((!move_torso TorsoUpPosture)
     (!pick_from_tray ?object ?arm)
     )
    ((Instance On ?on)
     (HasArea ?on ?trayArea)
     (HasPhysicalEntity ?on ?object)
     (Instance TrayArea ?trayArea)
     (Instance Arm ?arm)
     (HasGripper ?arm ?gripper)
     (not ((Instance Holding ?holding)
           (HasGripper ?holding ?gripper)
           (hasPassiveObject ?holding ?otherObject))
     )
     (HasArmPosture ?arm ?armposture)
     (Instance ArmToSidePosture ?armPosture)
     (Instance TorsoUpPosture ?torsoposture)
     (Instance ?objtype ?object)
     (not (hasAffordanceException ?objtype pick_from_tray))
     )
    ((!pick_from_tray ?object ?arm))
    ((Instance On ?on)
     (HasArea ?on ?trayArea)
```

```
      (HasPhysicalEntity ?on ?object)
      (not (Instance TrayArea ?trayArea))
      )
     ()
     )
```

A.4 Tray Manipulation

Listing A.6: Operator for placing an object on the tray

```
(:operator (!place_on_tray ?object ?arm ?trayArea)
    ((Instance TrayArea ?trayArea)
     (not ((Instance On ?trayOn)
           (HasArea ?trayOn ?trayArea)
           (HasPhysicalEntity ?trayOn ?otherObject))
     )
     (HasGripper ?arm ?gripper)
     (Instance Holding ?holding)
     (HasGripper ?holding ?gripper)
     (HasPassiveObject ?holding ?object)
     (Instance Arm ?arm)
     (HasArmPosture ?arm ?armposture)
     (Instance ArmToSidePosture ?armposture)
     (Instance TorsoUpPosture ?torsoposture)
     ; affordanceExceptions
     (Instance ?objtype ?object)
     (not (hasAffordanceException ?objtype place_on_tray))
     ; assignments
     (CNT ?cnt)
     (new_constant ?newOn On ?cnt)
     (new_constant ?newArmPosture ArmToSidePosture ?cnt ?arm)
     (inc_cnt ?cntn ?cnt)
    )
    ((Instance Holding ?holding)
     (HasGripper ?holding ?gripper)
     (HasPassiveObject ?holding ?object)
     (Instance ArmToSidePosture ?armposture)
     (hasArmPosture ?arm ?armposture)
     (CNT ?cnt))
    ; add
    ((Instance On ?newOn)
     (HasArea ?newOn ?trayArea)
     (HasPhysicalEntity ?newOn ?object)
     (Instance ArmToSidePosture ?newArmPosture)
     (hasArmPosture ?arm ?newArmPosture)
     (CNT ?cntn))
    )
```

Listing A.7: Operator for placing an object on the tray

```
(:operator (!pick_from_tray ?object ?arm)
    ((Instance On ?on)
     (HasArea ?on ?trayArea)
     (HasPhysicalEntity ?on ?object)
     (Instance TrayArea ?trayArea)
     (Instance Arm ?arm)
     (HasGripper ?arm ?gripper)
     (not ((Instance Holding ?holding)
           (HasGripper ?holding ?gripper)
           (HasPassiveObject ?holding ?otherObject)
           ))
     (HasArmPosture ?arm ?armposture)
     (Instance ArmToSidePosture ?armposture)
     (Instance TorsoUpPosture ?torsoposture)
     ; affordanceExceptions
     (Instance ?objtype ?object)
     (not (hasAffordanceException ?objtype pick_from_tray))
     ; assignments
     (CNT ?cnt)
     (new_constant ?newHolding Holding ?cnt)
     (new_constant ?newArmPosture ArmToSidePosture ?cnt ?arm)
     (inc_cnt ?cntn ?cnt)
     )
    ; del
    ((Instance On ?on)
     (HasArea ?on ?trayArea)
     (HasPhysicalEntity ?on ?object)
     (Instance ArmToSidePosture ?armposture)
     (hasArmPosture ?arm ?armposture)
     (CNT ?cnt))
    ; add
    ((Instance Holding ?newHolding)
     (HasGripper ?newHolding ?gripper)
     (hasPassiveObject ?newHolding ?object)
     (Instance ArmToSidePosture ?newArmPosture)
     (hasArmPosture ?arm ?newArmPosture)
     (CNT ?cntn))
    )
```

A.5 Moving the Base

Listing A.8: Modified methods for moving the base

```
(:operator (!move_base ?to ?origin)
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?from)
     (CNT ?cnt)
     (new_constant ?newrobotat RobotAt ?cnt)
     (inc_cnt ?cntn ?cnt)
     )
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?from)
     (CNT ?cnt)
     )
    ((Instance RobotAt ?newrobotat)
     (HasRobot ?newrobotat trixi1)
     (HasArea ?newrobotat ?to)
     (CNT ?cntn)
     )
    )
(:operator (!move_base_param ?to ?speed ?origin)
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?from)
     (CNT ?cnt)
     (new_constant ?newrobotat RobotAt ?cnt)
     (inc_cnt ?cntn ?cnt)
     )
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?from)
     (CNT ?cnt)
     )
    ((Instance RobotAt ?newrobotat)
     (HasRobot ?newrobotat trixi1)
     (HasArea ?newrobotat ?to)
     (CNT ?cntn)
     )
    )
(:operator (!move_base_blind ?to ?origin)
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?from)
     (CNT ?cnt)
     (new_constant ?newrobotat RobotAt ?cnt)
     (inc_cnt ?cntn ?cnt)
     )
    ((Instance RobotAt ?robotat)
     (HasRobot ?robotat trixi1)
     (HasArea ?robotat ?from)
     (CNT ?cnt))
    ((Instance RobotAt ?newrobotat)
     (HasRobot ?newrobotat trixi1)
     (HasArea ?newrobotat ?to)
     (CNT ?cntn)
     )
    )
```

B Resulting plans

B.1 Plans for Attend Table 1 Scenario

Listing B.1: Plan A for attend table 1 scenario

```
Plan with 32.2 (32.2) cost and 3 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE1 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1- 9.8302608696
)
```

Listing B.2: Plan B for attend table 1 scenario

```
Plan with 38.5 (49.3) cost and 4 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
    (
        [
          -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
          -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
              10.7535128205
        ]
      -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE1 FAST
          PREMANIPULATIONAREAEASTCOUNTER1- 7.85908
    )
)
```

Listing B.3: Plan C for attend table 1 scenario

```
Plan with 40.5 (51.2) cost and 4 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
    (
        [
          -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
          -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
              10.7535128205
        ]
      -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE1 SLOW
          PREMANIPULATIONAREAEASTCOUNTER1- 9.8302608696
    )
)
```

Listing B.4: Plan with minimum amount of steps for attend table 1 scenario

```
Plan with 32.2 (32.2) cost and 3 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE1 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1- 9.8302608696
)
```

Listing B.5: Plan with shortest sequential duration for attend table 1 scenario

```
Plan with 32.2 (32.2) cost and 3 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE1 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1- 9.8302608696
)
```

Listing B.6: Plan with shortest parallel duration for attend table 1 scenario

```
Plan with 32.2 (32.2) cost and 3 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  −!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE1 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1−  9.8302608696
)
```

B.2 Plans for Attend Table 2 Scenario

Listing B.7: Plan A for attend table 2 scenario

```
Plan with 36.3 (36.3) cost and 3 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE2 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1- 13.9501304348
)
```

Listing B.8: Plan B for attend table 2 scenario

```
Plan with 40.8 (51.6) cost and 4 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
    (
        [
          -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
          -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
              10.7535128205
        ]
      -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE2 FAST
          PREMANIPULATIONAREAEASTCOUNTER1- 10.14848
    )
)
```

Listing B.9: Plan C for attend table 2 scenario

```
Plan with 44.6 (55.4) cost and 4 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
    (
        [
          -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
          -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
              10.7535128205
        ]
      – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE2 SLOW
          PREMANIPULATIONAREAEASTCOUNTER1– 13.9501304348
  )
)
```

Listing B.10: Plan with minimum amount of steps for attend table 2 scenario

```
Plan with 36.3 (36.3) cost and 3 steps:
(
  – !MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1– 11.6365
  – !TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE– 10.7535128205
  – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE2 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1– 13.9501304348
)
```

Listing B.11: Plan with shortest sequential duration for attend table 2 scenario

```
Plan with 36.3 (36.3) cost and 3 steps:
(
  – !MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1– 11.6365
  – !TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE– 10.7535128205
  – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE2 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1– 13.9501304348
)
```

Listing B.12: Plan with shortest parallel duration for attend table 2 scenario

```
Plan with 36.3 (36.3) cost and 3 steps:
(
  – !MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1– 11.6365
  – !TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE– 10.7535128205
  – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE2 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1– 13.9501304348
)
```

B.3 Plans for Attend Table 3 Scenario

Listing B.13: Plan A for attend table 3 scenario

```
Plan with 43.8 (43.8) cost and 3 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE3 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1- 21.361
)
```

Listing B.14: Plan B for attend table 3 scenario

```
Plan with 46.1 (56.8) cost and 4 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
    (
        [
          -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
          -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
              10.7535128205
        ]
      -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE3 FAST
          PREMANIPULATIONAREAEASTCOUNTER1- 15.4111538462
    )
)
```

Listing B.15: Plan C for attend table 2 scenario

```
Plan with 44.6 (55.4) cost and 4 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
    (
        [
          -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
          -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
              10.7535128205
        ]
        −!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE2 SLOW
            PREMANIPULATIONAREAEASTCOUNTER1− 13.9501304348
    )
)
```

Listing B.16: Plan with minimum amount of steps for attend table 3 scenario

```
Plan with 43.8 (43.8) cost and 3 steps:
(
  −!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1− 11.6365
  −!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE− 10.7535128205
  −!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE3 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1− 21.361
)
```

Listing B.17: Plan with shortest sequential duration for attend table 3 scenario

```
Plan with 43.8 (43.8) cost and 3 steps:
(
  −!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1− 11.6365
  −!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE− 10.7535128205
  −!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE3 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1− 21.361
)
```

Listing B.18: Plan with shortest parallel duration for attend table 3 scenario

```
Plan with 43.8 (43.8) cost and 3 steps:
(
  −!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1− 11.6365
  −!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE− 10.7535128205
  −!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE3 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1− 21.361
)
```

B.4 Plans for Attend Table 4 Scenario

Listing B.19: Plan B for attend table 4 scenario

```
Plan with 48.2 (58.9) cost and 4 steps:
(
  –!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1– 11.6365
     (
         [
            –!MOVE_TORSO TORSODOWNPOSTURE– 19.0144878049
            –!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE–
                10.7535128205
         ]
       –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
           PREMANIPULATIONAREAEASTCOUNTER1– 17.5291590909
     )
)
```

Listing B.20: Plan A for attend table 4 scenario

```
Plan with 49.3 (49.3) cost and 3 steps:
(
  –!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1– 11.6365
  –!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE– 10.7535128205
  –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1– 26.9018636364
)
```

Listing B.21: Plan C for attend table 4 scenario

```
Plan with 57.5 (68.3) cost and 4 steps:
(
  –!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1– 11.6365
     (
         [
            –!MOVE_TORSO TORSODOWNPOSTURE– 19.0144878049
            –!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE–
                10.7535128205
         ]
```

```
            –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
                PREMANIPULATIONAREAEASTCOUNTER1– 26.9018636364
        )
)
```

Listing B.22: Plan with minimum amount of steps for attend table 4 scenario

```
Plan with 49.3 (49.3) cost and 3 steps:
(
    –!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
        MANIPULATIONAREAEASTCOUNTER1– 11.6365
    –!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE– 10.7535128205
    –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
        PREMANIPULATIONAREAEASTCOUNTER1– 26.9018636364
)
```

Listing B.23: Plan with shortest sequential duration for attend table 4 scenario

```
Plan with 49.3 (49.3) cost and 3 steps:
(
    –!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
        MANIPULATIONAREAEASTCOUNTER1– 11.6365
    –!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE– 10.7535128205
    –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
        PREMANIPULATIONAREAEASTCOUNTER1– 26.9018636364
)
```

Listing B.24: Plan with shortest parallel duration for attend table 4 scenario

```
Plan with 48.2 (58.9) cost and 4 steps:
(
    –!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
        MANIPULATIONAREAEASTCOUNTER1– 11.6365
        (
            [
                –!MOVE_TORSO TORSODOWNPOSTURE– 19.0144878049
                –!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE–
                    10.7535128205
            ]
            –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
                PREMANIPULATIONAREAEASTCOUNTER1– 17.5291590909
        )
)
```

B.5 Plans for Door Scenario

Listing B.25: Plan A for door scenario

```
Plan with 63.8 (74.5) cost and 6 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
       MANIPULATIONAREAEASTCOUNTER2- 10.9211875
     (
          [
            -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
            -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
                10.7535128205
          ]
       -!MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY FAST
            PREMANIPULATIONAREAEASTCOUNTER2- 18.3392
       -!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT FAST
            NEARAREADOORNARROWHALLWAY- 9.4315
       -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
            NEARAREADOORNARROWRESTAURANT- 6.0564
     )
)
```

Listing B.26: Plan B for door scenario

```
Plan with 64.9 (75.7) cost and 6 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
       MANIPULATIONAREAEASTCOUNTER2- 10.9211875
     (
          [
            -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
            -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
                10.7535128205
          ]
       -!MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY FAST
            PREMANIPULATIONAREAEASTCOUNTER2- 18.3392
       -!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT SLOW
            NEARAREADOORNARROWHALLWAY- 10.6
       -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
            NEARAREADOORNARROWRESTAURANT- 6.0564
     )
)
```

Listing B.27: Plan C for door scenario

```
Plan with 66.0 (76.7) cost and 6 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
       MANIPULATIONAREAEASTCOUNTER2- 10.9211875
    (
        [
          -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
          -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
              10.7535128205
        ]
      -!MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY FAST
          PREMANIPULATIONAREAEASTCOUNTER2- 18.3392
      -!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT FAST
          NEARAREADOORNARROWHALLWAY- 9.4315
      -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
          NEARAREADOORNARROWRESTAURANT- 8.2623
    )
)
```

Listing B.28: Plan D for door scenario

```
Plan with 67.1 (77.9) cost and 6 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
       MANIPULATIONAREAEASTCOUNTER2- 10.9211875
    (
        [
          -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
          -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
              10.7535128205
        ]
      -!MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY FAST
          PREMANIPULATIONAREAEASTCOUNTER2- 18.3392
      -!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT SLOW
          NEARAREADOORNARROWHALLWAY- 10.6
      -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
          NEARAREADOORNARROWRESTAURANT- 8.2623
    )
)
```

Listing B.29: Plan E for door scenario

```
Plan with 71.4 (82.2) cost and 6 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
       MANIPULATIONAREAEASTCOUNTER2- 10.9211875
    (
      [
        -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
        -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
            10.7535128205
      ]
      -!MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY SLOW
          PREMANIPULATIONAREAEASTCOUNTER2- 26.0026
      -!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT FAST
          NEARAREADOORNARROWHALLWAY- 9.4315
      -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
          NEARAREADOORNARROWRESTAURANT- 6.0564
    )
)
```

Listing B.30: Plan F for door scenario

```
Plan with 72.6 (83.3) cost and 6 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
       MANIPULATIONAREAEASTCOUNTER2- 10.9211875
    (
      [
        -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
        -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
            10.7535128205
      ]
      -!MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY SLOW
          PREMANIPULATIONAREAEASTCOUNTER2- 26.0026
      -!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT SLOW
          NEARAREADOORNARROWHALLWAY- 10.6
      -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
          NEARAREADOORNARROWRESTAURANT- 6.0564
    )
)
```

Listing B.31: Plan G for door scenario

```
Plan with 73.6 (84.4) cost and 6 steps:
(
  −!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2− 10.9211875
    (
        [
          −!MOVE_TORSO TORSODOWNPOSTURE− 19.0144878049
          −!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE−
              10.7535128205
        ]
      −!MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY SLOW
          PREMANIPULATIONAREAEASTCOUNTER2− 26.0026
      −!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT FAST
          NEARAREADOORNARROWHALLWAY− 9.4315
      −!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
          NEARAREADOORNARROWRESTAURANT− 8.2623
    )
)
```

Listing B.32: Plan H for door scenario

```
Plan with 74.8 (85.6) cost and 6 steps:
(
  −!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2− 10.9211875
    (
        [
          −!MOVE_TORSO TORSODOWNPOSTURE− 19.0144878049
          −!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE−
              10.7535128205
        ]
      −!MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY SLOW
          PREMANIPULATIONAREAEASTCOUNTER2− 26.0026
      −!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT SLOW
          NEARAREADOORNARROWHALLWAY− 10.6
      −!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
          NEARAREADOORNARROWRESTAURANT− 8.2623
    )
)
```

Listing B.33: Plan I for door scenario

```
Plan with 82.2 (82.2) cost and 6 steps:
```

```
(
  – !MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2 – 10.9211875
  – !TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE – 10.7535128205
  – !MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY SLOW
      PREMANIPULATIONAREAEASTCOUNTER2 – 26.0026
  – !MOVE_TORSO TORSODOWNPOSTURE – 19.0144878049
  – !MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT FAST
      NEARAREADOORNARROWHALLWAY – 9.4315
  – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
      NEARAREADOORNARROWRESTAURANT – 6.0564
)
```

Listing B.34: Plan J for door scenario

```
Plan with 82.3 (93.0) cost and 6 steps:
(
  – !MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2 – 10.9211875
    (
        [
          – !MOVE_TORSO TORSODOWNPOSTURE – 19.0144878049
          – !TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE –
              10.7535128205
        ]
      – !MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY FAST
          PREMANIPULATIONAREAEASTCOUNTER2 – 27.4094736842
      – !MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT FAST
          NEARAREADOORWIDEHALLWAY – 8.9860555556
      – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
          NEARAREADOORWIDERESTAURANT – 15.959
    )
)
```

Listing B.35: Plan K for door scenario

```
Plan with 83.3 (83.3) cost and 6 steps:
(
  – !MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2 – 10.9211875
  – !TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE – 10.7535128205
  – !MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY SLOW
      PREMANIPULATIONAREAEASTCOUNTER2 – 26.0026
  –!MOVE_TORSO TORSODOWNPOSTURE– 19.0144878049
  –!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT SLOW
      NEARAREADOORNARROWHALLWAY– 10.6
  –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
      NEARAREADOORNARROWRESTAURANT– 6.0564
)
```

Listing B.36: Plan L for door scenario

```
Plan with 83.5 (94.2) cost and 6 steps:
(
  –!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2– 10.9211875
    (
        [
          –!MOVE_TORSO TORSODOWNPOSTURE– 19.0144878049
          –!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE–
              10.7535128205
        ]
      –!MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY FAST
          PREMANIPULATIONAREAEASTCOUNTER2– 27.4094736842
      –!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT SLOW
          NEARAREADOORWIDEHALLWAY– 10.1771052632
      –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
          NEARAREADOORWIDERESTAURANT– 15.959
    )
)
```

Listing B.37: Plan M for door scenario

```
Plan with 84.4 (84.4) cost and 6 steps:
(
  –!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2– 10.9211875
  –!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE– 10.7535128205
  –!MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY SLOW
      PREMANIPULATIONAREAEASTCOUNTER2– 26.0026
  –!MOVE_TORSO TORSODOWNPOSTURE– 19.0144878049
  –!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT FAST
      NEARAREADOORNARROWHALLWAY– 9.4315
  –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
      NEARAREADOORNARROWRESTAURANT– 8.2623
)
```

Listing B.38: Plan N for door scenario

```
Plan with 85.6 (85.6) cost and 6 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2- 10.9211875
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  -!MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY SLOW
      PREMANIPULATIONAREAEASTCOUNTER2- 26.0026
  -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
  -!MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT SLOW
      NEARAREADOORNARROWHALLWAY- 10.6
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
      NEARAREADOORNARROWRESTAURANT- 8.2623
)
```

Listing B.39: Plan O for door scenario

```
Plan with 92.7 (103.4) cost and 6 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2- 10.9211875
    (
        [
          -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
          -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
              10.7535128205
        ]
      -!MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY FAST
          PREMANIPULATIONAREAEASTCOUNTER2- 27.4094736842
      -!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT FAST
          NEARAREADOORWIDEHALLWAY- 8.9860555556
      -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
          NEARAREADOORWIDERESTAURANT- 26.3236842105
    )
)
```

Listing B.40: Plan P for door scenario

```
Plan with 93.8 (104.6) cost and 6 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2- 10.9211875
    (
            [
               −!MOVE_TORSO TORSODOWNPOSTURE− 19.0144878049
               −!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE−
                  10.7535128205
            ]
         −!MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY FAST
            PREMANIPULATIONAREAEASTCOUNTER2− 27.4094736842
         −!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT SLOW
            NEARAREADOORWIDEHALLWAY− 10.1771052632
         −!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
            NEARAREADOORWIDERESTAURANT− 26.3236842105
      )
)
```

Listing B.41: Plan Q for door scenario

```
Plan with 98.9 (109.7) cost and 6 steps:
(
   −!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
         MANIPULATIONAREAEASTCOUNTER2− 10.9211875
      (
            [
               −!MOVE_TORSO TORSODOWNPOSTURE− 19.0144878049
               −!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE−
                  10.7535128205
            ]
         −!MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
            PREMANIPULATIONAREAEASTCOUNTER2− 44.06175
         −!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT FAST
            NEARAREADOORWIDEHALLWAY− 8.9860555556
         −!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
            NEARAREADOORWIDERESTAURANT− 15.959
      )
)
```

Listing B.42: Plan R for door scenario

```
Plan with 100.1 (110.9) cost and 6 steps:
(
   −!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
         MANIPULATIONAREAEASTCOUNTER2− 10.9211875
      (
            [
              – !MOVE_TORSO TORSODOWNPOSTURE– 19.0144878049
              – !TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE–
                  10.7535128205
            ]
          – !MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
              PREMANIPULATIONAREAEASTCOUNTER2– 44.06175
          – !MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT SLOW
              NEARAREADOORWIDEHALLWAY– 10.1771052632
          – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
              NEARAREADOORWIDERESTAURANT– 15.959
    )
)
```

Listing B.43: Plan S for door scenario

```
Plan with 102.2 (102.2) cost and 5 steps:
(
  – !MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2– 10.9211875
  – !TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE– 10.7535128205
  – !MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
      PREMANIPULATIONAREAEASTCOUNTER2– 44.06175
  – !MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT SLOW
      NEARAREADOORWIDEHALLWAY– 10.1771052632
  – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
      NEARAREADOORWIDERESTAURANT– 26.3236842105
)
```

Listing B.44: Plan T for door scenario

```
Plan with 109.3 (120.1) cost and 6 steps:
(
  – !MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2– 10.9211875
    (
        [
          – !MOVE_TORSO TORSODOWNPOSTURE– 19.0144878049
          – !TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE–
              10.7535128205
        ]
      – !MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
          PREMANIPULATIONAREAEASTCOUNTER2– 44.06175
        –!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT FAST
            NEARAREADOORWIDEHALLWAY– 8.9860555556
        –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
            NEARAREADOORWIDERESTAURANT– 26.3236842105
    )
)
```

Listing B.45: Plan U for door scenario

```
Plan with 109.7 (109.7) cost and 6 steps:
(
    –!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
        MANIPULATIONAREAEASTCOUNTER2– 10.9211875
    –!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE– 10.7535128205
    –!MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
        PREMANIPULATIONAREAEASTCOUNTER2– 44.06175
    –!MOVE_TORSO TORSODOWNPOSTURE– 19.0144878049
    –!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT FAST
        NEARAREADOORWIDEHALLWAY– 8.9860555556
    –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
        NEARAREADOORWIDERESTAURANT– 15.959
)
```

Listing B.46: Plan V for door scenario

```
Plan with 110.5 (121.3) cost and 6 steps:
(
    –!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
        MANIPULATIONAREAEASTCOUNTER2– 10.9211875
    (
        [
            –!MOVE_TORSO TORSODOWNPOSTURE– 19.0144878049
            –!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE–
                10.7535128205
        ]
        –!MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
            PREMANIPULATIONAREAEASTCOUNTER2– 44.06175
        –!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT SLOW
            NEARAREADOORWIDEHALLWAY– 10.1771052632
        –!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
            NEARAREADOORWIDERESTAURANT– 26.3236842105
    )
)
```

Listing B.47: Plan W for door scenario

```
Plan with 110.9 (110.9) cost and 6 steps:
(
   −!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
        MANIPULATIONAREAEASTCOUNTER2− 10.9211875
   −!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE− 10.7535128205
   −!MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
        PREMANIPULATIONAREAEASTCOUNTER2− 44.06175
   −!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT SLOW
        NEARAREADOORWIDEHALLWAY− 10.1771052632
   −!MOVE_TORSO TORSODOWNPOSTURE− 19.0144878049
   −!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
        NEARAREADOORWIDERESTAURANT− 15.959
)
```

Listing B.48: Plan X for door scenario

```
Plan with 110.9 (110.9) cost and 6 steps:
(
   −!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
        MANIPULATIONAREAEASTCOUNTER2− 10.9211875
   −!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE− 10.7535128205
   −!MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
        PREMANIPULATIONAREAEASTCOUNTER2− 44.06175
   −!MOVE_TORSO TORSODOWNPOSTURE− 19.0144878049
   −!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT SLOW
        NEARAREADOORWIDEHALLWAY− 10.1771052632
   −!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
        NEARAREADOORWIDERESTAURANT− 15.959
)
```

Listing B.49: Plan Y for door scenario

```
Plan with 120.1 (120.1) cost and 6 steps:
(
   −!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
        MANIPULATIONAREAEASTCOUNTER2− 10.9211875
   −!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE− 10.7535128205
   −!MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
        PREMANIPULATIONAREAEASTCOUNTER2− 44.06175
   −!MOVE_TORSO TORSODOWNPOSTURE− 19.0144878049
   −!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT FAST
        NEARAREADOORWIDEHALLWAY− 8.9860555556
```

```
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
      NEARAREADOORWIDERESTAURANT- 26.3236842105
)
```

Listing B.50: Plan Z for door scenario

```
Plan with 121.3 (121.3) cost and 6 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2- 10.9211875
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  -!MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
      PREMANIPULATIONAREAEASTCOUNTER2- 44.06175
  -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
  -!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT SLOW
      NEARAREADOORWIDEHALLWAY- 10.1771052632
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
      NEARAREADOORWIDERESTAURANT- 26.3236842105
)
```

Listing B.51: Plan AA for door scenario

```
Plan with 121.3 (121.3) cost and 6 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2- 10.9211875
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  -!MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
      PREMANIPULATIONAREAEASTCOUNTER2- 44.06175
  -!MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT SLOW
      NEARAREADOORWIDEHALLWAY- 10.1771052632
  -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
      NEARAREADOORWIDERESTAURANT- 26.3236842105
)
```

Listing B.52: Plan with minimum amount of steps for door scenario

```
Plan with 102.2 (102.2) cost and 5 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2- 10.9211875
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  – !MOVE_BASE_PARAM NEARAREADOORWIDEHALLWAY SLOW
      PREMANIPULATIONAREAEASTCOUNTER2– 44.06175
  – !MOVE_BASE_PARAM NEARAREADOORWIDERESTAURANT SLOW
      NEARAREADOORWIDEHALLWAY– 10.1771052632
  – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
      NEARAREADOORWIDERESTAURANT– 26.3236842105
)
```

Listing B.53: Plan with shortest sequential duration for door scenario

```
        – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
            NEARAREADOORNARROWRESTAURANT– 6.0564
    )
)
```

Listing B.54: Plan with shortest parallel duration for door scenario

```
Plan with 63.8 (74.5) cost and 6 steps:
(
  – !MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER2
      MANIPULATIONAREAEASTCOUNTER2– 10.9211875
    (
        [
          – !MOVE_TORSO TORSODOWNPOSTURE– 19.0144878049
          – !TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE–
              10.7535128205
        ]
      – !MOVE_BASE_PARAM NEARAREADOORNARROWHALLWAY FAST
          PREMANIPULATIONAREAEASTCOUNTER2– 18.3392
      – !MOVE_BASE_PARAM NEARAREADOORNARROWRESTAURANT FAST
          NEARAREADOORNARROWHALLWAY– 9.4315
      – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
          NEARAREADOORNARROWRESTAURANT– 6.0564
    )
)
```

B.6 Plans for Pepper Mill Scenario

Listing B.55: Plan A for *pepper mill* scenario to table 1

```
Plan with 32.2 (32.2) cost and 3 steps:
(
  – !MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1 – 11.6365
  – !TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE – 10.7535128205
  – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE1 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1 – 9.8302608696
)
```

Listing B.56: Plan B for *pepper mill* scenario to table 1

```
Plan with 126.0 (152.3) cost and 9 steps:
(
  – !MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1 – 11.6365
  – !PICK_FROM_TRAY PEPPERMILL RIGHTARM1 – 22
    (
        [
          – !MOVE_ARMS_TO_CARRYPOSTURE – 13.5361914894
          – !MOVE_TORSO TORSODOWNPOSTURE – 19.0144878049
        ]
      – !MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE1 FAST
          PREMANIPULATIONAREAEASTCOUNTER1 – 7.85908
        (
            [
              – !MOVE_ARM_TO_SIDE RIGHTARM1 – 12.7538043478
              – !MOVE_TORSO TORSOUPPOSTURE – 21.7206086957
            ]
          – !PLACE_ON_TRAY PEPPERMILL RIGHTARM1 TRAYAREARIGHT1 –
              33
          – !TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE –
              10.7535128205
        )
    )
)
```

Listing B.57: Plan A for *pepper mill* scenario to table 2

```
Plan with 36.3 (36.3) cost and 3 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE2 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1- 13.9501304348
)
```

Listing B.58: Plan B for *pepper mill* scenario to table 2

```
Plan with 128.3 (154.6) cost and 9 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
  -!PICK_FROM_TRAY PEPPERMILL RIGHTARM1- 22
      (
          [
            -!MOVE_ARMS_TO_CARRYPOSTURE- 13.5361914894
            -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
          ]
        -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE2 FAST
            PREMANIPULATIONAREAEASTCOUNTER1- 10.14848
          (
              [
                -!MOVE_ARM_TO_SIDE RIGHTARM1- 12.7538043478
                -!MOVE_TORSO TORSOUPPOSTURE- 21.7206086957
              ]
            -!PLACE_ON_TRAY PEPPERMILL RIGHTARM1 TRAYAREARIGHT1-
                33
            -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
                10.7535128205
          )
      )
)
```

Listing B.59: Plan A for *pepper mill* scenario to table 3

```
Plan with 43.8 (43.8) cost and 3 steps:
(
   -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
        MANIPULATIONAREAEASTCOUNTER1- 11.6365
   -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
   -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE3 SLOW
        PREMANIPULATIONAREAEASTCOUNTER1- 21.361
)
```

Listing B.60: Plan B for *pepper mill* scenario to table 3

```
Plan with 133.5 (159.8) cost and 9 steps:
(
   -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
        MANIPULATIONAREAEASTCOUNTER1- 11.6365
   -!PICK_FROM_TRAY PEPPERMILL RIGHTARM1- 22
      (
           [
              -!MOVE_ARMS_TO_CARRYPOSTURE- 13.5361914894
              -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
           ]
         -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE3 FAST
              PREMANIPULATIONAREAEASTCOUNTER1- 15.4111538462
           (
                 [
                    -!MOVE_ARM_TO_SIDE RIGHTARM1- 12.7538043478
                    -!MOVE_TORSO TORSOUPPOSTURE- 21.7206086957
                 ]
              -!PLACE_ON_TRAY PEPPERMILL RIGHTARM1 TRAYAREARIGHT1-
                    33
              -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
                    10.7535128205
           )
      )
)
```

Listing B.61: Plan A for *pepper mill* scenario to table 4

```
Plan with 49.3 (49.3) cost and 3 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
  -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE- 10.7535128205
  -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 SLOW
      PREMANIPULATIONAREAEASTCOUNTER1- 26.9018636364
)
```

Listing B.62: Plan B for *pepper mill* scenario to table 4

```
Plan with 135.7 (161.9) cost and 9 steps:
(
  -!MOVE_BASE_BLIND PREMANIPULATIONAREAEASTCOUNTER1
      MANIPULATIONAREAEASTCOUNTER1- 11.6365
  -!PICK_FROM_TRAY PEPPERMILL RIGHTARM1- 22
    (
      [
        -!MOVE_ARMS_TO_CARRYPOSTURE- 13.5361914894
        -!MOVE_TORSO TORSODOWNPOSTURE- 19.0144878049
      ]
    -!MOVE_BASE_PARAM PREMANIPULATIONAREASOUTHTABLE4 FAST
        PREMANIPULATIONAREAEASTCOUNTER1- 17.5291590909
      (
        [
          -!MOVE_ARM_TO_SIDE RIGHTARM1- 12.7538043478
          -!MOVE_TORSO TORSOUPPOSTURE- 21.7206086957
        ]
        -!PLACE_ON_TRAY PEPPERMILL RIGHTARM1 TRAYAREARIGHT1-
          33
        -!TUCK_ARMS ARMTUCKEDPOSTURE ARMTUCKEDPOSTURE-
          10.7535128205
      )
    )
)
```

Acknowledgement

This thesis was created at group TAMS, Department of Informatics, University of Hamburg.

I would like to thank Prof. Jianwei Zhang for the trust reposed in me by supervising my work and giving me the opportunity to work on the PR2 and Sebastian Rockel for supervising this thesis. His feedback and the critical review provided valuable improvements.

I would also like to thank Hannes Bistry for his ideas and suggestions.

On a personal note, I would like to thank my parents, Peter and Barbara Einig, for giving me the opportunity to study unburdened, and my brother Kolja Einig for his helpful suggestions on my writing. I also would like to thank Claudia Ringelhan for supporting and motivating me throughout the writing process. Another word of thanks is adressed to all the helpful colleagues who revised this thesis.

At last I would like to thank all RACE partners for creating the excellent system resulting from the project.