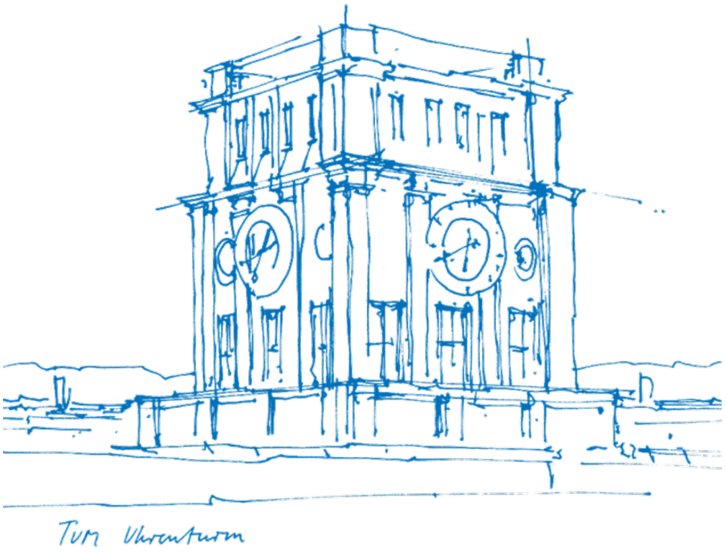
Sensor Based Robotic Systems and Intelligent Assistance Systems
School of Computation, Information and Technology
Technical University of Munich

Interactive/Reactive Robot Mission Scheduling

For Human-in-the-Loop Mission Control

Sofía La Rota López



Interactive/Reactive Robot Mission Scheduling

For Human-in-the-Loop Mission Control

Sofía La Rota López

Sensor Based Robotic Systems and Intelligent Assistance Systems
School of Computation, Information and Technology
Technical University of Munich

Interactive/Reactive Robot Mission Scheduling

For Human-in-the-Loop Mission Control

Sofía La Rota López

Thesis for the attainment of the academic degree

Master of Science (M.Sc.)

at the School of Computation, Information and Technology of the Technical University of Munich.

Examiner:
Prof. Dr. Alin Albu-Schäffer

Supervisors:
Ryo Sakagami, Andreas Dömel

Submitted:
Munich, 01.02.2025

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Munich, 01.02.2025 Sofía La Rota López

Abstract

One inherent problematic in the development and operation of complex robotic systems is the considerable skill barrier level associated with them. This is particularly problematic in the context of human-robot cooperation, especially with regards to non-roboticist human experts from other fields. In order to mitigate some frustrations associated with mission generation and modification involved for remotely operated scientific exploration missions, we developed and tested an interactive and reactive scheduling program capable of enforcing system and precedence constraints while providing human-understandable feedback as part of an iterative human-in-the-loop refinement process. An early failure warning during mission execution to enable preventive recovery behavior was also provided. As part of this development, we formulated a “good enough” search heuristic and decision process for clear result presentation to the user. The program demonstrated reasonably good results with simulated test missions and is being developed further for deployment on a real robotic system.

Acknowledgements

I would like to thank Andreas Dömel for his guidance. Above all, I'm extremely grateful to Ryo Sakagami for his major support as a mentor, encouragement as a researcher, and kindness as a friend. I want to thank Hayden for his unwavering support during this odyssey, as well as Nef, for (emphatically) reminding me to occasionally pause and take a break.

This work would not have been possible without the support of everyone at the institute. Working amongst you all has been a privilege. I thank you all.

Contents

1 Introduction

1.1 Cognition in Robotics

Cognitive load [9], a term coined in the 1980s, is used to describe the amount of information capable of being processed at any given time. This load is partly determined by the inherent difficulty of a task, but can also vary depending on the level of familiarity or perceived difficulty. Besides the applications of cognitive load theory in pedagogy and psychology, the study of information processing capacity has been extensively used in other areas as well. For example, great efforts are expended in the design of consumer goods and technology in order to provide a "user-friendly" and "intuitive" experiences. However, this is often not the case with emergent technologies and highly specialized settings with a high skill entry barrier.

Sadly, the inherently complex workings of robotics can easily fall prey to this dynamic, which has a profound impact on the interaction with robots and non-roboticists, for example. It can also hinder the development of multidisciplinary studies involving the participation of non-roboticist experts, likely hindering scientific research and exploration. In the study of human-robot interaction, the paradigm of cognition is twofold: there is the non-trivial development and testing of intelligent artificial systems, on the one hand; On the other, there is the issue of interface with an organic conscience. Human information processing capacities can be overwhelmed by a sufficiently large mental load, impairing cognitive performance and having a discouraging effect on the person or people involved. No matter how intricately the internal logic of a robot is designed, little effective cooperation can be had without effective communication between human and machine.

One maxima of clear communication in organizational settings is the ability to provide as much information as necessary to make informed decisions, without overwhelming the recipient with excessive details. This carries over to human-robot communication in the context of autonomous or semi-autonomous mission control. Effective delegation of execution to an automated systems plays a significant role in reducing the user cognitive load. Ideally, a suitable human-robot interface should enhance synergy between a team of human and robot agents and take advantage of all of the human's skills, knowledge, and resources instead of reducing them to the role of a pilot.

1.2 Problem Context

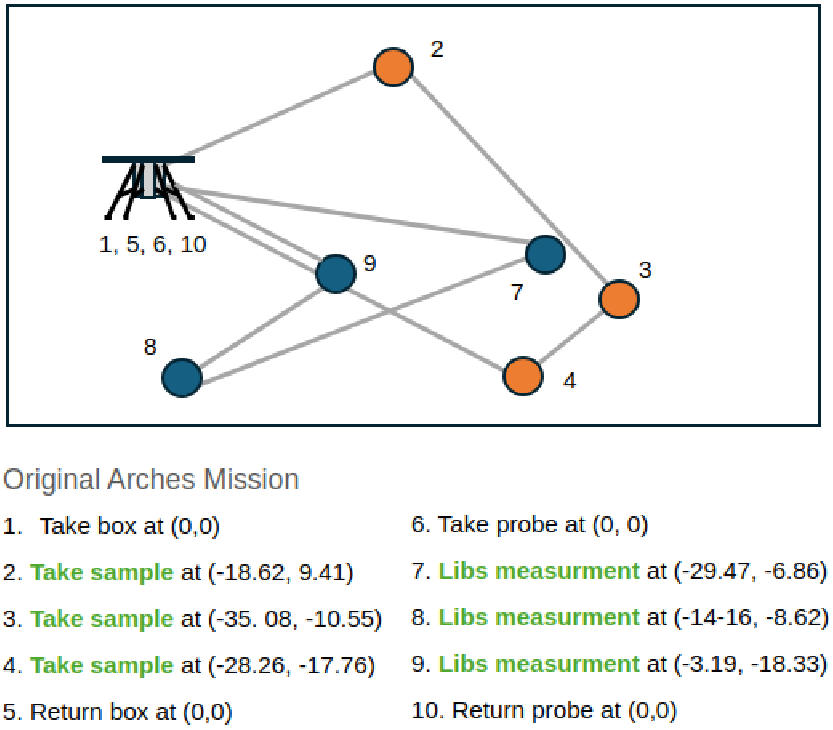
1.2.1 Human-in-the-loop Mission Control

the level of autonomy of a robot or robotic system can range from nonexistent (manual control) to completely autonomous. Irrespective of the system capacity, human supervision at a minimum is deemed critical for high-risk scenarios such as rescue or scientific operations as a safety measure and in order to safeguard operation robustness. Other reasons for including a human "in the loop" include ethical/organizational concerns about the level of decision-making power that an automated system should have. Finally, another advantage is the ability to rely on the non-encoded expertise that can be provided by a human, provided that a sufficiently efficient communication between the two is facilitated.

1.2.2 2022 Space-Analog Arches Mission

As part of the ARCHES project, a heterogeneous team of autonomous robots was deployed for a demonstration of scientific sampling mission on a Moon-analog site in Mt. Etna, Sicily [16]. An overview of this team can be found in Section 2.1. The Geological Mission I (GEO I) consisted of the semi-autonomous remote operation of two LRU robots to reach, inspect, and collect scientifically relevant samples. ROSMC, described in Section 2.1.2, was used as an interface for mission control between the robots and the mission control center located far from the remote location, in a hotel in Catania. This software allowed the operators to adapt the mission flexibly based on in-situ discoveries.

Equipped with several scientific cameras, LRU1 was able to collect panoramic scans for three different regions of interest (ROIs). Meanwhile, LRU2 successfully collected several soil and rock samples and deposited them in its collection bow, which it had previously picked up from the lander and put on its back. Afterwards, LRU2 performed three measurement readings using a Laser Induced Breakdown Spectroscopy (LIBS) payload module which it had, again, attached and placed on its back. A simplified diagram showing the result of the mission for LRU2 (not considering failed tasks) is shown in Figure 1.1.



Original Arches Mission

1. Take box at (0,0)
2. Take sample at (-18.62, 9.41)
3. Take sample at (-35. 08, -10.55)
4. Take sample at (-28.26, -17.76)
5. Return box at (0,0)
6. Take probe at (0, 0)
7. Libs measurment at (-29.47, -6.86)
8. Libs measurment at (-14-16, -8.62)
9. Libs measurment at (-3.19, -18.33)
10. Return probe at (0,0)

Figure 1.1 Overview of ROSMC's GUI.

Although one of the technical core challenges of the Arches project included automated task planning, no planning algorithm was implemented for use with the heterogeneous robot team for the mission in Mt. Etna, nor has any integration with mission control happened to date. The mission composition itself was planned manually, and the investigation on multi-robot mission scheduling carried on as derivative work thereafter.

A diagram showing an overview of the mission control dynamic as suggested by Sakagami et al. can be seen in Figure 1.2, showing a single instance of an automated mission schedule component being used. However, there is more that could to be done in order to improve the overall user experience when tasks fail, which can often happen in these types of environments. This vulnerability is compounded by the architecture behind RAFCON, the autonomous task executor behind the robots (Section 2.1.1). Despite its advantages, RAFCON lacks the context of the whole mission, and thus any autonomous recovery behavior can only occur in a reactive rather than of preventive manner.

During my time working at DLR as a student, I have been learning the intricacies of these systems and detected this project opportunity while maintaining DLR's ROSMC, expanding its functionality to provide interactive and reactive scheduling assistance and attempt to overcome these issues.

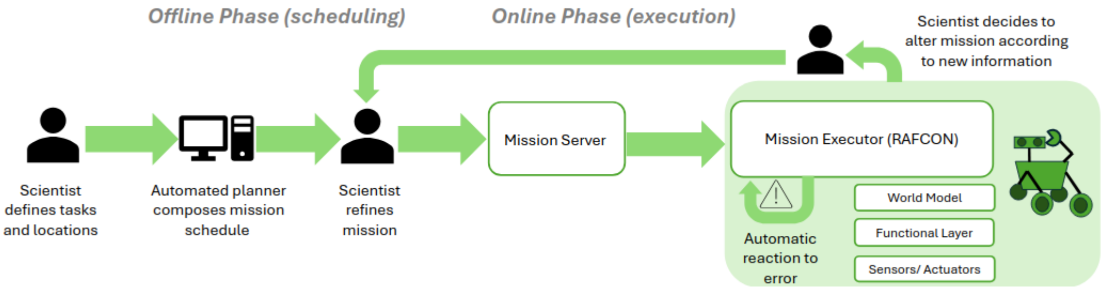

Offline Phase (scheduling)
Online Phase (execution)
Scientist decides to alter mission according to new information
Scientist defines tasks and locations
Automated planner composes mission schedule
Scientist refines mission
Mission Server
Mission Executor (RAFCON)
World Model
Functional Layer
Sensors/ Actuators
Automatic reaction to error

Figure 1.2 Schematic of the mission control for a remote scientific mission proposed under the assumption of an automated schedule planner implementation.

1.3 Objectives

1.3.1 Thesis Statement

We aim to develop and test a mission control program capable of providing interactive and reactive high-level skill task scheduling in for its use with collaborative human-in-the-loop robotic systems. The diagram in Figure 1.3 shows the proposed changes to the mission control dynamic, proposing a cooperative mission composition and preventive failure detection, as well as maintaining the capacity for flexible mission composition. Concrete objectives, as well as relevant functional requirements, are mentioned below in detail.

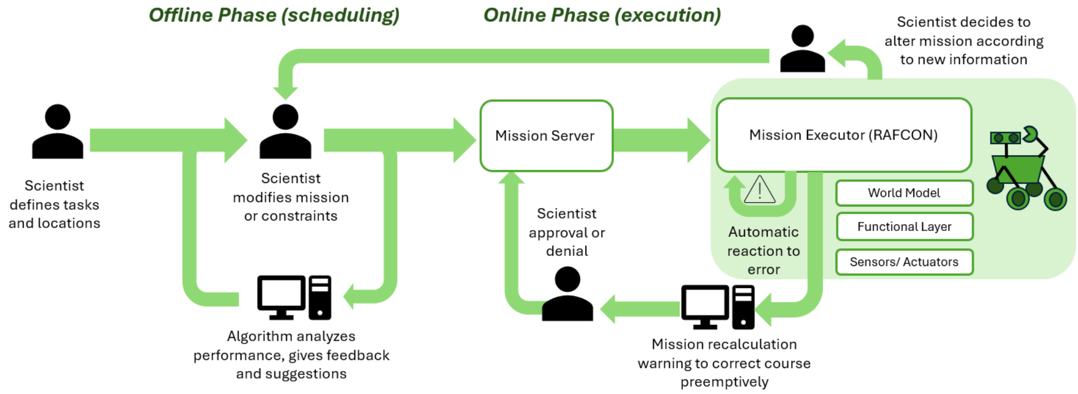

Offline Phase (scheduling)
Online Phase (execution)
Scientist decides to alter mission according to new information
Scientist defines tasks and locations
Scientist modifies mission or constraints
Algorithm analyzes performance, gives feedback and suggestions
Mission Server
Scientist approval or denial
Mission Executor (RAFCON)
World Model
Functional Layer
Sensors/ Actuators
Automatic reaction to error
Mission recalculation warning to correct course preemptively

Figure 1.3 Modified schematic of the mission control.

1.3.2 Project Objectives

- Develop and validate a robot mission scheduling algorithm
  - Select or develop a suitable heuristic
  - Generate acceptable mission sequences as solutions
  - Comparison and assessment of solutions (or lack thereof) in human-understandable terms
  - Integrate user-generated constraints or preferences
  - Develop and validate an algorithm for mission recalculation
- Develop and implement a robot scheduling program
  - Generate an intuitive interface for mission scheduling

  - Interactive negotiation process during mission composition
  - Provide passive mission feasibility recalculations during execution
  - Provide preemptive failure detection capabilities during execution
  - Provide mission recalculation capabilities during execution
- Deploy the scheduling program with the RMC heterogenous team of robots
  - Integrate with existing systems
  - Deploy the program for the ground station
  - Demonstrate the program capabilities via a demonstration with a LRU2

1.3.3 Functional Requirements

- Scheduling Algorithm
  - Enforce search in a feasibility region limited by battery
  - Enforce task prerequisite constraints
  - Comparison and assessment of solutions (or lack thereof) in human-understandable terms
  - Consider user-generated constraints or preferences
  - Provide transparent reasoning behind outputs.
  - System-agnostic formulation
- Scheduling Program
  - Guarantee the capacity for human override at all times (human led, robot assisted interaction)
  - Enforce task prerequisite constraints
  - Communicate information clearly, in intuitive terms
  - Consider user-generated constraints or preferences
  - System- and robot-agnostic formulation to consider scalability and implementation with other systems
  - Eventual scalability to a multi-robot formulation
  - Capable of being integrated and deployed with the systems used in the RMC lab at DLR

2 Related Work

2.1 System Specifics of the Heterogeneous Team of Mobile Robotics

The heterogenous team of mobile robotics (Figure 2.1) was developed by DLR to demonstrate technologies required for executing complex, partially autonomous missions on planetary surfaces for project ROBEX (Robotic Exploration for Extreme environments) [15, 8, 17, 12]. The team consists of two lightweight rovers and one aerial drone; as well as ground facilities including the lander and ground station; and equipment such as the Wi-Fi expander, sample collection box, and LIBS probe.

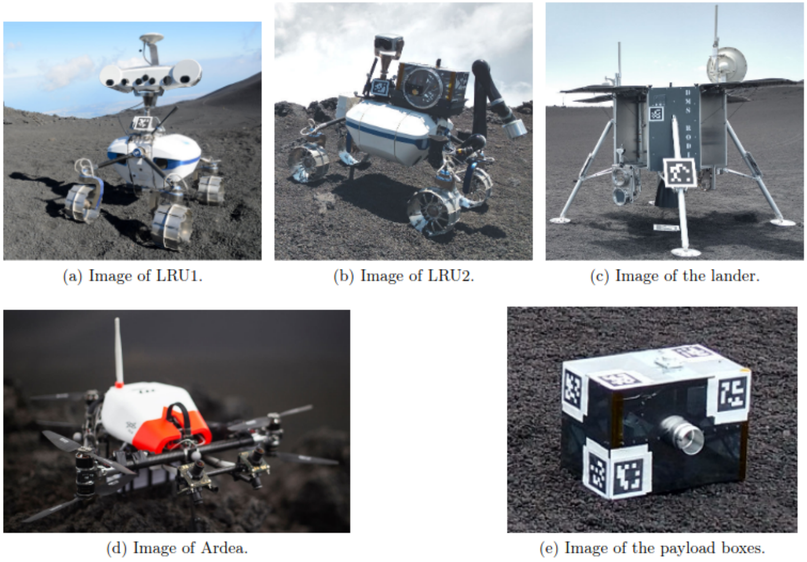

(a) Image of LRU1.
(b) Image of LRU2.
(c) Image of the lander.
(d) Image of Ardea.
(e) Image of the payload boxes.

Figure 2.1 team of the heterogeneous robots. Image taken from [12].

As their name implies, the Lightweight Rover Units, LRU1 and LRU2, weight approximately 40kg. More importantly, they each have complementary capabilities, as the first robot is equipped with three navigation cameras and the ScienceCam, which itself consists of a pair of wide-angle cameras (WACs) (Manta G-145B NIR), a narrow-angle camera (Manta G-505C), and a thermal camera (Xenics Serval-640). Its main function is to map unknown environments. Meanwhile, LRU2 is equipped with a force-controlled Jacko2 manipulator and the ability to carry payload boxes. Ardea, the hexacopter drone, with two pairs of wide-angle stereo cameras, is ideal for unknown area reconnaissance and coarse map creation. The lander functions as a global landmark and central operation base for the mobile robots, containing the scientific equipment and providing a place to recharge batteries. The versatility of the heterogenous team provides them with capabilities that make their operations both more robust and versatile as compared to a single robot.

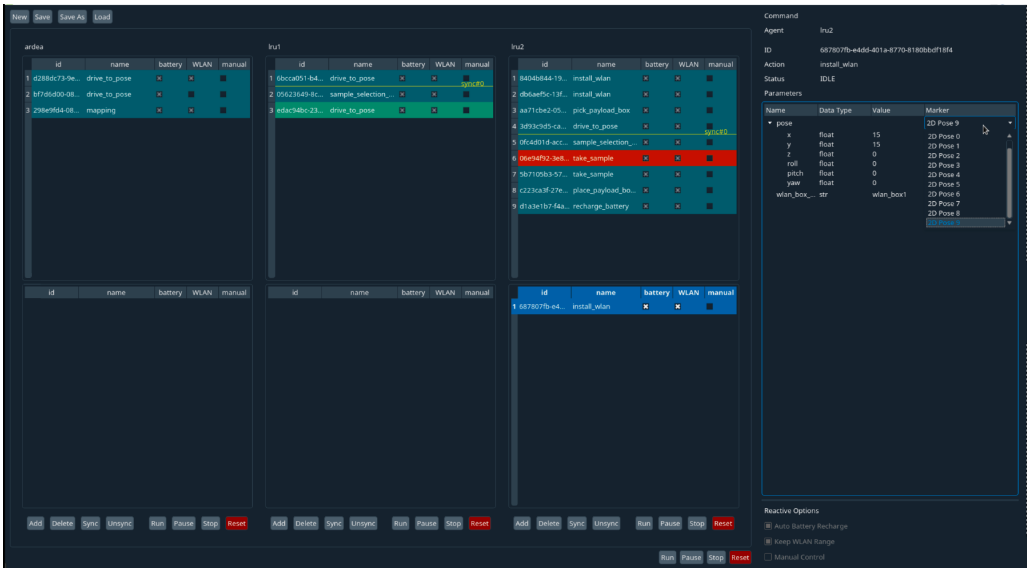
2.1.1 RMC Advanced Flow Control (RAFCON)

RAFCON (RMC advanced flow control) is a piece of software developed at DLR that allows users to program robotic tasks through the use of hierarchical state machines (HSM). One of the main benefits of a hierarchical architecture is the ability to atomize a highly complex problem into a series of simpler sub-problems with varying layers of detail, which partly explains the prevalence of hierarchical state machines

(HSM) as a common approach to robot control. One of the most distinctive characteristics of RAFCON is its visual approach to mission composition and prototyping, which enables the creation of high-level skill libraries that can be used to automate robotic systems with relative ease, even by users that are not expert robot programmers [3]. RAFCON is used as the main executive component of most, if not all, robots at DLR, including the team of mobile robotics.

2.1.2 ROS Mission Control (ROSMC)

The development of ROS Mission Control (ROSMC) [13] presented a mission operation framework for heterogeneous team of robots collaborating remotely with a human operator, and explicitly recognized the usability barrier present in robot mission control present in multidisciplinary settings, as well as the role of the operator's mental workload. The author addressed these issues in the context of a non-roboticist expert involved in a remote human-in-the-loop scientific exploration mission for the previously described team of heterogeneous mobile robots. The main GUI developed for ROSMC can be seen in Figure 2.2.



ardea

	id	name	battery	WLAN	manual
1	d288dc73-9e...	drive_to_pose	☒	☒	□
2	bf7d6d00-08...	drive_to_pose	☒	□	□
3	298e9fd4-08...	mapping	☒	☒	□

lru1

	id	name	battery	WLAN	manual
1	6bcca051-b4...	drive_to_pose	☒	☒	□
2	05623649-8c...	sample_selection_...	☒	☒	□
3	edac94bc-23...	drive_to_pose	☒	☒	□

lru2

	id	name	battery	WLAN	manual
1	8404b844-19...	install_wlan	☒	☒	□
2	db6aef5c-13f...	install_wlan	☒	☒	□
3	aa71cbe2-05...	pick_payload_box	☒	☒	□
4	3d93c9d5-ca...	drive_to_pose	☒	☒	□
5	0fc4d01d-acc...	sample_selection_...	☒	☒	□
6	06e94f92-3e8...	take_sample	☒	☒	□
7	5b7105b3-57...	take_sample	☒	☒	□
8	c223ca3f-27e...	place_payload_bo...	☒	☒	□
9	d1a3e1b7-f4a...	recharge_battery	☒	☒	□

	id	name	battery	WLAN	manual
1	687807fb-e4...	install_wlan	☒	☒	□

Command

Agent lru2

ID 687807fb-e4dd-401a-8770-8180bbdf18f4

Action install_wlan

Status IDLE

Parameters

Name	Data Type	Value	Marker
pose			2D Pose 9
x	float	15	
y	float	15	
z	float	0	
roll	float	0	
pitch	float	0	
yaw	float	0	
wlan_box_...	str	wlan_box1	

Reactive Options

☒ Auto Battery Recharge

☒ Keep WLAN Range

☐ Manual Control

Figure 2.2 Overview of ROSMC's GUI.

A mission is composed in two phases: The first phase consists of the mission design, during which an operator can compile a formerly agreed upon mission which consist of a series of high-level skills, some of which were collaborative in nature. Afterward, the skills are synchronized to the respective robots, after which the second phase, the mission execution, could begin. Given the high operation uncertainty, the author recognized the need for flexible on-the-spot decisions and mission modifications, as well as the necessity of reactive behavior to safeguard the robot's autonomy, especially with regard to battery recharge operations. The specialized interface allows the scientists to supervise the mission execution, directly engage with the robots and make mission alterations based on their expertise, all without the need to handle a line of code. During preliminary user studies, however, the author described how users could experience frustration as skills failed, predominantly due to task preconditions (i.e. picking up a Wi-Fi box required LRU2 not to be carrying anything on its back) [12].

2.2 Advances in Robot Task Scheduling

Bischoff et. al. proposed automated planning algorithm was as part of the ARCHES project [2] [16]. It proposed calculating a preliminary solution using a greedy algorithm based partially on the foundational work of Zhang and Parker, who first proposed the concept of robot coalitions in 2013 to tackle a multi-robot task scheduling problem with their work, "Multi Robot Task Scheduling" [19], citing a lack of efficient algorithms in existence for multi-robot task allocation (MRTA). The authors proposed a formulation refined using an optimization heuristic based on a vehicle routing algorithm [14], a feasibility criterion and a local improvement heuristic stemming from an initial solution found by a greedy search using a cost function with both static and dynamic criteria. Afterwards, the authors would publish their findings on heuristic reoptimization for time-extended multi-robot task allocation problems [1], in which the problem of dynamic mission recalculation in real time is discussed. The cheapest maximum cost insertion heuristic, a modification of the cheapest insertion heuristic, is introduced.

In 2018, Roehr proposed a constraint-based mission planning approach for reconfigurable, modular multi-robot systems [5]. In this work, the author discusses the role of modularity and mssion flexibility on the system capacity to react to unforeseen circumstances. Yang et. al. also proposed a similar self-reactive planning algorithm with dynamic task assignments on 2019 [18]. In this case, the authors formalize the problem of dynamic task fulfillment using state transitions represented through a behavior tree. Although their formulation is worthy of mention, both proposals, however, are directed towards swarm-like systems of many similar agents performing simple tasks, instead of a couple of complex robots performing high-level skills.

It is no surprise that most development has been theoretical in nature, and applied research has been mainly developed for industrial and market-based settings. In these cases, a series of bidding algorithms for robot task allocation are often used, omitting the scheduling problem altogether [18]. To the best of our knowledge, however, no work about multi-robot scheduling tool with human-in-the-loop mission control has been implemented.

3 Theory

The following chapter contains theoretical framework of the thesis. It gives a brief overview of the main driving concepts behind its development, but is by no means an exhaustive list. It also provides an insight into the nomenclature used, and potentially provide some clarity with regards to terminology with different implications regardless of name similarity, and which should not be used interchangeably.

3.1 Embedded Artificial Intelligence

In its broadest sense, artificial intelligence (AI) concerns itself with rational decision making, i.e., the ability of an agent to act so as to attempt to achieve the best possible expected outcome with the information given [11]. The study of robotics concerns itself with the subset of artificial embodied rational agents, robots, which are capable of partially perceiving and interacting their environment, and sometimes, with other agents.

There are a multitude of cognitive models which, as the name implies, model cognitive processes such as perception, memory, action and, of course, reasoning, each with advantages and drawbacks. For example, Knowledge-based systems (KBS) are named as such because they solve complex programs using an explicitly represented knowledge base (KB) in addition to a inference engines. However, some notable drawbacks to symbolic inference include the difficulty in acquiring and maintaining a large knowledge base. Despite this, inference inevitably fails if it falls outside of the scope of its knowledge base, which makes exhaustive contingency planning a necessity, and even then cannot fully provide a fallback in case of an unforeseen event. Symbolic inference is very computationally expensive [3].

3.2 The Scheduling Problem

Classical automated planning (not to be confused with motion planning) is a combinatorial optimization problem defined as the task of finding a sequence of actions to accomplish a concrete goal [11]. The main advantage of automatic over manual planning is a reducing workload and, potentially, the ability to include longer task sequences and more complex constraints than could be comfortably handled by a human in a reasonable time.

3.2.1 State Space Search Notation

Space state search (not to be confused with state space representation from control theory) provides a suitable representation for agents performing a series of actions in an episodic, dynamic environments [11]. It represents a process in which successive system configurations are explored recursively with the intention of finding a desired goal state. Given an agent capable of a finite set of actions $A = \{x_1, x_2, ...x_m\}$, its state s_k while performing the k-eth action of a sequence S in a deterministic environment can be described by the tuple

$$s_k : \langle A(s_k), a_k, R(s_k, a_k), U(s_k, a_k) \rangle$$

where $A(s_k) \subseteq A$ is the set of actions reachable from s_k. Furthermore, a_k represents a chosen action for the state k, which then generates a result $R(s_k, a_k, s_k + 1)$. $U(s_k, a_k)$ can represent a heuristic value function mapping the utility from performing said action that is context-specific.

The properties of a heuristic value function heavily determine search strategy election and efficacy. For example, dynamic programming provides an elegant formulation to search problem solutions if the heuristic value function is concave, as greedy best-first search is guaranteed to find an optimal solution, if it exists. However, problems without an optimal sub-structure do not have this advantage. A greedy search is still useful in practice, however, if one desires to find a solution quickly, optimality notwithstanding.

3.2.2 The Travelling Salesman

The travelling Salesman Problem (TSP) is perhaps the most classical and well-researched, NP-hard combinatorial optimization problem, the study of which has served to develop and serve as a benchmark for optimization algorithms of computationally difficult problems [7]. The solution to this problem requires finding the shortest possible route to visit a series of given points (cities) exactly once, then returning to the home city. It was first formulated formally in 1930, although an algorithmic approach to find an optimal solution without calculating all possible sequences (brute-forcing) would not be found until decades later. The mathematical generalization of this problem has provided a base to formulate and resolve similar combinatorial optimization problems, including scheduling problems.

Branch and bound (BnB), a method first proposed in 1960, continues to be one of the most prominent used algorithms for solving TSP and other NP-hard problems [6]. As with dynamic programming, BnB works by breaking the problem down into smaller sub-problems. In this tree search, branches are expanded and then trimmed using a bounding function. Solutions are discarded if they are not able to perform better than the best one found so far. The efficacy of this method, however, is heavily reliant on a suitable heuristic and ability to find a bounding heuristic. In the worst case scenario, a BnB degenerates to an exhaustive search.

3.2.3 Action Description Language

Several attempts have been made in an attempt to standardize knowledge representation and reasoning for robot planning languages. Action Description Language (ADL), proposed in 1987, provided a way to encode robot actions, their respective parameters, effects on the robot and/or environment, as and their preconditions [10]. Although several variants exists, in its more general form, ADL syntaxis encodes the preconditions needed to perform a given action, as well as its effects. One of the main advantages of ADL format include reusability and human-legibility, and has been used as a starting point to develop current knowledge representation and reasoning formulations used today; for example, Planning Domain Definition Language (PDDL) [4].

3.3 Remarks on Optimality

In 1978, Herbert A. Simon would win the Nobel Prize in Economics for his interdisciplinary approach to operations research by contemplating elements of cognitive psychology in process optimization by recognizing the concept of bounded rationality. Simon recognized the practical importance of using a non-optimal solution as an alternative to utilizing excessive resources to calculate a mathematically pure optimum [11]. Instead, he proposed a feasibility boundary, within which solutions can be considered good enough for practical purposes.

4 Practical development

4.1 Overview

In paper, the goal of an automated planner is similar to the travelling salesman problem: to find a suitable task execution sequence, after which the robot should return to a goal configuration. The idea behind the algorithms execution was to provide a negotiation process, enabling the user to refine or modify the sequence until satisfied, aided by the technical feedback provided. We attempted to develop a process that could answer the following questions:

- Taking into account the robot's current state and future synchronized tasks in queue (if any), is a given task sequence feasible?
- If not, why? How could it be made feasible?
- Are there any alternatives that could improve the mission performance? How do they differ?
- Is the ongoing mission still feasible?
- If not, why and when could it fail?
- How can this be corrected?

Figure 4.1 shows an overview of the enhanced version of ROSMC developed as a result of this project. Initially, the program was conceived as a stand-alone analog to its predecessor, but as evidence mounted that the functionalities and dependencies between the two were far too intertwined, a decision was made to merge both projects.

4.2 Skill Set and Configuration

The skill set A of a robot includes the collection of its actionable high-level skills (RAFCON state machines) that have been fitted for scheduling compatibility. For this purpose, a configuration file loosely following ADL syntaxis was provided in order to add the necessary information for scheduling purposes. For each skill, a name corresponding with the desired state machine is provided, as well as the expected execution time in minutes, default parameters, preconditions and effects. An example for LRU2's skill take box skill is shown below:

take box: (Pick a collection box located in the given coordinates)

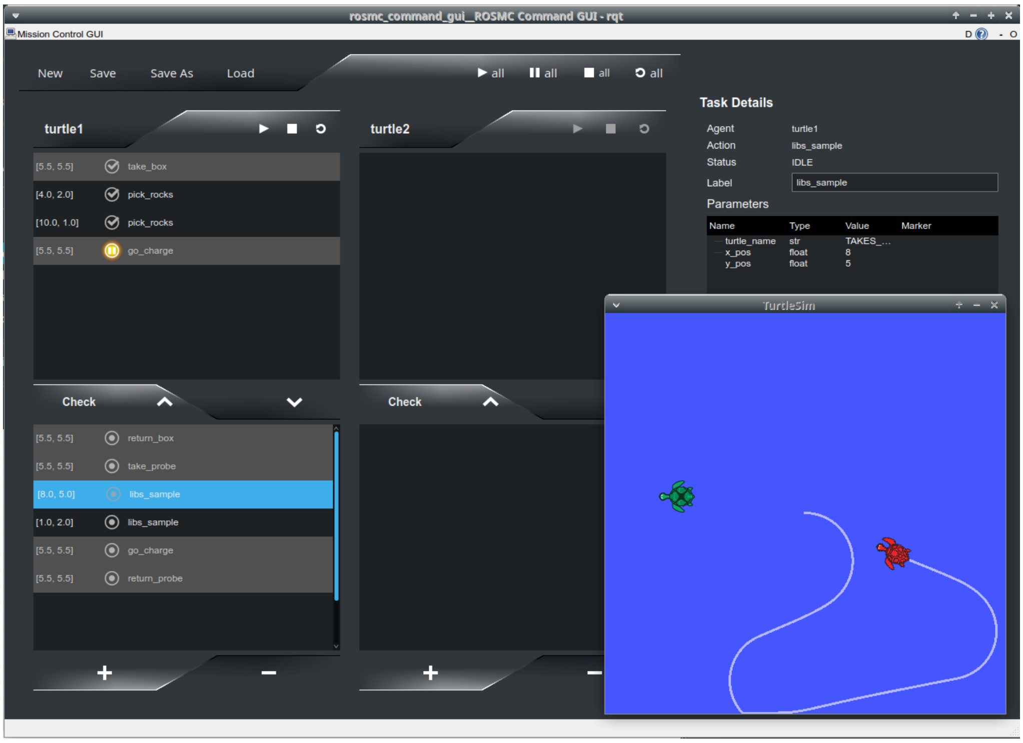
Execution time: 5
Default parameters:

- x: 0.0
- y: 0.0

Preconditions:

- has box: False
- has probe: False

Effects:



[5.5, 5.5]	✓	take_box
[4.0, 2.0]	✓	pick_rocks
[10.0, 1.0]	✓	pick_rocks
[5.5, 5.5]	❚❚	go_charge

[5.5, 5.5]	◉	return_box
[5.5, 5.5]	◉	take_probe
[8.0, 5.0]	◉	libs_sample
[1.0, 2.0]	◉	libs_sample
[5.5, 5.5]	◉	go_charge
[5.5, 5.5]	◉	return_probe

Task Details

Agent turtle1
Action libs_sample
Status IDLE
Label libs_sample

Parameters

Name	Type	Value	Marker
turtle_name	str	TAKES_...	
x_pos	float	8	
y_pos	float	5	

Figure 4.1 Screen capture of the enhanced version of ROSMC.

- Update x
- Update y
- has box = True
- Update battery

The full skill set description created for LRU2 can be found in Appendix A.1.

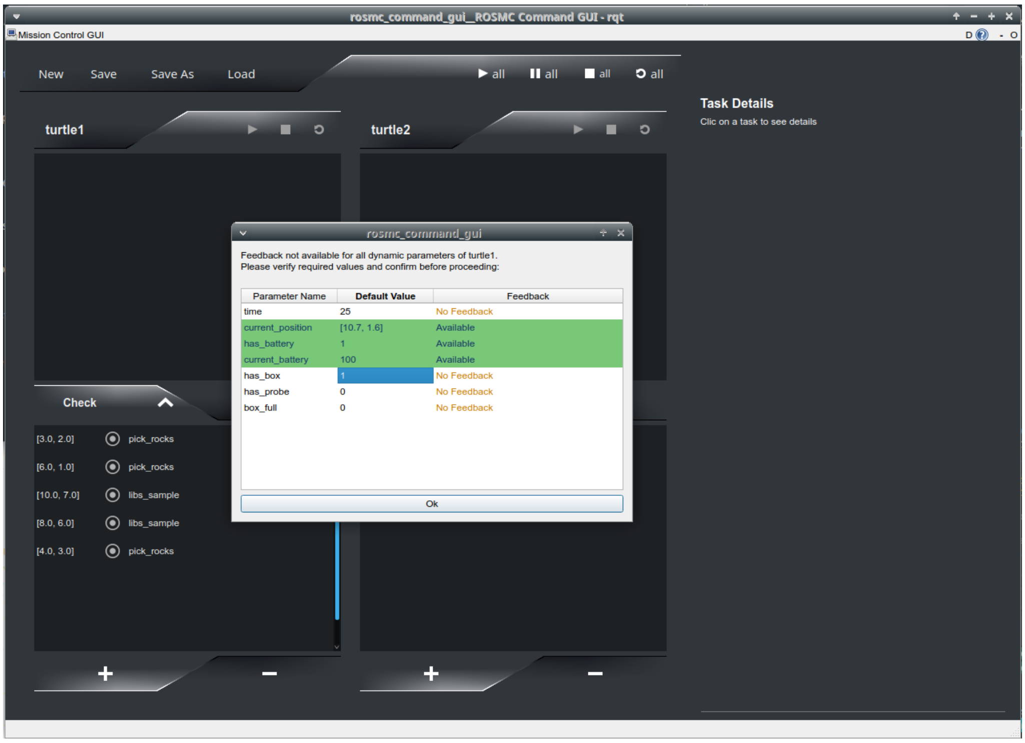
4.2.1 Tasks and Support Skills

The mission tasks are defined as the skill subset $Q \subset A$ containing an organized list of skills queried by the user with ROSMC. Any skill not contained in Q is considered a support skill. Because of how the state search is carried out, they should only be added to the mission sequence if they have a positive contribution to the net mission performance either in terms of performance criteria or by handling task prerequisites. The mechanics of how this is implemented will be explained in the following sections.

4.2.2 Skill Set Relationship to Configuration

A robot's configuration $X = \{x_1, x_2, ...x_n\}$ can be defined by the minimal set of parameters needed to specify all of the skills in their skill set plus the parameters used as performance indicators for the heuristic value function. In essence, the configuration is a reduced subset of a robot's world view. However, no actual connection to a robot's world model was established directly due to the project's limitations. Instead, an artificial divide between static and dynamic parameters was used depending on how they were handled internally given ROS architecture. Each robot's static parameters (presumed to remain constant during a mission execution) were imported once from the relevant ROS parameters during initialization, whereas information from dynamic parameters was obtained through the form of subscribers to the relevant ROS topic.

There were cases in which no feedback existed for some relevant parameters; particularly, LRU2 has no sensor to detect if it successfully equipped a collection box or libs measurment instrument on its back. To handle this, default values were provided and user verification was required at startup. Figure 4.2 shows a screen capture with the window used to for this purpose.



[3.0, 2.0]	pick_rocks
[6.0, 1.0]	pick_rocks
[10.0, 7.0]	libs_sample
[8.0, 6.0]	libs_sample
[4.0, 3.0]	pick_rocks

Feedback not available for all dynamic parameters of turtle1.
Please verify required values and confirm before proceeding:

Parameter Name	Default Value	Feedback
time	25	No Feedback
current_position	[10.7, 1.6]	Available
has_battery	1	Available
current_battery	100	Available
has_box	1	No Feedback
has_probe	0	No Feedback
box_full	0	No Feedback

Figure 4.2 Informative window opened after clicking the name for Turtle1.

4.3 State Representation

Having explained the skill set and parameters of a robot, the state search representation used to represent robot's configuration explicitly at a given state given by Eq. (4.1). The result term $R(s_k, a_k, X_k) = X_{k+1}$ updates the robot's configuration in accordance to the skill a_k effects discussed previously.

$$s_k : \langle X_k, A(s_k), a_k, R(s_k, a_k, X_k), U(s_k, a_k)\rangle \tag{4.1}$$

The set of reachable actions $A(s_k)$ of each state is constructed depending on the skill set A, the mission tasks queried Q, and how many of those tasks have already been scheduled.

Using a tree representation, the current robot configuration at state s_0 in the root node begins with $A(s_0) = A$. As children nodes are expanded and evaluated, those whose skill prerequisites are inconsistent with X_0 fail. If a node corresponding to a scheduled task is completed, this tasks is removed from the set of reachable actions and all subsequent states. For a support skill, however, A_s remains the same, implying any support skill can be utilized indefinitely. Figure 4.3 illustrates this behavior.

This implicit way of handling A_s was implemented considering the Markovian nature of state space search. If state transition was handled based on skill prerequisites alone, then support skills would need to be scheduled either immediately before or after the required task, which is already handled by RAFCON's hierarchical state machines. Instead, by handling these constraints implicitly, causal constraints can be resolved while considering the context of the whole mission. Although, in theory, a less restricted domain could cause the resulting tree branching factor to become exponentially large, in practice, this can be handled by a suitable search method and heuristic.

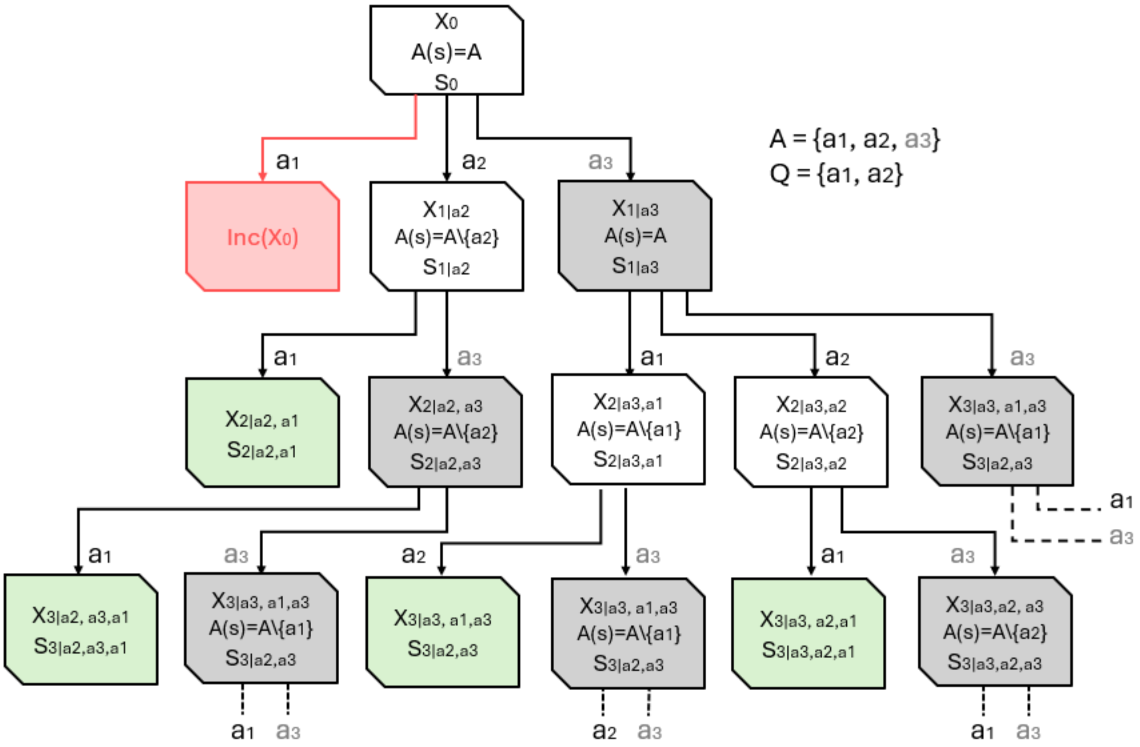


Figure 4.3 Example tree search for a set of three skills, two of which are queried as tasks. Also shown is the consequence of task a_1's incompatibility with the start parameters, rendering the expanded state node as failed.

The result generates support tasks creation handled without direct input from the user, and instead allowing for a suitable search algorithms to "fill in the gaps", which more closely resembles intuitive human communication in this regard. For example, even if it is not explicitly stated, most humans would consider it logical that a sample collection box must be procured before heading out to pick samples, that this must be carried out anytime before picking samples (but not necessarily just before). It also has the added benefit of providing flexibility if several combinations of support skills can provide the similar results.

4.4 Search Algorithms and Differential Analysis

The search strategy and posterior diagnostic will be detailed in this chapter. A tree search structure was implemented, with each successive node coinciding with an additional skill or task added to the sequence, as discussed before. A greedy best-first search strategy using a heuristic value function was used.

Some considerations worth mentioning: during an offline planning stage, the actual trajectory between two tasks is unknown, and so is the terrain. Instead, to provide an admissible heuristic, the Cartesian distance between the current and previous locations $d_{n-1,n}$ is assumed. The minimum possible driving time T_{travel} is calculated given the robot's average driving speed and autonomy time at full charge. For the case with LRU2, these parameters were obtained from its documentation; however, they could also be acquired experimentally if needed.

4.4.1 Heuristic Value Functions

The idea behind a value function was attempting to optimize a sequence according to non-system-specific relevant evaluation criteria, the most obvious of which would be total expected execution time. Furthermore, given the context of a battery-powered robot, the effect of available charge was taken into consideration as a reward modifier, functionally working as a pseudo-discount factor and creating a situation of diminishing return as the battery begins to deplete. Eventually, the battery depletion creates either a search finish condition, or the tipping point for a battery recharge operation, if it is available in the skill set. After some preliminary testing, Eq. (4.2) was chosen to calculate the utility for a state n given a skill a with an expected execution time of T_a, maximum autonomy time of T_{max} with a full charge, current battery percentage of c_n and minimum admissible battery percentage of c_{min}.

$$U_{(n,a)} = (\alpha\beta R_a) - P_a \tag{4.2}$$

$$\alpha = \begin{cases} 1 & a \in Q \\ 0 & a \notin Q \end{cases} \tag{4.3a}$$

$$\beta = (c_n - c_{min})^{1/4} \tag{4.3b}$$

$$R_a = T_a + T_{max} \tag{4.3c}$$

$$P_a = T_a + T_{travel} \tag{4.3d}$$

Modifier Terms

Eqns. (4.3) describe the binary reward activation term α, implying no reward is obtained for the selection of unscheduled support skills. As such, a search algorithm is unlikely to schedule support skills unless they contribute to the overall sequence feasibility. The battery dependant term β is constrained between 1 for a fully charged robot and 0 for a minimum admissible battery percentage. The formulation for the task reward R_a and penalty P_a have a normalizing factor over the maximum and minimum skill utility regardless of execution times. Figure 4.4 illustrates the relationship between utility value and battery charge for performing tasks with different expected execution times.

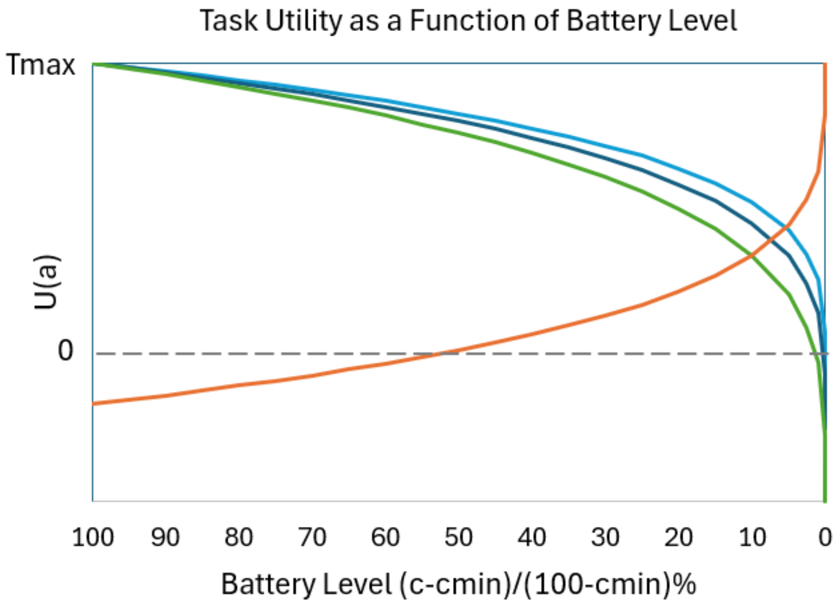


Figure 4.4 The utility as a function to effective battery charge $\frac{c_n - c_{min}}{100 - c_{min}}$ for three different tasks (blue). For all three, the maximum reward for a fully charged battery is equal to T_{max} and drops down to a value of $-T_{task}$ for $c_n = c_{min}$. The orange line describes the behavior of recharge operations and has the opposite behavior.

$$U_{(n,a)} = (\alpha(1-\beta)R_a) - P_a \tag{4.4}$$

Utility Variation for Battery Recharge Operations

During testing, the issue of correctly handling the possibility of programming a battery recharge operation was present. It was determined that a slight variation of the utility function for a recharge operation, in which the utility was inversely proportional to the battery charge percentage, solved more problems than it created. For this particular task,

$$U_{(n,recharge)} = (1-\beta)R_{recharge} - P_{recharge}$$

One issue stemming from this formulation is the possibility of the search failing to terminate if no solution exists, and the option to recharge the battery only adds to this problem. An arbitrary depth limit of three times the task sequence was added as a safeguard.

4.4.2 Failure Analysis

In the context of this problem, understanding why and where a search failed is almost as important as procuring a solution, if there is one. A search is terminated without a solution if there are no more nodes in the frontier. If this is the case, failure analysis from the children can be performed by considering only three failure causes are possible. The reasoning goes as follows:

For every child node:

- If the depth limit is reached, the node failed by reaching the depth limit.
- Otherwise, if the node failed due to low battery.
- Otherwise, the node must have failed because of inconsistent skill prerequisites.

Then, the following inference is made:

- If at least one child failed by reaching the depth limit, then the search failed inconclusively by reaching the depth limit.
- Otherwise, if at least one child failed due to battery, then the search failed due to low battery.
- Otherwise, all of the children failed because of inconsistent skill prerequisites, and thus the branch failed for that reason.

4.4.3 Offline, Planning Phase

The offline, planning phase encompasses the moments in which a user adds and modifies tasks before synchronizing them to the mission server. The process as a whole attempts to answer the following questions for the user:

- Taking into account the robot's current state and future synchronized tasks in queue (if any), is a given task sequence feasible?
- If not, why?
- How could it be made feasible, if possible?
- Are there any alternatives that could improve the mission performance? How do they differ?

"Exact" Sequence

The first search consists of evaluating the task query exactly as input by the user. If this search fails, the offending task in the sequence, as well as the failure reason, are extracted.

"Strict" Sequence

The second search proceeds assuming a strict precedence constraint with the assumption that queried tasks should be carried out in strict order, but allowing for padding with as many support skills as needed. If both solutions exist and are equal, then this serves to validate the efficiency of the original plan. If both solutions exist and differ, this can offer insights for a potential trade-off scenario in which a longer task sequence could offer a better overall solution (for example, if scheduling a preemptive recharging operation could make a mission less risky). If this search succeeds while the previous one failed due to inconsistency, then it essentially "fills in the holes" necessary to perform the original mission without altering the task sequence order. Failure of this search implies failure of the latter search.

"Greedy" Sequence

The final search ignores any precedence constraints and allows the use of support skills. This search has the potential of finding yet another different solution that can be considered, if it is different from the two previously generated sequences. Failure of this search implies failure of all searches.

Result Comparison and Presentation to the User

The combined results of the tree searches are used as a base to infer additional information. For example, a "Strict" search succeeding while an "Exact" search did not, implies the necessity of adding support skills to the original task sequence. The failure point and criteria of the "Exact" search directly explains what task failed and why, while the additional support skill padding provided by the "Strict" sequence provide insight as to how this can be fixed. Furthermore, if the "Greedy" search returns a result that is not identical to the latter, then at least one viable alternative exists. For each successful search, the user is presented with an overall performance in human-understandable terms: the total expected mission duration, total Cartesian distance traveled, and minimum battery level encountered. This overview helps differentiate solutions in trade-off scenarios.

Depending on the combined search results, a pop-up message with a summary of the results shows up, followed by a window that contains an overview of one to three search results (avoiding redundant or unnecessary information). Figure 4.5 contains a diagram with the main graphical elements of this window, while Figure 4.6 illustrates the decision diagram behind the content displayed for each of the eight possible scenarios, which are detailed in Table 4.7. Table 4.8 contains the possible messages corresponding to each failure scenario.

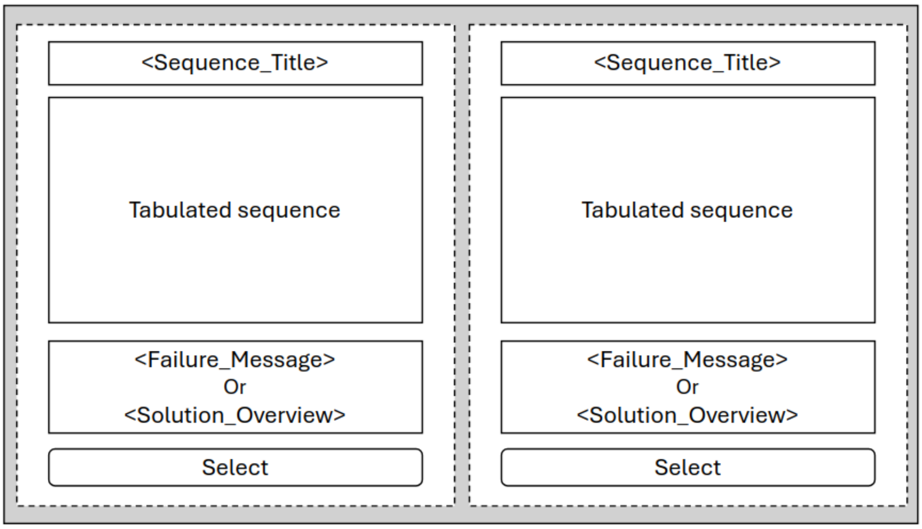


Figure 4.5 Schematic of the overview window to show two sequences.

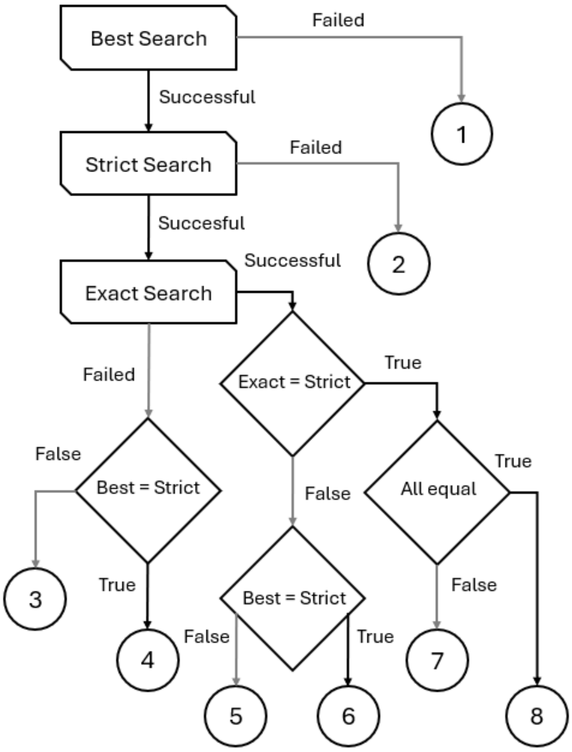
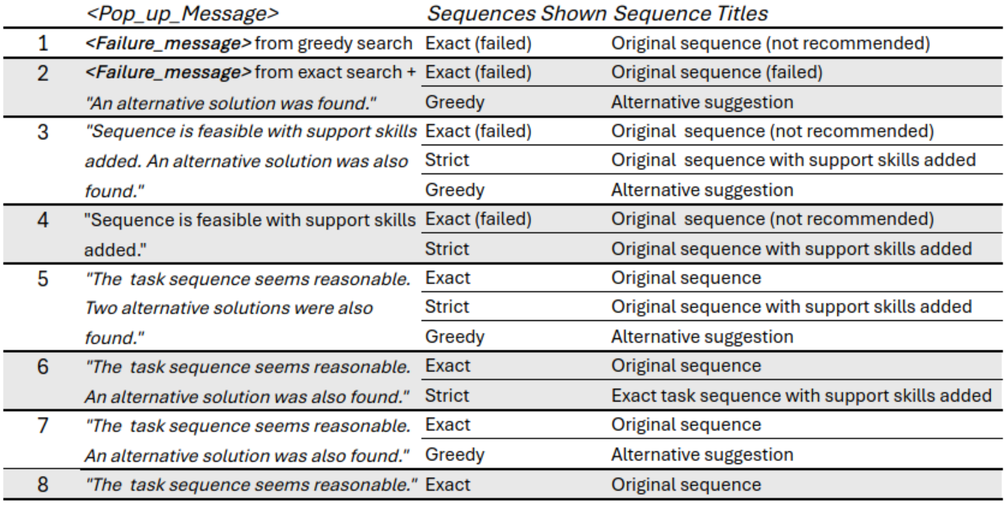


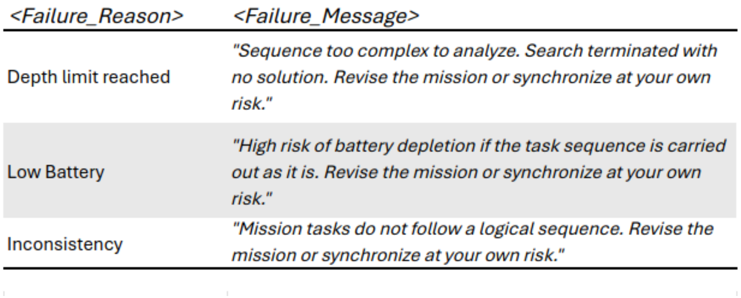
Figure 4.6 Decision diagram for result classification into one of eight possible scenarios.



	<Pop_up_Message>	*Sequences Shown*	*Sequence Titles*
1	***<Failure_message>*** from greedy search	Exact (failed)	Original sequence (not recommended)
2	***<Failure_message>*** from exact search + *"An alternative solution was found."*	Exact (failed)	Original sequence (failed)
		Greedy	Alternative suggestion
3	*"Sequence is feasible with support skills added. An alternative solution was also found."*	Exact (failed)	Original sequence (not recommended)
		Strict	Original sequence with support skills added
		Greedy	Alternative suggestion
4	"Sequence is feasible with support skills added."	Exact (failed)	Original sequence (not recommended)
		Strict	Original sequence with support skills added
5	*"The task sequence seems reasonable. Two alternative solutions were also found."*	Exact	Original sequence
		Strict	Original sequence with support skills added
		Greedy	Alternative suggestion
6	*"The task sequence seems reasonable. An alternative solution was also found."*	Exact	Original sequence
		Strict	Exact task sequence with support skills added
7	*"The task sequence seems reasonable. An alternative solution was also found."*	Exact	Original sequence
		Greedy	Alternative suggestion
8	*"The task sequence seems reasonable."*	Exact	Original sequence

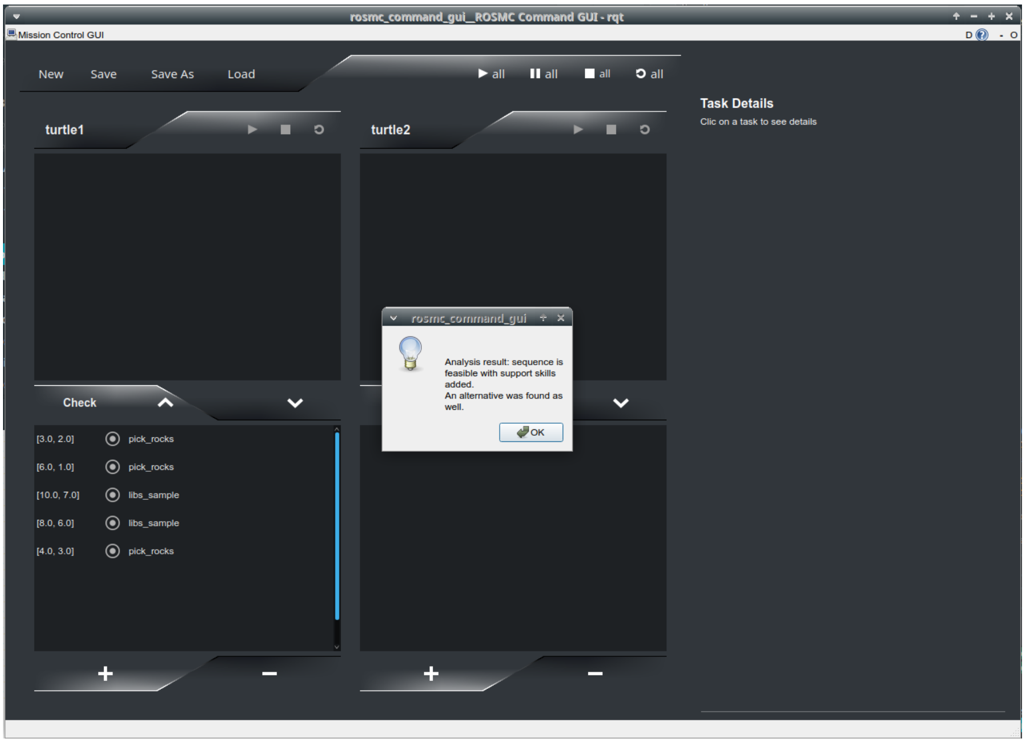
Figure 4.7 Table showcasing the content of the pop-up message and overview window according to the combined result. Depending on the case, one to three results are showcased and their details displayed.

Although it would be tempting to make an automated selection of the best sequence by a direct comparison of their utilities, it was decided that a providing the user with a maximum of three possible solutions from which to choose was a better choice, as the human operator could very well have additional information not contained in the robot's knowledge base, as they could have a specific reasoning behind a non-optimal execution as far as the robot is concerned. For example, they might be aware of important terrain considerations not taken into account in this model. Ultimately, the decision was made to provide the user with an informed suggestion they can choose to ignore. Finally, Figures 4.9 and 4.10 show screen captures of the final implementation.



<Failure_Reason>	*<Failure_Message>*
Depth limit reached	*"Sequence too complex to analyze. Search terminated with no solution. Revise the mission or synchronize at your own risk."*
Low Battery	*"High risk of battery depletion if the task sequence is carried out as it is. Revise the mission or synchronize at your own risk."*
Inconsistency	*"Mission tasks do not follow a logical sequence. Revise the mission or synchronize at your own risk."*

Figure 4.8 Informative message according to search failure reason.



Analysis result: sequence is feasible with support skills added.
An alternative was found as well.

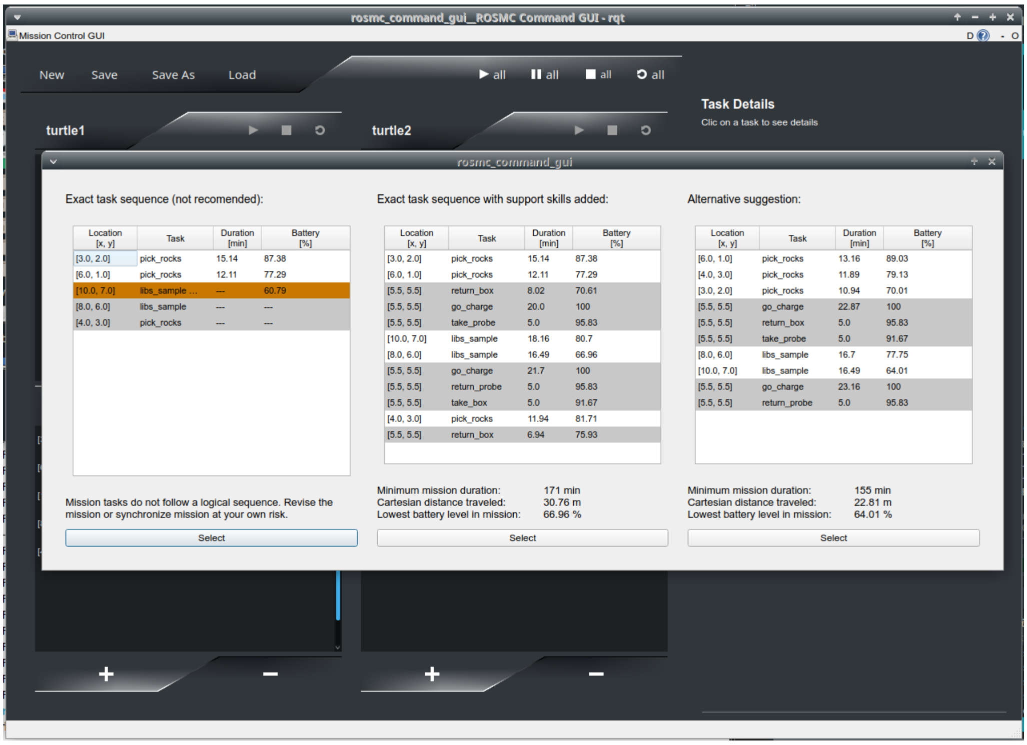
[3.0, 2.0]	pick_rocks
[6.0, 1.0]	pick_rocks
[10.0, 7.0]	libs_sample
[8.0, 6.0]	libs_sample
[4.0, 3.0]	pick_rocks

Figure 4.9 Screen capture of informative pop-up after an offline search.

After selection, the data in the mission server is automatically updated to reflect the user's selection, including a clear visual distinction between human-generated tasks and auto-generated support tasks, if any. The user is then free to continue editing the mission if they desire before synchronizing it to a robot.

4.4.4 Online, Reactive Phase

The reactive behavior is activated as soon as there are any synchronized tasks pending on the mission server. Its main functionality is to provide a quick reassessment of mission feasibility using updated parameter values as the mission progresses. Each time the reassessment is triggered, the updated robot parameters are used to account for deviations. For the time being, the search is triggered after a change to the synchronized mission is detected in the server, i.e. when a task fails, is completed successfully, or if a new task is synchronized or unsynchronized (Figure 4.11). If this search were to fail, this is communicated to the user with an informative pop-up (Figure 4.12), followed by an overview regarding the failure reason which the user can make use of to take corrective measures (Figure 4.13). For example, a task execution taking longer than usual, involving rough terrain or otherwise requiring more intensive battery consumption than was expected, and an additional recharge operation might be advised.



Task Details

Clic on a task to see details

Exact task sequence (not recomended):

Location [x, y]	Task	Duration [min]	Battery [%]
[3.0, 2.0]	pick_rocks	15.14	87.38
[6.0, 1.0]	pick_rocks	12.11	77.29
[10.0, 7.0]	libs_sample ...	---	60.79
[8.0, 6.0]	libs_sample	---	---
[4.0, 3.0]	pick_rocks	---	---

Mission tasks do not follow a logical sequence. Revise the mission or synchronize mission at your own risk.

Select

Exact task sequence with support skills added:

Location [x, y]	Task	Duration [min]	Battery [%]
[3.0, 2.0]	pick_rocks	15.14	87.38
[6.0, 1.0]	pick_rocks	12.11	77.29
[5.5, 5.5]	return_box	8.02	70.61
[5.5, 5.5]	go_charge	20.0	100
[5.5, 5.5]	take_probe	5.0	95.83
[10.0, 7.0]	libs_sample	18.16	80.7
[8.0, 6.0]	libs_sample	16.49	66.96
[5.5, 5.5]	go_charge	21.7	100
[5.5, 5.5]	return_probe	5.0	95.83
[5.5, 5.5]	take_box	5.0	91.67
[4.0, 3.0]	pick_rocks	11.94	81.71
[5.5, 5.5]	return_box	6.94	75.93

Minimum mission duration: 171 min
Cartesian distance traveled: 30.76 m
Lowest battery level in mission: 66.96 %

Select

Alternative suggestion:

Location [x, y]	Task	Duration [min]	Battery [%]
[6.0, 1.0]	pick_rocks	13.16	89.03
[4.0, 3.0]	pick_rocks	11.89	79.13
[3.0, 2.0]	pick_rocks	10.94	70.01
[5.5, 5.5]	go_charge	22.87	100
[5.5, 5.5]	return_box	5.0	95.83
[5.5, 5.5]	take_probe	5.0	91.67
[8.0, 6.0]	libs_sample	16.7	77.75
[10.0, 7.0]	libs_sample	16.49	64.01
[5.5, 5.5]	go_charge	23.16	100
[5.5, 5.5]	return_probe	5.0	95.83

Minimum mission duration: 155 min
Cartesian distance traveled: 22.81 m
Lowest battery level in mission: 64.01 %

Select

Figure 4.10 Screen capture of assessment overview after an offline search.

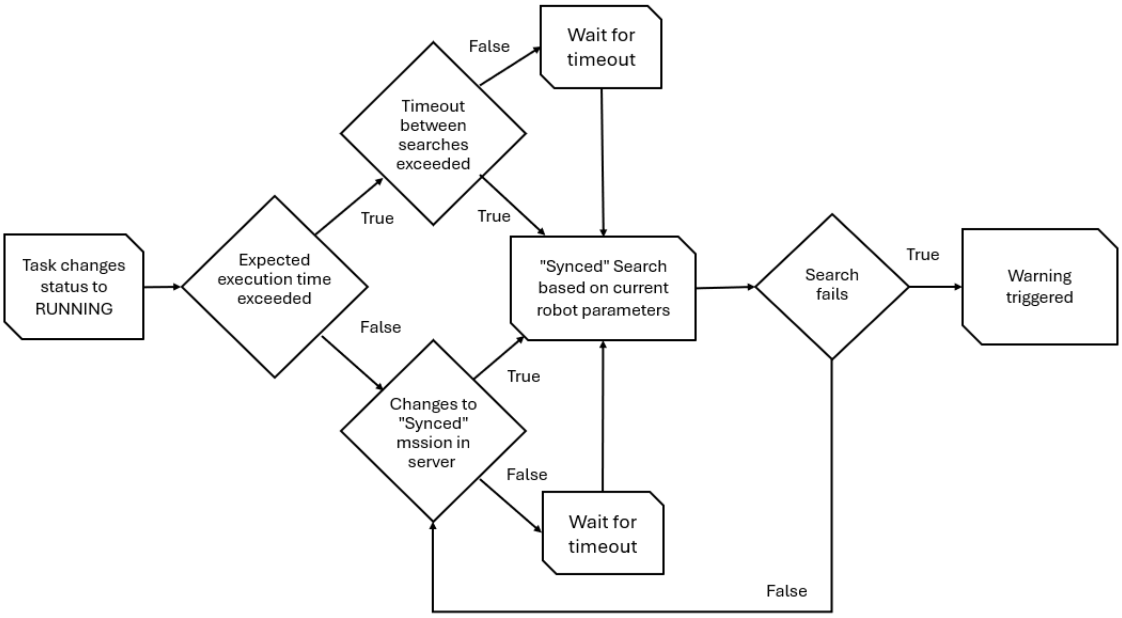
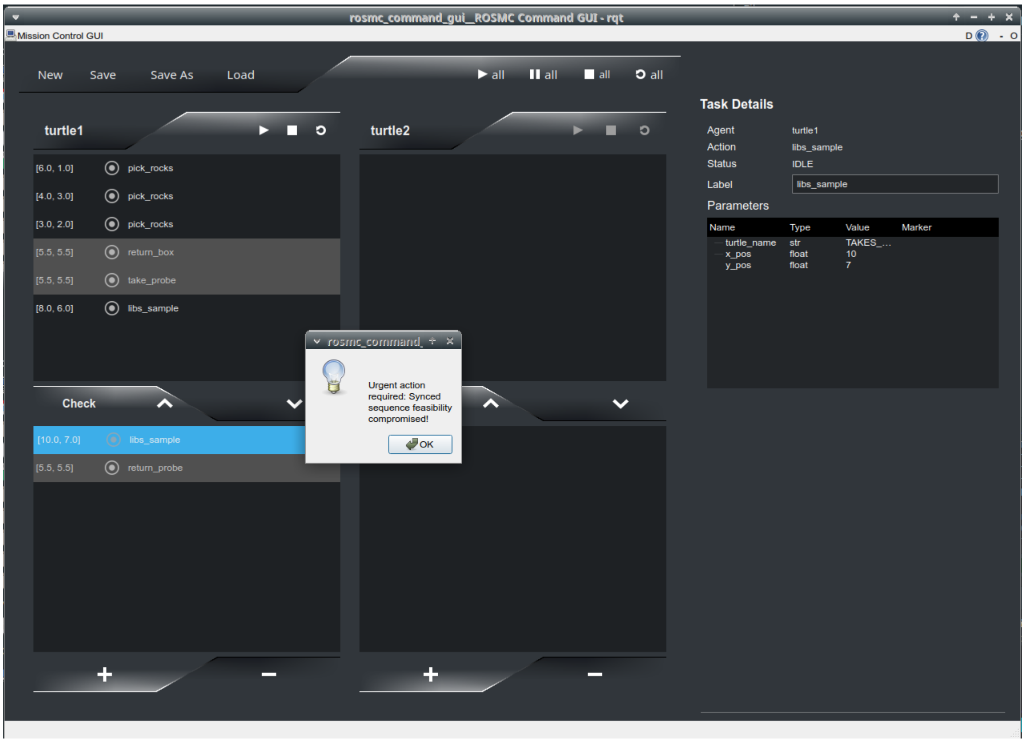

Wait for timeout
False
Timeout between searches exceeded
True
True
Task changes status to RUNNING
Expected execution time exceeded
"Synced" Search based on current robot paremeters
Search fails
True
Warning triggered
False
True
Changes to "Synced" mssion in server
False
Wait for timeout
False

Figure 4.11 Decision process behind the online search execution and warning trigger

On-the-fly Mission Modificatons

A secondary function of the reactive behavior is calculating the expected starting parameters for the offline planner in case appending new tasks is desired when a mission is already underway, thus preserving the flexible nature of ROSMC and the ability to perform on-the-fly mission changes.



[6.0, 1.0]	pick_rocks
[4.0, 3.0]	pick_rocks
[3.0, 2.0]	pick_rocks
[5.5, 5.5]	return_box
[5.5, 5.5]	take_probe
[8.0, 6.0]	libs_sample

Check

[10.0, 7.0]	libs_sample
[5.5, 5.5]	return_probe

Task Details

Agent	turtle1
Action	libs_sample
Status	IDLE
Label	libs_sample

Parameters

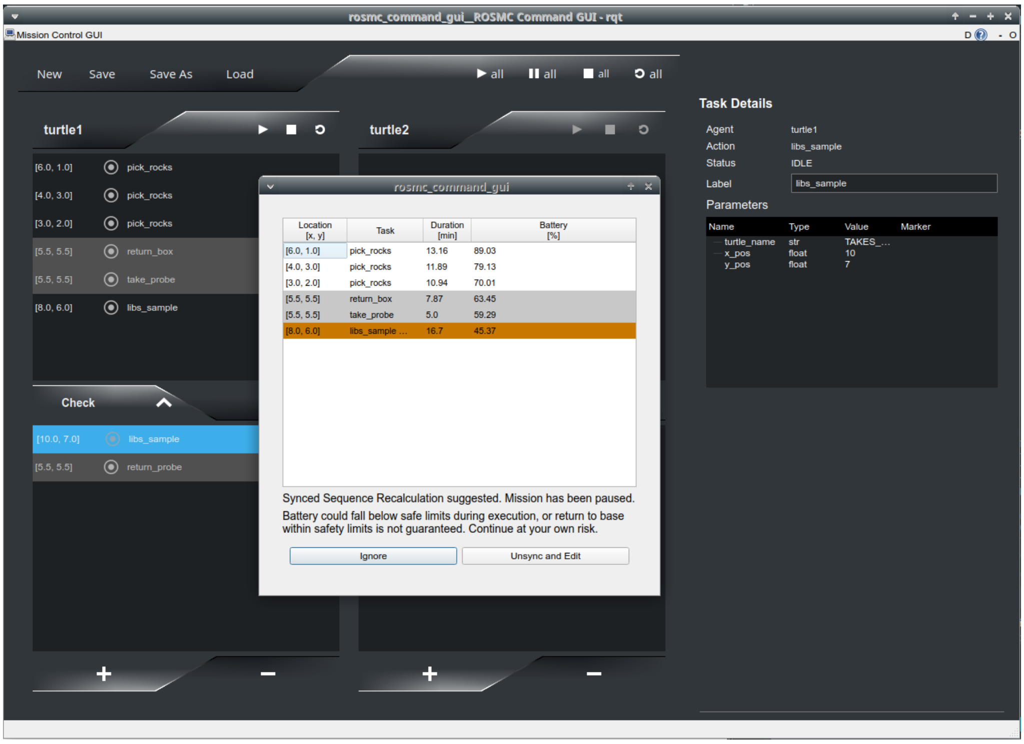
Name	Type	Value	Marker
turtle_name	str	TAKES_...	
x_pos	float	10	
y_pos	float	7	

rosmc_command_

Urgent action required: Synced sequence feasibility compromised!

OK

Figure 4.12 Screen capture of informative pop-up after a failed search.



[6.0, 1.0]	pick_rocks
[4.0, 3.0]	pick_rocks
[3.0, 2.0]	pick_rocks
[5.5, 5.5]	return_box
[5.5, 5.5]	take_probe
[8.0, 6.0]	libs_sample

Check

[10.0, 7.0]	libs_sample
[5.5, 5.5]	return_probe

Task Details

Agent	turtle1
Action	libs_sample
Status	IDLE
Label	libs_sample

Parameters

Name	Type	Value	Marker
turtle_name	str	TAKES_...	
x_pos	float	10	
y_pos	float	7	

rosmc_command_gui

Location [x, y]	Task	Duration [min]	Battery [%]
[6.0, 1.0]	pick_rocks	13.16	89.03
[4.0, 3.0]	pick_rocks	11.89	79.13
[3.0, 2.0]	pick_rocks	10.94	70.01
[5.5, 5.5]	return_box	7.87	63.45
[5.5, 5.5]	take_probe	5.0	59.29
[8.0, 6.0]	libs_sample ...	16.7	45.37

Synced Sequence Recalculation suggested. Mission has been paused.
Battery could fall below safe limits during execution, or return to base within safety limits is not guaranteed. Continue at your own risk.

Ignore | Unsync and Edit

Figure 4.13 Screen capture of assessment overview after a failed search.

Discussion

Several discussions were had regarding the scope of the online search. At first, we considered augmenting the search to behave similarly to the offline case and enhance its ability to provide suggestions, or even offer to modify the entire synchronized mission if a better solution was detected. However, the idea of altering an already synchronized mission was ultimately dropped because mission robustness could be affected, especially when considering contexts with significant communication delay. In any case, the current implementation allows the user to un-synchronize the mission and use the offline search as an alternative. This also keeps the reactive search as agile as possible and avoids using unnecessary computational resources during mission execution.

5 Testing and Validation

The development of both our algorithm and program were highly iterative in nature, and were refined, in part, by the use of judicious trial and error. The task sequence of the Arches Mission in Mt. Etna was used as a benchmark to validate the algorithm performance and behavior using data from the LRU2 robot. During the actual program development, several test missions of increasing difficulty were devised to test, not only specific capabilities, but prototype the overall interaction process between the user, the GUI and code. In this case, ROS turtlesim was used to emulate a simplified version of LRU2 for deveopment simplicity and agility. Finally, the program was finally deployed for its execution with the multirobot system.

5.1 Algorithm validation using Data from the Arches Mission

The performance of the search algorithm was first validated using the mission data for the Arches Mission in Mt. Etna and comparing the search results with the original sequence. The tasks were first randomised to avoid skewing the results towards the original sequence. This process was repeated a total of five times.

5.1.1 Description of the Test Scenario

The lander, situated at [0, 0], contains sample boxes and instruments that can be retrieved from and returned to. We assume battery recharge operations also take place at the lander.

Skill set: pick_rocks, libs_sample, take_box, take_probe, return_box, return_probe, go_charge, move_to_position

Static Parameters:

- Driving speed (meters/minute): 60
- Battery duration (minutes): 120
- Minimum battery percentage: 50

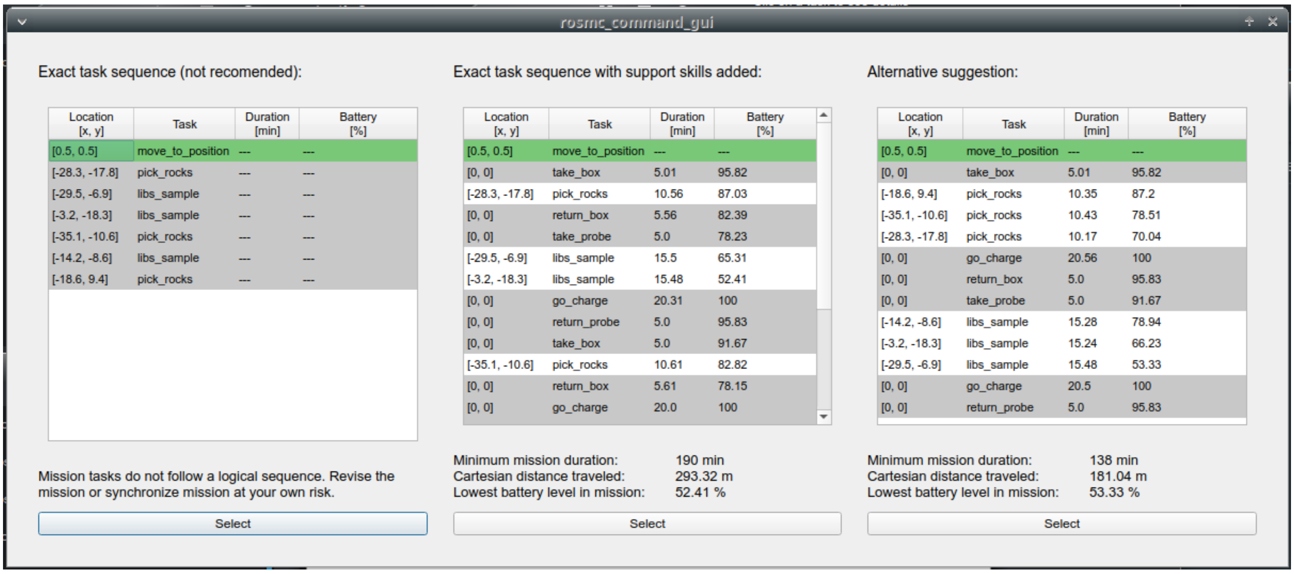
Dynamic Parameters with Initial Values:

- x: 0
- y: 0
- yaw: 0
- time: 0
- battery_percentage: 100
- has_box: False
- has_probe: False

5.1.2 Validation Results

The result overview window as a result of one of the searches can be found in Figure 5.1. Although predictably, the results from the Exact searches were all different according to the randomization, all greedy searches converged to the same result, which is similar but not identical to the original mission sequence.

Figure 5.2 shows the results after analyzing the original sequence, which allows the discrepancies to be seen: due to the greedy nature of our search, the libs_sample task located at [-14.2, -8.6] was favored first because it was the closest to the lander after the skill take_probe was scheduled. The result is a compounded travel distance that is roughly equivalent with regard to battery management and duration, but 11.29 meters longer. When put into context, however, a performance drop of less than 10 percent is still negligible when compared to the uncertainties present during trajectory planning. We consider this trade-off is more than acceptable if we consider the very few seconds required to compute a solution using our algorithm, which could potentially save precious time during a real mission.



Exact task sequence (not recomended):

Location [x, y]	Task	Duration [min]	Battery [%]
[0.5, 0.5]	move_to_position	---	---
[-28.3, -17.8]	pick_rocks	---	---
[-29.5, -6.9]	libs_sample	---	---
[-3.2, -18.3]	libs_sample	---	---
[-35.1, -10.6]	pick_rocks	---	---
[-14.2, -8.6]	libs_sample	---	---
[-18.6, 9.4]	pick_rocks	---	---

Mission tasks do not follow a logical sequence. Revise the mission or synchronize mission at your own risk.

Exact task sequence with support skills added:

Location [x, y]	Task	Duration [min]	Battery [%]
[0.5, 0.5]	move_to_position	---	---
[0, 0]	take_box	5.01	95.82
[-28.3, -17.8]	pick_rocks	10.56	87.03
[0, 0]	return_box	5.56	82.39
[0, 0]	take_probe	5.0	78.23
[-29.5, -6.9]	libs_sample	15.5	65.31
[-3.2, -18.3]	libs_sample	15.48	52.41
[0, 0]	go_charge	20.31	100
[0, 0]	return_probe	5.0	95.83
[0, 0]	take_box	5.0	91.67
[-35.1, -10.6]	pick_rocks	10.61	82.82
[0, 0]	return_box	5.61	78.15
[0, 0]	go_charge	20.0	100

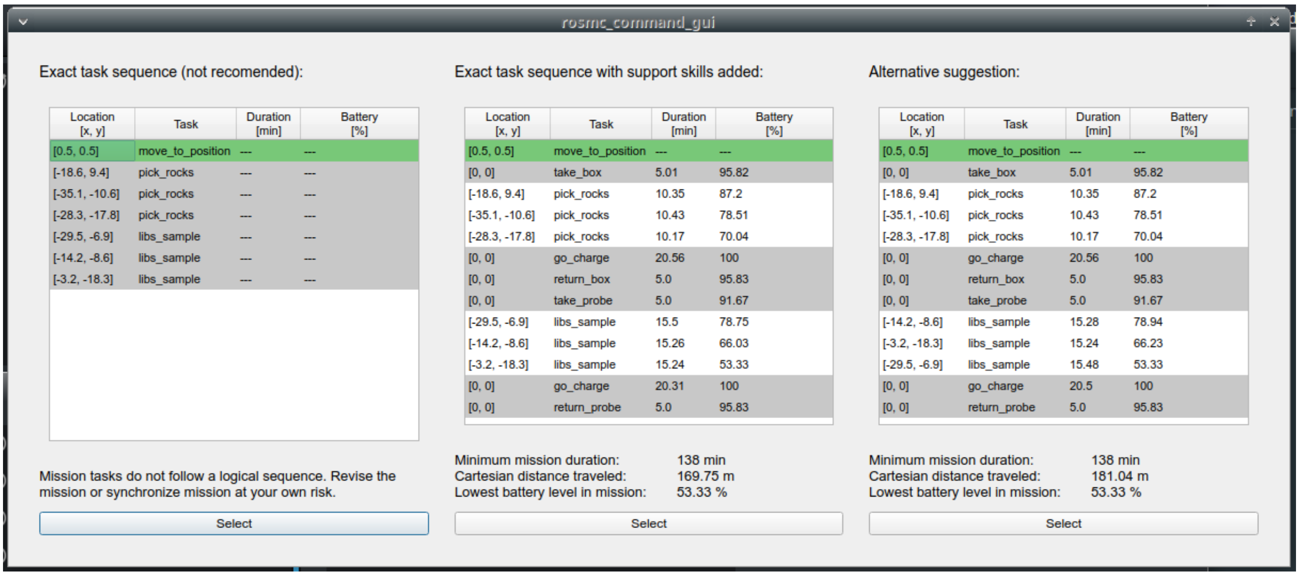
Minimum mission duration: 190 min
Cartesian distance traveled: 293.32 m
Lowest battery level in mission: 52.41 %

Alternative suggestion:

Location [x, y]	Task	Duration [min]	Battery [%]
[0.5, 0.5]	move_to_position	---	---
[0, 0]	take_box	5.01	95.82
[-18.6, 9.4]	pick_rocks	10.35	87.2
[-35.1, -10.6]	pick_rocks	10.43	78.51
[-28.3, -17.8]	pick_rocks	10.17	70.04
[0, 0]	go_charge	20.56	100
[0, 0]	return_box	5.0	95.83
[0, 0]	take_probe	5.0	91.67
[-14.2, -8.6]	libs_sample	15.28	78.94
[-3.2, -18.3]	libs_sample	15.24	66.23
[-29.5, -6.9]	libs_sample	15.48	53.33
[0, 0]	go_charge	20.5	100
[0, 0]	return_probe	5.0	95.83

Minimum mission duration: 138 min
Cartesian distance traveled: 181.04 m
Lowest battery level in mission: 53.33 %

Figure 5.1 Overview window displaying results using randomized tasks from the Arches mission. The green task corresponds to a previously executed task meant to initialize the robot-turtle near the lander.



Exact task sequence (not recomended):

Location [x, y]	Task	Duration [min]	Battery [%]
[0.5, 0.5]	move_to_position	---	---
[-18.6, 9.4]	pick_rocks	---	---
[-35.1, -10.6]	pick_rocks	---	---
[-28.3, -17.8]	pick_rocks	---	---
[-29.5, -6.9]	libs_sample	---	---
[-14.2, -8.6]	libs_sample	---	---
[-3.2, -18.3]	libs_sample	---	---

Mission tasks do not follow a logical sequence. Revise the mission or synchronize mission at your own risk.

Exact task sequence with support skills added:

Location [x, y]	Task	Duration [min]	Battery [%]
[0.5, 0.5]	move_to_position	---	---
[0, 0]	take_box	5.01	95.82
[-18.6, 9.4]	pick_rocks	10.35	87.2
[-35.1, -10.6]	pick_rocks	10.43	78.51
[-28.3, -17.8]	pick_rocks	10.17	70.04
[0, 0]	go_charge	20.56	100
[0, 0]	return_box	5.0	95.83
[0, 0]	take_probe	5.0	91.67
[-29.5, -6.9]	libs_sample	15.5	78.75
[-14.2, -8.6]	libs_sample	15.26	66.03
[-3.2, -18.3]	libs_sample	15.24	53.33
[0, 0]	go_charge	20.31	100
[0, 0]	return_probe	5.0	95.83

Minimum mission duration: 138 min
Cartesian distance traveled: 169.75 m
Lowest battery level in mission: 53.33 %

Alternative suggestion:

Location [x, y]	Task	Duration [min]	Battery [%]
[0.5, 0.5]	move_to_position	---	---
[0, 0]	take_box	5.01	95.82
[-18.6, 9.4]	pick_rocks	10.35	87.2
[-35.1, -10.6]	pick_rocks	10.43	78.51
[-28.3, -17.8]	pick_rocks	10.17	70.04
[0, 0]	go_charge	20.56	100
[0, 0]	return_box	5.0	95.83
[0, 0]	take_probe	5.0	91.67
[-14.2, -8.6]	libs_sample	15.28	78.94
[-3.2, -18.3]	libs_sample	15.24	66.23
[-29.5, -6.9]	libs_sample	15.48	53.33
[0, 0]	go_charge	20.5	100
[0, 0]	return_probe	5.0	95.83

Minimum mission duration: 138 min
Cartesian distance traveled: 181.04 m
Lowest battery level in mission: 53.33 %

Figure 5.2 Overview window displaying results using the exact task sequence used during the Arches mission. The Greedy search in this scenario converges to the same result.

5.2 Interactive-Reactive Mission Control

The following series of tests impose increasingly difficult search scenarios for the automated scheduler to solve. Rather than presenting several unrelated search queries, the next four search scenarios happen sequentially, using the output of the latter as a starting point for the next. We decided this would illustrate the capabilities and intended use of the program, in which a flexible, back-and-forth mission refinement between human and computer can take place.

5.2.1 Description of the Test Scenario

The simulation scenario created for ROSMC is constrained in a window measuring 11x11 units. Therefore, task coordinates and driving speed were reduced accordingly. Driving speed was also reduced to reflect this. The spawning coordinates of [5.5, 5.5] were used as the default location of the hypothetical lander from which sample boxes and instruments can be retrieved, returned, and the battery recharge operations can take place.

Static Parameters:

- Driving speed (meters/minute): 1
- Battery duration (minutes): 120
- Minimum battery percentage: 50

Dynamic Parameters with Initial Values:

- x: 5.5
- y: 5.5
- yaw: 0
- time: 0
- battery_percentage: 100
- has_box: False
- has_probe: False

5.2.2 Test 1: Filling in the gaps and refining an Inefficient Task Sequence

In this initial scenario, an inefficient sequence of three pick_rock tasks was created.

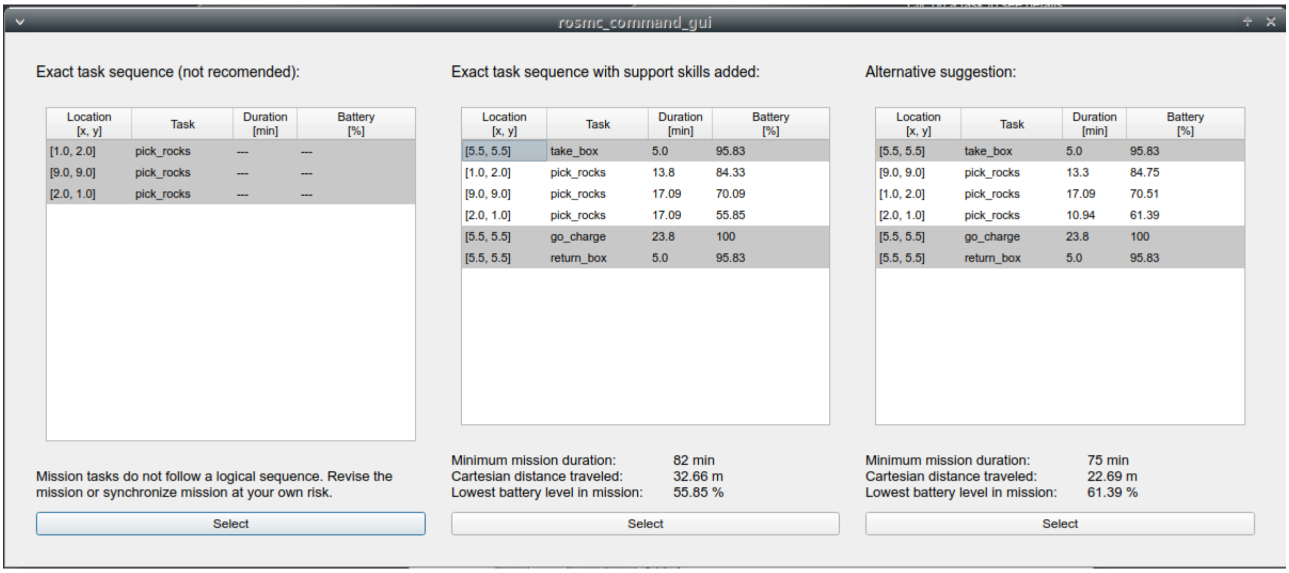
Expected outcome

Exact sequence failure due to battery when charge drops below 50 percent. Strict sequence result starting with pick_box support task, with occasional go_recharge operations in between, and ending in place_box. Greedy sequence generating a more efficient task sequence that Strict result in terms of execution time and travel speed.

Test Result

The expected outcome can be seen in Figure 5.3. Predictably, the first task selected by the greedy algorithm was the one closest to the starting location.

The greedy search trees generated for this query can be seen in Figure 5.4, where the effect of the reward modifiers can be seen: task rewards decrease as the battery is drained, and how that contributes to go_recharge skill scheduling. The heuristic also proved efficient at avoiding exponential branching.



Exact task sequence (not recomended):

Location [x, y]	Task	Duration [min]	Battery [%]
[1.0, 2.0]	pick_rocks	---	---
[9.0, 9.0]	pick_rocks	---	---
[2.0, 1.0]	pick_rocks	---	---

Mission tasks do not follow a logical sequence. Revise the mission or synchronize mission at your own risk.

Exact task sequence with support skills added:

Location [x, y]	Task	Duration [min]	Battery [%]
[5.5, 5.5]	take_box	5.0	95.83
[1.0, 2.0]	pick_rocks	13.8	84.33
[9.0, 9.0]	pick_rocks	17.09	70.09
[2.0, 1.0]	pick_rocks	17.09	55.85
[5.5, 5.5]	go_charge	23.8	100
[5.5, 5.5]	return_box	5.0	95.83

Minimum mission duration:	82 min
Cartesian distance traveled:	32.66 m
Lowest battery level in mission:	55.85 %

Alternative suggestion:

Location [x, y]	Task	Duration [min]	Battery [%]
[5.5, 5.5]	take_box	5.0	95.83
[9.0, 9.0]	pick_rocks	13.3	84.75
[1.0, 2.0]	pick_rocks	17.09	70.51
[2.0, 1.0]	pick_rocks	10.94	61.39
[5.5, 5.5]	go_charge	23.8	100
[5.5, 5.5]	return_box	5.0	95.83

Minimum mission duration:	75 min
Cartesian distance traveled:	22.69 m
Lowest battery level in mission:	61.39 %

Figure 5.3 Overview window displaying results for the first scenario.

5.2.3 Test 2: Mission with Mixed Tasks and Prerequisite Constraints

Using the greedy result of the previous query as a base, two additional libs_sample tasks were created and shuffled between the pick_rock tasks, generating an alternating sequence.

Expected outcome

Exact sequence failure due to task inconsistencies Strict sequence result generating support skills to pick up a collection box or LIBS probe when appropriate, with occasional go_recharge operations in between, or failing if depth limit is reached. Greedy sequence result grouping similar tasks together

Test Result

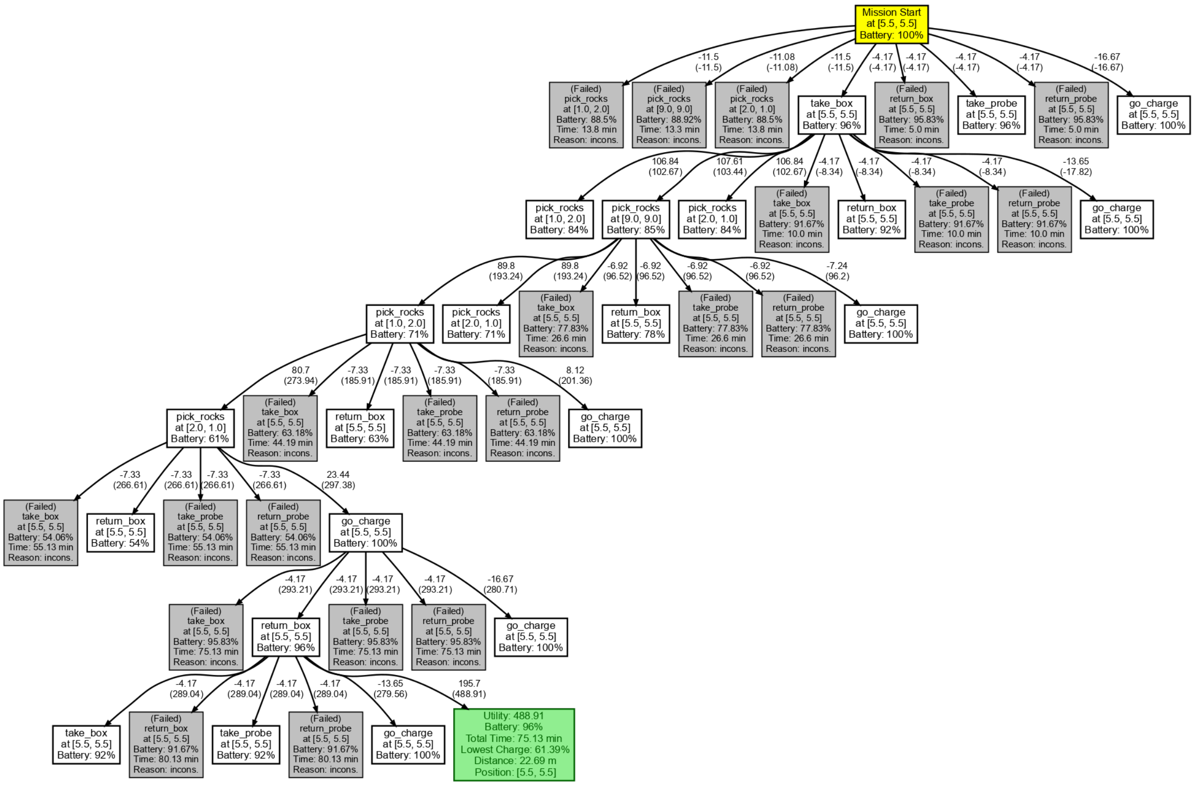
The results shown in Figure 5.5 showcase the search algorithm's ability to produce the a solution grouping tasks with similar prerequisites together. Furthermore, this successfully tested the iterative nature of the program, as the result from a previous query can be elaborated or modified if a user decides to do so. The auto-generated support skills created in the context of the previous mission are ignored in the new search.

5.2.4 Test 3: Offline Planning with Synchronized Tasks in Server

Once again, the greedy sequence result is selected and the first four tasks are synchronized. Afterwards, one more pick_rock and one libs_sample tasks will be created before searching.

Expected outcome

The synchronized tasks should show up in all three sequences Strict solution should schedule return_box and take_probe skills before executing the firs tlibs_sample task. Greedy sequence should schedule the unsynced pick_rock task nearest to the location of the synced task first, and should not schedule a take_box task before it.



Mission Start at [5.5, 5.5] Battery: 100%
-11.5 (-11.5)
-11.08 (-11.08)
-11.5 (-11.5)
-4.17 (-4.17)
-4.17 (-4.17)
-4.17 (-4.17)
-4.17 (-4.17)
-16.67 (-16.67)
(Failed) pick_rocks at [1.0, 2.0] Battery: 88.5% Time: 13.8 min Reason: incons.
(Failed) pick_rocks at [9.0, 9.0] Battery: 88.92% Time: 13.3 min Reason: incons.
(Failed) pick_rocks at [2.0, 1.0] Battery: 88.5% Time: 13.8 min Reason: incons.
take_box at [5.5, 5.5] Battery: 96%
(Failed) return_box at [5.5, 5.5] Battery: 95.83% Time: 5.0 min Reason: incons.
take_probe at [5.5, 5.5] Battery: 96%
(Failed) return_probe at [5.5, 5.5] Battery: 95.83% Time: 5.0 min Reason: incons.
go_charge at [5.5, 5.5] Battery: 100%
106.84 (102.67)
107.61 (103.44)
106.84 (102.67)
-4.17 (-8.34)
-4.17 (-8.34)
-4.17 (-8.34)
-4.17 (-8.34)
-13.65 (-17.82)
pick_rocks at [1.0, 2.0] Battery: 84%
pick_rocks at [9.0, 9.0] Battery: 85%
pick_rocks at [2.0, 1.0] Battery: 84%
(Failed) take_box at [5.5, 5.5] Battery: 91.67% Time: 10.0 min Reason: incons.
return_box at [5.5, 5.5] Battery: 92%
(Failed) take_probe at [5.5, 5.5] Battery: 91.67% Time: 10.0 min Reason: incons.
(Failed) return_probe at [5.5, 5.5] Battery: 91.67% Time: 10.0 min Reason: incons.
go_charge at [5.5, 5.5] Battery: 100%
89.8 (193.24)
89.8 (193.24)
-6.92 (96.52)
-6.92 (96.52)
-6.92 (96.52)
-6.92 (96.52)
-7.24 (96.2)
pick_rocks at [1.0, 2.0] Battery: 71%
pick_rocks at [2.0, 1.0] Battery: 71%
(Failed) take_box at [5.5, 5.5] Battery: 77.83% Time: 26.6 min Reason: incons.
return_box at [5.5, 5.5] Battery: 78%
(Failed) take_probe at [5.5, 5.5] Battery: 77.83% Time: 26.6 min Reason: incons.
(Failed) return_probe at [5.5, 5.5] Battery: 77.83% Time: 26.6 min Reason: incons.
go_charge at [5.5, 5.5] Battery: 100%
80.7 (273.94)
-7.33 (185.91)
-7.33 (185.91)
-7.33 (185.91)
-7.33 (185.91)
8.12 (201.36)
pick_rocks at [2.0, 1.0] Battery: 61%
(Failed) take_box at [5.5, 5.5] Battery: 63.18% Time: 44.19 min Reason: incons.
return_box at [5.5, 5.5] Battery: 63%
(Failed) take_probe at [5.5, 5.5] Battery: 63.18% Time: 44.19 min Reason: incons.
(Failed) return_probe at [5.5, 5.5] Battery: 63.18% Time: 44.19 min Reason: incons.
go_charge at [5.5, 5.5] Battery: 100%
-7.33 (266.61)
-7.33 (266.61)
-7.33 (266.61)
-7.33 (266.61)
23.44 (297.38)
(Failed) take_box at [5.5, 5.5] Battery: 54.06% Time: 55.13 min Reason: incons.
return_box at [5.5, 5.5] Battery: 54%
(Failed) take_probe at [5.5, 5.5] Battery: 54.06% Time: 55.13 min Reason: incons.
(Failed) return_probe at [5.5, 5.5] Battery: 54.06% Time: 55.13 min Reason: incons.
go_charge at [5.5, 5.5] Battery: 100%
-4.17 (293.21)
-4.17 (293.21)
-4.17 (293.21)
-4.17 (293.21)
-16.67 (280.71)
(Failed) take_box at [5.5, 5.5] Battery: 95.83% Time: 75.13 min Reason: incons.
return_box at [5.5, 5.5] Battery: 96%
(Failed) take_probe at [5.5, 5.5] Battery: 95.83% Time: 75.13 min Reason: incons.
(Failed) return_probe at [5.5, 5.5] Battery: 95.83% Time: 75.13 min Reason: incons.
go_charge at [5.5, 5.5] Battery: 100%
-4.17 (289.04)
-4.17 (289.04)
-4.17 (289.04)
-4.17 (289.04)
-13.65 (279.56)
195.7 (488.91)
take_box at [5.5, 5.5] Battery: 92%
(Failed) return_box at [5.5, 5.5] Battery: 91.67% Time: 80.13 min Reason: incons.
take_probe at [5.5, 5.5] Battery: 92%
(Failed) return_probe at [5.5, 5.5] Battery: 91.67% Time: 80.13 min Reason: incons.
go_charge at [5.5, 5.5] Battery: 100%
Utility: 488.91 Battery: 96% Total Time: 75.13 min Lowest Charge: 61.39% Distance: 22.69 m Position: [5.5, 5.5]

Figure 5.4 Greedy tree generated for scenario 1. Failed nodes are colored in gray.

Test Result

Figure 5.6 illustrates how the state changes from the synchronized mission are carried over and considered for the unsynchronized mission.

5.2.5 Test 4: Automatic, Online Search after Synchronizing an Ill-advised Mission

For the last time, the greedy sequence from scenario is chosen as a base. However, all instances of go_recharge are deleted from the sequence before synchronizing the remaining tasks.

Expected Outcome

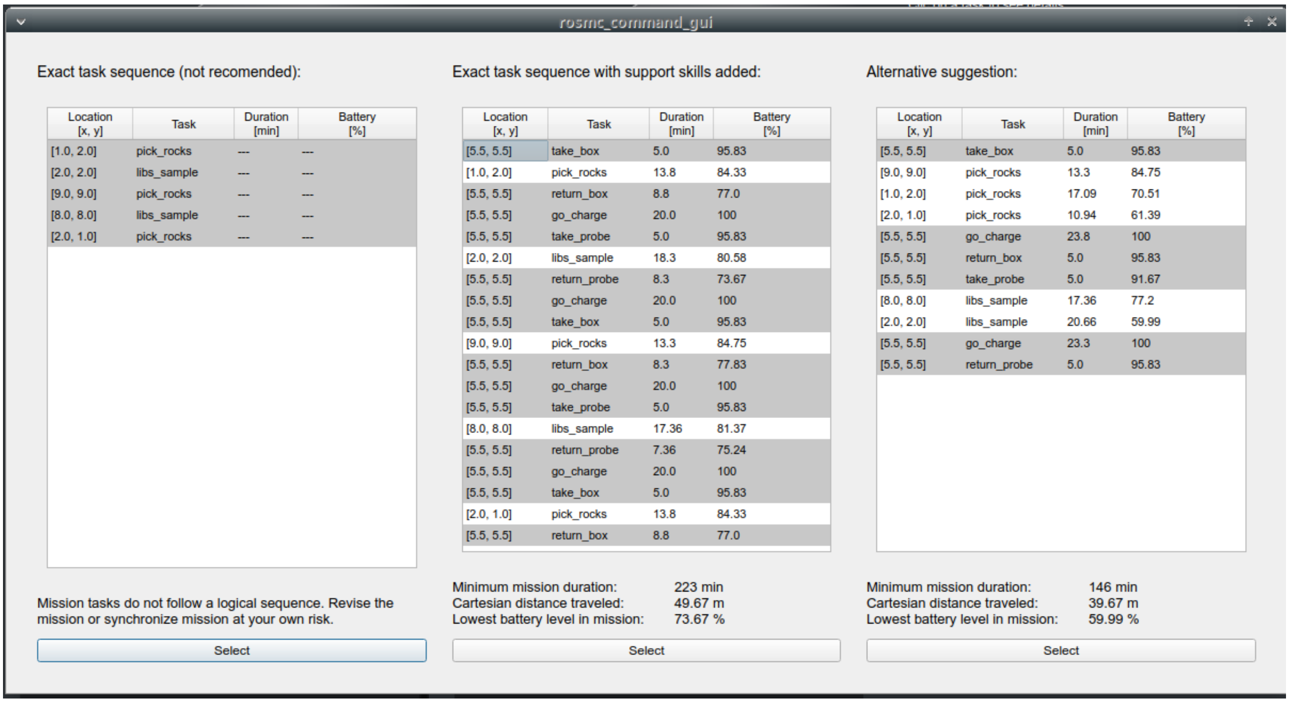
The warning pop-up should emerge as soon as task synchronizing is done the overview window should show a failure in due to low battery

Test Result

As expected, a warning pop-up showed up as soon as the mission was done synchronizing, followed by an overview window seen in Figure 5.7, indicating that the search was triggered automatically and detected a problem.

5.2.6 Special Mention: Waypoints

One interesting possibility is using the skill move_to_position to create waypoints, which could be used to better plan a path between regions of interests and provide a more accurate assessment from the beginning. To test this, a single libs_sample task was created. Intermediate instances of move_to_position tasks were used to generate waypoints to and from the lander.



Exact task sequence (not recomended):

Location [x, y]	Task	Duration [min]	Battery [%]
[1.0, 2.0]	pick_rocks	---	---
[2.0, 2.0]	libs_sample	---	---
[9.0, 9.0]	pick_rocks	---	---
[8.0, 8.0]	libs_sample	---	---
[2.0, 1.0]	pick_rocks	---	---

Mission tasks do not follow a logical sequence. Revise the mission or synchronize mission at your own risk.

Exact task sequence with support skills added:

Location [x, y]	Task	Duration [min]	Battery [%]
[5.5, 5.5]	take_box	5.0	95.83
[1.0, 2.0]	pick_rocks	13.8	84.33
[5.5, 5.5]	return_box	8.8	77.0
[5.5, 5.5]	go_charge	20.0	100
[5.5, 5.5]	take_probe	5.0	95.83
[2.0, 2.0]	libs_sample	18.3	80.58
[5.5, 5.5]	return_probe	8.3	73.67
[5.5, 5.5]	go_charge	20.0	100
[5.5, 5.5]	take_box	5.0	95.83
[9.0, 9.0]	pick_rocks	13.3	84.75
[5.5, 5.5]	return_box	8.3	77.83
[5.5, 5.5]	go_charge	20.0	100
[5.5, 5.5]	take_probe	5.0	95.83
[8.0, 8.0]	libs_sample	17.36	81.37
[5.5, 5.5]	return_probe	7.36	75.24
[5.5, 5.5]	go_charge	20.0	100
[5.5, 5.5]	take_box	5.0	95.83
[2.0, 1.0]	pick_rocks	13.8	84.33
[5.5, 5.5]	return_box	8.8	77.0

Minimum mission duration: 223 min
Cartesian distance traveled: 49.67 m
Lowest battery level in mission: 73.67 %

Alternative suggestion:

Location [x, y]	Task	Duration [min]	Battery [%]
[5.5, 5.5]	take_box	5.0	95.83
[9.0, 9.0]	pick_rocks	13.3	84.75
[1.0, 2.0]	pick_rocks	17.09	70.51
[2.0, 1.0]	pick_rocks	10.94	61.39
[5.5, 5.5]	go_charge	23.8	100
[5.5, 5.5]	return_box	5.0	95.83
[5.5, 5.5]	take_probe	5.0	91.67
[8.0, 8.0]	libs_sample	17.36	77.2
[2.0, 2.0]	libs_sample	20.66	59.99
[5.5, 5.5]	go_charge	23.3	100
[5.5, 5.5]	return_probe	5.0	95.83

Minimum mission duration: 146 min
Cartesian distance traveled: 39.67 m
Lowest battery level in mission: 59.99 %

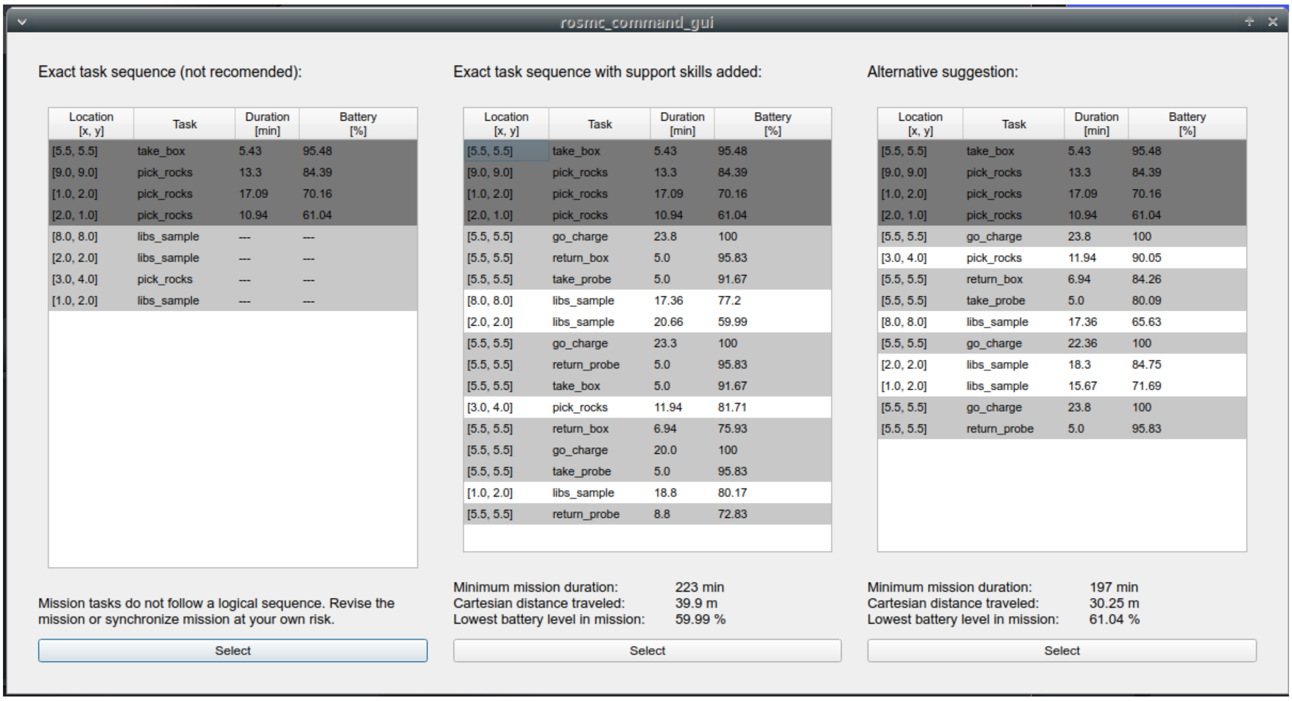
Figure 5.5 Overview window displaying results for the second scenario. Although the original sequence (with support skills) is feasible, the greedy search provided a far more reasonable answer.

Test Result

The results of the search can be seen in Figure 5.8, in which the limitations of the search algorithm can be seen. In the second sequence, the search greedily attempts to fulfill a task as early as possible before, generating a sub-optimal result. The last result makes no sense in the overall context of the mission.

5.3 Implementation with LRU2

Due to time constraints and considerable delays due to system failures with the robot, a demonstration was not possible as of the current date.



Exact task sequence (not recomended):

Location [x, y]	Task	Duration [min]	Battery [%]
[5.5, 5.5]	take_box	5.43	95.48
[9.0, 9.0]	pick_rocks	13.3	84.39
[1.0, 2.0]	pick_rocks	17.09	70.16
[2.0, 1.0]	pick_rocks	10.94	61.04
[8.0, 8.0]	libs_sample	---	---
[2.0, 2.0]	libs_sample	---	---
[3.0, 4.0]	pick_rocks	---	---
[1.0, 2.0]	libs_sample	---	---

Mission tasks do not follow a logical sequence. Revise the mission or synchronize mission at your own risk.

Exact task sequence with support skills added:

Location [x, y]	Task	Duration [min]	Battery [%]
[5.5, 5.5]	take_box	5.43	95.48
[9.0, 9.0]	pick_rocks	13.3	84.39
[1.0, 2.0]	pick_rocks	17.09	70.16
[2.0, 1.0]	pick_rocks	10.94	61.04
[5.5, 5.5]	go_charge	23.8	100
[5.5, 5.5]	return_box	5.0	95.83
[5.5, 5.5]	take_probe	5.0	91.67
[8.0, 8.0]	libs_sample	17.36	77.2
[2.0, 2.0]	libs_sample	20.66	59.99
[5.5, 5.5]	go_charge	23.3	100
[5.5, 5.5]	return_probe	5.0	95.83
[5.5, 5.5]	take_box	5.0	91.67
[3.0, 4.0]	pick_rocks	11.94	81.71
[5.5, 5.5]	return_box	6.94	75.93
[5.5, 5.5]	go_charge	20.0	100
[5.5, 5.5]	take_probe	5.0	95.83
[1.0, 2.0]	libs_sample	18.8	80.17
[5.5, 5.5]	return_probe	8.8	72.83

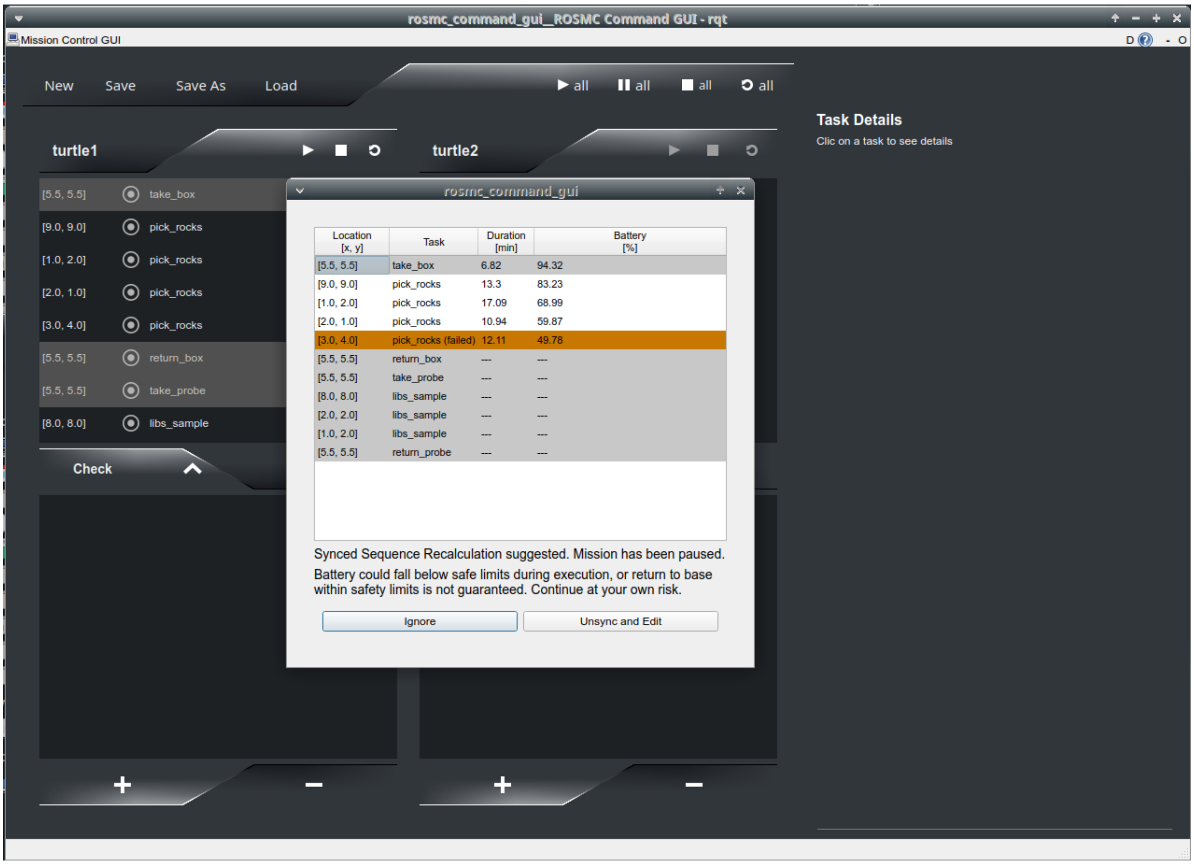
Minimum mission duration: 223 min
Cartesian distance traveled: 39.9 m
Lowest battery level in mission: 59.99 %

Alternative suggestion:

Location [x, y]	Task	Duration [min]	Battery [%]
[5.5, 5.5]	take_box	5.43	95.48
[9.0, 9.0]	pick_rocks	13.3	84.39
[1.0, 2.0]	pick_rocks	17.09	70.16
[2.0, 1.0]	pick_rocks	10.94	61.04
[5.5, 5.5]	go_charge	23.8	100
[3.0, 4.0]	pick_rocks	11.94	90.05
[5.5, 5.5]	return_box	6.94	84.26
[5.5, 5.5]	take_probe	5.0	80.09
[8.0, 8.0]	libs_sample	17.36	65.63
[5.5, 5.5]	go_charge	22.36	100
[2.0, 2.0]	libs_sample	18.3	84.75
[1.0, 2.0]	libs_sample	15.67	71.69
[5.5, 5.5]	go_charge	23.8	100
[5.5, 5.5]	return_probe	5.0	95.83

Minimum mission duration: 197 min
Cartesian distance traveled: 30.25 m
Lowest battery level in mission: 61.04 %

Figure 5.6 Results for the third search scenario. The first four tasks are shown in dark gray, signaling their status as synchronized. The fact that the robot is expected to hold a collection box is acknowledged in how the task prerequisites are handled.



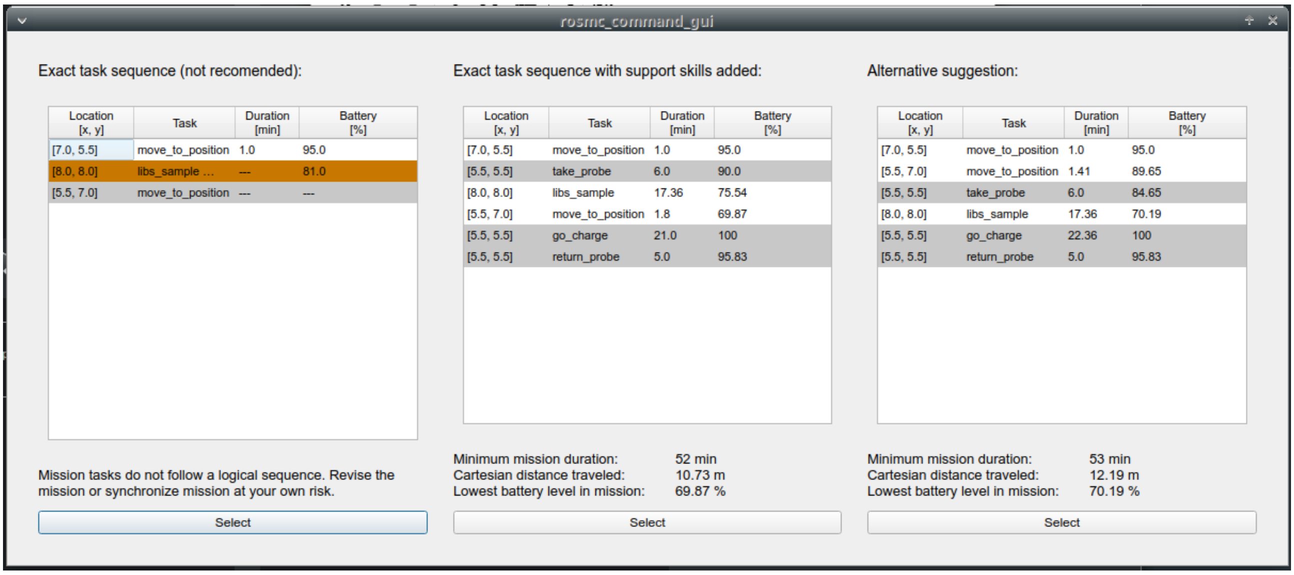
[5.5, 5.5]	take_box
[9.0, 9.0]	pick_rocks
[1.0, 2.0]	pick_rocks
[2.0, 1.0]	pick_rocks
[3.0, 4.0]	pick_rocks
[5.5, 5.5]	return_box
[5.5, 5.5]	take_probe
[8.0, 8.0]	libs_sample

Location [x, y]	Task	Duration [min]	Battery [%]
[5.5, 5.5]	take_box	6.82	94.32
[9.0, 9.0]	pick_rocks	13.3	83.23
[1.0, 2.0]	pick_rocks	17.09	68.99
[2.0, 1.0]	pick_rocks	10.94	59.87
[3.0, 4.0]	pick_rocks (failed)	12.11	49.78
[5.5, 5.5]	return_box	---	---
[5.5, 5.5]	take_probe	---	---
[8.0, 8.0]	libs_sample	---	---
[2.0, 2.0]	libs_sample	---	---
[1.0, 2.0]	libs_sample	---	---
[5.5, 5.5]	return_probe	---	---

Synced Sequence Recalculation suggested. Mission has been paused.

Battery could fall below safe limits during execution, or return to base within safety limits is not guaranteed. Continue at your own risk.

Figure 5.7 Results for the automatic online search triggered after mission synchronization, indicating precisely when and why it is likely to fail.



Exact task sequence (not recomended):

Location [x, y]	Task	Duration [min]	Battery [%]
[7.0, 5.5]	move_to_position	1.0	95.0
[8.0, 8.0]	libs_sample ...	---	81.0
[5.5, 7.0]	move_to_position	---	---

Mission tasks do not follow a logical sequence. Revise the mission or synchronize mission at your own risk.

Exact task sequence with support skills added:

Location [x, y]	Task	Duration [min]	Battery [%]
[7.0, 5.5]	move_to_position	1.0	95.0
[5.5, 5.5]	take_probe	6.0	90.0
[8.0, 8.0]	libs_sample	17.36	75.54
[5.5, 7.0]	move_to_position	1.8	69.87
[5.5, 5.5]	go_charge	21.0	100
[5.5, 5.5]	return_probe	5.0	95.83

Minimum mission duration:	52 min
Cartesian distance traveled:	10.73 m
Lowest battery level in mission:	69.87 %

Alternative suggestion:

Location [x, y]	Task	Duration [min]	Battery [%]
[7.0, 5.5]	move_to_position	1.0	95.0
[5.5, 7.0]	move_to_position	1.41	89.65
[5.5, 5.5]	take_probe	6.0	84.65
[8.0, 8.0]	libs_sample	17.36	70.19
[5.5, 5.5]	go_charge	22.36	100
[5.5, 5.5]	return_probe	5.0	95.83

Minimum mission duration:	53 min
Cartesian distance traveled:	12.19 m
Lowest battery level in mission:	70.19 %

Figure 5.8 Results for a search containing waypoints.

6 Discussion

The following section compiles the most important reflections with regard to the results compiled in the previous chapter, in order from concrete to systematic. First, a discussion regarding the search algorithms used, as well as their implications, is presented. Following that, the program itself is commented. Finally, the discussion ends with an overview of the interaction between the program in the context of its relationship to the other systems it is related to.

6.1 Search Algorithm

6.1.1 Effectiveness

Perhaps the most pressing issue regarding the implicit handling of $A(s_k)$ is its capacity for explosive branching, as the base branching factor of the tree search is the size of the skill set itself. While this could be a factor for a robot with a skill set in the hundreds or thousands, this would be very unwieldy in reality and unlikely to occur. The effective branching factor is also reduced significantly in practice due to both a reasonable search heuristic and prerequisite inconsistency constraints. Therefore, unfeasible solutions are trimmed immediately. It is, however, prudent to acknowledge how this algorithm is intended for the mission control of a relatively short list of complex skills. It is not intended to be used to optimize a large amount of simple actions, for which there are better tools available.

6.1.2 Limitations of the Greedy Best-First Search

The effectiveness of an automated scheduler is only as effective as its core search algorithm, and in this case, the limitations of using a greedy search have been evident, although by no means disastrous. The most obvious way to bypass these limitations would be to implement, for example, a branch-and-bound approach as seen to solve classical travelling salesman formulations. However, finding a good heuristic to determine an upper bound is not nearly as trivial, and was out of the scope of this work due to time constraints. The main drawback of a greedy best-first approach was most evident when faced with a task hidden behind costly prerequisites versus tasks which were not, regardless of total travel distance. Another way in which this could be potentially offset is by considering an additional "potential" term in the value function to reward support skills that enable tasks, for example.

6.1.3 Tractability and Human-Legibility

The main advantage of the space state search formulation in itself is its tractability, and this extends to our end result: If desired, the whole decision tree for every search could be analyzed, and every parameter value change could be observed in detail if needed, which was especially helpful during the generative phase of the project as it made troubleshooting less complex. In this case, the main challenge consisted of finding a reasonable way of presenting an adequate amount of information. Although the study would have benefited from structured user studies, this was impossible due to time constraints, and is certainly recommended for future development.

6.2 Program Functionality

6.2.1 Functionality with Regard to Objectives

The current program implementation definitely allows for interactive mission generation. We consider the program provides effective and detailed suggestions every time it is prompted to do so, which gives the user sufficient information with which to iterate or make final decisions without ever being overridden. So far, the only functionality which does not behave as expected is the use of waypoints. Other corner cases, however, remain to be tested in more detail.

As to the reactive mission recalculation capabilities, the decision was made to forego this feature in favor of a more agile and robust behavior, especially given that the capacity to reconfigure a mission is already possible as part of the functionalities of ROSMC and the offline searches.

Regarding a practical, scalable application, there are still several questions that still need to be resolved. The first question is how the skill set files are created, by whom, and according to which standards. This issue is particularly important for the skill set creation of multiple robots, especially considering the possibility of multi-robot mission scheduling. Furthermore, it could also be useful to explore a more direct connection to a robot's world model.

6.2.2 Waypoints

So far, the program is unable to make a reasonable use of waypoints because they are not correctly modeled in the algorithm formulation, and the implied logic behind their use is not implemented. One way in which this could be solved without an overcomplicated heuristic would be to give users the ability to create groups of tasks, essentially working like sub-problems in a search context. From a usability standpoint, grouped tasks could be dragged and dropped as a single entity if desired, and would provide the user with the ability to specify that a portion or portions of the search must follow a strict sequence. This formulation could have the added benefit of improving search efficiency as well.

6.3 System Functionality and Deployment

Even though a full mission demonstration with a real system was ultimately not achieved, the program is already deployed using the same system architecture: it is integrated with the mission server, RAFCON state machine library and executor and other components of the multirobot team system. With regard to the real LRU2, the program can effectively receive state updates from LRU2's position and battery status. The necessary changes to RAFCON state machines and libraries have also been deployed, as have been the updated configuration files used for multirobot mission containing skill set and scheduler-related information, so the system could, theoretically, run. Testing with the real robot should be carried out in the near future as a priority. In fact, it would be interesting to consider other robot systems with different contexts, or even different mission scenarios with the same robots, in order to fully test the program, its scalability, and discover new opportunities for improvement.

7 Conclusions

- Search algorithm validation carried out using the data from the ARCHES mission in Mt. Etna confirmed the latter's capacity to produce reasonable sequences within the feasibility region delimited both by the skill set constraints and the robot battery. Although slightly sub-optimal, the results are good enough for practical purposes when considering a real mission execution uncertainty.
- The program proved it is capable of providing interactive mission composition and reactive mission assessment, as well as providing informative feedback and reasoning behind its reasoning in terms that were easy to understand and compare.
- The program has been successfully initiated with the heterogeneous team system, but full deployment is still not complete.
- It would be important to further discuss how the skill set could be standardized for scalability and, eventually, how the program could interface directly with a robot world model.

7.1 Limitations

Time was, by far, the most important limitation in this project and its scope. As such, the scope of this project has been limited to the deployment for one robot, LRU2. Ultimately, robot deployment was hindered by multiple successive technical problems with the robot which not only impeded development time, but also provided a bottleneck scenario with many interested parties eager to utilize the system when available.

7.2 Future Work

- A more detailed assessment and comparison of the search heuristics could be suggested to overcome the problems associated with a greedy search heuristic.
- Work on the successful deployment and testing with LRU2 and the heterogeneous robot system at RMC is still pending and will continue. A practical demonstration involving the box and sample picking tasks is underway.
- Maturation of the enhanced ROSMC capabilities are still necessary before any releases are made. At this stage, the program is a functional prototype. With more time, its detailed behavior, especially in unforeseen circumstances, needs to be tested and evaluated in detail.
- More practical testing could be done to refine the reactive search behavior. The author believes this endeavor would highly benefit from experimentation with real robotic systems.
- Moreover, in order to correctly assess if some of the difficulties associated with the usability of the original system have been mitigated, a comparable user study must be undertaken.
- Finally, the author expresses her desire to continue working on this project, especially towards the development of multi-robot capabilities.

A Appendix

A.1 LRU2 Skill Set

take_box: (Pick a collection box located in the given coordinates)

Execution time: 5
Default parameters:

- x: 0.0
- y: 0.0

Preconditions:

- has_box: False
- has_probe: False

Effects:

- Update x
- Update y
- has_box = True
- Update battery_percentage
- Update time

return_box: (Place a held collection box on the given coordinates)

Execution time: 5
Default parameters:

- x: 0.0
- y: 0.0

Preconditions:

- has_box: True
- has_probe: False

Effects:

- Update x
- Update y
- has_box = False
- Update battery_percentage
- Update samples_stored
- Update time

take_probe: (Pick a LIBS instrument located in the given coordinates)

Execution time: 5
Default parameters:

- x: 0.0
- y: 0.0

Preconditions:

- has_box: False
- has_probe: False

Effects:

- Update x
- Update y
- has_probe = True
- Update battery_percentage
- Update time

return_probe: (Place a held LIBS instrument on the given coordinates)

Execution time: 5
Default parameters:

- x: 0.0
- y: 0.0

Preconditions:

- has_box: False
- has_probe: True

Effects:

- Update x
- Update y
- has_probe = False
- Update battery_percentage
- Update_time

take_sample: (Pick a rock or sand sample and place kit in a collection box)

Execution time: 10
Default parameters:

- x: 0.0
- y: 0.0

Preconditions:

- has_box: True
- box_full: False

Effects:

- Update samples_stored
- Update x
- Update y
- has_probe = False
- Update battery_percentage
- Update time

pick_rocks: (Perform LIBS measurement in the given coordinates)

Execution time: 15
Default parameters:

- x: 0.0
- y: 0.0

Preconditions:

- has probe: True

Effects:

- Update x
- Update y
- Update battery_percentage
- Update time

go_recharge: (Drive back to the lander and exchange batteries)

Execution time: 20
Default parameters:

- x: 0.0
- y: 0.0

Preconditions:

- (None)

Effects:

- Update x
- Update y
- Update battery_percentage
- Update time

move_to_pose: (Drive to desired pose)

Execution time: 5
Default parameters:

- x: 0.0
- y: 0.0
- yaw: 0.0

Preconditions:

- (None)

Effects:

- Update x
- Update y
- Update yaw
- Update battery_percentage
- Update time

move_to_position: (Drive to desired position)

Execution time: 5
Default parameters:

- x: 0.0
- y: 0.0

Preconditions:

- (None)

Effects:

- Update x
- Update y
- Update battery_percentage
- Update time

mission_finish: (Desired end configuration)

Execution time: 0
Default parameters:

- x: 0.0
- y: 0.0

Preconditions:

- x=0.0
- y=0.0
- has_probe: False
- has_ox: False

Effects:

- Update x
- Update y
- Update battery_percentage
- Update time

Bibliography

[1] Esther Bischoff et al. “Heuristic reoptimization of time-extended multi-robot task allocation problems”. In: *Networks* 84.1 (2024), pp. 64–83. DOI: https://doi.org/10.1002/net.22217. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.22217. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.22217.

[2] Esther Bischoff et al. “Multi-Robot Task Allocation and Scheduling Considering Cooperative Tasks and Precedence Constraints”. In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2020, pp. 3949–3956. DOI: 10.1109/SMC42975.2020.9283215.

[3] Sebastian G. Brunner et al. “RAFCON: A graphical tool for engineering complex, robotic tasks”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 3283–3290. DOI: 10.1109/IROS.2016.7759506.

[4] Maria Fox and Derek Long. “PDDL+: Modeling continuous time dependent effects”. In: *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*. Vol. 4. 2002, p. 34.

[5] Nam Eung Hwang, Hyung Jun Kim, and Jae Gwan Kim. “Centralized Mission Planning for Multiple Robots Minimizing Total Mission Completion Time”. In: *Applied Sciences* 13.6 (2023). ISSN: 2076-3417. DOI: 10.3390/app13063737. URL: https://www.mdpi.com/2076-3417/13/6/3737.

[6] A. H. Land and A. G. Doig. “An Automatic Method of Solving Discrete Programming Problems”. In: *Econometrica* 28.3 (1960), pp. 497–520. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1910129 (visited on 01/19/2025).

[7] E.L. Lawler. *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization. John Wiley & Sons, 1985. URL: https://books.google.de/books?id=qbFlMwEACAAJ.

[8] Peter Lehner et al. “Mobile manipulation for planetary exploration”. In: *2018 IEEE Aerospace Conference*. 2018, pp. 1–11. DOI: 10.1109/AERO.2018.8396726.

[9] Giulia Orrú and Luca Longo. “The Evolution of Cognitive Load Theory and the Measurement of Its Intrinsic, Extraneous and Germane Loads: A Review”. In: *International Symposium on Human Mental Workload*. 2018. URL: https://api.semanticscholar.org/CorpusID:86587936.

[10] Edwin P. D. Pednault. “ADL and the State-Transition Model of Action”. In: *Journal of Logic and Computation* 4.5 (Oct. 1994), pp. 467–512. ISSN: 0955-792X. DOI: 10.1093/logcom/4.5.467. eprint: https://academic.oup.com/logcom/article-pdf/4/5/467/6322988/4-5-467.pdf. URL: https://doi.org/10.1093/logcom/4.5.467.

[11] Stuart Russell and Peter Norvig. *Artificial Intelligence, Global Edition A Modern Approach*. Pearson Deutschland, 2021, p. 1168. ISBN: 9781292401133. URL: https://elibrary.pearson.de/book/99.150005/9781292401171.

[12] Ryo Sakagami. *Cooperative Operation Framework for Heterogeneous Robotic Teams in Planetary Exploration*. Tech. rep. The University of Tokyo, Feb. 2020. URL: https://elib.dlr.de/134199/.

[13] Ryo Sakagami et al. “ROSMC: A High-Level Mission Operation Framework for Heterogeneous Robotic Teams”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 5473–5479. DOI: 10.1109/ICRA48891.2023.10161133.

[14] Martin W. P. Savelsbergh. "The Vehicle Routing Problem with Time Windows: Minimizing Route Duration". In: *ORSA Journal on Computing* 4.2 (1992), pp. 146–154. DOI: 10.1287/ijoc.4.2.146. eprint: https://doi.org/10.1287/ijoc.4.2.146. URL: https://doi.org/10.1287/ijoc.4.2.146.

[15] Martin J Schuster et al. "Towards autonomous planetary exploration: The Lightweight Rover Unit (LRU), its success in the SpaceBotCamp challenge, and beyond". In: *Journal of Intelligent & Robotic Systems* 93.3 (2019). DOI: https://doi.org/10.1007/s10846-017-0680-9. URL: https://link.springer.com/article/10.1007/s10846-017-0680-9#Ack1.

[16] Martin J. Schuster et al. "The ARCHES Space-Analogue Demonstration Mission: Towards Heterogeneous Teams of Autonomous Robots for Collaborative Scientific Sampling in Planetary Exploration". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5315–5322. DOI: 10.1109/LRA.2020.3007468.

[17] Armin Wedler et al. "First Results of the ROBEX Analogue Mission Campaign: Robotic Deployment of Seismic Networks for Future Lunar Missions". In: Sept. 2017.

[18] Qin Yang et al. "Self-Reactive Planning of Multi-Robots with Dynamic Task Assignments". In: *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. 2019, pp. 89–91. DOI: 10.1109/MRS.2019.8901075.

[19] Yu Zhang and Lynne E. Parker. "Multi-robot task scheduling". In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 2992–2998. DOI: 10.1109/ICRA.2013.6630992.