UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

PATH PLANNING FOR A MARS ROVER: A PROGNOSTIC BASED OFFLINE DECISION-MAKING APPROACH

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL ELÉCTRICA

MARIANA IGNACIA SALINAS CAMUS

PROFESOR GUÍA:
MARCOS ORCHARD CONCHA

MIEMBROS DE LA COMISIÓN:
CONSTANZA AHUMADA SANHUEZA
CHETAN KULKARNI

SANTIAGO DE CHILE
2022

RESUMEN DE LA TESIS PARA OPTAR AL GRADO
DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA,
MENCIÓN ELÉCTRICA Y MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERA CIVIL ELÉCTRICA
POR: MARIANA IGNACIA SALINAS CAMUS
FECHA: 2022
PROF.GUÍA: MARCOS ORCHARD CONCHA

PLANIFICACIÓN DE RUTA PARA UN ROVER EN MARTE: UN SISTEMA DE TOMA DE DECISIONES FUERA DE LÍNEA BASADO EN PRONÓSTICO

Un rover en Marte debe visitar puntos de ruta para tomar muestras científicas. Debido a la falta de fiabilidad de las telecomunicaciones, el rover necesita un sistema de toma de decisiones autónomo. La planificación de ruta se puede separar en dos etapas, en línea y fuera de línea, para reducir el costo computacional. En esta tesis, se describe un sistema de toma de decisiones de pronósticas (PDM) fuera de línea. El sistema PDM se formuló como un problema de optimización, utilizando un algoritmo genético (GA) para encontrar soluciones factibles. El sistema considera el estado de carga de las baterías, el número de puntos de ruta visitados y el perfil del terreno. El GA decide el número de puntos de ruta a visitar, en qué orden se visitan y cuándo recargar las baterías. Se implementó y probó el sistema PDM bajo diferentes mapas de terreno, realizando un estudio de sensibilidad de los parámetros. El sistema encuentra soluciones factibles eficientemente, evita la sobrecarga de consumo energético y planifica la recarga de la batería para satisfacer la política de SoC. El análisis del impacto de fuentes de incertidumbre del modelo muestra que el enfoque adoptado da un grado de holgura para operar.

RESUMEN DE LA TESIS PARA OPTAR AL GRADO
DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA,
MENCIÓN ELÉCTRICA Y MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERA CIVIL ELÉCTRICA
POR: MARIANA IGNACIA SALINAS CAMUS
FECHA: 2022
PROF.GUÍA: MARCOS ORCHARD CONCHA

PATH PLANNING FOR A MARS ROVER: A PROGNOSTIC BASED OFFLINE DECISION-MAKING APPROACH

A rover mission consists of visiting waypoints to take scientific samples. Due to telecommunication unreliability, the rover must have an autonomous decision-making system. The path-planning problem can be separated into two stages — online and offline — to reduce computational cost. In this thesis, an offline prognostic decision-making (PDM) system is described. The PDM problem was formulated from an optimization point of view that, unlike previous approaches, uses a genetic algorithm (GA) to find feasible solutions. The optimization problem is mathematically developed and accounts for the battery state of charge, the number of waypoints visited, and the terrain profile. The GA decides the number of waypoints to visit, in which order the rover will visit them, and when it needs to recharge batteries. The PDM system was implemented and tested through simulations under different terrain maps and a study of the parameters sensitivity was performed. Results showed that the system can efficiently find feasible solutions in different scenarios, prevents energy consumption overload and plans battery recharge to satisfy the SoC policy. An analysis of the impact of different sources of uncertainty in the model is performed, which demonstrates that the approach taken gives a degree of slack to operate.

Dedicado a la memoria de
Ximena Camus Ortega

Agradecimientos

Me gustaría agradecer a la vida, a todas las personas que de alguna u otra forma han sido parte de mi camino universitario y a las instituciones en donde me he educado. Me considero una persona muy afortunada por haber tenido la oportunidad de explorar distintas áreas del conocimiento, por poder estudiar y trabajar en lo que me apasiona.

Quiero agradecer a mi familia por su apoyo durante todos los años en la etapa escolar y universitaria. A Luis, mi papá, por enseñarme la importancia de la determinación y la responsabilidad. A Macarena, mi hermana, por su apoyo y por ser mi eterna profesora. A Ximena, mi mamá, por enseñarme a ser una persona bondadosa y por el inmenso cariño que siempre me entregó; se que estaría muy feliz por este logro.

A mis amigos, algunas quienes me acompañan desde el liceo y otros ya en la universidad, les agradezco enormemente por todos los buenos momentos que hemos pasado, por el apoyo en los momentos díficiles y por tantas risas que hemos compartido durante estos años; no lo habría podido lograr sin ustedes. Quiero mencionar especialmente aquí a Pablo y a Kay, quienes me han apoyado mucho durante la realización de esta tesis con consejos de programación y redacción.

Quiero agradecer también a mis compañeros de universidad y a los miembros del SPEL, donde empecé a conocer y a encantarme con el mundo de la investigación. En especial a Gustavo Díaz, de quien aprendí mucho realizando mi práctica y otros proyectos.

Por supuesto también agradezco al profesor Marcos Orchard, por la confianza que ha depositado en mí, por abrirme puertas y porque su apoyo ha sido fundamental para la realización de esta tesis.

Finalmente, quiero agradecer al departamento de Ingeniería Eléctrica por financiar el arancel de mis estudios de magíster al otorgarme la beca Luz Chuaqui. Espero que con los años estos espacios de investigación e ingeniería sean lugares mas diversos, con mayor participación de mujeres y disidencias.

Table of Content

List of Tables

List of Figures

Chapter 1

Introduction

1.1. Context and Motivation

One of the most frequently asked question is: When and how did life began on Earth?. This question has fueled missions that seek to explore the universe and find life in other remote places, to understand our own planet. One of the most accepted theories is that life can develop in a planet with water, carbon dioxide, and an adequate temperature. In this way, scientists have laid the foundations for exploration on Mars, the planet closest to Earth that seems to meet these characteristics and that may even be an incubator for microbial life [1].

Mars has characteristics similar to Earth. It is a rocky planet, with a rotational speed on its axis almost equal to that of Earth. A Martian day, which is called sol, is 40 minutes longer than an Earth day, but a Martian year is equivalent to 687 Earth days. Although it is about half the diameter of the Earth, its surfaces are similar to the continents of our planet. The Red Planet currently has a thin atmosphere of carbon dioxide and cannot support life as we know it. However, from the studies that have been done since the last century, there is evidence that suggests that billions of years ago there were rivers and streams of water flowing on the Martian surface, which leads one to think that Mars did have an atmosphere capable of keeping the water in a liquid state. This leads us to ask ourselves, what happened during the evolution of Mars so that it is now a dry and dusty planet?. The study of the red planet allows scientists to understand that drastic changes in climate can alter all life on a planet and of course, the possibility that one day humans may inhabit Mars.

The exploration of Mars began in 1960, when two probes where sent but exploded before reaching Earth's orbit. During the following decades there were different missions of spacecraft and probes that orbited the red planet in order to be able to photograph the Martian surface, learn about the topography, analyze characteristics of the atmosphere, among others.

In 1997, NASA's first rover landed, Sojourner. This was an experimental vehicle intended to prove that unmanned technology could be used to explore another planet. Sojourner managed to complete the planned scientific tests and remained operational on Mars for 85 Earth days [2]. NASA has subsequently sent four more rovers: Spirit and Opportunity (2004), Curiosity (2012) and Perseverance (2021). Although they share some similarities, their technologies and designs have been adapted to carry out increasingly complex missions [3].

In general terms, a rover is made up of 8 subsystems: traction and mobility, autonomous control and navigation, mechanical structure, data and command handling, communications, energy generation and storage, thermal control, and scientific experiments. These subsystems are constrained by power, time, computational capacity and weight limitations, as well as the communication delays involved in sending signals between Earth and Mars [4]. These communication delays make autonomous navigation a critical subsystem because Earth-Mars communication takes from 8 to 42 minutes and is only possible during communication windows. Hence, when an unexpected situation occurs, the rover enters safe mode and stops its movements until it receives instructions from Earth. Engineers must then analyze the data and determine new instructions. This entire process can take days, delaying the mission, incurring further economic costs, and risking the lifespan of the rover [5].

One of the challenges in developing an autonomous navigation system is that rovers have limited computational capacity. The computational cost has been reduced in some frameworks by separating the system into two stages: offline and online. The first is the offline stage, so it has no computational restrictions and consists of calculating the "best path" given the already known information about the terrain, the rover's health status and the waypoints it needs to visit to complete its scientific experiments. The second stage is online and is responsible for making corrections to the path in case an unexpected obstacle or terrain is encountered [6][7].

One approach to tackle the offline autonomous navigation stage is Prognostic Decision-Making (PDM) [8]. This approach includes modules for fault diagnosis and prognosis of the power and system health to make decisions and plan a route based on this information. With this approach, decision making becomes a complex optimization problem to determine the set of actions that the rover must follow. This optimization problem can be solved by different methods, such as gradient descent, dynamic programming or stochastic methods. Although some of these methods have been implemented in the past, more efficient and robust methods to preserve the health of rover's components would be useful. One candidate is the evolutionary or genetic algorithm (GA), which is based on natural selection obtained through iterations of the individuals or solutions that solve the problem efficiently [9].

In this thesis, we describe the design, implementation and testing of a Decision-Making System to solve the rover's offline path planning problem on Mars. The path planning is formulated as an optimization problem, which is divided into two sub-problems, GA-1 and GA-2. Therefore, the Decision-Making System consists of two GA. The first GA decides the number of waypoints and the order to visit them and, the second GA uses that information to plan a path between a pair of subsequent waypoints.

1.2. Hypothesis

This thesis is focused on the study and design of a Prognostic Decision-Making system (PDM). The PDM system's goal is to obtain an strategy to allow a Mars rover, which is an explorer vehicle, to move autonomously. Therefore, the hypothesis of this work are the following.

1. The path planning of locations that a rover needs to visit, without communications delay between Earth and Mars, can be solved through a offline PDM system that takes into account waypoints, terrain characterization and state of charge of the batteries.

2. A genetic algorithm (GA) can solve the optimization problem formulated by the offline PDM system to plan the route.

1.3. Objectives

1.3.1. Main Objective

The main objective of this thesis is to design, implement and test an offline strategy to solve the problem of planning of autonomous movement for an explorer rover on Mars, based on the approach of PDM, using a GA that can solve the optimization problem.

1.3.2. Specific Objectives

1. To design a decision-making system that takes into account state of charge of the batteries, probabilistic characterization of the terrain and waypoints.

2. To design and implement a GA to solve the optimization problem

3. To test and compare the performance of the PDM system in different simulated scenarios.

1.3.3. Thesis Outline

The thesis structure is as follows. Chapter 2 provides a theoretical framework of general aspects of a Mars rover's and documentation on the terrain characterization for Mars. It also provides a description of a state-space model for the state of charge estimation of batteries, describes the mathematical approach of the path planning problem and reviews GAs and their principal operators and considerations. Chapter 3 states the offline Decision-Making System and the implementation as a GA. Chapter 4 conducts three set of experiment to analyze the performance and the parameters sensitivity. Finally, Chapter 5 presents the conclusions and future work.

Chapter 2

Theoretical Framework

2.1. General Aspects of a Rover

A rover is a motorized exploration vehicle. Generally, a rover has 4, 6, or 8 wheels and can vary in size and weight. In 1997 the first rover landed on Mars, *Sojourner*, and subsequently, diverse missions have been performed by different agencies and research centers. NASA Jet Propulsion Laboratory (JPL) has sent rovers *Spirit*, *Opportunity*, *Curiosity*, and *Perseverance*. The European Space Agency (ESA), in collaboration with Roscosmos (Russian Space Agency), is developing the rover *ExoMars*, which was expected to be launched in 2022, however, due to the Russo-Ukrainian War the mission was postponed indefinitely. The launch of the explorer robot *MELOS* (Mars Exploration of Life and Organism Search) made by JAXA (Japan Aerospace Exploration Agency) is expected to be in 2022 but the agency has not published updates.

Therefore, the leadership of NASA in terms of space exploration on the Martian surface is clear. NASA has not only successfully sent rovers, but also they have been in operation for years. For example, *Opportunity* was active for nearly 15 years [10]. Thus, in this thesis, the analysis of systems and technologies of rovers is based on NASA's documentation. In Figure 2.1 *Perseverance* rover is shown, which has an almost identical design and characteristics to the rover *Curiosity*. In the same way, *Curiosity* looks superficially similar to the twin rovers *Spirit* and *Opportunity*, but it has many improvements regarding the internal systems that allow its operation.

The main systems necessary for the operation of the rover will be detailed in the following subsections. It is worth to have a general knowledge of each system to understand the requirements and limitations when developing the PDM system. The information presented is primarily based on the engineering of *Curiosity* [3].

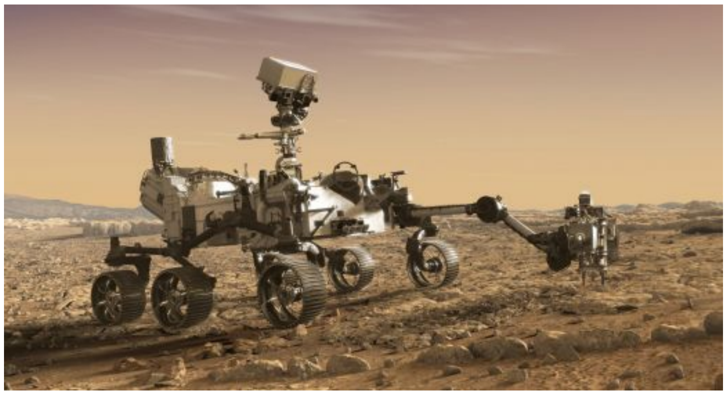


Figure 2.1: Perseverance was developed by NASA JPL and landed on Mars on February of 2021 [51].

2.1.1. Power System

Rovers *Curiosity* and *Perseverance* obtain energy from the Multi-Mission Radioisotope Generator (MMRTG), which can be visualized in Figure 2.2. Previous rovers used solar panels, but that technology is less reliable on Mars. It is unviable to charge batteries nor to continue the operation during the night, during sand storms, and in places where sunlight does not reach.

The MMRTG can reach a weight of $40[kg]$ and generates electrical energy using the heat produced by the radioactive decay of plutonium dioxide. The plutonium has a mean lifetime of 87.7 years, so the energy production will decay with time and, eventually, will cause a reduction in the rover's activity. MMRTG was designed to survive accidents that might occur in the launch; therefore, it does not release radioactive material to Earth.
As was stated before, this generator provides a more stable energy source; however, it is not the most efficient in transforming heat into electricity. It converts around $114[W]$ to electricity while the rover's heat rejection system uses the remaining $1900[W]$ of heat to maintain the electrical components to an adequate temperature, which can range from $-40°C$ to $50°C$. The rest of the heat is released into the Martian atmosphere.

A rover needs 45 to $70[W]$ when on sleep mode. Once is awake it needs a minimum of $150[W]$, with a peak of $500[W]$, when the rover is moving. Since the MMRTG only converts around $114[W]$ of electricity, the rover relies heavily on the energy stored in its batteries to perform its operations, which is why the robot is awake for only 6 hours of the day and the rest of the time is in sleep mode to charge its batteries [11].

The last four rovers developed by NASA have utilized Lithium-ion batteries due to the advantages that this technology offers in regards of mass, volume, and temperature range [12]. In the power system, an array of two Lithium-ion batteries are used, with a capacity of $1200[Wh]$. The batteries used in these missions have a good performance, during charge and discharge, even in a wide range of temperatures (from $30°C$ to $-25°C$). The batteries can also provide sufficient energy during launch and operations on the Martian surface, tolerate altitude changes during the trip, and have adequate characteristics regarding the batteries' life cycle. [13].

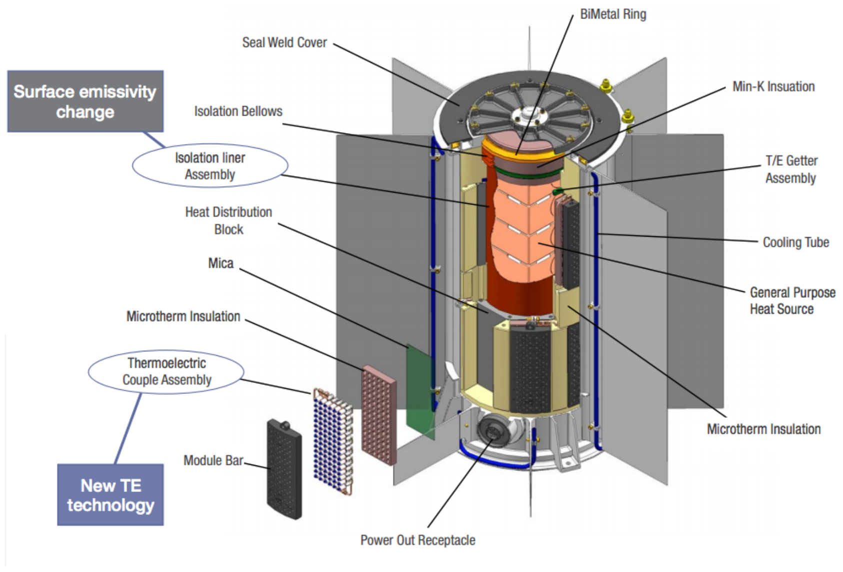


Figure 2.2: Multi-Mission Radioisotope Generator (MMRTG) [52].

2.1.2. Thermal Control

The thermal control system is critical due to the wide fluctuations of temperature presented on the Martian surface, which is a consequence of the planet's thin atmosphere. The Rover Environmental Monitoring Station (REMS) measures the temperature in the air just above the floor. According to REMS data, during the night, the temperature varies from $-85°C$ to $-65°C$, with an average of $-75°C$. During the day, the average temperature is $-10°C$, within a range of $-30°C$ to $5°C$.

Each external part of the rover was designed to endure the previously mentioned temperatures; however, each internal part has different thermal requirements. For example, the Warm Electronics Box (WEB), which holds the internal electronics of the rover, keeps the temperature between $-40°C$ to $50°C$ so the components do not fail. The remaining heat of the MMRTG is used to maintain the temperature within this range, as mentioned in subsection 2.1.1. This process is in charge of the Rover Heat Rejection System (RHRS), which pumps Freon (a liquid commonly used as a refrigerant) through a tube that passes near the MMRTG to obtain the residual heat and finally use it to heat the internal parts of the rover. When necessary to cool the vehicle, the fluid is pumped from the rover's avionics mounting plate to the cold plates in the exterior of the heat exchangers and just below the rover's upper deck.

Therefore, it is clear that the RHRS is essential to avoid component wear and failure of the whole robot. To ensure the system's continuous operation, RHRS has two redundant pumps, two mixing valves, and two redundant divider valves. The valves allow the RHRS to cool or heat the Freon as necessary.

2.1.3. Telecommunication

One of the most limiting aspects of a space mission is the broadband to send data. The operations performed by the rover are determined by the windows through which the communication can occur between Mars-Earth. The telecommunication system of the rover has a Small Deep-Space Transponder (SDST) that is a transponder designed by JPL for space missions and operates in the X-band, a Solid-State Power Amplifier (SSPA), and a UHF transceiver, which allocates in WEB.

The downlink is performed by the UHF transceiver and orbiters Odyssey and Mars Reconnaissance Orbiter, as shown in Figure 2.3. The telecommunication link is defined to send data that is not critical to telemetry, so 60 % of the data goes through this channel. On average, 1.8 potential communications can be performed during a sol (martian day), with a data rate from the rover to the orbiters of 128 kbps [14]. The uplink has a data rate of 1 to 2 kbps; thus, sending a series of commands, which usually means sending around 225 kilobits, needs to be planned with 15 minutes of anticipation.

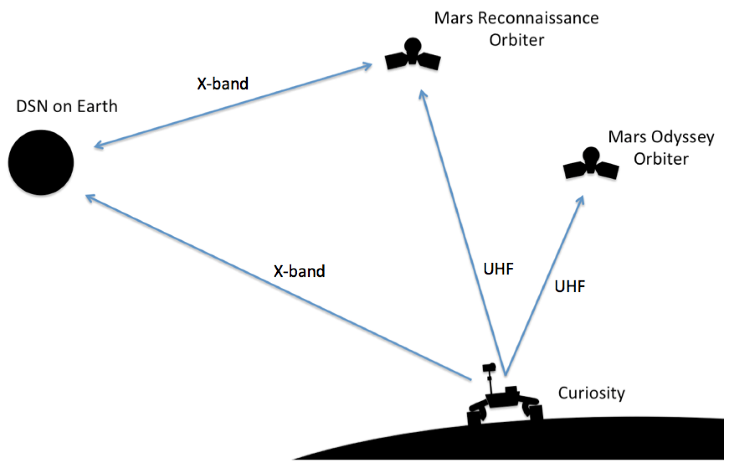


Figure 2.3: Scheme of connection between the rover and the Earth [53].

A known communication problem is when the Sun is between Earth and Mars, also known as solar conjunction. When solar conjunction occurs, the uplink communication channel is not reliable. During this situation, the rover (and all the other spacecraft on Mars) enter a 'low activity' mode because if the rover gets in a risky situation, the engineers on Earth cannot send commands to solve the problem. However, the solar conjunction only occurs every 26 months, and when it happens, the blockage lasts between 2 to 5 weeks.
Another problem, although infrequent, is that the weather on Earth can affect both the uplink and the downlink communication channels. Nevertheless, the Deep Space Network (DSN), which is composed of 3 ground stations, works satisfactorily 95 % of the time.

2.1.4. Mobility System

The last four rovers sent by NASA utilize a mechanical architecture called Rocker-Bogie, which connects six wheels to the robot's body. This system has three principal components [17], as shown in Figure 2.4.

- Differential: Connects rockers from the left and right side to the vehicle's body with the axis located in the rover's upper part. The differential is inside WEB, which is the principal structure of the rover's body.

- Rocker: Rocker: One is on the left side, and another is on the rover's right side. These connect the wheels with the differential and the bogie.

- Bogie: Connects the middle and rear wheels with the rocker.

The rocker-bogie suspension system is used for these types of vehicles because it allows the rover to traverse obstacles (rocks or ditches) of 25[cm]. It is worth noting that, with this system, the rover has mobility stability even with a tilt of 45°. The design can absorb much of the impact loads the rover experiences during lander departure.

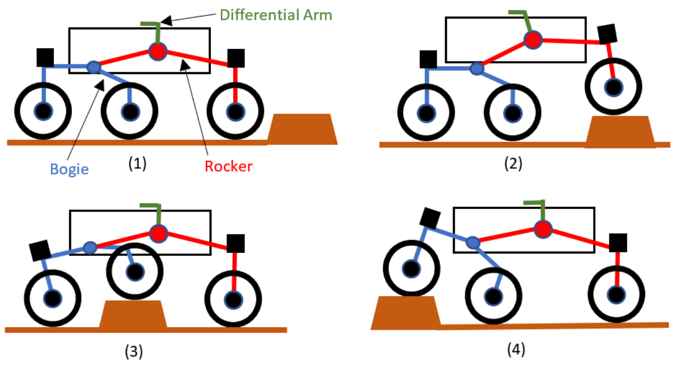


Figure 2.4: Diagram of the Rocker-Bogie suspension system.

As shown in Figure 2.4, the rocker-bogie system keeps all wheels in touch with the surface, even when facing obstacles or bumpy terrain. The latter is a fundamental characteristic because the rover's weight is always distributed between all the wheels and is not overload one of them when the vehicle is on soft terrain, like sand, causing the rover to sink. Another situation where all wheels must be in contact with the soil is when it needs to 'climb' through an extremely irregular surface [15].

The rocker-bogie system is considered a passive suspension system; therefore, it has a low complexity when it comes to controlling the rover because the mechanical structure is the one that adapts to the changes of the external loads and the terrain. However, there are configurations of active suspension systems. One example is the active suspension system of the rover *Sherpa* [16]. The active suspension system has a higher cost regarding the control adaptive system that they require but has more benefits than a passive suspension. The active suspension can get involved in the rover's movements and increment the traction on the soil. For rovers with wheels mounted on structures similar to legs, it is possible to have a configuration that allows the rover to change from driving to walking mode.

Figure 2.5 shows an example of the benefits of an active suspension system. In this case, the robot has four independent legs; each leg has a wheel. When *Sherpa* encounters larger obstacles, it utilizes its manipulator arm as support while lifting its wheels to overcome the obstacle.

Other important parts of the mobility system are the wheels and the actuators. The wheels are made of aluminum and have a spiral pattern on the surface which helps the suspension system absorb the energy and reduce the impact of the load. As mentioned above, each wheel has its motor, and each motor has an encoder. The encoder is used to control the velocity and is used to determine the steering angle and the wheel's odometry. If the vehicle has a very low load, the velocity is $4.6[cm/sec]$. If the vehicle has a maximum load, the drivers will have a current of $2[A]$ with $28[V]$, resulting in a velocity of $2.6[cm/sec]$. The whole mobility system must be able to operate on Mars at a minimum temperature of $-55°C$.

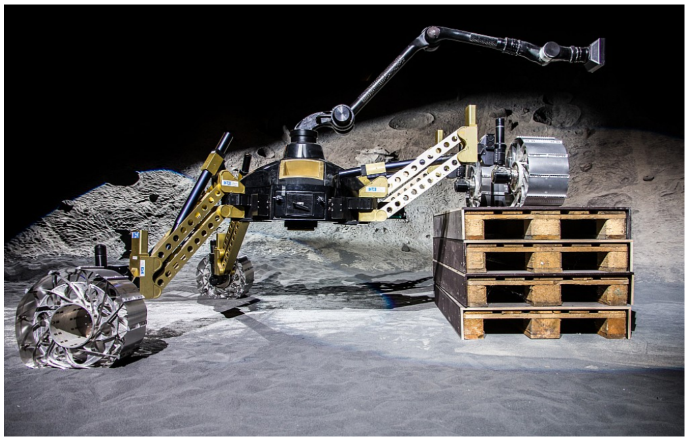


Figure 2.5: *Sherpa* testing its active suspension system [16].

2.1.5. Flight Software

The Flight Software (FS) is responsible for most of the rover's functionalities during all the mission phases. The infrastructure of the FS includes control and maintenance of the processor, more than a thousand bus and devices, command processing, sequence execution, telemetry processing, data compression, and time management of the spacecraft, among other functionalities. It is worth mentioning that FS controls, practically, all the activities during the phases of Entry, Descent, and Landing.

This subsection details the FS functionalities only needed during the operation phase on the Martian surface. Specifically, the FS developed for the rovers *Spirit* and *Opportunity* serves as a reference [18]. The above is because the characteristics of the rovers sent subsequently are similar; at the end of this subsection, their peculiarities are detailed.

The FS is written mainly in C, with some aspects programmed in C++. The FS, in addition to the operative system, has an executable image of approximately 11 MB. The onboard memory consists of 128 MB of Dynamic Random-Access Memory (DRAM) and 3 MB of Electrically Erasable Programmable Read-Only Memory (EEPROM) [17]. *Perseverance* and *Curiosity* have a RAD750 processor [19].

The FS controls the rover's mobility through the execution of basic commands. If the rover is in autonomous mode, the FS is in charge of processing images to identify the obstacle and calculate the safest path to reach the goal location. Another important activity that FS performs is handling the scientific equipment to capture the data required later for research.

One of the requirements of the FS is to coordinate activities because the hardware that controls the motor can only execute a certain amount of tasks simultaneously. For onboard activities, there is an arbitrariness mechanism that prioritizes the requests and allows the use of computational resources according to the task's priority.

Perseverance has an FS with a substantial difference from its predecessors because the FS can be updated with improvements during the mission. Another unique feature is that it has two computer modules called Rover Compute Element (RCE). One of these modules is usually turned off and is turned on only when the other RCE is facing problems. Because the two RCEs are identical, both represent the rover's 'brain'. Figure 2.6 shows the location of the 'brain' of the rover.

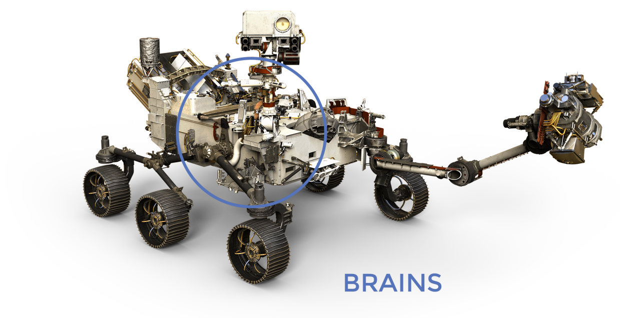


Figure 2.6: Location of the 'brain' of *Perseverance* [51].

2.1.6. Autonomous Navigation

As mentioned several times, the communication instability between Mars and Earth is an important issue; therefore, it is critical to have an autonomous navigation system to plan the route for each sol. During each sol, the rover navigates between 30 and 70 meters. A team of 3 engineers plans the path, which takes around 8 to 10 hours of work for each sol; this reflects the complexity of the task and the system integrated into the rover [20].

Spirit, *Opportunity* and *Curiosity* have three main navigation modes: blind-drive, hazard avoidance, and hazard avoidance and full-time visual odometry [21].

- Blind-drive: In this mode, there are a sufficient amount of terrain images of its current location to determine if there is a safe path. It can also use a series of commands to traverse a few meters. Due to the low computational cost of this mode, it is the one that allows the rover to move at a greater velocity in comparison to the other modes.
- Hazard avoidance: In this mode, the rover can choose the path, but it has to stop frequently to analyze if there is an obstacle nearby; however, this mode is for longer distances. The algorithm used for this mode is called GESTALT, which is described later in this section.
- Hazard avoidance and full-time visual odometry: This mode is used when the task requires high precision to reach the goal position or when the rover is on a slippery or

steep path. It is similar to the hazard avoidance mode, but the difference is that the rover stops every half-length of the vehicle to carry out a visual odometry analysis. Contrary to the blind-drive, hazard avoidance and full-time visual odometry has a massive computational cost due to their complexity, which is why in this mode is when the rover moves at the slowest velocity.

Spirit and *Opportunity* were the first rovers to use stereo vision, local map-based obstacle avoidance, and visual odometry in their navigation system. They were also the first to use computer vision in a real-time autonomous navigation system during landing [22]. Below, there is a detailed explanation of these techniques.

Both rovers have three pairs of stereo cameras: a pair of “hazcams” (hazard cameras) looking forward, another pair of hazcams looking backward, and a pair of “navcams” (navigation cameras) on the mast. The hazcams have a wide field of view (FOV) that allows seeing more than the width of the rover at a short distance of roughly four meters. That is the reason why hazcams are convenient for obstacle avoidance. On the other hand, navcams have a narrower FOV and a wider baseline; however, their FOV is enough to analyze the traversability of a candidate path many meters beyond the vehicle [22].

The information gathered from the cameras is used in the GESTALT algorithm [23], which aims to avoid obstacles in the rover’s path. GESTALT transforms stereo images into grid maps of traversability. The algorithm generates a set of allowed turns and movements by the traversability map for each driving-cycle. Once the driving-cycle is over, the rover stops to take new images and starts a new driving cycle. Each cycle considers a distance between 35 [cm] to 1 [m], and it takes around 70 seconds to calculate.

The visual odometry, which maintains the local navigation map, is executed by using dead reckoning based on the wheels encoders and the IMU, along with the updates made by the sun sensors. A significant challenge for visual odometry in Mars terrains is the slippage. The slippage makes it difficult for the encoders and the IMU to estimate the rover’s position effectively; thus, these sensors are not enough. Therefore, the images from the stereo cameras are used to update the position. However, due to its high computational cost, visual odometry is used only when necessary [21].

Perseverance uses a new software for autonomous navigation, which is called Enhanced Navigation (ENav) [24]. ENav uses 2D generated heightmaps, which includes: base heightmap, ground undulations, constant slope and rock distributions. These heightmaps are based on the orbital data and are representative of the environments that *Perseverance* faces in Mars. ENav is constantly being updated by the stereo images collected by the rover when its moving, therefore, the main source of error in the heightmap is the error in stereo matching which results in elevation error.

ENav has Path Planner Wrapper functions to choose the next arc that the rover will drive. To summarize the wrapper functions do the following: (1) receives the 3D point cloud taken by the stereo cameras, (2) updates the planner’s costmap based on the new heightmap and rover position, (3) chooses the next arc to drive given the updated heightmap and costmap, rover pose, and goal by using a local tree-search planner and a global planner, which uses

Dijkstra's algorithm, and (4) evaluates the arcs by sorting them based on their cost and safety.

2.2. Energy Storage

Robots sent by NASA have three types of batteries: primary, thermal, and rechargeable; each has different functions. Primary batteries are in the Lander; they are responsible for supporting the operations of Entry, Descent, and Landing. Thermal batteries are in charge of pyro events during the Descent and opening of the parachute. Rechargeable batteries support the launch, correct anomalies during the trip, and aid during the operations on the surface [26]; therefore, rechargeable batteries are vital for the autonomous navigation system. For that reason, this section focuses only on rechargeable batteries.

Rovers *Spirit* and *Opportunity* were the first to use an array of lithium-ion batteries for surface operations. Initially, these robots were projected to operate for 90 sols; however, *Spirit* was operative for 2623 sols and *Opportunity* for more than 5000 sols, far exceeding expectations. This success can be attributed, for the most part, to the lithium-ion battery array used [13].

2.2.1. Basic Concepts and Considerations

Many types of batteries have been developed over the years. For aerospace applications, the most used are nickel-hydrogen batteries and lithium-ion batteries. The nickel-hydrogen batteries stand out because of their long useful life; however, their energy density is only a third of what a lithium-ion battery has [27]. For explorers robots, cells have a mixture of nickel cobalt oxide cathode, MCMB graphitic anode, and a low-temperature electrolyte with a ternary mixture of solvents [26].

Basic terminology used in the study of batteries is presented below.

- State of Charge (SoC): It measures the quantity of energy that is available $E(t)$ in the battery relative to its maximum capacity C. As a convention, when the battery is full, SoC is equal to 1 (or 100 %) and is equal to 0 when it is empty [32].

$$SoC = \frac{E(t)}{C}$$

- End of Life (EOL): Moment in which the battery can no longer satisfy the requirements or the desired performance [25].

- Remaining Useful Life (RUL): Remaining time until the battery falls onto the failure threshold [25].

- Depth of Discharge (DoD): Amount of charge drawn from the battery Q_d due to its maximum available charge C [28].

$$DoD = \frac{Q_d}{C} \cdot 100\,\% = 100\,\% - SoC$$

- State of Health (SoH): Battery's actual condition and the capacity to meet certain performance when comparing it to a new battery. Thus, it describes the relationship between the current capacity of the battery C_a and the nominal capacity C_N [28].

$$SoH_a = \frac{C_a}{C_N} \cdot 100\,\%$$

Batteries degrade over time; this is worth taking into account since degradation causes power and capacity to fade, which could reduce the efficiency of the rover. Different operating factors cause degradation, such as high/low temperature, high/low SoC, and high charge and discharge rate [29]. The internal electronics of the rover are kept between -40°C to 50°C (as explained in section 2.1.2); thus, a faster degradation of the battery is expected.

However, an autonomous navigation system can consider a SoC policy to reduce degradation. A SoC policy constrains the SoC to operate within a certain range. For example, according to [29], it is stated that a good SoC policy is between 20 % to 80 %. In [19], states that for *Curiosity* the SoC policy chosen was between 40 % to 100 %. Figure 2.7 shows the energy capacity degradation of the battery by using different SoC policies.

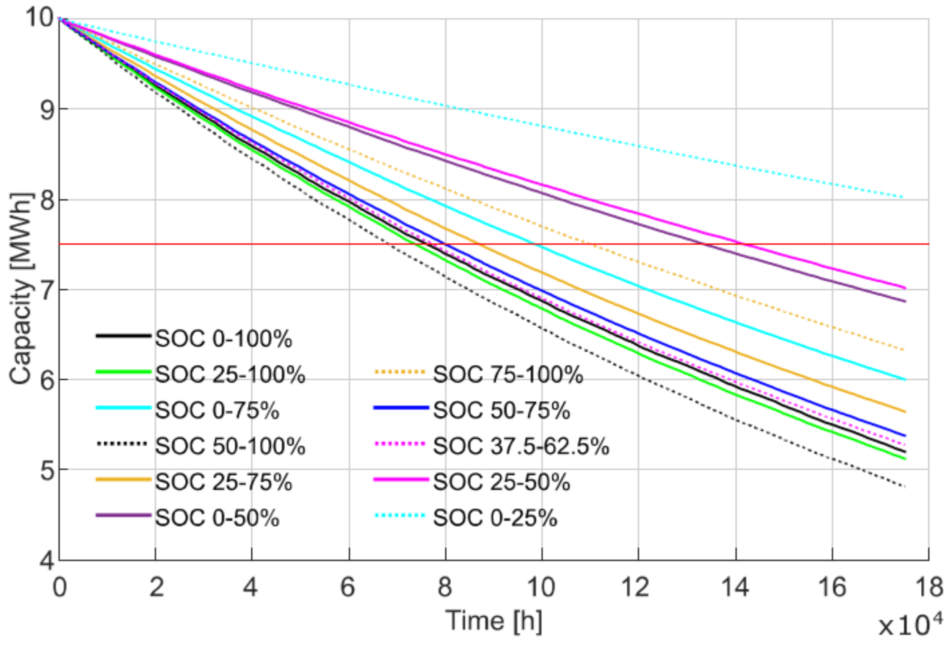


Figure 2.7: Energy capacity degradation for different operational policies [30].

2.2.2. Requirements for a Rover

Rovers *Spirit* and *Opportunity* have batteries that meet the following requirements — rovers subsequently sent by NASA have similar requirements — as stated in [26].

- Operation voltage from 24 to 36 V.

- Provide energy of 220 Wh during launch and 283 Wh/sol when temperature is at 0°C during surface operation.
- Have a life cycle of at least 270 cycle and 50 % of DoD at 0°C.
- Being able to admit pulses of 30 A at 50 ms in a wide range of temperatures.
- The battery array must be compact and lightweight.

2.2.3. State-Space Model

In the context of space exploration, it is critical to have a Battery Health Monitoring (BHM) system because a failure in the batteries could provoke the end of the mission. The BHM system can determine SoC, RUL and EOL, while considering SoH as a constant to simplify the mathematical problem. This is why, the BHM system is especially important for decision-making and for path planning [31].

Table 2.1: State-Space Model nomenclature

Nomenclature	Definition
ω_1	Process noise (internal resistance)
ω_2	Process noise (SoC)
η	Measurement noise
Δt	Sampling time
E_{crit}	Maximum nominal energy delivered by the energy storage device
i	Battery discharge current
v	Battery voltage

The model-based prognosis paradigm consists of 2 steps, estimation of the current state of the system and prediction, which based on hypotheses about the future inputs of the system. In the following, the system is presented in the state space representation based on [33] and [34] proposal.

$$x_1(k+1) = x_1(k) + \omega_1(k) \tag{2.1}$$
$$x_2(k+1) = x_2(k) - v(k) \cdot i(k)\Delta t E_{crit}^{-1} + \omega_2(k) \tag{2.2}$$

$$\begin{aligned} v(k) = v_l + (v_o - v_l)\exp\{\gamma(x_2(k) - 1\} + \alpha v_l(x_2(k) - 1) \\ +(1-\alpha)v_l(\exp\{-\beta\} - \exp\left\{-\beta\sqrt{x_2(k)}\right\} - i(k) \cdot x_1 + n(k) \end{aligned} \tag{2.3}$$

The evolution of the internal resistance of the battery is described in (2.1), where x_1 represents the internal resistance of the battery and ω_1 is the process noise. The evolution of the state of charge of the battery, SoC, is described in terms of the power delivered by the battery to the motor in equation (2.2), where x_2 is the SoC value (in a range from 0 to 1), v is the measured voltage battery in the terminals at time k, i is the current measured at time k, E_{crit} is the expected total energy delivered and ω_2 is the process noise for the SoC evolution. The observation equation is shown in equation (2.3), this expression characterizes the voltage measured as a function of SoC and the internal resistance. The parameters $\alpha, \beta, \gamma, v_l$ (y-intercept of the curve) and v_o (open circuit voltage when the battery is fully charged) are obtained empirically and depend on the array of lithium-ion cells used in the battery pack. These parameters characterize the open circuit voltage curve, shown in Figure 2.8, which is divided in three zones. Zone 2 is modeled as $v_L + \alpha \cdot v_L(x_2(k) - 1)$, thus, $v_L \cdot \alpha$ represent the slope of the curve. Zone 1 and Zone 3 have a different behavior which is why parameters γ and β are added to the equation to minimize MSE for each zone respectively. Equations (2.1), (2.2) and (2.3) are used when the battery is in the process of discharge, when the battery is charging the sign of the current must be inverted.

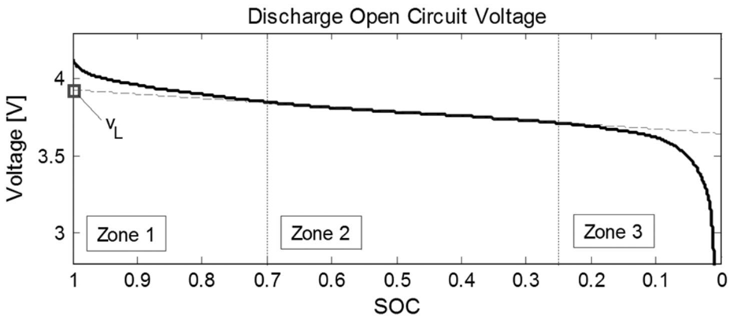


Figure 2.8: Example of Li-ion cell discharge open circuit voltage as a function of SoC [33].

2.3. Terrain Classification

Terrain identification is critical when developing an autonomous navigation system so the robot can operate safely and minimize the risk of mission failure. For example, *Spirit* took a path with deformable soil that had not been detected in orbital or terrestrial images. In addition, *Curiosity* experimented with mobility difficulties when traversing a terrain of sharp rocks that damaged its wheels.

The uncertainty about the terrain's condition makes it difficult to plan the route that the rover should take. An important concept is traversability. Traversability is the ability of the terrain to hold up a vehicle without reaching the failure point, which depends on both the terrain and the rover. Up to date, many different approaches have been used to tackle this problem. These approaches reckon with limitations of time, energy, and computational cost. One of the most common approaches used to determine the traversability of the terrain is the estimation and prediction of slippage.

Due to technological advances, specifically in Artificial Intelligence, new techniques are getting more promising results. In 2016, NASA JPL developed a new software called Soil Property and Object Classification (SPOC) that is based on Deep Learning and can identify the different types of terrain on the Martian surface using computer vision [35]. This model was trained using the high-resolution images of HiRISE (High Resolution Imaging Science Experiment). In particular, the images of *Columbia Hills* — candidate landing site for mission M2020 — were used to evaluate the performance of SPOC. This site has a size of $40x40km$, which corresponds to 25 gigapixels. On *Columbia Hills*, they identified eight terrain types. The results are detailed in Table 2.2 with their respective percentage of instances found in the dataset.

Table 2.2: Terrain classification of *Columbia Hills*

Terrain type	% of data
Smooth regolith (SR)	19.8 %
Smooth outcrop (SO)	5.4 %
Sparse ripples firm (SRF)	2.5 %
Rough regolith (RR)	11.8 %
Rough outcrop (RO)	9.6 %
Sparse ripples sandy (SRS)	20.8 %
Dense linear ripples (DLR)	6.0 %
Polygonal ripples (PR)	24.1 %

In [7] an algorithm for path planning is proposed. The first stage of this algorithm works offline and generates a set of paths that can be recalculated during the second stage. This second stage works online and recalculates the path if unforeseen obstacles are encountered. Because this thesis aims to develop an offline decision-making system, only the first stage of the algorithm proposed in [7] is considered.

To define a set of feasible and optimal paths, let us consider that the exploration site is divided as a grid, where each of the cells belongs to a terrain category. Each category considers the properties of the terrain; therefore, a cell can be traversed at a certain estimated velocity.
This estimation of the velocity at which the rover can move is calculated by combining maps of Cumulative Fractional Area (CFA), which corresponds to the fraction of the area covered in rocks, the Digital Elevation Model (DEM), and the traversability of the terrain. The characteristics of each category and their respective estimated velocities are detailed below.

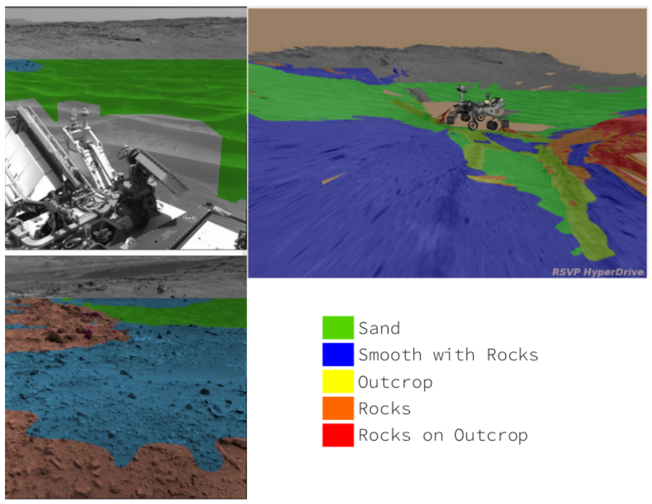


Figure 2.9: Example of terrain classification using SPOC [35].

- **Category A** consists of benign terrains (SR +SO) with a CFA between 0 and 7 % and an elevation less than 15°. The expected velocity of the rover in this terrain is $64.8\frac{m}{hr}$.
- **Category B** consists of benign terrains with a CFA between 0 to 7 % if the elevation goes from 15 to 20°. If the CFA is between 7 and 15 %, the slope can range from 0 to 15°. The expected velocity is $52.5\frac{m}{hr}$.
- **Category C** composes rough terrains (RR+RO) with a CFA from 0 to 15 % and a slope from 0 to 20°. The range of velocities of this category is 24.2-$48.5\frac{m}{hr}$.
- **Category D** includes sandy terrains (SRF+SRS), with a CFA from 0 to 15 % and a slope less than 10°. The expected velocity is in the range of 10.9-$40.8\frac{m}{hr}$.
- **Category E** includes benign, rough, and sandy terrains if the slope is between 20° to 25° with a CFA from 0 to 15 %. The expected velocity is $10.9\frac{m}{hr}$.
- **Category F** composes sandy terrains with a slope from 10 to 20°, regardless of the CFA. It also includes any category of terrain with a CFA greater than 15 % and a slope greater than 25°. This category is untraversable, which is why any cell that belongs to this category cannot be included in a potential path.

2.4. Path Planning

Nowadays, robots are becoming increasingly more important in our daily lives, from robots in the industry to smart home devices. An important aspect to ensure a near optimal performance is to successfully solve the navigation challenges that a robot faces in its respective environment. Path planning, also called motion planning, refers to the task of finding a path when the goal position is known [50]. The nature of the path planning depends on the context in which the robot its performing the task, in the following a brief description of the factors to consider when solving a path planning problem is shown.

- **Environment nature**: The problem can consider either a static or dynamic environment. In a static problem the position are fixed and obstacles do not change overtime. On the other hand, in a dynamic problem the position of the goal position and/or the obstacle might vary through the search problem; therefore, a dynamic environment represents a more complex problem.
- **Map knowledge**: Depending on the knowledge of the map, the problem can be considered a global path planning or a local path planning problem. In a global path planning situation, the robot has previous knowledge and the environment is modeled as a map; thus, a path is generated before moving towards the goal position, which means that the process is done offline. On the other hand, in a local path planning situation the robot is uncertain about the environment and relies on its sensors to estimate a map in real time while also performing a search process to avoid obstacles and find an optimal path to the goal position.

- **Completeness**: A path planning algorithm can be exact, if it can find an optimal solution or if it can prove that there is not a feasible solution, or it can be heuristic, which means that the algorithm search for a good quality solution.

2.4.1. Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a widely known problem in optimization and it has many applications and variations. In the field of robotics, it has gain attention due to its use for path planning problems [36].

Lets start by giving a brief description of TSP. Assume we have a set $\{c_1, c_2, ..., c_N\}$ of cities and each pair of cities has a given distance $d(c_i, c_j)$. The goal of this problem is to find an ordering π of the cities so each city is visited only once and finishing the path in the same city where it started [38]. Therefore, the goal is to minimize the expression shown in (2.4), in which $d(c_{\pi(i)}, c_{\pi(i+1)})$ represents the distance between two subsequent cities and $d(c_{\pi(N)}, c_{\pi(1)})$ represents the distance between the last and first city, since TSP requires that the trip must finish where it started.

$$\sum_{i=1}^{N-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(N)}, c_{\pi(1)}) \tag{2.4}$$

TSP is a non-trivial problem and it can be computational expensive since the number of possible round trips is $(N-1)!$. For example, if we have 11 locations that we want to visit, then there are $3,628,800$ possible round trips. Therefore, different strategies are used in order to find an optimal solution to TSP, such as, greedy algorithm, ant colony, genetic algorithms, etc [37]. Figure 2.10 shows an example of a visualization of TSP considering 7 locations to visit.

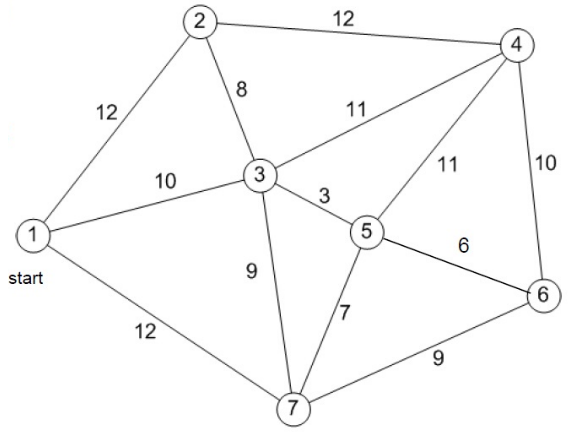


Figure 2.10: Example of a diagram for TSP [36].

2.4.2. The Shortest Path Problem

One common problem in robotics is to find the shortest path between a start point and a target point. The shortest path problem (SPP) has many variants, one of them is to find the shortest path while avoiding obstacles. In the following subsection, we present a brief description of SPP based on the formulation of [39] from vertex to vertex.

Lets consider a directed graph $G = (V, A)$, where V is the vertex set and A is the arc set. The arc set corresponds to the weight to travel from vertex v_i to v_j, this weight can represent energy consumption, travel time or distance, to name a few. The directed graph G is described by the matrix $D = [c_{i,j}]$, where $c_{i,j}$ is the cost of the arc that joins v_i and v_j; thus, $c_{i,i}$ is equal to zero. If there is no arc from i to j, then $d_{i,j} = \infty$.

Lets remember that, by definition, a path cannot have a vertex more than once, which means that a path does not intersect itself. A visual example of a path in a directed graph is shown in Figure 2.11, the directed graph has 5 vertex and the path is drawn by the solid lines. Therefore, the cost of the path C_p is the sum of all the arcs cost in the path P. In the example this would mean that $C_p = c_{13} + c_{34} + c_{45}$. Finally the SPP aims to find an optimal path that minimizes C_p.

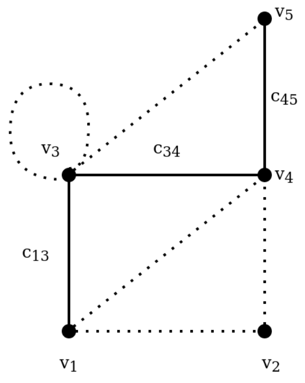


Figure 2.11: Example of a path in graph theory.

2.5. Genetic Algorithms

There are many current problems where it is required to optimize or improve the performance of a particular system or application. For example, you can lower costs, improve efficiency, or extend the useful life of a component. The cost function represents what is being optimized. In this way, an optimization problem can be modeled where an optimum point is sought that can maximize or minimize the cost function.

Metaheuristic is a group of methods focused on solving problems known as 'hard optimization'. Among the metaheuristic methods are tabu search, ant colony algorithms, simulated annealing, and genetic algorithms, to name a few [40].

Genetic Algorithms (GA), also called Evolutionary Algorithms (EA), are inspired by biology, specifically by Darwin's theory. A GA randomly generates several individuals that constitute the initial population. Each individual represents a solution to the optimization problem. The individuals that have a better performance, which means that it is a more optimal solution to the problem, are the ones that pass their genes to the next generation. The number of n generations — also known as criteria termination — determines the number of times the algorithm is iterated. The criteria termination defines when a satisfactory solution has been encountered; however, there are different ways to set the criteria termination as explained in [48] and [49], which will be discussed below. The generic GA can be visualized in Figure 2.12. In the following, the operators are explained in detail, along with considerations for this particular problem. The explanation and description of the operators are based mainly on [40] and [43].

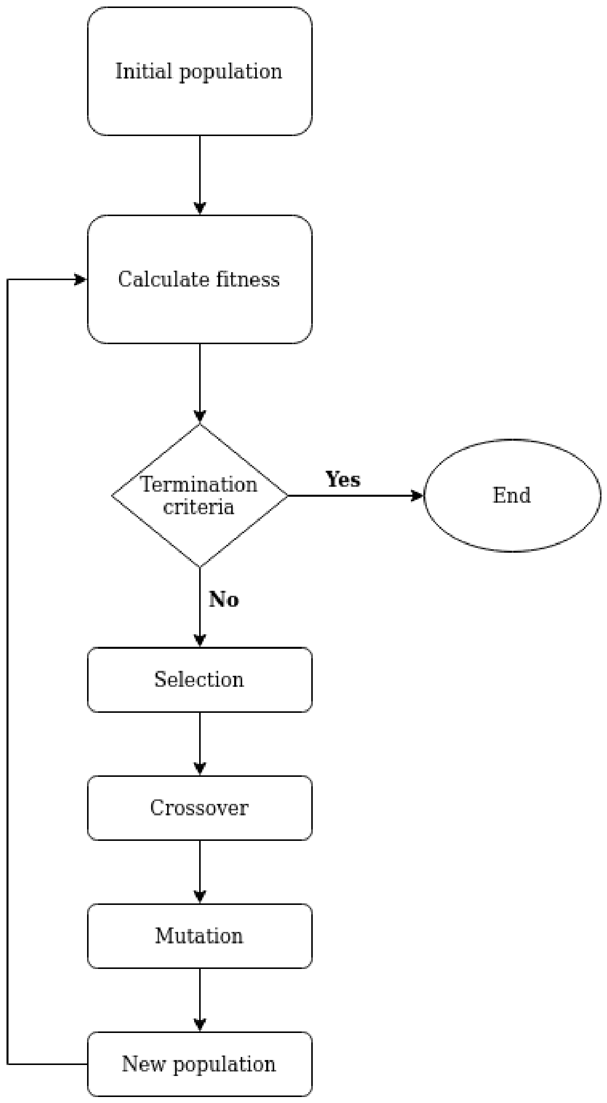


Figure 2.12: Diagram of a generic GA.

2.5.1. Selection

During each iteration, a group of individuals is selected for reproduction or replacement. The individuals with a higher fitness value (in a maximization problem) are the ones with higher probabilities of being selected. In this way, the selection operator aims to improve the average fitness of the next generation. It is worth noting that the selection operator can keep the diversity in the individuals to reduce the risk of a premature convergence [41]. There are different methods to select individuals; they are described below.

- **Proportional selection:** Each individual has a probability p_i of being selected. This probability is calculated based on the fitness value f_i of the individual.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2.5}$$

- **Tournament selection:** A group of t individuals is selected randomly from the population. The individuals of this group compete with each other, so the winner is one with the highest fitness value. The advantage of this method is that it maintains a high variability among the individuals of the population and is easier to implement when compared to the proportional selection method.

- **Truncation selection:** This is the easiest method to implement because, from a sorted list of fitness values, the best n individuals are selected.

2.5.2. Crossover

Two individuals are chosen, which are called parents. These parents will breed one or two children who will pass to the next generation. Usually, this operator is stochastic; thus, if the operator is used with the same pair of parents, it would produce different children.

Regarding the path planning problem, the chromosome is made from a sequence of nodes that constitute the path. The first step is to choose a random node of the chromosome. This node splits the path into two parts. The first child is generated by combining the first part of the first parent with the second parent. While the second child is generated in the same way respectively [44] [45] [46].

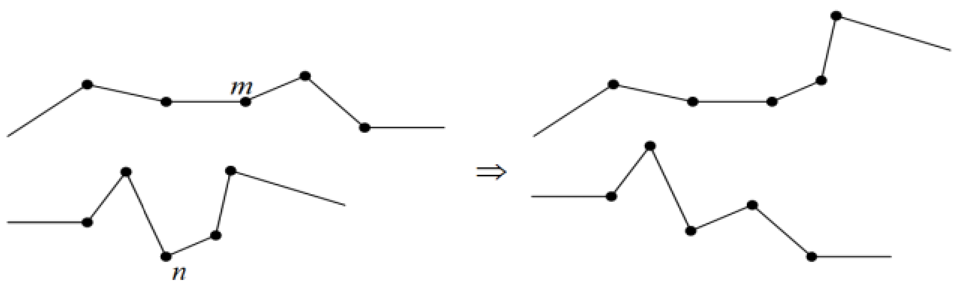


Figure 2.13: Crossover operator [47].

2.5.3. Mutation

This operator generates a random individual that replaces one of the offspring previously generated by the crossover operator. The goal of the mutation is to maintain a minimum of diversity in the population, which can not be guaranteed only using the crossover operator. In general terms, the mutation strategies modify an individual, so the new one is close to the original one. In other words, this operator executes a local search in the neighborhood of each individual to perform the mutation. The above is critical in problems where several local optimal exist. The crossover operator keeps looking in the neighborhood of the local solutions, while the mutation operator searches in new regions where the global optimum might reside.

For the path planning problem, there exist a few variations to this operator. A generic visualization on how this operator works can be seen in Figure 2.14. In [44] a node from the path is chosen randomly (it cannot be the start or goal node), and it is replaced with a node that is not in the path. In [46] a similar approach is taken, but the mutation window, which is the range of the node changes, decreases as the feasibility ratio, the percentage of feasible paths in the population, increases.

Both approaches do not reckon with the possibility that changing a node can transform a feasible path into an infeasible one. A common way to deal with this problem is to check the path's feasibility after the mutation, and if the new path is unfeasible, then apply a new mutation until a feasible one is generated. However, in [45], a new mutation procedure is proposed. After randomly choosing one node to mutate, a set that contains all feasible neighbor nodes of the mutation node is defined. This set of nodes generates a set of paths, and their fitness value is calculated. The original mutation node is replaced with the mutated node that produces the path with the best fitness value. This method has a more fast convergence than other mutation methods.

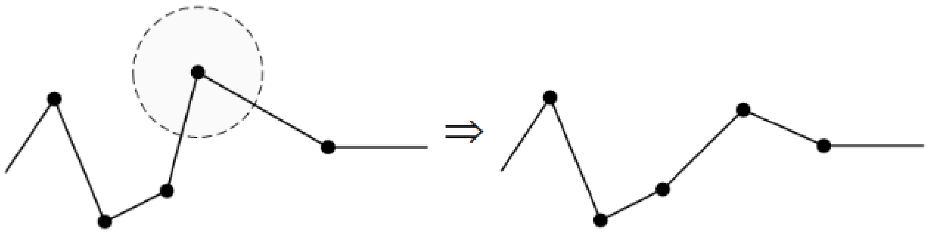


Figure 2.14: Mutation operator [47].

2.5.4. Additional Operators

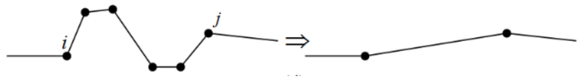
The three operators explained above are enough to build a GA. Nevertheless, for the path planning problem, additional operators are used as mentioned in the literature [44], [45], [46] and [47].

- **Node repair**: This operator moves a node that falls on an obstacle by performing a local search in the neighborhood around it.

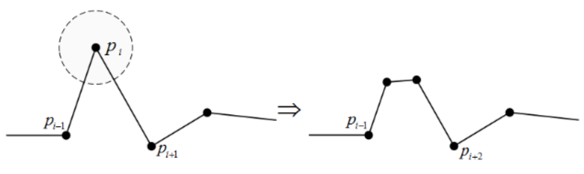
- **Line repair**: It fixes an infeasible line segment. A local search is performed to insert a suitable node between the two nodes of the segment.

- **Deletion**: This operator, which is also called shortcut, is applied to both feasible and infeasible paths. If the path is infeasible and after the deletion of a randomly chosen node, it is transformed into a feasible one, then the random is deleted. In the case of a feasible path, if the deletion of the node increases the fitness value of the individual, then said node is removed, as shown in Figure 2.15.a.

- **Smooth**: It is applied to feasible paths. The operator, shown in Figure 2.15.b, tries to smooth sharp turns of a path to reduce the cost of the solution. The operator acts based on the value of the angle θ of the turn, and it may insert a node to accomplish this.



(a) Shortcut operator.



(b) Smooth operator.

Figure 2.15: Additional operators [47].

2.5.5. Penalty Functions

As mentioned above, GA solves a hard optimization problem. This type of problem typically has constraints that can be linear or non-linear and unequal o equal. These constraints define the feasible region where the solution can take place. For general purposes, let us define a non-linear programming problem as,

$$\begin{aligned} \min_{x} \quad & f(\vec{x}) && \text{(2.6a)} \\ \text{subject to} \quad & g_i(\vec{x}) \leq 0, \quad i = 1, ..., n && \text{(2.6b)} \\ & h_j(\vec{x}) = 0. \quad j = 1, ..., p && \text{(2.6c)} \end{aligned}$$

If you consider this, a question arises: How to handle constraints when programming a GA? One idea is to consider penalty functions. These functions transform a constrained-optimization problem into an unconstrained one by adding a value to the objective function, as shown by (2.7). The most used penalty functions are the ones that use an *exterior* method. The exterior method starts with an infeasible solution, and from there, it moves towards the feasible region [42].

$$\phi(\vec{x}) = f(\vec{x}) \pm \left[\sum_{i=1}^{n} r_i \times G_i + \sum_{j=1}^{p} c_j \times L_j \right] \tag{2.7}$$

The expression of (2.7) is a generalization, where ϕ is the new objective function, r_i and

c_j are penalty factors, and G_i and L_j are functions of their respective constraints. The most common functions are shown in (2.8) and (2.6c)

$$G_i = max[0, g_i(\vec{x})]^{\beta} \tag{2.8}$$

$$L_j = |h_j(\vec{x})|^{\gamma} \tag{2.9}$$

Death Penalty [43] is the most used penalty function due to its simple implementation and computational efficiency. This penalty rejects all infeasible individuals by assigning them a fitness equal to zero if the solution violates a constraint. Although Death Penalty is used widely, it has some drawbacks. For example, it is limited to problems in which the feasible search space is convex, and the feasible search space is big in the proportion of the whole search space.

There is a great variety of types of penalty functions that are more useful depending on the characteristics of the problem. Some of them are static penalties, dynamic penalties, annealing penalties, adaptive penalties, and co-evolutionary penalties, to name a few. Each type also has multiple approaches, as is well documented in [42].

We are going to focus on two types of penalty functions, static penalties, and dynamic penalties. Static penalties are the ones where the penalty factors remain constant throughout the process of the algorithm. Instead, dynamic penalties include in their function a parameter that allows computing the function's value using the number of the current generation. In most cases, the value of the function increases with the number of generations to eliminate all infeasible individuals at the end of the evolutionary process.

2.5.6. Termination Criteria

Since the GA is an iterative process, it is important to choose appropriate termination criteria so that GA does not end prematurely nor compromise the computational budget available. There are different types of termination criteria; the main ones are direct and derived [48][49].

Direct termination criteria is a group of termination criteria where no further calculations are needed, and the stop condition is predefined. The most used ones are described in the following.

- **Maximal number of generations:** GA stops when a number of N_{max} generations is reached.

- **Hitting a bound:** This criterion is fulfilled when the best value of the fitness value is equal to or greater than a previously defined bound. To use this criterion is necessary to know beforehand an approximate value of the global optimum.

- **K-iterations:** It is satisfied if there is no improvement in the best fitness values through a K number of iterations. This criterion requires that the user assumes that after a number of iterations with no improvements in the fitness value, it is impossible to obtain better results.

Derived termination criteria calculate auxiliary values using data obtained through the evolutionary process. This group includes termination criteria such as running mean, standard deviation, best-worst, phi, kappa, POP-Var, ϵ-Variance. Below, the most common criteria are described.

- **Running mean:** It is fulfilled if the absolute value of the difference between the best objective value f_*^n of the current generation n and the average of the best objective values of the last n_l generations are equal to or less than a given ϵ.

$$TC_{mean} = |f_*^n - \frac{1}{n_{last}} \sum_{i=n-n_l}^{n-1} f_*^i| \leq \epsilon \tag{2.10}$$

- **Standard deviation:** This criterion is satisfied when the standard deviation of all fitness values of the current generation of μ individuals is equal to or less than a given ϵ.

$$TC_\sigma = \sqrt{\frac{1}{\mu} \sum_{i=1}^{\mu} (f_i^n - \frac{1}{\mu} \sum_{i=1} \mu f_i^n)^2} \leq \epsilon \tag{2.11}$$

Chapter 3

Decision-Making System

3.1. Architecture

A rover's mission consists of visiting various waypoints to perform science tasks. A set $\mathcal{W}$ of M waypoints w_i is defined as $\{(x_i, y_i), r_i\}_{i=1}^{M}$, where (x_i, y_i) is the cartesian coordinate of the waypoint and r_i is the reward associated with the visit to the waypoint i [8] [25].

This work developed and implemented by simulation a decision-making system for a Mars rover. The system was able to determine the path that the vehicle must follow based on the optimization of different criteria, regarding energy, state of health of components, time, and reward of the waypoints that need to be visited. Figure 3.1 shows the architecture of the PDM system.

Due to the complexity of the problem, it was divided into two sub-problems. The first sub-problem, GA-1, decides the amount of N ($N \leq M$) waypoints to be visited and the order, with the limitation of the maximum time T_{end} given for the mission. Therefore, M represents the number of all the waypoints and N represents number of waypoints that the PDM systems chooses to visit given the constraints. The second sub-problem, GA-2, receives two subsequently waypoints, (w_i, w_{i+1}), and finds an optimal path between them; thus, the GA-2 algorithm is iterated $N-1$ times. After $N-1$ iterations, all sub-routes are joined to form the complete route that traverse the N waypoints. The complete route is used to calculate the charging plan by estimating the SoC through the path. The charging plan is a list of nodes where the rover stops to charge its batteries.

3.2. Problem Formulation

Consider a map grid $\mathcal{M}$ with length m and width n. The set $\mathcal{A}$ contains all the arcs that connect each point, i.e $\mathcal{A} = (i, j) : \forall i, j \in \mathcal{M}, i \neq j$. The set $\mathcal{W}$ contains M point coordinates (a, b) that correspond to waypoints; each waypoint has an associated reward $r_{a,b}$.

$$r_{a,b} = \begin{cases} 0, & \text{if } (a,b) \notin \mathcal{W} \\ > 0, & \text{if } (a,b) \in \mathcal{W} \end{cases} \tag{3.1}$$

The rover's mission is to visit all the chosen waypoints in a given time T_{end}, which is the

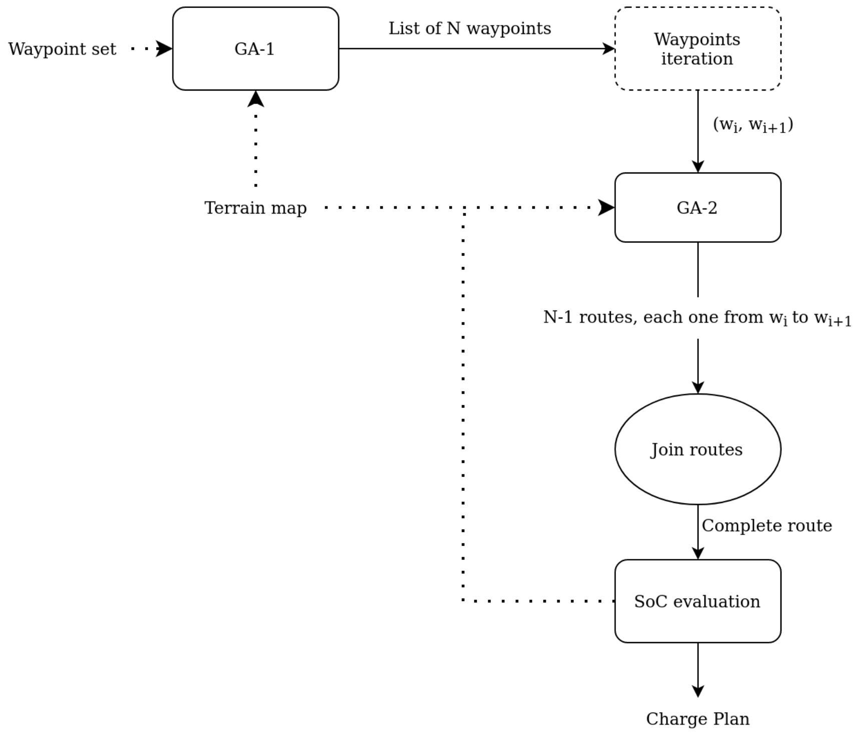


Figure 3.1: System architecture.

expected time to fulfill a mission. Typically it has a range from 90 sols to 1 martian year. It is important to state that is not necessary that the vehicle ends the mission where it started. The rover's position is denoted as $p_{a,b}$.

Each arc (i, j) belongs to a certain terrain class, defined in the section 2.3. This class gives information about the velocity with which the vehicle can move and the amount of energy required to do so. Therefore, the arc (i, j) defines the rover's travel time $t_{i,j}$ and the energy $E_{i,j}$ that the battery must provide.

Taking into account section 2.2, certain operating factors cause a faster degradation to the battery. Thus, a SoC policy is included so the batteries operate in a range that preserves their functionality and health. The policy is defined by the range $[\alpha^-, \alpha^+]$, where $\alpha^-, \alpha^+ \in [0, 1]$ such that $\alpha^+ > \alpha^-$.

The rover can stop at any place to recharge the batteries from the energy provided by MMRTG, since the energy that is transformed to electricity is constant. When it needs to recharge, the robot enters sleep mode, and all the activities are halted except for such functions as maintaining the right temperature inside the rover, as was explained in section 2.1.1. The SoC of each battery while recharging and the time it needs to do so is calculated using the same expression to discharge the battery but with the opposite sign in the current. Once the rover stops, it recharges its batteries until a SoC equal to 1 is reached. The MMRTG converts constantly $114[W]$ (every hour) to electricity but the rovers electricity needs on sleep mode can vary from 45 to $70[W]$; therefore, the electricity requirement is modeled as a uniform distribution.

Three variables define the entire operation of the rover: the sequence of nodes $Path$, which defines the route; the visited waypoints CP, which allows us to measure the level of success or failure of the mission; and the charging plan B, which makes the schedule for when and where the batteries need to be charged.

The problem is to find a set of Pareto optimal routes to fulfill the mission considering all the constraints. The criteria to choose this set of routes is to minimize the time and the energy required and to maximize the rewards associated with each waypoint. The problem solution must deliver the optimal sequence of nodes $Path^*$ and the optimal set CP^* of visited waypoints. A feasible solution satisfies the following:

- A waypoint cannot be visited more than once.
- The route cannot consider traverse in the terrain of class F.
- The rover stops its operations once it fall outside the allowed SoC.

3.2.1. GA-1

GA-1 optimizes the rewards associated when visiting a waypoint and the time that it takes to visit them. The result of GA-1 is $CP = [(w_{a,b})_0, (w_{a,b})_1, ..., (w_{a,b})_n]$. Therefore, the accumulated reward of the entire path is defined by (3.2).

$$J_1 = \sum_{i=0}^{N} (r_{a,b})_i \tag{3.2}$$

To calculate the time $t_{i,j}$, which is the time it takes to travel through the arc (i,j), it is needed to know the velocity at which the rover can move through a certain cell of the map. The velocity is estimated depending on the category of the arc (i,j). Below, the velocities are defined for each category. Categories C and D are defined by an uniform distribution denoted as $\mathcal{U}$.

$$v_{i,j} = \begin{cases} 64.8[\frac{m}{hr}], & \text{if the arc } (i,j) \in \text{category A} \\ 52.4[\frac{m}{hr}], & \text{if the arc } (i,j) \in \text{category B} \\ \mathcal{U}(24.5, 48.5)[\frac{m}{hr}], & \text{if the arc } (i,j) \in \text{category C} \\ \mathcal{U}(10.9, 40.8)[\frac{m}{hr}], & \text{if the arc } (i,j) \in \text{category D} \\ 10.9[\frac{m}{hr}], & \text{if the arc } (i,j) \in \text{category E} \\ 0[\frac{m}{hr}], & \text{if the arc } (i,j) \in \text{category F} \end{cases} \tag{3.3}$$

For each movement, the time $t_{i,j}$ which is the time it takes to get from position p_i to position p_{i+1} is stored in a vector T of length l. The total time travel also takes into account the time of operation of each waypoint. Thus, the total time is defined by (3.4).

$$J_2 = \sum_{k=0}^{l} (t_{i,j})_k \tag{3.4}$$

Since we want to maximize the accumulated reward J_1 and minimize the total travel time J_2, the multiplicative inverse of the accumulated reward is taken into account, as shown in

(3.5). The weights ω_1 and ω_2 tune the importance of the variables.

$$J_{GA-1} = \omega_1 \sum_{i=0}^{N} \frac{1}{(r_{a,b})_i} + \omega_2 \sum_{k=0}^{l} (t_{i,j})_k \tag{3.5}$$

The constraints for this sub-problem are the following.

$$p_0 = (a_{landing}, b_{landing}) \quad \text{Rover starts the path in the landing position.} \tag{3.6}$$

$$\sum_{k=0}^{l} (t_{i,j})_k \leq T_{end} \quad \text{Maximum allowed time of the operation.} \tag{3.7}$$

3.2.2. GA-2

The second problem, GA-2, only considers two adjacent waypoints in the set CP, recursively until the last point. For example, it takes the first two $(w_{a,b})_0$ and $(w_{a,b})_1$, to solve the path planning problem considering that $(w_{a,b})_0$ as the start point and $(p_{a,b})_1$ as the goal point, without dealing with any other waypoint. This second sub-problem optimizes the energy and time, with the constraint of the SoC of the batteries and the outcome is a fine-tuned route between the waypoints $(w_{a,b})_i$ and $(w_{a,b})_{i+1}$, which is a sub-set P_i. Accordingly, the optimal set of nodes P^* is the result of the concatenation of all the sub-sets P_i^*.

The energy $E_{i,j}$ used for moving the rover through the arc (i, j) depends on the category of the terrain and the velocity. The rover has a peak of $500[Wh]$ while driving and a minimum of $150[Wh]$. Therefore, if the terrain is rougher then more energy will be consumed. The range $[150, 500]Wh$ is evenly divided into 5 categories of terrain (category F is not considered in a feasible path). Then, a ratio is calculated between the actual velocity at which the rover moves and the nominal velocity for that category of terrain as given by (3.3). Finally, the energy $E_{i,j}$ is the result of the linear equation $E_{i,j} = x \cdot m + b$, where x is the ratio previously mentioned, m is the slope of the curve and b is the offset, which is given by each category. The total energy consumption of each sub-path is defined in (3.8).

$$J_3 = \sum_{k=0}^{l} (E_{i,j})_k \tag{3.8}$$

The total time of each sub-path is calculated in a similar way to the GA-1 algorithm. The only difference is that the operation time is not considered because the GA-2 aims to optimize a path between waypoints, as shown in equation (3.9). It is worth mentioning that the sub-paths between two waypoints are not the same as the path that GA-1 obtains because GA-2 does a fine-tuning for each sub-path; meanwhile, GA-1 aims to find an optimal set of waypoints to visit.

$$J_4 = \sum_{k=0}^{l} (t_{i,j})_k \tag{3.9}$$

GA-2 goal is to minimize both the energy consumption $E_{i,j}$ and the time $t_{i,j}$ for each sub-path. Expression (3.11) shows the cost function for the optimization problem.

$$J_{GA-2} = \omega_3 J_3 + \omega_4 J_4 \tag{3.10}$$

$$J_{GA-2} = \omega_3 \sum_{k=0}^{l} (E_{i,j})_k + \omega_4 \sum_{k=0}^{l} (t_{i,j})_k \tag{3.11}$$

The constraint of this problem is the SoC of the batteries, as mentioned before. The SoC is calculated using the state space representation shown in section 2.2.3. Once the value of SoC is equal to or below α^-, the second battery is used until it falls below the policy. When the two batteries have a SoC less than the lower bound of the policy α^-, then the rover must stop to recharge its batteries. The batteries are recharged until they reach a SoC equal to α^+; the time it takes to recharge batteries is T_{charge}. Equations from (3.12) to (3.17) represent all the constraint of the GA-2 problem.

$$p_s = (w_{a,b})_i \quad \text{The path starts at waypoint i} \tag{3.12}$$
$$p_g = (w_{a,b})_{i+1} \quad \text{The path ends at waypoint i+1} \tag{3.13}$$
$$SoC_1(0) = \alpha^+ \quad \text{Battery 1 SoC at the landing position} \tag{3.14}$$
$$SoC_2(0) = \alpha^+ \quad \text{Battery 2 SoC at the landing position} \tag{3.15}$$
$$\alpha^- \leq SoC_1(k) \leq \alpha^+ \quad \text{SoC policy for battery 1} \tag{3.16}$$
$$\alpha^- \leq SoC_2(k) \leq \alpha^+ \quad \text{SoC policy for battery 2} \tag{3.17}$$

3.3. Map Generation

The terrain map is a weighted matrix, which represents the cell category. Each cell represents 1 square meter and can have a value from 0, which is category A (the most benign terrain category), to 5, which is category F (an untraversable terrain in AutoNav mode that is considered an obstacle).

The matrix is generated by taking into account the probability of each terrain category and the set of waypoints for the mission. First, the matrix is filled with the terrain categories of the waypoints. So when a cell is a waypoint the probability of category A is given by $1 - (p_B + p_C + p_D + p_E)$, since is impossible that a waypoint is at the same time an obstacle (category F). The three nearest neighbors of the waypoint cell take the same category. If a cell is not a waypoint then the terrain category is given by randomly choosing a number based on the probability of occurrence of each category $p_A, p_B, p_C, p_D, p_E, p_F$. The three nearest neighbors take the same category if they are not already given a category.

As mentioned before, the rover can travel at a certain speed that is given by the terrain category. We considered the autonomous navigation problem is stochastic; thus, we prognosticate the expected velocity for each terrain category.

Categories C and D are within a range of values $[24.5, 48.5]$ and $[10.9, 40.8]$, respectively. Therefore, it is assumed that a uniform distribution describes the behavior of the rover's speed in these terrains. However, when evaluating a path we cannot take a speed given by a uniform distribution since the results may be misleading. For example, let us consider that we have two candidate paths; each one of them is made up of three nodes. Path 1 has two nodes that belong to category C, and one belongs to D. Path 2 has one node that belongs to category C, and two nodes belong to category D. It is logical to think that Path 1 is better and should be the one chosen by the GA. Nonetheless, it can occur that by taking the speeds randomly sampled by the uniform distribution the nodes get the following speeds, Path 1: $25.0, 26.7, 12.3$ and Path 2: $45.3, 36.5, 40.2$. In the latter example, the faster and less energy-consuming path is Path 2, but it will not be representative if we take many simulations of both paths. Thus, it is necessary to take a statistic of the speed for each terrain.

An option to compute the statistic may be the mean of the distribution; however, in this context is not appropriate since we want to make a conservative estimation considering that the cost of underestimating the time or energy that a path takes may be critical for the mission. Taking that into account, we take the speed value that marks the 10 % lower probability of occurrence, which is calculated using the cumulative distribution function. Since categories A, B and E have a fixed speed value, we add Gaussian noise and take the same statistic. Finally, the speed used in GA-1 and GA-2 to generate paths is shown in Table 3.1.

Table 3.1: Speed used for each terrain in Decision-Maker System

Terrain	Speed $[\frac{m}{hr}]$
A	62.33
B	50.01
C	26.62
D	13.57
E	8.35

3.4. GA Programming

3.4.1. GA-1

Initial Population

The proposed algorithm is implemented in Python. The GA uses the set of waypoints that need to be visited to create a set of N individuals. To do that, a random number n is chosen, which represents the individual's length. Let us remember that an individual would not necessarily consider all the waypoints of the original set because using all the waypoints might violate a time restriction. Then, the individual is generated by taking a random sample of size n. It is important to notice that every individual has the landing position in the first gene.

Crossover Operator

The crossover operator takes as input two individuals, which from now on are called parents. Each parent is previously chosen by a tournament selection of size 2. It randomly chooses two numbers representing a start position and an end position of the genes (in this case, a gene is a tuple of cartesian coordinates of a waypoint) of the parent A. The genes of parent A within the given range are chosen for the new individual, from now on called the child. The genes of parent B, which are not present in the child, are now passed to the child to ensure that a waypoint is only once in each individual. As mentioned before, all individuals must have the landing position as the first gene.

One particular issue with this operator is that it diminishes the diversity of length of the individuals, which means that usually, the entire population converges to have all the waypoints in their genes. The latter is a problem since it often provokes a constraint violation of the travel time. To solve this issue, each time a child is generated there is a 20 % chance of eliminating a random number of their genes. Algorithm 1 summarizes how the crossover operator works.

Algorithm 1 Crossover operator GA-1

```
Input: Parent A $PA$, Parent B $PB$
Output: Child
gene1 ← $random(1, len(PA))$
gene2 ← $random(1, len(PA))$
startGene ← $min(gene1, gene2)$
endGene ← $max(gene1, gene2)$
childPA ← $PA[startGene : endGene]$
childPB ← []
for gene in PB do
    if gene not in childPA then
        insert gene into childPB
    end if
end for
child ← [landingPosition] + childPA + childPB
/* Randomly eliminate genes */
coin ← $random(0, 1)$
if coin < 0.2 then
    nLen ← $random(2, len(waypointSet))$
    if len(child) > nLen then
        for i in range(len(child) - nLen) do
            eliminate ← $random(1, len(child) - 1)$
            del child[eliminate]
        end for
    end if
end if
return child
```

Mutation Operator

The mutation operator swaps genes within an individual according to a mutation rate. The first step is to iterate through the genes of an individual. In each iteration, there is a change of swapping the gene, given by the mutation rate. If the gene is chosen to be swapped, then a random gene is chosen to be interchanged with.

Fitness Evaluation

The fitness of each individual is calculated by using the travel time T, the accumulated reward R, and the number of collisions with obstacles $n_{obstacles}$ of the individual. The accumulated reward is the sum of all the rewards of the individual, which are given by the waypoints w_i visited.

The travel time T is calculated by extending the route, which means that a simple step-by-step route is performed to connect all the waypoints within the route. For example, if an individual has the following waypoints $[(1,1),(3,5),(10,5)]$ is necessary to know which are the points that the rover will traverse when going from the waypoint $(1,1)$ to waypoint $(3,5)$, and so on. In this example, the route that connects these two waypoints could be $(2,2),(3,3),(3,4),(3,5)$, where every point is only one step (horizontally, diagonally or vertically) away of the previous point. Figure 3.2 shows a visualization of two different valid paths to connect waypoints $(1,1)$ and $(3,5)$. In this way, for every individual an extended route is calculated, which is later used to calculate the time that the rover will take to complete a route that visits the set of waypoints given by the individual.

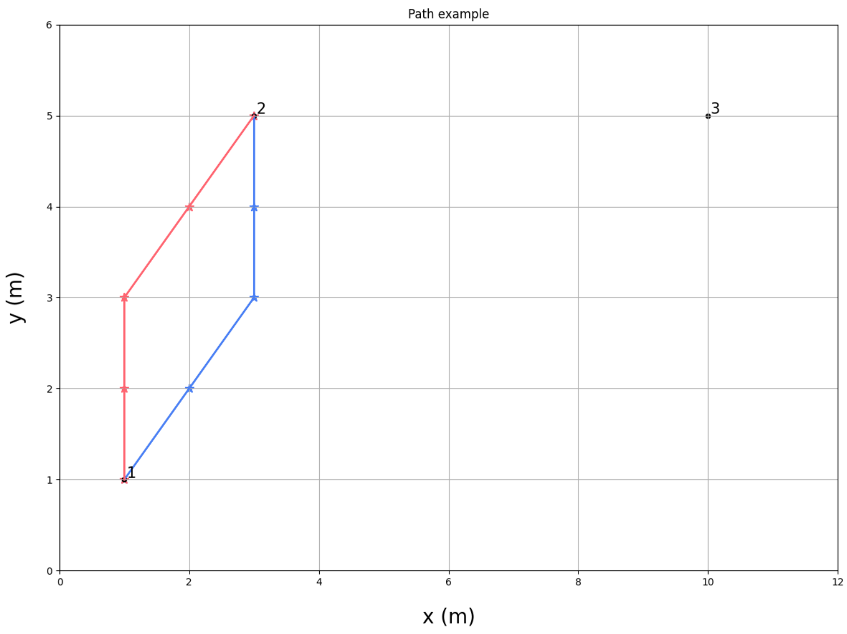


Figure 3.2: Example of paths between two waypoints.

The extended route mentioned before does not take into consideration if one of the points is an obstacle or not; therefore, the amount of collisions is calculated because it is used as penalization when evaluating the fitness of each individual. This penalization is considered because later GA-2 may be able to avoid the obstacles encountered by making a detour; however, this detour may take more time.

It is worth mentioning that the accumulated reward and the travel time have different magnitude orders causing issues of sensitivity when changing the weights of the cost function. As a consequence, it is required to perform a normalization of these variables. The normali-

zation is performed by the function MinMaxScaler from the library sklearn. MinMaxScaler takes an array of values to calculate the range in which the variables are. After the scaler is fitted, it can receive a value and transform it into the feature range.
Since the range of the travel time and accumulated reward is unknown, it is assumed that the values of these variables calculated for the initial population are representative to fit the MinMaxScaler. Thus, two scalers are created (one for the travel time variable and the other for the accumulated reward variable); each one is fitted with an array that contains all the values calculated in the initial population. It is worth mentioning that the multiplicative inverse of the accumulated reward R is used to fit the scaler.

Taking the above into account, the fitness of an individual is calculated as shown in (3.18). The variables $\tilde{R}$ and $\tilde{T}$ are the normalized variables accumulated reward and travel time, respectively. This GA has a death penalty for the time constraint; hence, if the travel time T exceeds the maximum travel time entered by the user — meaning that the time constraint is violated — then the fitness of that individual is equal to $-\infty$, which for purposes of programming is equal to $-1e30$.

$$fitness = -(\omega_1 \tilde{R} + \omega_2 \tilde{T}) * (1 + 2 * n_{obstacles}) \tag{3.18}$$

Algorithm 2 Algorithm GA-1

```
Input: WaypointList $\mathcal{W}^0$; Hyper-parameters $\mathcal{H}$; Landing position $\mathcal{L}$; Map $\mathcal{M}$
Output: Optimal amount of waypoints ordered in a list
$P \leftarrow$ initialPopulation-1($\mathcal{H}[size], \mathcal{W}^0, \mathcal{L}$);
for generation $\leftarrow$ 1 to $\mathcal{H}[max]$ do
    Fitness Rank (fr) $\leftarrow$ rankRoutes($\mathcal{P}, \mathcal{M}$)
    /* crossover */
    $\hat{\mathcal{P}} \leftarrow []$
    for $i = 1, ..., \mathcal{H}[max]$ do
        $parent1, parent2 \leftarrow$ tournamentSelection($\mathcal{P}, fr$)
        $child \leftarrow$ crossover-1($parent1, parent2$)
        insert child into $\hat{\mathcal{P}}$
    end for
    /* mutation */
    for individual in $\hat{\mathcal{P}}$ do
        individual $\leftarrow$ mutate-1($individual, \mathcal{H}[mut\_rate]$)
    end for
    insert $eliteIndividuals$ in $\hat{\mathcal{P}}$
    $\mathcal{P} \leftarrow \hat{\mathcal{P}}$ new generation
end for
$bestIndividual \leftarrow$ Individual in $\mathcal{P}$ with the highest fitness
$CP \leftarrow bestIndividual[CP]$
return $CP$
```

3.4.2. GA-2

Initial Population

Each individual of the initial population must be feasible and have a fixed start and end point. An individual consist of a random path created by the function `createIndividual`. This function takes the start point and searches for the start point position in a dictionary that has a list of feasible neighbors (meaning that it does not include obstacles as neighbors) for each position. One neighbor is chosen randomly from the list given by the dictionary for that position. Then the function checks if the chosen node is already in the path, which means that the path has a loop, which would transform the path into an unfeasible one. If this condition is fulfilled, it tries with the other neighbors on the list. If all neighbors fulfill this condition, the function returns `None`. This process is repeated with each node, until the current node is the same as the goal node, meaning that the path is complete. If a number of maximum iterations are reached, and the last node is not the same as the goal node, then the function returns `None`.

Crossover Operator

The crossover operator for the GA-2 receives two parents, which were previously chosen by a tournament selection of size 2 since it was proved empirically that for a tournament of that size converges faster than size 3 or 4. A random number is generated in range from 1 to $int(max(len(parentA), len(parentB)))/2$. The upper bound of the range was chosen from doing several experiments, with other bounds the GA losses diversity among the individuals. This number represents an index that splits the path of parent A and parent B into two pieces. The first child is given by the combination of the first part of parent A and the second part of parent B. The second child is then the combination of the first part of parent B and the second part of parent A. Figure 3.3 shows an example of the crossover operation, in this case, the number that splits the paths into two is equal to 3.

One issue with this operator is shown in Figure 3.3 in Child 2. The last node of the first part of parent B is $(5, 2)$, and the first node of the second part of parent A is $(6, 4)$; these two nodes are not neighbors of each other. If we want to make this path feasible, we need to connect these two nodes by adding extra nodes. In this example, we can add $(5, 3)$ or $(6, 3)$ to solve the issue. Thus, after generating a child the function adds extra genes to connect nodes that are not neighbors. This function does not always find a feasible connection between two nodes, which means that the function may add nodes that are obstacles.

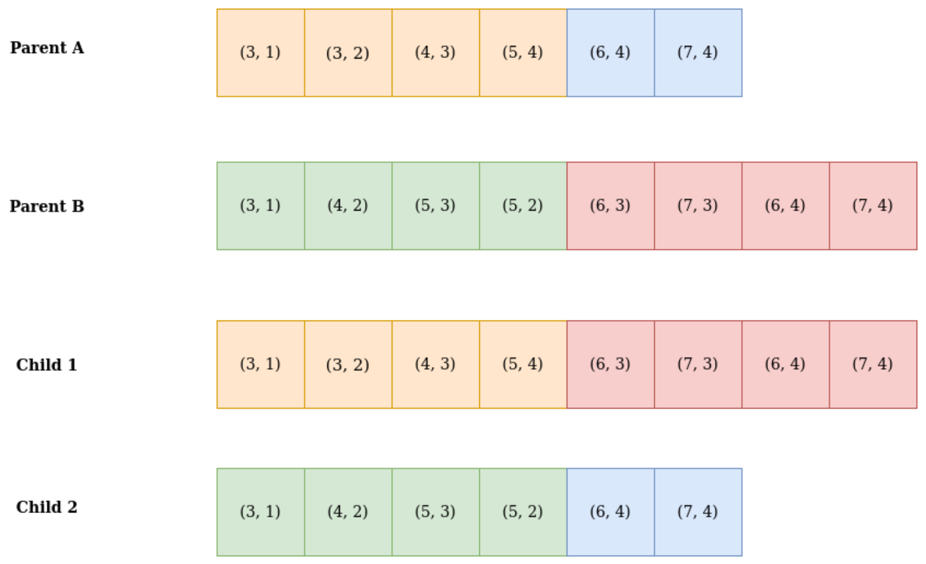


Figure 3.3: Crossover operation for GA-2.

Mutation Operator

The mutation operator iterates through each node of the individual. In each iteration, there is a chance of changing the node, given by the mutation rate. If node i is chosen to be mutated then the neighbors of the node $i-1$ are searched. The node i is then replaced with every neighbor found, creating several candidate individuals, which is equal to the number of feasible neighbors. The function that connects paths is used to add extra nodes if a candidate individual requires to do so. The fitness of each one of these candidate is calculated. The final individual is the one with the highest fitness. If the original individual has a higher fitness than the mutated individual then mutation of that gene is not performed.

Termination Criteria

The number of generations required to encounter a near-optimal solution varies greatly depending on the distance between the start point and the endpoint and their location of them. Therefore, having a fixed termination criterion, such as the number of generations, may lead to a sub-optimal solution or take extra computational resources if an optimal solution is found before reaching the required number of generations.

A solution for this issue is using the running mean termination criteria, which was explained in section 2.5. After 10 generations, the running mean is evaluated. For this problem is considered that an optimal solution is found when considering the average of the last 5 generations and taking $\epsilon = 5e-5$. The value of ϵ is chosen by testing values with different orders of magnitude, $5e-5$ was the one that showed better performance by giving a faster convergence but not compromising the quality of the results. If the running mean condition is not fulfilled after 150 generations then GA-2 ends.

Fitness Evaluation

Fitness is calculated by using the path time T and the total energy consumption E. Similar to GA-1, both variables are normalized using `MinMaxScaler`. Both scalers are fitted by using the values of time and energy calculated for every individual in the initial population.

The travel time between two nodes is calculated by dividing the euclidean distance between the two nodes by the average speed. For example, if the terrain category of node i is A and the terrain category of node $i+1$ is B then the velocity for that cell is the average of 64.8 and 52.4. Thus, the total travel time is calculated by iterating through the entire path. To obtain the path time we must calculate the time required to charge batteries. The amount of times the rover needs to charge batteries and the hours required to do so is calculated by using the expressions shown in section 2.2.3. Finally, the path time T is the sum of the total travel time and the sum of the total time required to charge batteries.
The total energy consumption is calculated by multiplying the time of going from node i to node $i+1$ with the energy consumption variable E for the given terrain, which is calculated as explained in section 3.2.2.

The fitness is then computed as shown in equation (3.19). GA-2 uses Death Penalty; therefore, if a path has an obstacle at a node or does not end at the goal point then its fitness is equal to $-\infty$.

$$fitness = -(\omega_3 \tilde{T} + \omega_4 \tilde{E}) \tag{3.19}$$

Algorithm 3 summarizes how GA-2 is programmed and its inputs and outputs.

Algorithm 3 Algorithm GA-2

```
Input: Start point $w_i$; End point $w_{i+1}$; Hyper-parameters $\mathcal{H}$;Map $\mathcal{M}$
Output: Optimal path $Path$; Charging plan $B$
$P \leftarrow$ initialPopulation-2$(w_i, w_{i+1}, \mathcal{H}[size], \mathcal{M})$;
for generation $\leftarrow$ 1 to $\mathcal{H}[max]$ do
    Fitness Rank (fr) $\leftarrow$ rankRoutes$(\mathcal{P}, \mathcal{M})$
    /* crossover */
    $\hat{\mathcal{P}} \leftarrow []$
    for $i = 1, ..., \mathcal{H}[size]$ do
        $parent1, parent2 \leftarrow$ tournamentSelection$(\mathcal{P}, fr)$
        $child \leftarrow$ crossover-2$(parent1, parent2)$
        if checkContinuity$(child)$ then
            insert child into $\hat{\mathcal{P}}$
        else
            $fixedChild \leftarrow$ fixChild(child)
            if isfeasible$(fixedChild)$ then
                insert $fixedChild$ into $\hat{\mathcal{P}}$
            end if
        end if
    end for
    /* mutation */
    for individual in $\hat{\mathcal{P}}$ do
        individual $\leftarrow$ mutate-2$(individual, \mathcal{H}[mut\_rate])$
    end for
    insert $eliteIndividuals$ in $\hat{\mathcal{P}}$
    $\mathcal{P} \leftarrow \hat{\mathcal{P}}$      /*new generation*/
    if $TC_{mean} \leq \epsilon$ then
        break      /*early convergence*/
    end if
end for
$bestIndividual \leftarrow$ Individual in $\mathcal{P}$ with the highest fitness
$Path \leftarrow bestIndividual[Path]$
$B \leftarrow bestIndividual[B]$
return $Path, B$
```

Chapter 4

Simulation Tests

4.1. Experimental Setting

We conducted three sets of experiments to test the performance of the decision-making system. For all the sets of experiments we considered three maps of $100m^2$; these maps were generated using the values shown in Table 4.1. It is also worth noting that we used the same landing position, which is (3, 1), and the same set of waypoints and rewards for every experiment, as shown in Table 4.2. Appendix 5 shows a visualization of the three maps used for experiments, with waypoints and the reward of each one.

Table 4.1: Terrain probabilities in maps generation

Map	% of obstacles	p_A	p_B	p_C	p_D	p_E	p_F
#1	1 %	0.3	0.19	0.15	0.15	0.2	0.01
#2	3 %	0.3	0.17	0.15	0.15	0.2	0.03
#3	5 %	0.3	0.15	0.15	0.15	0.2	0.05

The first set of experiments aimed to study the sensitivity of GA-1 to different scenarios, different weight combinations in the fitness function, and time constraints. The second set of experiments was similar and aimed to analyze the sensitivity of GA-2 to different maps and weight combinations. Finally, the last set of experiments considered the whole decision-making system and analyzed the performance, considering the number of times that the rover needs to stop to charge its batteries, constraint violation, and timely execution of the system.

All experiments were performed on the IDS server provided by the Electrical Engineering Department. IDS has an Intel Core i9-9900KF 3.60$[GHz]$ processor and 32$[GB]$ of RAM. The GA algorithms were programmed in Python 3.8.3.

Table 4.2: Waypoint set

Waypoint	Reward
(90, 24)	10
(15, 15)	2
(47, 59)	6
(10, 30)	1
(80, 37)	20
(8, 98)	8
(37, 60)	12
(64, 17)	7
(73, 20)	3
(53, 62)	20
(27, 3)	16
(93, 73)	40
(81, 94)	10
(87, 64)	14

Curiosity has two batteries of $43[Ah]$ nominal capacity and a nominal voltage between 24 to $32.8V$. Since we used state-space model to evaluate SoC and the output voltage, we needed a battery that was already studied and had the parameters $\alpha, \beta, \gamma, v_l$ and v_o empirically calculated. In [34] a battery with similar characteristics, as shown in Table 4.3, is used in their experiments. In the study, they evaluated the desired parameters for the state-space model, which are detailed in Table 4.4.

Table 4.3: Parameters of the battery

Battery bank	8 Li-Ion cells (GBS-LFMP40Ah) in serial connection
Nominal capacity	40 AH
Nominal voltage	25.6 V
Nominal power	4 kW
R	1 Ω
L1	2.5 mH
L2	2.5 mH

Table 4.4: Parameters for state-space model

Mode	v_L	v_o	γ	α	β
Discharging	4.971	26.792	0.556	-2.631	0.508
Charging	5.033	26.750	0.556	2.733	0.567

4.2. GA-1 Analysis

In this section, we analyze the performance of GA-1 in different maps, constraints, and the sensitivity of the algorithm when changing the weights of the fitness function. In every experiment, the hyperparameters used for GA-1 were the same and are detailed in Table 4.5, which were chosen after running several pre-tests.

The set of experiments consisted of the following, for every map three different T_{max} constraints were tested with three different combinations of weights. Thus, the set of experiments consisted of 27 tests in total. The time constraints are $T_{max} \leq 20000[s]$, $T_{max} \leq 40000[s]$ and $T_{max} \leq 80000[s]$. It is worth mentioning that GA-1 does not include the time that it takes to recharge the rover's batteries; therefore, a simple approximation is used to calculate the time that it takes to perform the entire operation. The approximation is based on the documentation of *Curiosity* [11], so for every 6 hours of driving it takes 18 hours of recharging. In other words, T_{max} is a constraint on the time that the rover is driving. The combination of weights is summarized in Table 4.6. Let us remember that w_1 multiples the variable of accumulated reward and w_2 multiplies the variable of travel time, as shown in equation 3.5.

Table 4.5: Hyperparameters of GA-1

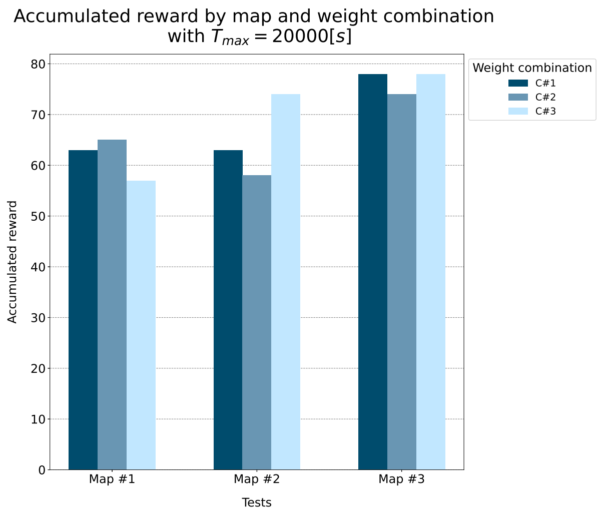
Population Size	200
Generations	20
Mutation Rate	0.05
Elite Size	10

Table 4.6: Weight combination

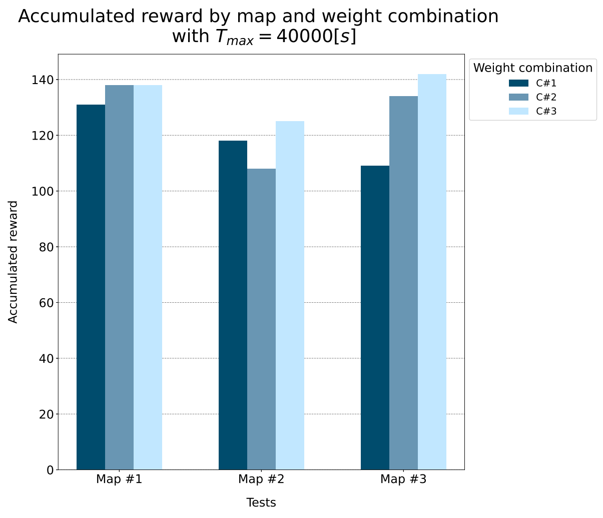
Combination	w_1	w_2
c_1	0.9	0.1
c_2	0.5	0.5
c_3	0.1	0.9

Figure 4.1 summarizes the results obtained for the accumulated reward in each experiment grouped by time constraint. As expected, when T_{max} was higher, GA-1 found solutions that had higher accumulated rewards. However, it calls the attention that in Figure 4.1.a we see that Map #3 — the map with the highest percent of obstacles — got routes with higher accumulated rewards. For the time constraint, $T_{max} \leq 20000[s]$, the routes generated by Map #1 were longer than the ones found by the algorithm for Map #2 and Map #3, yet the waypoints chosen in Map #1 had lower rewards. This behavior may be explained because the fitness function penalizes collisions with obstacles. So visiting more waypoints increases the amount of encountering an obstacle; thus, GA-1 chooses shorter routes with higher rewards to compensate. The differences in the length of the routes are important because it shows that the addition done in the crossover operator (the elimination of a random number of genes) maintains a certain diversity in the population, which allows for finding optimal solutions.

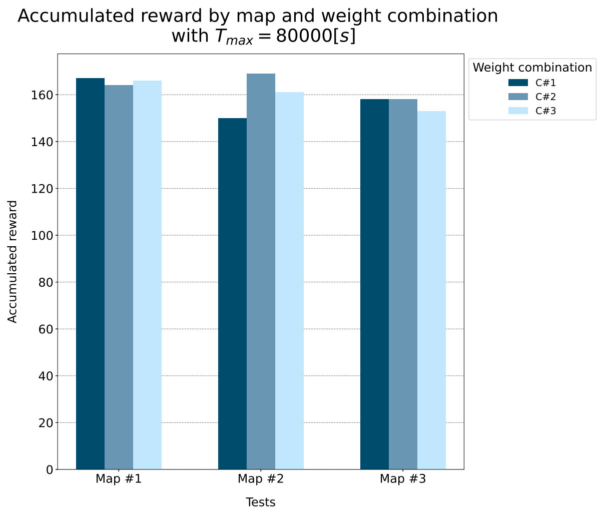
An important issue is that there is not a clear relationship between the weight combination and the value of the accumulated reward. We would expect to see higher values for combination C#1 and lower for C#3. However, in the more relaxed time constraint, we see that the accumulated rewards do not vary much between maps or weight combinations. This may be explained by the fact that the maximum accumulated reward that any route can get, is fixed and less fluctuating. However, the time variable has more space to be optimized and after reaching a certain value of the accumulated reward, it is the time variable the one that plays a bigger role in improving the fitness value, regardless of the weight combination. Further experiments need to be performed to analyze the influence of the normalization used, the mathematical formulation of the fitness function, and the sensitivity to other weight combinations, given that the normalization is done on the multiplicative inverse of the accumulated reward.



(a)



(b)

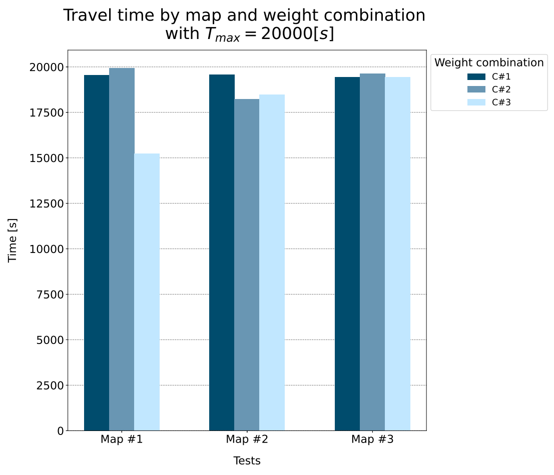


(c)

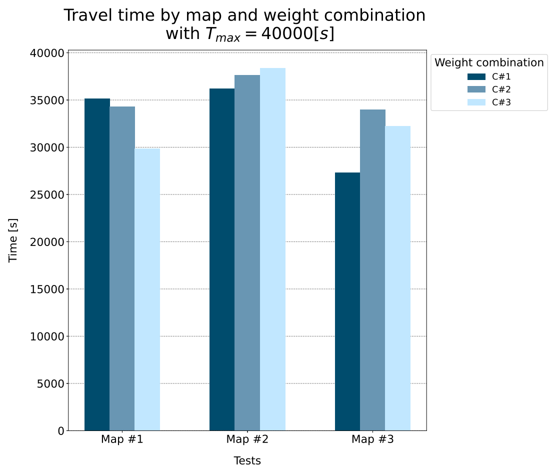
Figure 4.1: Accumulated reward analyzed by time constraint.

Figure 4.2 shows the results of the travel time for each experiment. For $T_{max} \leq 20000[s]$ all routes, except by one, took more than $18000[s]$ to complete. The solutions were close to the time restriction because it was a strict constraint that did not allow to visit many waypoints; thus, to maximize the fitness function the algorithm must choose routes that were near the maximum time allowed to provide a few rewards. For $T_{max} \leq 40000[s]$ and $T_{max} \leq 80000[s]$ we see more variation in the time, meaning that the algorithm tries to find a balance between rewards and time.

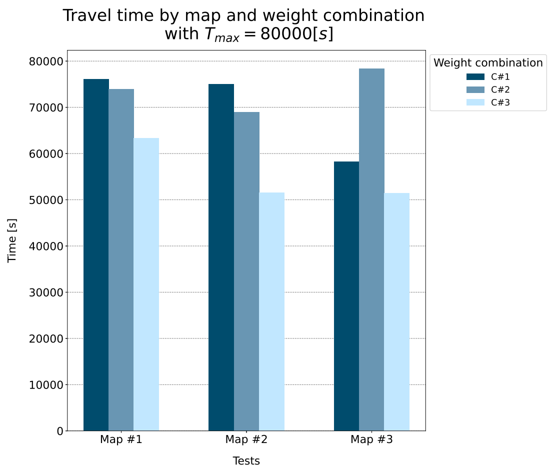
For the travel time, we see a clearer relationship between the weight combination and the value of the travel time. Results obtained with the C#1 combination show more time-consuming routes than the ones obtained with the C#3 combination, except for the results in Figure 4.2.b for Map #2 and Map #3. Let us remember that the GA algorithm does not always converge to a global optimum and can get stuck in arbitrary points or local optima. Therefore, a few outliers are expected in a limited set of 27 experiments. The results obtained with combination C#2 were more variable, meaning that when giving the same importance to the accumulated reward and the travel time the solutions were flexible regarding the context and the initial population, which influences the convergence of the algorithm.



(a)



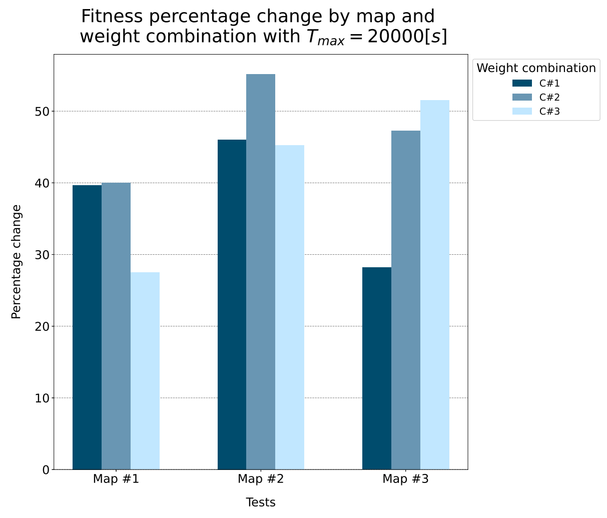
(b)



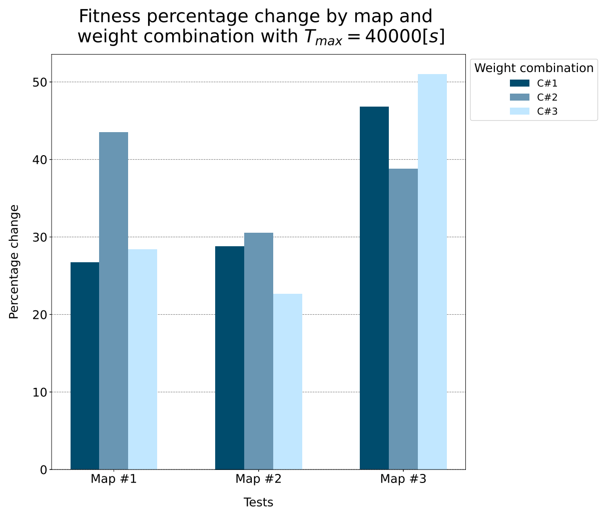
(c)

Figure 4.2: Travel time analyzed by time constraint.

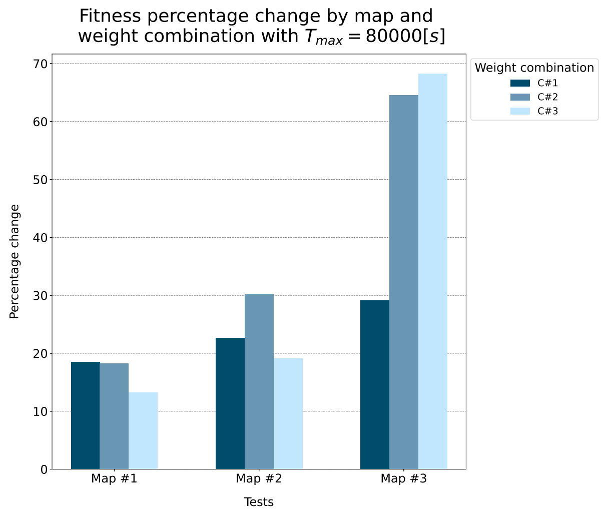
Another important aspect to analyze is the improvement of the function. Figure 4.3 shows the percentage improvement of the fitness function by calculating the percentage change between the fitness of the best individual of the first generation with the best individual of the last generation. As expected, in every experiment there was an improvement in the fitness function. For $T_{max} \leq 20000[s]$ and $T_{max} \leq 40000[s]$ there is between 20 % to 50 % change, with four out of six experiments showing more improvement by using combination C#2 when comparing to other combination tested in the same respective map. For $T_{max} \leq 80000[s]$ there was a huge improvement in the scenario of Map #3 when using combinations C#2 and C#3. The latter can be explained because in the first generation random routes were generated. Therefore, several routes may have encountered many obstacles. The number of obstacles rises more in the case of having a more relaxed constraint, which allows for visiting more waypoints, but at the same time visiting more waypoints means more obstacles than the ones found in cases with a more strict time constraint.



(a)



(b)



(c)

Figure 4.3: Fitness percentage change analyzed by time constraint.

Finally, we analyze the frequency in which the waypoints were chosen in each result obtained from the 27 experiments performed. In Figure 4.4, waypoints are sorted from the ones with the highest reward (from the left) to the ones with lower rewards (to the right), and they are grouped by maps. We can see that there is a relationship between the value of the reward and the number of times in which the waypoint is chosen. The waypoint $(93, 73)$, the one with the highest reward, appears in 88 % of the routes generated, even though it is far from the landing position $(3, 1)$.

Waypoints $(81, 94)$ and $(8, 98)$ do not follow the trend since we would expect them to appear more often in the routes. However, this behavior is explained because both of these waypoints are isolated (see Figures 5.1, 5.2 and 5.3 for a visual representation); thus, the reward for visit them does not compensate the time that it takes to reach those locations. On the other hand, we have waypoints $(47, 59)$, $(73, 20)$, and $(15, 15)$ that appear often when considering their low rewards. The latter can be explained because these waypoints are near waypoints with high rewards, and in the case of $(15, 15)$, which has a reward of the value of 2, it is the closest waypoint to the landing position.

Lastly, every waypoint is chosen at least once by each map scenario. However, there is an exception for waypoint $(10, 30)$ when using Map #3. In Map #3 (see Figure 5.3), this waypoint is surrounded by obstacles, and as mentioned before, it also has a low reward. Thus, the reward value does not compensate for the penalization that occurs if a collision happens, or in other words, the detour that GA-2 will have to take to avoid these obstacles.

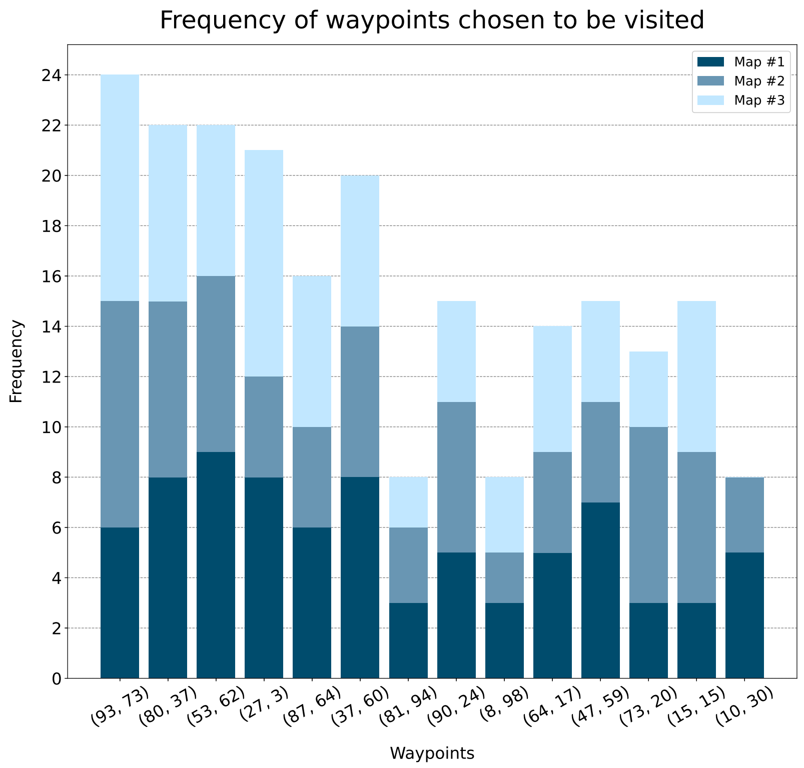


Figure 4.4: Waypoints chosen to be visited.

4.3. GA-2 Analysis

In this section, we analyze the performance of GA-2 in different maps and the sensitivity of the solutions when changing the fitness function's weights. All experiments in GA-2 used the hyperparameters in Table 4.7. It is worth remembering that ϵ is used as a termination criterion, as explained in Section 3.4.2.

Similar to the experiments performed in Section 4.2, the set of experiments for GA-2 consisted of three maps used before and three different combinations of weights. Therefore, nine experiments were performed. In this case, all nine experiments used the same start position $P_s = (3, 1)$, which corresponds to the landing position, and end position $P_e = (27, 3)$ to be able to compare the performance. The weights combination were c_1: $w_3 = 0.1, w_4 = 0.9$, c_2: $w_3 = 0.5, w_4 = 0.5$ and c_3: $w_3 = 0.9, w_4 = 0.1$, where w_3 multiplies the total travel time and w_4 multiples the energy consumption.

Table 4.7: Hyperparameters of GA-2

Population Size	100
Generations	90
Mutation Rate	0.01
Elite Size	10
ϵ	5e-5

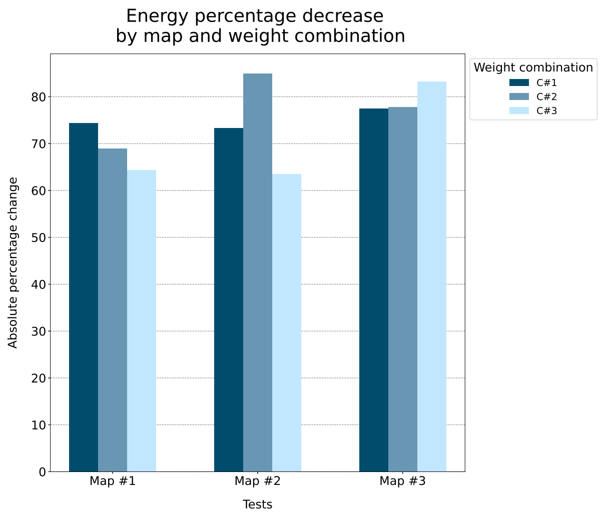
Table 4.8 summarizes the results of the set of experiments grouped by map. The values of energy consumption and travel time correspond to the best individual of the last generation. The last column of the table shows the number of generations needed to converge to these solutions. It is worth noticing that all experiments needed fewer generations than the fixed termination criterion, which is 90; therefore, all experiments were terminated because the running mean termination criterion was fulfilled. The latter is important since it gives flexibility to the algorithm and reduces the computational cost of running the same number of generations for every scenario, which is especially variable for GA-2.

Table 4.8: GA-2 experiments results grouped by map.

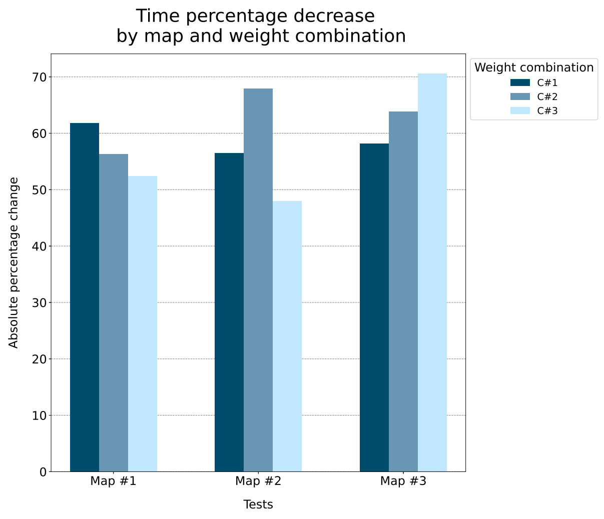
Map	Weight combination	Energy [Wh]	Time [sec]	N° of generations
#1	c_1	257.79	3089.03	45
	c_2	264.46	3606.11	35
	c_3	269.91	4370.71	18
#2	c_1	232.04	2599.48	39
	c_2	183.26	2134.46	39
	c_3	278.95	3526.43	30
#3	c_1	207.65	3145.9	31
	c_2	234.059	2457.78	44
	c_3	237.51	2423.99	42

The results of energy consumption show a clear relationship between the weight combination and the value of the variable in the study, as expected. The only exception occurs for Map #2 with combination c_2 where the energy value was less than the one obtained for combination c_2. This is explained because if we see the value reached for the travel time, we can notice that there is also a great reduction in that variable. Since the energy consumption and the total travel time are influenced by each other, we can suspect that the initial population for that experiment allowed us to converge to a more optimal solution.

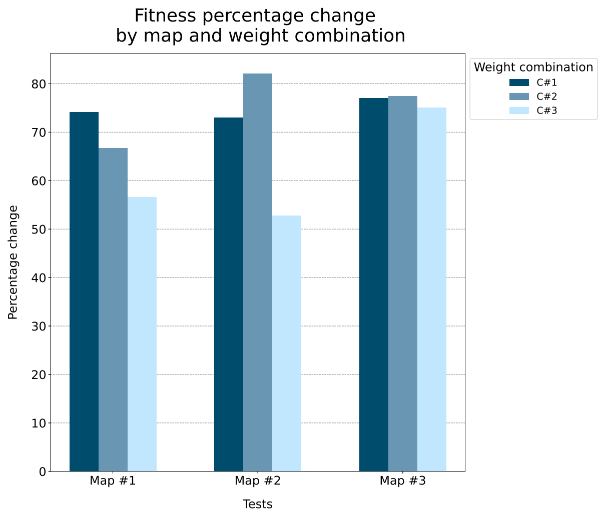
When it comes to the travel time variable there is an inverse pattern to the one expected. We would expect to see lower travel times for combination c_3 and more time-consuming for combination $c1$. However, the travel time follows the same trend as the energy consumption, and for every map, we see the worst result when reducing the weight value that multiplies the energy consumption variable. Hence, energy consumption optimization has a greater influence on the fitness function and overall results.



(a) Energy consumption.



(b) Travel time.



(c) Fitness.

Figure 4.5: Energy consumption, travel time and fitness percentage change.

Figure 4.5 shows the percentage change between the best individual of the first generation and the best individual of the last generation. Since both energy consumption and travel time experimented with a decrease we would expect to see the percentage change with negative values, but for purposes of visualization, the absolute value is shown.

By looking at the graphs in Figure 4.5 we can first notice that, as mentioned before, the experiment that used combination c_2 and Map #2 does not follow the trend and is outstandingly better. There is also a greater improvement (on average) in the experiments performed using Map #3 when compared to other maps since the first generation usually takes longer detours to avoid obstacles, which means more time and energy-consuming paths.

4.4. Decision-Making System Analysis

In this section, we analyze the entire operation of the Decision-Making System. To achieve this we selected one of the outputs generated by GA-1 in section 4.2. Later, we used GA-2 to find the path between waypoint i to waypoint $i+1$ — meaning that GA-2 was performed 8 times to produce one entire path — and we finally evaluated the number of times the rover had to stop in order to fulfill the SoC policy. The route selected is shown in Table 4.9 with its respective characteristics.

Table 4.9: GA-1 route selected

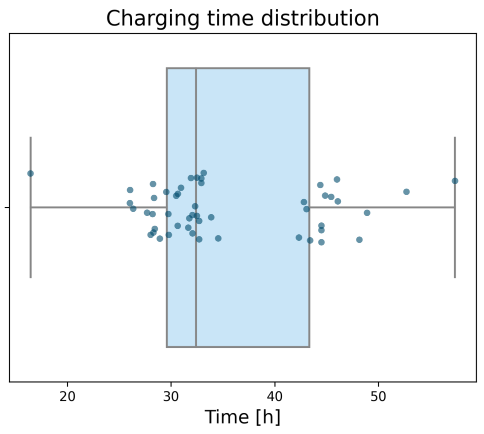
Route	(3,1), (15,15), (10,30), (80,37), (93,73), (90,24), (73,20), (37,60), (53,62)
Travel time	37640.7900 [s]
Travel time w/ recharging	46.5 [h]
Accumulated reward	108

Since the Decision-Making System is stochastic, we ran 50 simulations for the same route shown above to obtain a realistic representation using Map #2. The hyperparameters used for GA-2 are the same as the ones shown previously in Table 4.7 for every pair of waypoints. The weight combination used is c_1 due to the better results obtained in general with this combination. It is worth mentioning that running the entire Decision-Making System has a huge computational cost, especially when generating the initial population. An approach to reduce the execution time of the 50 simulations was to create 200 individuals for every GA-2 iteration, meaning that there was 8 initial population that consisted of 200 individuals. Thus, when GA-2 started, it selected a random sample of 100 individuals from the 200 individuals previously created for the corresponding pair of waypoints.

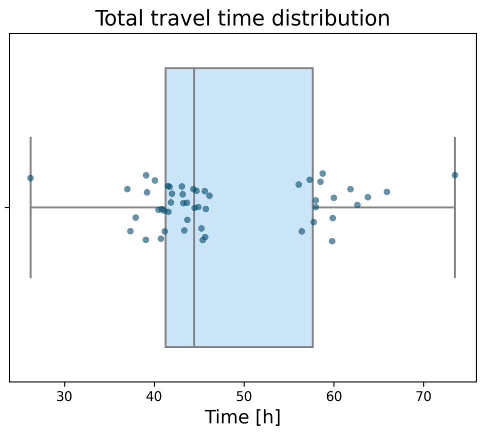
Figure 4.6 shows boxplots summarizing the results obtained for the time that it took to recharge the rover, and the total travel time (which includes the recharging time), and the energy consumption. Regarding the total travel time, we can notice that the median, which is $43.68[h]$, is lower than the approximation of the total travel time obtained from GA-1. The results show that 30 % of the simulations surpass the travel time estimated by GA-1, which surpasses from 10 to 20 hours and the main source of this extended time is the recharging time.

By looking at the charging time distribution we can see that most of the time that it took to complete the path was spent recharging the batteries, as expected from the documentation on the last rovers sent by NASA [3], [10] and [22]. The median of time spent recharging the batteries is $32.31[h]$.

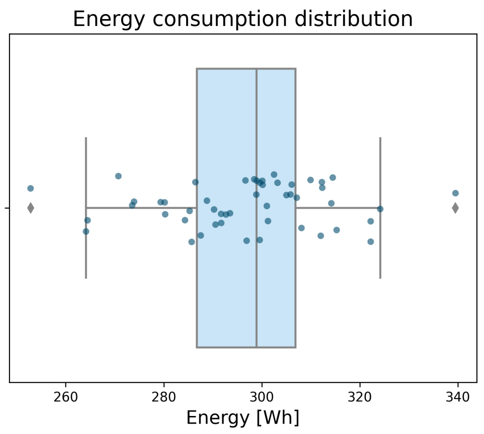
When it comes to energy consumption, there are two outliers one on the lower extreme and one on the higher side. The median energy consumption distribution is $298.89[Wh]$. It is worth noticing that none of the simulations had an energy consumption near the peak of the rover consumption energy, which is $500[Wh]$. Therefore, the Decision-Maker system was able to generate paths that optimize the energy, avoiding a possible failure due to operating constantly at the peak value allowed.



(a) Charging time.



(b) Travel time.



(c) Energy.

Figure 4.6: Energy consumption, total travel time and charging time distributions.

One important aspect to analyze is the terrain types chosen in a path. For every path, we calculated the number of occurrences of each terrain and normalized it by the length of the corresponding path. Figure 4.7 shows the results for the 5 terrains. We can see that terrain A makes the majority of a path, which was expected given that it allows us to drive faster and consume less energy due to the smoothness of the terrain. Overall, the Decision-Making System was able to generate paths that favor more benign terrains, which is beneficial for the rover's component's health.

The distribution of terrain E has a higher median, 0.1, than the one of terrain D, 0.08. This behavior is explained on one side because in Map #2 there is a 20 % chance of encountering terrain E versus a 15 % chance for terrain D. On the other side, terrain D has a range of speeds that the rover can drive at. However, the Decision-Making System has a prognostic approach, meaning that we are taking the speed value that marks the 10 % lower probability of occurrence causing that the speed value for terrain D and E are not too far apart, with 13.57 and 8.35 respectively.

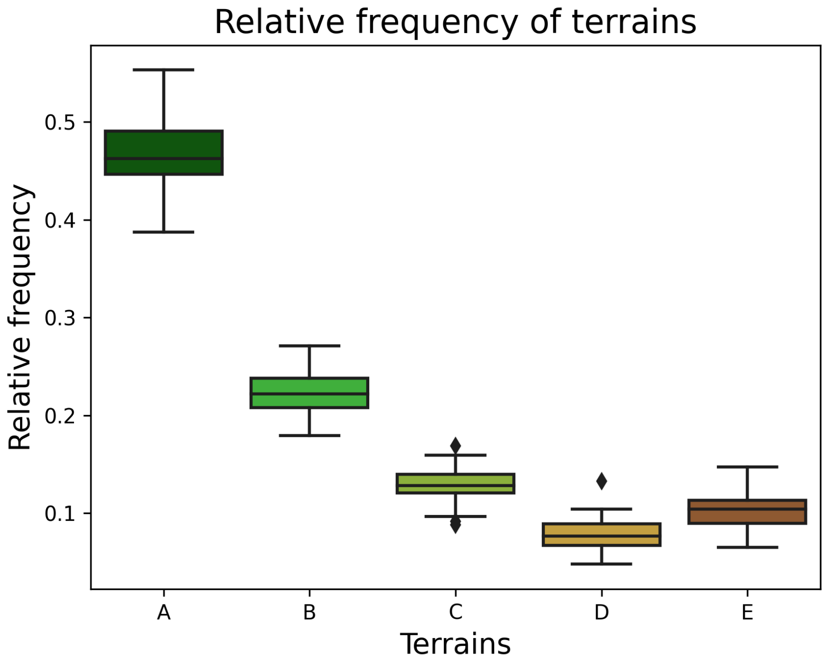
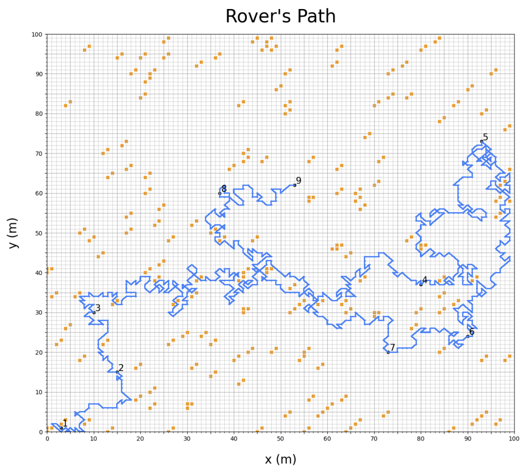


Figure 4.7: Relative frequency of each terrain category.

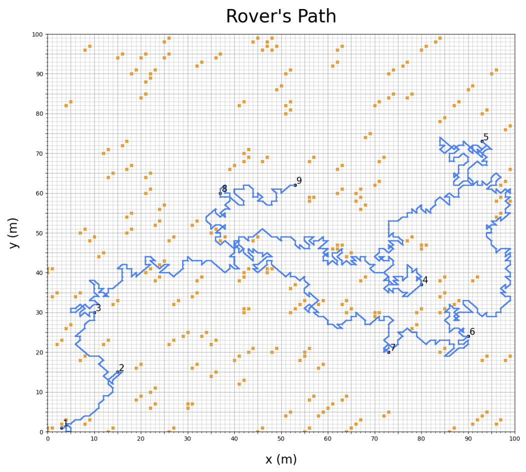
Figure 4.8 and 4.9 shows the first and last generation of paths generated by the Decision-Making System, at the right it shows the worst path, which is an outlier in terms of energy, and at the left, it shows the best paths in terms of time and energy consumption. If we compare the two paths we can see that Figure 4.9.a has few sharp curves to traverse terrains that are more benign. The sharp curves do not represent a problem for the rover since the size of the grid is $1m^2$, which means that it has space to take these curves more smoothly if the locomotion control takes into account a strategy to prevent drastic turns and twists. On the other hand, Figure 4.9.b has loops inside and redundant curves that show that for that particular simulation the solution was exceptionally sub-optimal. Another criterion can be added to prevent and diminish the chances of getting sub-optimal solutions.

One particular aspect that is important to analyze is the impact that uncertainty sources have on the decisions made by the system. The uncertainty sources are represented by the process noise in equations (2.1) and (2.2) of the state-space model of the batteries. The parameters ω_1 and ω_2 indicate the amount of noise we are considering for the internal resistance and the SoC evolution, respectively.

To analyze the impact of these parameters, one path generated by the PDM system is selected and the performance is studied by changing the parameters ω_1 and ω_2. It is important to remember, that the time it takes to recharge the batteries fluctuates since the power provided by the MMRTG is modeled as a uniform distribution. Therefore, the performance of the path with the original parameters is simulated 50 times, and the same is done when testing different values for ω_1 and ω_2. The baseline case is using ω_1 equal to $5 \cdot 10^{-8}$ and ω_2 equal to $1 \cdot 10^{-6}$.

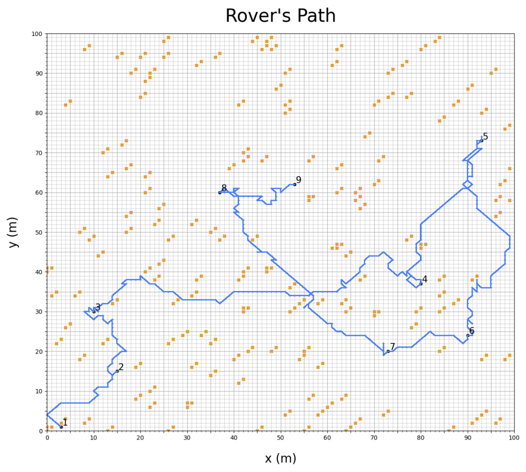


(a) Best path during first generation.

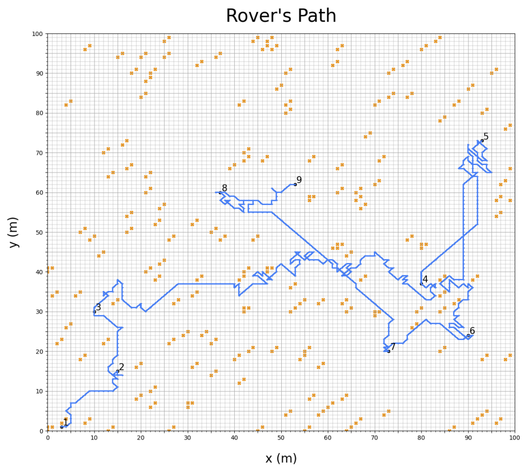


(b) Worst path during first generation.

Figure 4.8: Example of two paths generated by the Decision-Making System during first generation.



(a) Best path.



(b) Worst path.

Figure 4.9: Example of two paths generated by the Decision-Making System at the last generation.

Figure 4.10 shows the recharging time distributions of the entire path for different values on the process noise of the internal resistance equation while maintaining the same value for the process noise of the SoC evolution equation. It is worth noticing that regardless of the value of ω_1, the PDM system indicates that the rover must stop two times to satisfy the SoC policy, and the locations chosen to stop are near the locations chosen by the baseline case, within an area of approximately $20[m^2]$ for each stop. Results show that increasing the noise value decreases the recharging time of the batteries. However, for values $\omega_1 = 5 \cdot 10^{-1}$ and $\omega_1 = 5 \cdot 10^{-4}$ the simulations show more dispersion and the median value is far from the experimental expected value indicated by the documentation for rovers *Curiosity* and *Perseverance*, which is approximately 18 hours per recharge. On the other side, values $\omega_1 = 5 \cdot 10^{-12}$

and $\omega_1 = 5 \cdot 10^{-16}$ have a similar median value to the baseline case but have a higher standard deviation, which can be detrimental for the PDM system.

The same experiment was performed, but on this occasion, we vary the ω_2, which is the process noise of the evolution of SoC. In this experiment we tested ω_2 with the following values, $1 \cdot 10^{-2}$, $1 \cdot 10^{-4}$, $1 \cdot 10^{-6}$ (baseline case), $1 \cdot 10^{-8}$ and $1 \cdot 10^{-10}$. For values, $1 \cdot 10^{-2}$ and $1 \cdot 10^{-4}$, the PDM system indicates that the rover must stop only once, and the recharging times do not have physical sense because it shows that the rover can recharge the two batteries in a range from 30 minutes to 2 hours. Therefore, Figure 4.11 only shows the results of the baseline case and two test cases.

From the visualization, we can notice that the baseline case has a lower median value; however, this value is only 2 to 3 hours less than the median of other cases. The baseline case has also a slightly smaller standard deviation, which is $2.2393[h]$, when compared to the test cases, which is $2.6552[h]$ for $\omega_2 = 1 \cdot 10^{-8}$ and $2.5738[h]$ for $\omega_2 = 1 \cdot 10^{-10}$. For the three cases shown in Figure 4.11, the PDM system determines that the rover must stop two times. The exact coordinates of the locations vary, but for all three cases, they are within an area of $7.5[m^2]$.

The study of the influence of the uncertainty sources demonstrates the importance of choosing the parameters ω_1 and ω_2 correctly since it has a great impact on the standard deviation of the simulation's distribution. If the parameters have greater values than the baseline case then the PDM system estimates lower recharging time, to the point where it does not have physical sense according to real data on Mars rovers. However, the PDM system is more sensitive to changes in ω_2 because it reduces the number of stops and produces errors in a few simulations.

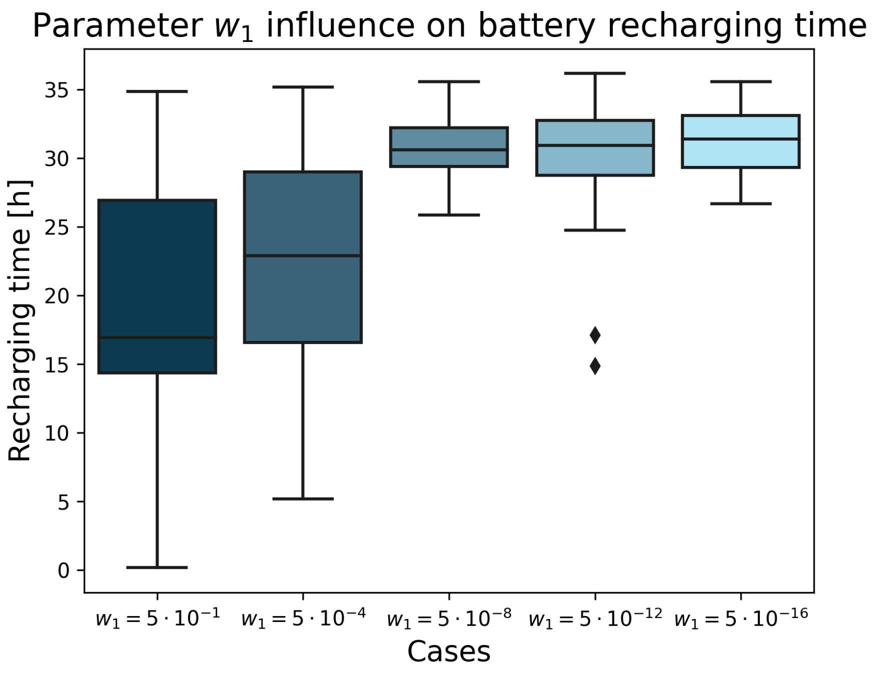


Figure 4.10: Parameter ω_1 influence on battery recharging time.

As mentioned before, the Decision-Making System considers a statistic for the speed at which the rover can drive at a certain terrain category. Thus, one question arises, how much does the performance of one path change if we consider the real ranges of speed for each terrain? Does that performance change represents a problem for the rover's mission?.

To answer those questions, we took the path that had the best performance, which is the one shown in Figure 4.9.a, and had an energy consumption of $264[Wh]$ and a total travel time of $94410[s]$. The path in the study was simulated 50 times but for each terrain, the speed considered the original ranges as uniform distributions, as stated before in (3.3). The simulation results are summarized in Figure 4.12. The results for the total travel time show that the median of the distribution is equal to $89728.9[s]$, which is less than the predicted value when using a conservative approach, and $70\,\%$ of the simulations have a shorter total travel time than the predicted. When it comes to energy consumption, all simulations needed less energy than predicted.

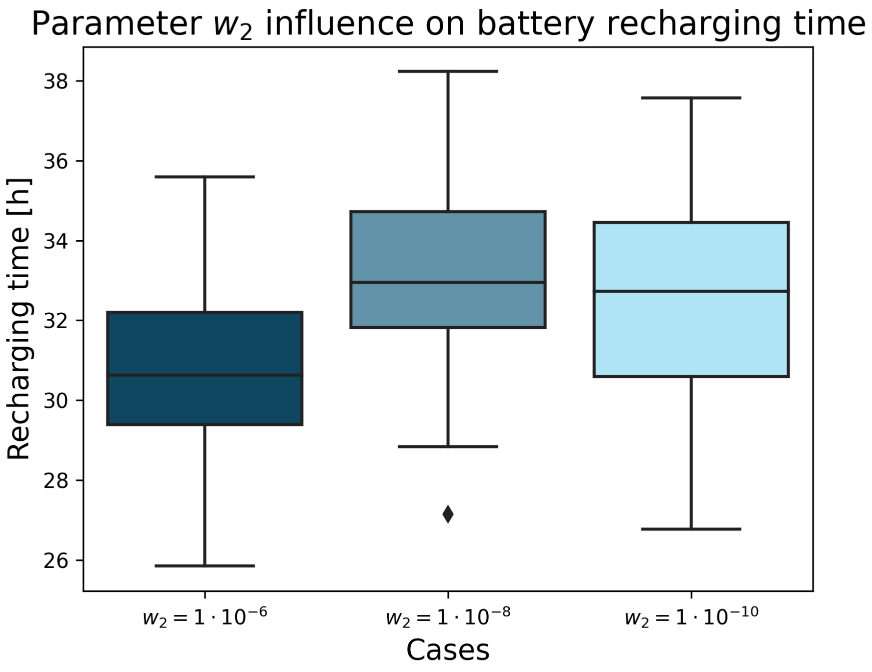
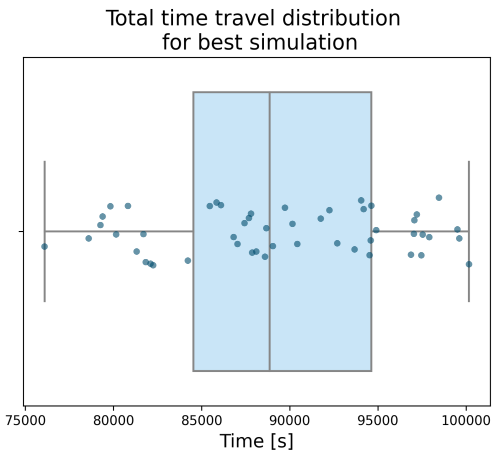
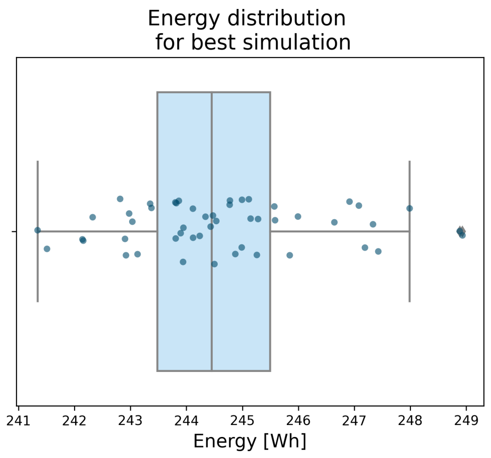


Figure 4.11: Parameter ω_2 influence on battery recharging time.

In this sense, it is worth carrying out the base modeling considering a conservative speed scenario because when facing a real situation (or close to a real context), there is a degree of slackness concerning the robot's performance along the way. Having this level of slack is beneficial, mainly in situations involving a significant degree of uncertainty associated with factors, such as terrain irregularities, component failure, command delay sent by the flight software, and encountering big or sharp rocks, among others. All of the above is more difficult to handle when considering the context of a Mars rover, where the robot operates in an extremely isolated and hostile environment.



(a) Travel time.



(b) Energy.

Figure 4.12: Energy consumption and total travel time distribution when testing the same individual.

Chapter 5

Conclusions and Future Work

In this thesis work, an offline Decision-Making System for path planning in the context of a Mars rover was developed. The Decision-Making System consists of two GA, GA-1 and GA-2, and uses a state-space model to evaluate the SoC of the rover's batteries. GA-1 works with a set of waypoints and rewards and evaluates the number of waypoints to visit and in which order to do so, given a maximum driving time constraint. GA-2 finds a path between a start point and an end point while avoiding untraversable terrains and taking into account a SoC policy. Therefore, GA-2 is iterated for each pair of subsequent waypoints to generate a path that considers all waypoints selected by GA-1.

The Decision-Making System was designed by stating a mathematical formulation as an optimization problem. It uses a state-space model as proposed in [33] to estimate the SoC, internal resistance, and output voltage of the batteries. The system has a SoC policy because it helps to reduce the effects of battery degradation, according to the last rover's missions on Mars. Following the proposal made in [7], the problem is formulated as a map grid with each cell belonging to a certain terrain category. Each terrain category defines the speed, either as the range or value at which the rover can drive. Since the problem has a stochastic approach to considering the uncertainty of the rover's mission, the speed values have noise added and the speed ranges are modeled as probability distributions. The waypoints are generated randomly and each of them has a reward associated.

As mentioned before, the Decision-Making System was divided into two parts to solving the problem. GA-1 maximizes the accumulated reward of the waypoints visited and minimizes the travel time that it takes to drive through the route while considering a time constraint. However, the travel time is an approximation since GA-2 is in charge of performing a fine-tuning of the path between a pair of waypoints. A function to eliminate some genes was added in the initial population and crossover operator to help maintain the diversity of the population to satisfy the time constraint. GA-1 considers a Death Penalty when it comes to the time constraint and a static penalty for the number of obstacles encountered in a route.

GA-2 minimizes the energy consumption and the total travel time. The energy consumption is given by the terrain type and a linear function, which depends on the ratio between the actual velocity and the nominal velocity. The total travel considers the time needed to recharge batteries. Since the velocity of terrain may vary even if they belong to the same category, a statistic of the probability distribution is taken when planning a path. The statistic is the speed value that marks the 10 % of the cumulative probability distribution of each terrain. Therefore, GA-2 has a conservative approach to preventing mission failures.

The implementation of GA-2 considers a function to connect paths after the crossover operator, since this operator may generate paths where the nodes are not neighbors of each other. The mutation operator is inspired by the procedure proposal made in [45], which allows for faster convergence. The running mean termination criterion is added since it reduces the computational cost of performing extra generations and gives flexibility to the GA. The latter is important considering that depending on the location of the waypoints, the generations needed for convergence fluctuate. GA-2 considers a Death Penalty if an individual considers an untraversable terrain cell in the path.

Three sets of experiments were performed to analyze the performance of the Decision-Making System. The first set of an experiment aimed to study GA-1 by testing the algorithm to different weight combinations in the fitness function and time constraints. Results showed that GA-1 can find feasible solutions and can favor waypoints with higher rewards while taking into consideration their location and the travel time of the entire route. Further improvement is needed for tuning the weights to give more importance to the accumulated reward since the experiments show that GA-1 is not sensitive enough when it comes to benefiting the reward variable during the optimization.

With similar factors tested, the second set of experiments analyzed the performance of GA-2. Results showed that the running mean termination criterion was successful in reducing the number of generations when given different conditions to the algorithm. GA-2 was able to plan an optimal path between two points and take detours when an obstacle was encountered. The tests demonstrated that the energy consumption variable has a greater influence on the overall improvement of the fitness function.

Finally, the last set of experiments consisted of running 50 simulations of the Decision-Making System taking one route generated by GA-1. Results showed that the system can generate paths that optimize the energy and the total travel time while preventing an overload consumption of energy and plans when the rover must stop to satisfy the SoC policy. Since the paths were generated by considering the prognosticated speed value, an evaluation of the performance of a path is done when taking the uncertainty of the real speed values. The latter showed that the prognosis approach gives us a degree of slackness to operate, which is beneficial when considering the hostility and uncertainty in an extraplanetary mission. Another important aspect is the study of the influence of the uncertainty sources in the state-space model of the batteries, since results show that higher values of noise cause estimations of batteries recharging times that do not correspond to real data. On the other hand, lower values of these parameter cause a greater standard deviation on the simulations.

Further improvements can be made to the system. One of them is to add the slippage predicted for a terrain category. This addition allows us to calculate the energy from the velocity of the wheels rather than from the velocity at which the rover is moving. Another addition that could be made to the system is a locomotion control module. The locomotion module could provide a set of commands for the motor that controls the wheels based on the movements planned by the decision-maker system.

Bibliography

[1] D. Rapp, “Why Explore Mars?,” in Human Missions to Mars, Cham: Springer International Publishing, 2016, pp. 1–15. doi: 10.1007/978-3-319-22249-3_1.

[2] D. Shayler, A. Salmon, and M. D. Shayler, Marswalk One: first steps on a new planet. Berlin; New York: Chichester, UK: Springer; In association with Praxis Publishing, 2005.

[3] E. Lakdawalla, The Design and Engineering of Curiosity: How the Mars Rover Performs Its Job, 1st ed. 2018. Cham: Springer International Publishing: Imprint: Springer, 2018. doi: 10.1007/978-3-319-68146-7.

[4] A. Ellery, Planetary Rovers: Robotic Exploration of the Solar System, 1st ed. 2016. Berlin, Heidelberg: Springer Berlin Heidelberg: Imprint: Springer, 2016. doi: 10.1007/978-3-642-03259-2.

[5] M. Bajracharya, M. W. Maimone, and D. Helmick, “Autonomy for Mars Rovers: Past, Present, and Future,” Computer, vol. 41, no. 12, pp. 44–50, Dec. 2008, doi: 10.1109/MC.2008.479.

[6] M. Ono, T. J. Fuchs, A. Steffy, M. Maimone, and Jeng Yen, “Risk-aware planetary rover operation: Autonomous terrain classification and path planning,” in 2015 IEEE Aerospace Conference, Big Sky, MT, Mar. 2015, pp. 1–10. doi: 10.1109/AERO.2015.7119022.

[7] G. Hedrick, N. Ohi, and Y. Gu, “Terrain-Aware Path Planning and Map Update for Mars Sample Return Mission,” IEEE Robot. Autom. Lett., vol. 5, no. 4, pp. 5181–5188, Oct. 2020, doi: 10.1109/LRA.2020.3005123.

[8] S. Narasimhan et al., “Autonomous Decision Making for Planetary Rovers Using Diagnostic and Prognostic Information,” IFAC Proceedings Volumes, vol. 45, no. 20, pp. 289–294, Jan. 2012, doi: 10.3182/20120829-3-MX-2028.00243.

[9] E. Balaban, J. J. Alonso, and K. F. Goebel, “An Approach to Prognostic Decision Making in the Aerospace Domain,” p. 19, 2012.

[10] T. de J. Mateo Sanguino, “50 years of rovers for planetary exploration: A retrospective review for future directions,” Robotics and Autonomous Systems, vol. 94, pp. 172–185, Aug. 2017, doi: 10.1016/j.robot.2017.04.020.

[11] M. A. Gross and G. Cardell, “An overview of Nasa’s mars science laboratory,” Nasa.gov. [Online]. Available: https://trs.jpl.nasa.gov/bitstream/handle/2014/42086/11-1313.pdf?sequence=1&isAllowed=y. [Accessed: 01-Dec-2021]

[12] R. Bugga, M. Smart, J. Whitacre, and W. West, “Lithium Ion Batteries for Space Applications,” in 2007 IEEE Aerospace Conference, Big Sky, MT, USA, 2007, pp. 1–7. doi: 10.1109/AERO.2007.352728.

[13] M. C. Smart, B. V. Ratnakumar, R. C. Ewell, S. Surampudi, F. J. Puglia, and R. Gitzendanner, "The use of lithium-ion batteries for JPL's Mars missions," Electrochimica Acta, vol. 268, pp. 27–40, Apr. 2018, doi: 10.1016/j.electacta.2018.02.020.

[14] J. Taylor, A. Makovsky, A. Barbieri, R. Tung, P. Estabrook, and A. G. Thomas, "Mars Exploration Rover Telecommunications," in Deep Space Communications, J. Taylor, Ed. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2016, pp. 251–358. doi: 10.1002/9781119169079.ch7.

[15] R. A. Lindemann and C. J. Voorhees, "Mars Exploration Rover Mobility Assembly Design, Test and Performance," in 2005 IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, HI, USA, 2005, vol. 1, pp. 450–455. doi: 10.1109/ICSMC.2005.1571187.

[16] G. Reina and M. Foglia, "An Adaptive Suspension System for Planetary Rovers," IFAC Proceedings Volumes, vol. 43, no. 16, pp. 199–204, 2010, doi: 10.3182/20100906-3-IT-2019.00036.

[17] M. von Ehrenfried, Perseverance and the Mars 2020 mission: follow the science to Jezero Crater. 2022. Accessed: Mar. 28, 2022. [Online]. Available: https://doi.org/10.1007/978-3-030-92118-7

[18] G. E. Reeves and J. F. Snyder, "An Overview of the Mars Exploration Rovers' Flight Software," in 2005 IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, HI, USA, 2005, vol. 1, pp. 1–7. doi: 10.1109/ICSMC.2005.1571113.

[19] R. Welch, D. Limonadi, and R. Manning, "Systems engineering the Curiosity Rover: A retrospective," in 2013 8th International Conference on System of Systems Engineering, Maui, HI, USA, Jun. 2013, pp. 70–75. doi: 10.1109/SYSoSE.2013.6575245.

[20] M. Ono, T. J. Fuchs, A. Steffy, M. Maimone, and Jeng Yen, "Risk-aware planetary rover operation: Autonomous terrain classification and path planning," 2015 IEEE Aerospace Conference. IEEE, Mar. 2015. doi: 10.1109/aero.2015.7119022.

[21] M. Bajracharya, M. W. Maimone, and D. Helmick, "Autonomy for Mars Rovers: Past, Present, and Future," Computer, vol. 41, no. 12. Institute of Electrical and Electronics Engineers (IEEE), pp. 44–50, Dec. 2008. doi: 10.1109/mc.2008.479.

[22] M. Maimone, A. Johnson, Y. Cheng, R. Willson, and L. Matthies, "Autonomous Navigation Results from the Mars Exploration Rover (MER) Mission," Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, pp. 3–13, 2006. doi: 10.1007/11552246_1.

[23] S. B. Goldberg, M. W. Maimone, and L. Matthies, "Stereo vision and rover navigation software for planetary exploration," Proceedings, IEEE Aerospace Conference. IEEE. doi: 10.1109/aero.2002.1035370.

[24] O. Toupet, T. Del Sesto, M. Ono, S. Myint, J. vander Hook, and M. McHenry, "A ROS-based Simulator for Testing the Enhanced Autonomous Navigation of the Mars 2020 Rover," in 2020 IEEE Aerospace Conference, Big Sky, MT, USA, Mar. 2020, pp. 1–11. doi: 10.1109/AERO47225.2020.9172345.

[25] E. Balaban et al., "Development of a Mobile Robot Test Platform and Methods for Validation of Prognostics-Enabled Decision Making Algorithms," IJPHM, vol. 4, no. 1, Oct. 2020, doi: 10.36001/ijphm.2013.v4i1.1449.

[26] B. V. Ratnakumar, M. C. Smart, A. Kindler, H. Frank, R. Ewell, and S. Surampudi, “Lithium batteries for aerospace applications: 2003 Mars Exploration Rover,” Journal of Power Sources, vol. 119–121. Elsevier BV, pp. 906–910, Jun. 2003. doi: 10.1016/s0378-7753(03)00220-9.

[27] M. Morris and S Tosunoglu, "Survey of rechargeable batteries for robotic applications,"Florida Conf. Recent Adv. Robot., May. 2012.

[28] W. Waag and D. U. Sauer, “SECONDARY BATTERIES – LEAD– ACID SYSTEMS | State-of-Charge/Health,” Encyclopedia of Electrochemical Power Sources. Elsevier, pp. 793–804, 2009. doi: 10.1016/b978-044452745-5.00149-0.

[29] X. Han et al., “A review on the key issues of the lithium ion battery degradation among the whole life cycle,” eTransportation, vol. 1, no. 100005, p. 100005, 2019.

[30] A. Perez, R. Moreno, R. Moreira, M. Orchard, and G. Strbac, “Effect of Battery Degradation on Multi-Service Portfolios of Energy Storage,” IEEE Trans. Sustain. Energy, vol. 7, no. 4, pp. 1718–1729, Oct. 2016, doi: 10.1109/TSTE.2016.2589943.

[31] M. Daigle and C. S. Kulkarni, “A battery health monitoring framework for planetary rovers,” in 2014 IEEE Aerospace Conference, 2014.

[32] M. Daigle and C. S. Kulkarni, “Electrochemistry-based Battery Modeling for Prognostics,” p. 13, 2013.

[33] D. A. Pola et al., “Particle-Filtering-Based Discharge Time Prognosis for Lithium-Ion Batteries With a Statistical Characterization of Use Profiles,” IEEE Trans. Rel., vol. 64, no. 2, pp. 710–720, Jun. 2015, doi: 10.1109/TR.2014.2385069.

[34] C. Burgos-Mellado, M. E. Orchard, M. Kazerani, R. Cárdenas, and D. Sáez, “Particle-filtering-based estimation of maximum available power state in Lithium-Ion batteries,” Applied Energy, vol. 161, pp. 349–363, Jan. 2016, doi: 10.1016/j.apenergy.2015.09.092.

[35] B. Rothrock, R. Kennedy, C. Cunningham, J. Papon, M. Heverly, and M. Ono, “SPOC: Deep Learning-based Terrain Classification for Mars Rover Missions,” presented at the AIAA SPACE 2016, Long Beach, California, Sep. 2016. doi: 10.2514/6.2016-5539.

[36] A. Nedjatia and B. Vizvárib, “Robot Path Planning by Traveling Salesman Problem with Circle Neighborhood: modeling, algorithm, and applications,” 2020, doi: 10.48550/ARXIV.2003.06712.

[37] J. M. Wiener, N. N. Ehbauer, H. A. Mallot, "Path planning and optimization in the traveling salesman problem: nearest neighbor vs. region-based strategies,"Dagstuhl Seminar Proceedings, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Nov. 2006.

[38] D. S. Johnson, “Local optimization and the Traveling Salesman Problem,” in Automata, Languages and Programming, vol. 443, M. S. Paterson, Ed. Berlin/Heidelberg: Springer-Verlag, 1990, pp. 446–461. doi: 10.1007/BFb0032050.

[39] N. Deo, Graph theory with applications to engineering computer science, Dover edition. Mineola, New York: Dover Publications, Inc, 2016.

[40] J. Dréo, Ed., Metaheuristics for hard optimization: methods and case studies. Berlin: Springer, 2006.

[41] T. Blickle and L. Thiele, “A Comparison of Selection Schemes Used in Evolutionary Algo-

rithms," Evolutionary Computation, vol. 4, no. 4, pp. 361–394, Dec. 1996, doi: 10.1162/evco.1996.4.4.361.

[42] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," Computer Methods in Applied Mechanics and Engineering, vol. 191, no. 11–12, pp. 1245–1287, Jan. 2002, doi: 10.1016/S0045-7825(01)00323-1.

[43] O. Kramer, Genetic Algorithm Essentials, vol. 679. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-52156-5.

[44] Yanrong Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, New Orleans, LA, USA, 2004, pp. 4350-4355 Vol.5. doi: 10.1109/ROBOT.2004.1302402.

[45] A. Tuncer and M. Yildirim, "Dynamic path planning of mobile robots with improved genetic algorithm," Computers Electrical Engineering, vol. 38, no. 6, pp. 1564–1572, Nov. 2012, doi: 10.1016/j.compeleceng.2012.06.016.

[46] A. Ghorbani, S. Shiry, and A. Nodehi, "Using Genetic Algorithm for a Mobile Robot Path Planning," in 2009 International Conference on Future Computer and Communication, Kuala Lumpar, Malaysia, Apr. 2009, pp. 164–166. doi: 10.1109/ICFCC.2009.28.

[47] M. Naderan-Tahan and M. T. Manzuri-Shalmani, "Efficient and safe path planning for a Mobile Robot using genetic algorithm," in 2009 IEEE Congress on Evolutionary Computation, Trondheim, Norway, May 2009, pp. 2091–2097. doi: 10.1109/CEC.2009.4983199.

[48] S. N. Ghoreishi, A. Clausen, and B. N. Joergensen, "Termination Criteria in Evolutionary Algorithms: A Survey:," in Proceedings of the 9th International Joint Conference on Computational Intelligence, Funchal, Madeira, Portugal, 2017, pp. 373–384. doi: 10.5220/0006577903730384.

[49] B. Jain, H. Pohlheim, and J. Wegener, "On termination criteria of evolutionary algorithms," 2001, pp. 768–768.

[50] A. Koubaa et al., Ïntroduction to Mobile Robot Path Planning", Robot path planning and cooperation. New York, NY: Springer Berlin Heidelberg, 2018, pp 3-13.

[51] "Mars 2020 perseverance rover," NASA. [Online]. Available: https://mars.nasa.gov/mars2020/. [Accessed: 07-Nov-2022].

[52] "Archival content: Enhanced Multi-Mission Radioisotope Thermoelectric Generator (eMMRTG) concept – NASA RPS: Radioisotope Power Systems," NASA, 20-Jan-2022. [Online]. Available: https://rps.nasa.gov/resources/56/archival-content-enhanced-multi-mission-radioisotope-thermoelectric-generator-emmrtg-concept/. [Accessed: 07-Nov-2022].

[53] "Communications with Earth," NASA. [Online]. Available: https://mars.nasa.gov/msl/mission/communications/. [Accessed: 07-Nov-2022].

Annex

Map Generation

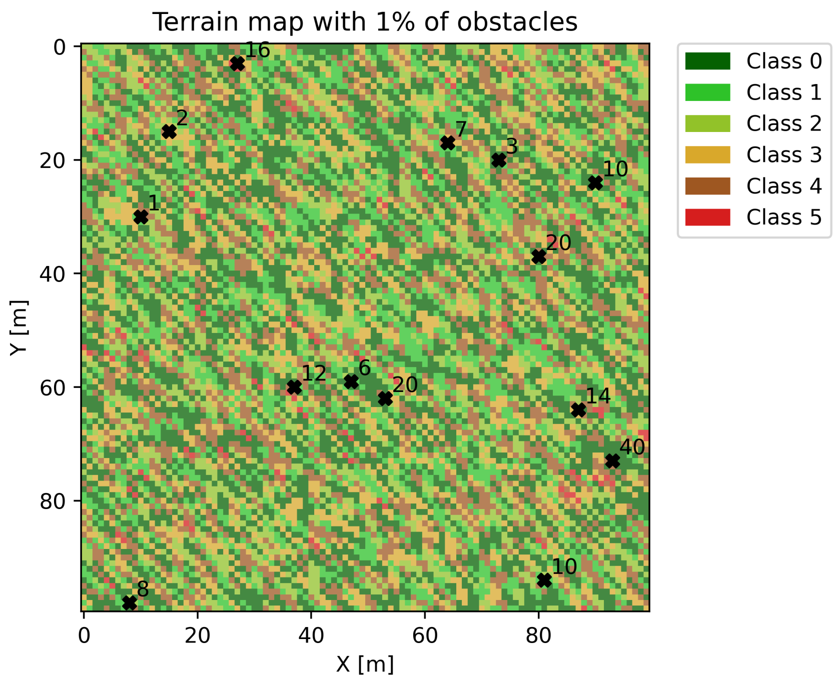


Figure 5.1: Map #1 with 1 % of obstacles.

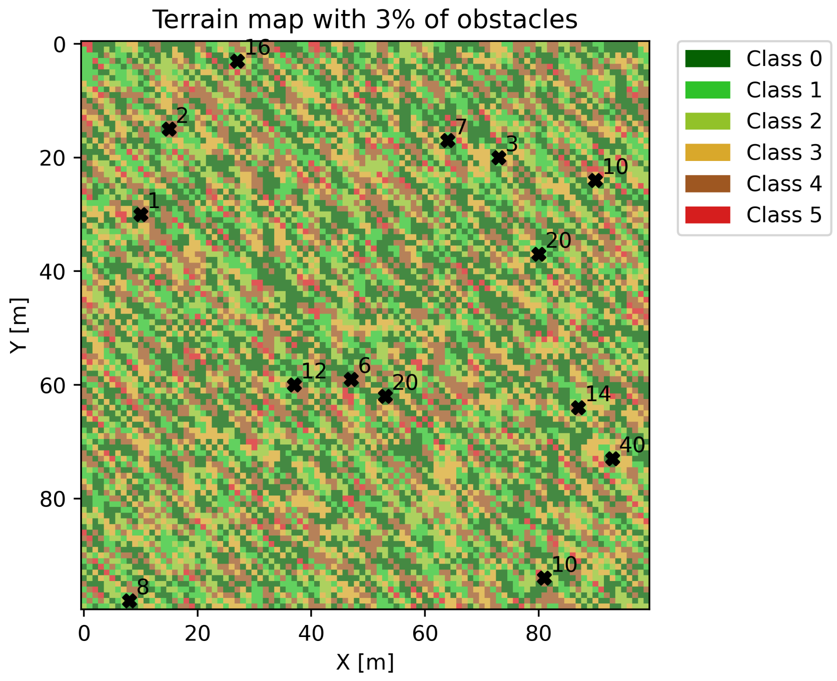


Figure 5.2: Map #2 with 3 % of obstacles.

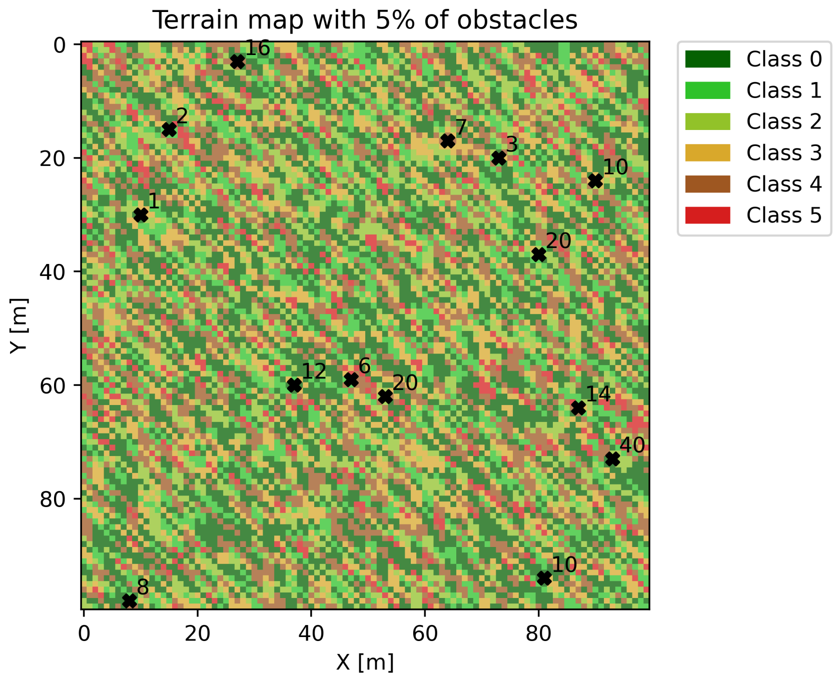


Figure 5.3: Map #3 with 5 % of obstacles.