University of Córdoba

Computer Science Engineering Degree
3rd course

Metaheuristics

PERSEVERANCE in Mars

Valentín Gabriel Avram Aenachioei
Alexandra-Maria Borsan
Aicha Bracken-Soliman
Radu Ciurca
Barbara Kereselidze
Víctor Rojas Muñoz
Davit Tchanturia

Academic year 202-2023
Córdoba, June 9, 2023

Contents

List of algorithms

[0, 2, 0, 2, 0, 0, 0, 3, 0, 0, 2, 0, 0, 2, 2, 2, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 3, 3, 0, 2, 1, 1, 1, 1, 3, 3, 3, 1, 1, 1, 2, 1, 2, 2, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

List of Tables

List of Figures

In this paper, we will set out how we have done the third practical assignment, *Perseverance on Mars*, developing a metaheuristic approach to the control of the Perseverance rover on Mars.

1 Introduction

The objective of this study is to assess the challenges and requirements of the Mars 2020 Mission and optimise it using a metaheuristic approach, focusing on the exploration of the red planet by NASA's Perseverance rover. The main goal of the mission is to collect valuable rocks for future expeditions. Perseverance has been designed to perform accurate and efficient explorations, optimising resources for subsequent missions. To achieve this, the rover must identify a route with the highest number of valuable rocks to analize, while minimising actions and avoiding obstacles such as holes or hard-to-reach areas to conserve resources and energy.

In this context, our theoretical analysis examines the optimization problem associated with determining the optimal path for Perseverance, considering the limited knowledge of the planet's terrain and constraints related to fuel consumption. Moreover, the study investigates metaheuristic approaches that can be employed to address this optimization, taking into account known limitations concerning movement and potential obstacles.

To identify the most effective path, we have analyzed various algorithms that can generate solutions closest to the optimal outcome.

2 Metaheuristic analysis

In our analysis, we employed both non-metaheuristic and metaheuristic approaches to derive the most optimal solution. Initially, we utilized non-metaheuristic algorithms such as a simple Shortest Path algorithm and Map Exploration using A* to gain a deeper understanding of the problem at hand. Subsequently, we adopted a metaheuristic approach by implementing Particle Swarm Optimization (PSO), which yielded promising results. Although we also implemented a Genetic Algorithm, the PSO approach outperformed it. To further optimize our solution, we ultimately decided to integrate PSO with the Genetic Algorithm.

2.1 Non-metaheuristic approaches

2.1.1 Shortest Path

The algorithm creates a vector that encodes the movements necessary to reach each intermediate position, and then combines the vectors to create the final path. However, the code does not take into account the possibility of local optima, and may not find the global optimal solution. This is the reason we haven't gone with this approach.

In terms of results and knowing that the best possible solution is *9320* fitness value, with this approach we obtain fitness values of *2500* as best.

2.1.2 A* algorithm

Algorithm divided into two parts, **Map exploration** and **A***. The **Map exploration** defines several functions to solve a metaheuristic problem of finding treasures on a map with rocks and holes. It generates a vector that explores a whole portion of the map, finds the rocks and holes, and returns their positions, based on the fitness of the .
With the positions, it calculates the distances between the rocks. Then, it tries to find the shortest path between the actual position of the rover and the closest rock, until it unites all.
The code had several issues and limitations. For example, it relies on external modules that are not provided, and there are no comments or documentation to explain the code's purpose and usage.

Once we know the position of the rocks and the holes on the map, we tried to implement the **A* search algorithm** to find the shortest path between two points on a grid. It uses the number of movements as heuristic function and store the path in a list of tuples. It then translates the list into a path vector, which is passed to a fitness function. The problem with this algorithm as mentioned before, is that it doesn't use a metaheuristic, nor does it take into account the fact that there are holes along the way, which means it can be optimized.
In terms of results and knowing that the best possible solution is *9320* fitness value, with this approach we obtain fitness values of *4500* as best.

2.2 Metaheuristic approach

After the non-metaheuristic approaches, we have moved on to the metaheuristic ones, starting with **PSO**.

2.2.1 Particle Swarm Optimization

We implemented a **Particle Swarm Optimization** (PSO) algorithm combined with a **Multi-Objective Evolutionary Algorithm** (MOEA) as a first metaheuristic approach. The PSO algorithm seeks to find the input that maximizes this fitness function, subject to some constraints defined by the bounds of the input.

The code initializes a population of particles, where each particle represents a potential solution. It then updates the positions and velocities of the particles iteratively based on the particles' previous positions, the global best position found so far, and the best position each particle has found so far. The algorithm also applies hill climbing to refine the solution for each particle after updating its position. After each iteration, the code clusters and recombines the particles to generate a new set of particles.

The code prints the current best solution and its fitness value at each iteration. If the first obtained fitness is less than 3000, the algorithm is restarted. Finally, the code returns the best position found and its fitness value.

MOEA (Multi-Objective Evolutionary Algorithm) is being used because it is a powerful optimization technique that can handle multiple conflicting objectives simultaneously. This is particularly useful in the case of the problem at hand, as there are multiple objectives to consider, such as minimizing fuel consumption, avoiding obstacles, and maximizing the number of valuable rocks found. MOEA can help us find a set of solutions that balance these objectives effectively.

Using PSO we got the best results, but we continued to explore with also just a Genetic Algorithm.

2.2.2 Genetic Algorithm

We implemented a genetic algorithm to see if we can get better results than the PSO MOEA. The genetic algorithm starts by creating a population of individuals, where each individual is a potential solution to the problem. It then evaluates the fitness of each individual in the population using a fitness function, which is defined externally.
The algorithm then selects two individuals from the population to be parents, based on their fitness. It then performs crossover and mutation operations on the selected parents to create two new individuals, which are added to a new population. This process is repeated until a predetermined number of generations is reached.

First of all, we initialize a random population of 100 individuals, with the same vector length used in the PSO. The reason we have chosen to initialize the population with 100 random individuals is to explore a wide range of the search space in the early stages of the algorithm.
This helps to avoid getting stuck in local optima, which can limit the search for the global optimum.
After running the PSO algorithm iteratively, we select the best individuals from the population based on their fitness values and input them into the genetic algorithm. This maintains diversity in the population and ensures that the best solutions found by the PSO algorithm are not lost.
The genetic algorithm is then used to further refine the solutions and improve their fitness values. Overall, this two-stage approach helps to balance exploration and exploitation of the search space, leading to better performance of the optimization algorithm.

First of all, we select the best two parents to apply the genetic operators over them. Once we have the population, we evaluate every individual, selecting the best solutions based on their fitness value.

For each iteration, we apply genetic operators over the best individuals, **crossover** on each iteration, and a **mutation** based on a probability.

In the **crossover**, we select to random cross points in the length of the vector and segments of the bit string are exchanged between the parents at the defined crossover points. In particular, the segments between the lowest crossover point and the highest crossover point are exchanged.

In this way, a child takes the segment of the first parent before the crossover point and the segment of the second parent after the crossover point, while the other child takes the segment of the second parent before the crossover point and the segment of the first parent after the crossover point.

For the **mutation**, we randomly choose between **replacing** a movement for a random new movement, **inserting** a new movement, at the end if the vector is long enough or in a random position otherwise. We can also delete a random movement if the vector is long enough.

At the end of the algorithm, the best individual in the final population is returned as the solution to the problem. This code uses elitism, which means that the best individuals from the previous generation are directly copied to the next generation without any modification. This helps to ensure that the best solutions are not lost in the evolution of the population. However, we still have gotten better results with the PSO approach.

2.2.3 Main Approach

Seeing that the PSO approach worked best, but GA also provided promising results, we have decided to merge them. The final code includes a mix between PSO and GA.

The main idea of this implementation is to start the genetic algorithm with good individuals in its population, in this case, the best results from the PSO.

The pseudocode of this implementation can be seen in the algorithm 1

Algorithm 1 Main Approach pseudocode

```
 1: procedure Particle Swarm Optimization
 2:     for all iterations do
 3:         Update each particle's position and velocity
 4:         Apply Hill Climbing
 5:         Update the particle's best position and global best position

 6:         procedure MOEA
 7:             Generate a new set of particles by clustering and recombining
 8:             Update particle positions, velocities, and best positions
 9:     if $BestFitness \leq 3000$ then
10:         Restart algorithm
        return bestFitness & bestIndividual

11: procedure Genetic Algorithm
12:     Create an initial population based on PSO results
13:     for all generations do
14:         Calculate fitness of the population
15:         Select best parents
16:         Perform crossover and mutation to create a new population
17:         Apply elitism
        return bestFitness & bestIndividual
```

In order to test the impact of the starting population in the genetic algorithm, we tried different versions of this algorithm:

- Initial population composed of the best solution from PSO.
- A percentage of the Initial population composed of the best solution from PSO.
- Best solution of PSO included once in the population

The best results we got was obtained using the intial population composed only of the best PSO obtained solution. In this case, we are short of diversification, since we considered the PSO as a diversification factor and the Genetic Algorithm as an intesification of the previously obtained solution.

2.2.4 Performance and Results

In terms of performance, we will ignore the performance of of the non-metaheuristic approaches, as our priority has been metaheuristic methods. As a brief comparison between metaheuristic and non-metaheuristic methos can be seen in the table 1.

Algorithm used	**Best result**	**Average runtime (minutes)**
Shortest Path	2500	1
A*	4500	0,05
PSO	5320	4,30
Genetic	4340	5,45
PSO + Genetic	7370	10,15

Table 1: General comparison between approaches

As a brief conclusion, we can see that the non-metaheuristical approaches are way faster than PSO or Genetic algorithms. Even so, we cannot take into account these results, since the main goal of this study is to use metaheuristic methods.

We can also see this differences in result in the figure 1.

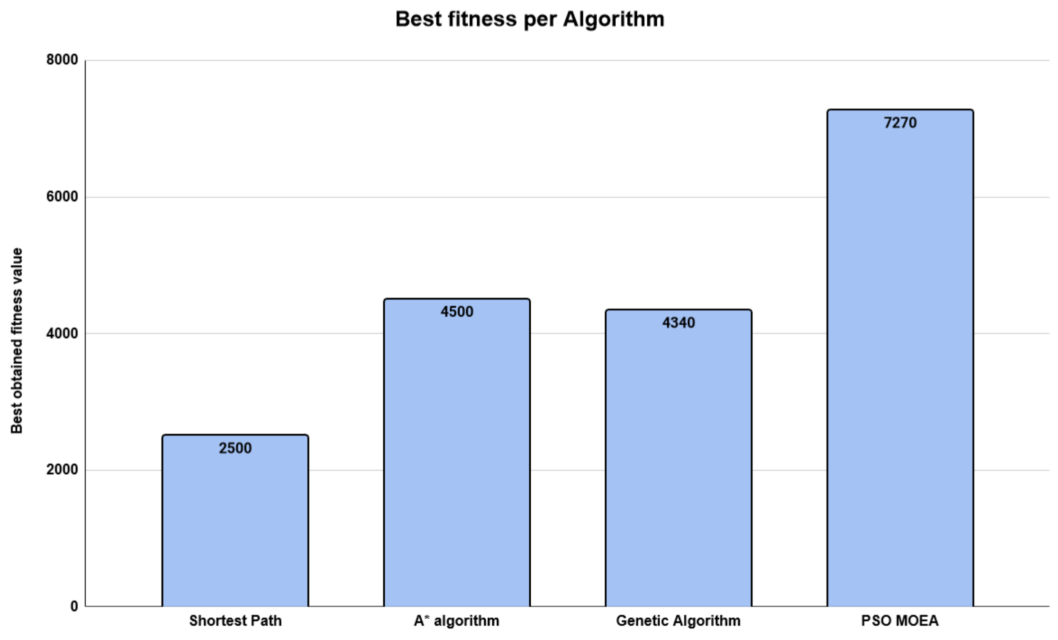

Figure 1: Fitness per algorithm

About the **Particle Swarm Optimization**, we consider it as the diversification component of our main approach. As it can be seen in the table 1, PSO obtains better results than Genetic, which means that, in this specific context, PSO explore a bigger portion of the search space, as it is capable of finding one more rock than the Genetic.

After some hyperparameter optimization, we have noted that the best performance is obtained with vectores of a size between 65 and 70 moves. Although the algorithm has more hyperparameters, the most representative one is the dimension number, i.e. the size of the vector.

We can see how the solutions improve in the PSO in the graph 2.

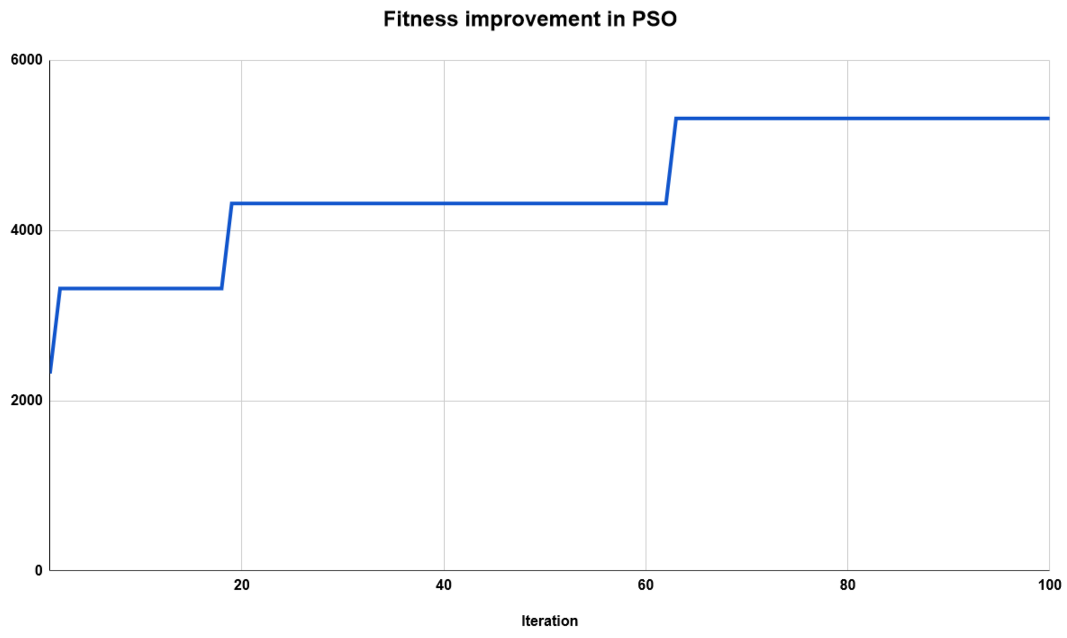

Figure 2: Fitness improvement in the PSO Algorithm

Once we have the best solution given, we can use it in the **genetic algorithm**.

This genetic algorithm is capable of improving any previous obtained solution, apart from starting from scratch. We can see how we can improve the initial results from any of the algorithm previously explained, as it can be seen in the graph 3.

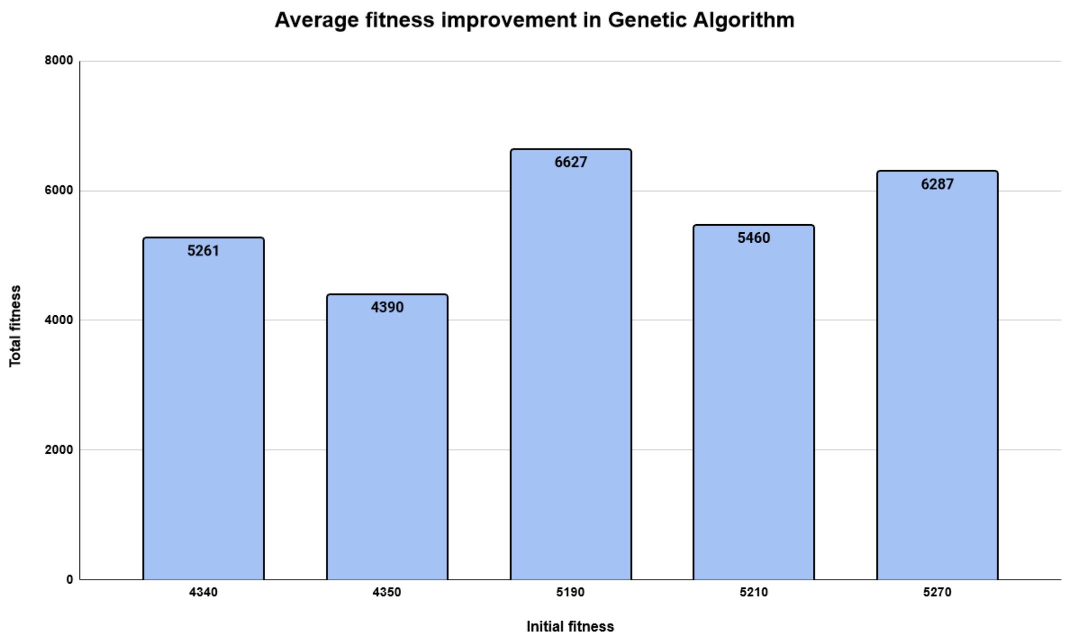

Figure 3: Fitness improvement in the Genetic Algorithm

We can see that the best final results are obtained starting from a "not that good" solution. That can be explained by the encoding of the solutions, where a sligthly better solution can be understood as a completely different path and search space.

As a conclusion to our main approach, we obtained a solution with a fitness value of 5190. After using this solution as part of the intial population in the genetic algorithm, we obtained a solution with a fitness value of 7370.

With that solution that represents a path in the map, and knowing the position of rocks and holes from the non-metaheurisitc approaches, we can graph the map and our solution. This map can be seen in the graph 4.

As an extra information, our best vector, which means our optimal path, can be encoded in the vector *path*:

$$path = \begin{bmatrix} 0 & 2 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 2 & 0 & 0 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 1 \\ 1 & 3 & 3 & 0 & 2 & 1 & 1 & 1 & 1 & 3 & 3 & 3 & 1 & 1 & 1 & 2 & 1 & 2 & 2 & 2 & 2 & 2 \end{bmatrix} \quad (1)$$

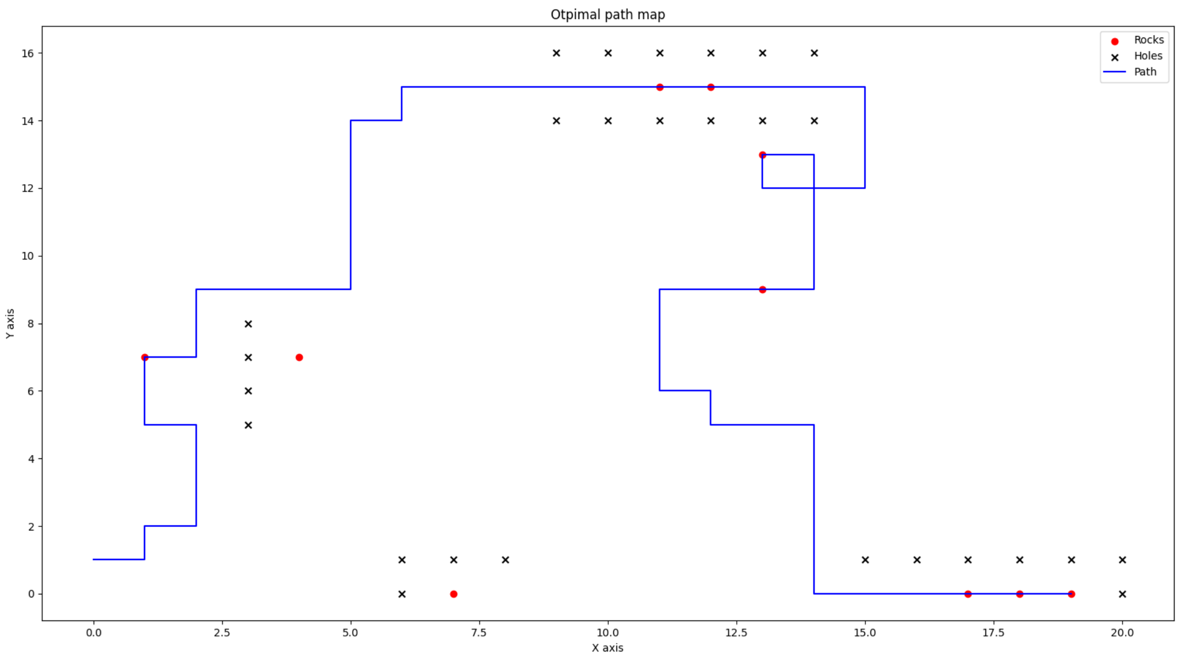

Figure 4: Map of our best solution

As a brief conclusion from this map, we observe that we are somehow lacking in diversification, because we are not finding every single rock in the map.

In addition, although we are able to avoid the holes successfully, we are not able to explore the area around these holes in depth, causing us to lose a rock along the way.

Finally, we see that this solution leads to a small loop in the path, making us make more moves than really necessary.

2.3 Conclusion

To summarise, the Mars perseverance 2020 mission poses some significant challenges regarding the optimization of the exploration while conserving resources and energy.

Our study has explored various non-metaheuristic and metaheuristic approaches to identify the optimal path for the Perseverance Mars rover. While non-metaheuristic algorithms like the Shortest Path algorithm and Map Exploration using A* have limitations, the metaheuristic approach of Particle Swarm Optimization (PSO) combined with a Multi-Objective Evolutionary Algorithm (MOEA) has yielded promising results.
We also tested a Genetic Algorithm, which provided interesting results but did not outperform the PSO + MOEA approach. Our findings suggest that the PSO MOEA algorithm is an effective solution for optimizing the exploration path of the Perseverance rover, using the PSO for diversification and the GA for intensification.
If the optimization process is to be even further improved in the future, the approach would continue to be using more metaheuristic or hybrid algorithms as we have proven in this study.