

A GENETIC ALGORITHM FOR SOLVING THE ROVER ROUTING PROBLEM

Wojciech Burzyński*, Mariusz Kaleta**

* Warsaw University of Technology
Doctoral School
Pl. Politechniki 1, 00-661 Warsaw, Poland

** Warsaw University of Technology
Institute of Control and Computational Engineering
Nowowiejska 15/19, 00-665 Warsaw, Poland

NASA’s Mars 2020 mission, featuring the Perseverance Rover and Ingenuity helicopter, demonstrated the viability of multi-vehicle extraterrestrial missions. In this paper, we present the Rover Routing Problem (RRP), a variant of the Team Orienteering Problem (TOP) tailored for routing a fleet of planetary rovers. The objective is to visit a subset of points of interest (PoIs) and perform research tasks while satisfying planning-horizon and energy constraints. We first introduce a mathematical formulation of the problem; however, exact solution methods struggle to scale beyond small instances (up to 8 PoIs). To address large-scale scenarios, we propose a genetic algorithm incorporating domain-specific mutation operators. Experimental results show that the proposed algorithm significantly outperforms a greedy baseline on instances up to 1,000 points of interest, yielding objective score improvements of up to 27% on dense domains and up to 52% on sparse domains within a 3-minute time limit. Furthermore, since unlimited battery capacity RRP is a relaxation of TOP, our algorithm consistently outperforms results in the published literature for standard benchmark TOP instances, confirming its effectiveness.

our algorithm also outperforms state-of-the-art results on benchmark TOP instances, confirming its quality.

1 Introduction

Ever since the *Sojourner* rover landed in 1997, NASA and other space agencies have sent multiple rovers, landers, and orbiters to survey the Martian surface to search for evidence of past life, locate water resources, and evaluate conditions for future human exploration. More recently, the *Mars 2020* mission successfully operated the *Perseverance* rover alongside the *Ingenuity* helicopter, demonstrating the operational viability of multi-vehicle extraterrestrial exploration. Inspired by these developments, this paper investigates the problem of routing and scheduling research tasks for a fleet of Martian surface vehicles.

In previous work Burzyński and Kaleta (2025) addressed this routing problem using a Mixed-Integer Linear Programming (MILP) formulation. However, due to computational complexity, exact optimal solutions were obtained only for small instances with up to 8 Points of Interest (PoIs), while instances exceeding 10 PoIs yielded relative optimality gaps above 40%. To overcome these

scalability limitations, this paper investigates an evolutionary approach capable of generating high-quality solutions for significantly larger problem instances.

From an algorithmic perspective, the Rover Routing Problem is closely related to the class of Team Orienteering Problems (TOP), where a fleet of vehicles collects rewards by visiting a subset of targets within a limited operational horizon. Because TOP is NP-hard, exact methods struggle on large instances, prompting extensive reliance on metaheuristic approaches. In the literature, several evolutionary algorithms have been applied to TOP variants, differing primarily in their solution encodings and operators. For instance, Karoum and Elbenani (2016) proposed a clonal selection algorithm featuring random node removal, node insertion, and local search to solve the conventional TOP. Mirzaei et al. (2017) investigated the NSGA-II genetic algorithm for the bi-objective TOP; alongside classic single-point crossover and random insertion, the authors introduced operators focused on minimizing score differences across paths, such as inserting nodes into the worst-scoring path or crossing over extremal parents. Mansfield et al. (2021) used a random-key encoding scheme to solve a TOP variant with service-time-dependent rewards. Additionally, Zhou et al. (2026) proposed a memetic algorithm incorporating two-point crossover and node swapping for a time-dependent TOP with multiple time windows. Overall, a diverse array of representations and genetic operators has been established in the literature.

The main contribution of this paper is the development of a specialized Genetic Algorithm (GA) featuring domain-specific mutation operators designed to effectively explore the solution space. Experimental evaluations demonstrate that our algorithm consistently outperforms a baseline greedy heuristic across large-scale benchmarks.

The remainder of this paper is organized as follows: Section 2 presents the mathematical formulation of the RRP. Section 3 details the proposed Genetic Algorithm, its domain-specific operators, and the reference greedy heuristic. Section 4 discusses the computational experiment results, and Section 5 concludes the paper with directions for future research.

2 Problem formulation

2.1 Problem statement

The Rover Routing Problem (RRP) considers a fleet of autonomous surface vehicles deployed to execute scientific research tasks across a set of PoIs. Given a fixed planning horizon and vehicle energy capacity limits, the goal is to schedule vehicle paths and research task executions to maximize total gathered research rewards.

In this work, we address a baseline deterministic formulation assuming a homogeneous vehicle fleet, deterministic travel times and research rewards, fixed task energy requirements, and a predetermined planning horizon. Moreover, for simplicity, but without losing generality, we assume only one research task at

each PoI, and non-interruptible travel between PoIs.

2.2 Mathematical model

We follow the approach of representing the problem with the Activity-on-Arc network, widely used in project scheduling. For that, we define three types of tasks (activities): research, battery charging, and movement. Let $E = \{(i, j) \mid (i, j) \in V^2 \wedge i \neq j\}$ represent the set of all tasks, and let $V \subset \mathbb{N}$ denote events, that is, the starting and completion of tasks, forming a directed graph $G = (V, E)$. Each task $(i, j) \in E$ is characterized by a duration τ_{ij} , a reward r_{ij} , and a task type z_{ij} . Let $K = \{1, \dots, |K|\}$ denote the set of vehicles, and let B denote maximum battery capacity. Each vehicle is initially located at node $\{0\} \in V$, which represents a common physical location for all rovers. The planning horizon spans from T_0 to T_{max} . The reader is referred to Burzyński and Kaleta (2025) for complete details on the underlying graph representation. The mixed-integer non-linear programming (MINLP) model is formulated as follows:

$$\max \sum_{(i,j) \in E} \sum_{k \in K} r_{ij} x_{ij}^k \quad (1)$$

s.t.:

$$y_0^k = 1 \quad \forall k \in K \quad (2)$$

$$\sum_{j \in V \setminus \{0\}} x_{0j}^k = 1 \quad \forall k \in K \quad (3)$$

$$\sum_{i \in V} x_{ij}^k = y_j^k \quad \forall j \in V \setminus \{0\}, \forall k \in K \quad (4)$$

$$\sum_{i \in V} x_{ji}^k - \sum_{i \in V} x_{ij}^k \leq 0 \quad \forall j \in V \setminus \{0\}, \forall k \in K \quad (5)$$

$$\sum_{k \in K} y_i^k \leq 1 \quad \forall i \in V \quad (6)$$

$$t_0^k = T_0 \quad \forall k \in K \quad (7)$$

$$t_j^k = \sum_{i \in V} x_{ij}^k (t_i^k + \tau_{ij}) \quad \forall j \in V \setminus \{0\}, \forall k \in K \quad (8)$$

$$T_0 \leq t_i^k \leq T_{max} \quad \forall k \in K, \forall i \in V \quad (9)$$

$$b_j^k = \sum_{i \in V} x_{ij}^k (b_i^k + \Delta E(t_i^k, \tau_{ij}, z_{ij})) \quad \forall j \in V \setminus \{0\}, \forall k \in K \quad (10)$$

$$0 \leq b_j^k \leq B \quad \forall j \in V \setminus \{0\}, \forall k \in K \quad (11)$$

$$y_i^k \in \{0, 1\}, x_{ij}^k \in \{0, 1\} \quad \forall i \in V, \forall j \in V, \forall k \in K \quad (12)$$

where the variables are defined as follows:

- x_{ij}^k - Binary decision variable; equals 1 if vehicle k executes task (i, j) , and 0 otherwise.
- y_j^k - Binary indicator variable; equals 1 if vehicle k completes task event j , and 0 otherwise.
- t_j^k - Continuous variable representing the completion time of task event j by vehicle k .
- b_j^k - Continuous variable representing the battery charge level of vehicle k upon completing task event j .

The objective function (1) maximizes the total reward collected across all completed tasks, where r_{ij} is the reward assigned to task (i, j) . Constraints (2)–(6) govern vehicle routing topology: constraints (2) and (3) force each vehicle to originate from the initial node 0, constraint (4) links completion event y_j^k to transit selection x_{ij}^k , constraint (5) ensures logical path continuity (a vehicle cannot exit a node without entering it), and constraint (6) guarantees that each PoI task is executed by at most one vehicle. Unlike standard VRP, visiting PoIs is optional. Furthermore, vehicles are not required to return to the initial node, nor to any other specified node.

Constraints (7)–(9) manage time progression: constraint (7) sets start times to T_0 , constraint (8) accumulates task durations along chosen routes, and constraint (9) enforces schedule adherence within $[T_0, T_{\max}]$.

Constraints (10)–(11) enforce battery state dynamics, where $\Delta E(t_i^k, \tau_{ij}, z_{ij})$ denotes net energy change during task (i, j) . This term incorporates both energy consumption (tool usage and transit) and energy generation (via solar panels or Radioisotope Thermoelectric Generators, RTGs), and is a source of non-linearity in the model. We elaborate on this further in the following section. Finally, constraint (12) enforces binary variable domains.

2.3 Energy model

Constraint (10) introduces non-linearity into the mathematical formulation. Let $P_-((i, j), t)$ denote power consumption, and $P_+((i, j), t)$ denote generator power output while performing task (i, j) of type z_{ij} starting at t . The net power rate $P((i, j), t) = P_+((i, j), t) + P_-((i, j), t)$ determines the net energy gain or loss $\Delta E((i, j), t)$ over duration τ_{ij} :

$$\begin{aligned} \Delta E((i, j), t) &= \int_t^{t+\tau_{ij}} P((i, j), s) \, ds \\ &= \int_t^{t+\tau_{ij}} P_+((i, j), s) \, ds + \int_t^{t+\tau_{ij}} P_-((i, j), s) \, ds \end{aligned} \quad (13)$$

Power consumption $P_-((i, j), t)$ is modeled via task-dependent constants:

$$P_-((i, j), t) = \begin{cases} -a & z_{ij} = M \\ -b & z_{ij} = R \\ -c & z_{ij} = C \end{cases} \quad (14)$$

where $a, b, c > 0$ represent fixed power draw rates, and M , R , and C are task types: movement, research, and charging, respectively. Power generation $P_+((i, j), t)$ can be modelled as constant for RTG systems or as a half-wave rectified sine wave for solar panels. Let T_d denote the length of a Martian solar day (sol), where 1 sol = 24 h 39 min 35 s. Let T_{dl} represent daylight duration and T_s denote sunrise time. For simplicity, we assume sunrise occurs at one-quarter of a sol ($T_s = T_d/4$) and daylight lasts for half of a sol ($T_{dl} = T_d/2$). Let I_0 be the peak solar irradiance ($I_0 = 430 \text{ W/m}^2$ for Mars), and let S denote the surface area of the solar panel. The cumulative energy generated up to time t is calculated as follows:

$$\int P_+((i, j), t) ds = \left(\sum_{i=0}^{n-1} \frac{2I_0 T_{dl}}{\pi} \right. \quad (15)$$

$$\left. + \frac{2I_0 T_{dl}}{\pi} \left[1 - \cos \left(\frac{\pi \min(T_{dl}, \max(0, T_{day} - T_s))}{T_{dl}} \right) \right] \right) S + C$$

$$n = \lfloor \frac{t}{T_d} \rfloor \quad (16)$$

$$T_{day} = t - nT_d \quad (17)$$

Equations (16) and (17) are auxiliary relations used to compute the number of full days completed prior to time t and the elapsed number of hours within the current day, respectively. Equation (15) consists of two components: the first sums total solar insolation (energy per unit area, solar irradiance integrated over time), while the second computes insolation in remaining day. The min and max operators enforce zero solar gain during nighttime hours. Finally, total integrated insolation is multiplied by the solar panel area S to yield total energy gain. Compared to models found in literature, such as Appelbaum and Flood (1990), this formulation does not take into account latitude and season of the year. However, local and temporary conditions can be modelled by adjusting values of I_0 , T_s and T_{dl} .

3 Methods

3.1 Genetic algorithm overview

The mixed-integer non-linear programming (MINLP) formulation presented in Section 2 has been linearized and transformed into a mixed-integer linear programming (MILP) problem, which was then solved with an MILP solver in Burzyński and Kaleta (2025); however, optimal solutions were obtained only

for small instances. To address large-scale instances efficiently, we propose a tailored Genetic Algorithm (GA).

Algorithm 1 outlines the overall evolutionary framework. The process initializes by generating a random population of N individuals. In each generation, a new population is constructed by preserving a small fraction $E_{\text{threshold}}$ of best individuals (elitism), followed by parent selection via roulette wheel selection. Offspring are generated through crossover and mutation operators before being added to the offspring pool.

To evaluate individuals, the population is sorted lexicographically by fitness: primary sorting maximizes collected research reward, while secondary sorting minimizes total schedule duration. The secondary criterion prefers compact schedules, increasing the likelihood that subsequent evolutionary operations yield feasible, high-reward offspring.

The evolution continues until the stopping condition is satisfied, which may be either defined by the number of generations or the wall time.

Algorithm 1 Genetic Algorithm Framework

```

1:  $P \leftarrow \emptyset$ 
2: for  $i = 1$  to  $N$  do
3:    $\text{ind} \leftarrow \text{GenerateRandomIndividual}()$ 
4:    $P \leftarrow P \cup \{\text{ind}\}$ 
5: end for
6: while  $\neg \text{StoppingCondition}()$  do
7:    $P_{\text{next}} \leftarrow \emptyset$ 
8:   for  $i = 1$  to  $E_{\text{threshold}}$  do ▷ preserve elite individuals
9:      $P_{\text{next}} \leftarrow P_{\text{next}} \cup \{P[i]\}$ 
10:  end for
11:  while  $|P_{\text{next}}| < N$  do
12:     $p_1, p_2 \leftarrow \text{RouletteSelection}(P)$ 
13:     $\text{offspring} \leftarrow \text{Crossover}(p_1, p_2)$ 
14:    if  $\text{ShouldMutate}()$  then
15:       $\text{offspring} \leftarrow \text{Mutation}(\text{offspring})$ 
16:    end if
17:     $P_{\text{next}} \leftarrow P_{\text{next}} \cup \{\text{offspring}\}$ 
18:  end while
19:   $\text{SortLexicographically}(P_{\text{next}})$ 
20:   $P \leftarrow P_{\text{next}}$ 
21: end while

```

3.2 Genotype representation and phenotype decoding

Solutions are encoded using an indirect, variable-length representation.

Genotype Structure (Chromosomal Encoding): An individual’s genotype is represented as a sequence of integer genes representing visited PoIs.

0	1	3	6	5	0	2	4	0	7	9	12
---	---	---	---	---	---	---	---	---	---	---	----

Table 1: Example genotype chromosomal representation ($|K| = 3$ vehicles).

The depot location is denoted by 0 and appears $|K|$ times in the chromosome. A vehicle trajectory (path) consists of the sequence of PoIs bounded between two consecutive 0 delimiters (or the chromosome boundary). Table 1 illustrates a genotype encoding for a fleet of $|K| = 3$ vehicles across three disjoint paths. This chromosomal representation allows PoIs to be reordered, added, or removed without violating topological solution validity.

Phenotype Expression and Decoding Procedure: To compute solution feasibility and evaluate fitness, a deterministic decoding function $\mathcal{D} : \text{Genotype} \rightarrow \text{Phenotype}$ maps the chromosomal sequence into an executable schedule of discrete task events. The resulting phenotype consists of an explicit timeline of three task activity types:

- **Movement (M):** Spatial movement between PoIs.
- **Research (R):** Scientific task execution at a target PoI.
- **Charging (C):** Stationary idle time required for solar or RTG power recharge.

During decoding, charging activities (C) are inserted greedily only when projected state-of-charge constraints would otherwise be violated. The duration of each inserted charging activity is calculated dynamically to provide just enough energy to reach and complete the subsequent task. Tasks extending beyond the maximum planning horizon $T_{\max}$ are truncated and discarded.

Table 2 demonstrates the phenotypic decoding for the first vehicle path in Table 1. At $t = 0$, the vehicle originates with an initial battery level of 23. Transit to PoI 1 requires 5 time units and consumes 10 energy units. Executing the research task at PoI 1 requires an additional 2 units of time and 2 units of energy. The vehicle then travels to PoI 3, consuming 4 time units and 8 energy units, followed by research execution requiring 2 units of time and energy. Because direct transit to PoI 6 requires 6 energy units but only 1 unit remains in the battery, a charging task (C) is dynamically inserted following research execution at PoI 3 to prevent depletion. As a result, the vehicle reaches PoI 6 at $t = 21$ and completes research at $t = 23$. Finally, because transit to PoI 5 requires an additional 5 time units—resulting in an arrival time of $t = 28 > T_{\max} = 25$ —the task exceeds the operational window and is truncated.

Fitness Function Formulation: The fitness function $\mathcal{F}(\text{ind})$ maps a decoded phenotype to a scalar objective value corresponding to Eq. (1):

$$\mathcal{F}(\text{ind}) = \sum_{(i,j) \in E_{\text{decoded}}} r_{ij} \quad (18)$$

$t_{\max} = 25$										
	0	1M	1R	3M	3R	C	6M	6R	5M	...
t	0	5	7	11	13	18	21	23	28	...
b	23	13	11	3	1	11	5	3		...

Table 2: Phenotypic decoding of the first path from Table 1 ($T_{\max} = 25, B = 23$). Gray highlights denote greedily inserted charging tasks (C); struck-out entries denote infeasible tasks truncated beyond $T_{\max}$.

where E_{decoded} is the set of valid tasks completed within $[T_0, T_{\max}]$ resulting from the decoding procedure.

3.3 Mutation Operators

We evaluate two distinct sets of mutation operators. The first set comprises standard operators adapted from the VRP and TOP literature (Sylejmani et al., 2012):

AddUnused: Selects an unvisited PoI at random and inserts it into a random position in the genotype.

Reinsert: Selects a visited PoI in the genotype and moves it to a different position.

SwapWithUnused: Swaps a randomly chosen PoI in the genotype with an unvisited PoI.

Preliminary experimentation indicated that single-element manipulations rarely produce immediate fitness improvements in constrained temporal windows (Section 4). To overcome this limitation, we introduced a suite of sequence-based heuristic mutation operators:

AddUnusedInverse: Selects a random unvisited PoI and inserts it at the specific chromosomal location that minimizes incremental schedule duration expansion.

BreakEnd: Removes a terminal sub-sequence of PoIs from a selected vehicle route and greedily appends unvisited PoIs until $T_{\max}$ is saturated.

BreakMiddle: Removes an intermediate sub-sequence of PoIs within a route and greedily re-fills the created gap with unvisited PoIs until $T_{\max}$ is reached.

Modified 2-Opt: Executes a full 2-opt local search procedure over the genotype. Rather than introducing unvisited PoIs, it optimizes the ordering of existing PoIs to minimize total transit time. The operator is adapted for open, multi-vehicle paths separated by K depot delimiters (0) rather than closed single-vehicle cycles. Segment inversions and swaps are performed both within individual routes (intra-route) and across distinct vehicle trajectories (inter-route).

3.4 Crossover operators

Two tailored crossover operators were implemented to combine parental genetic material:

Order Crossover (OX): Adapted from classical order crossover (Davis et al., 1985), modified to handle variable-length chromosomes and multiple depot delimiters (denoted by 0). A contiguous sub-sequence from Parent 1 is copied directly to the offspring, and remaining unique genes are filled in order from Parent 2.

Path-Mixing Crossover: Copies entire vehicle paths alternatingly between Parent 1 and Parent 2. Resulting duplicate PoIs are resolved by preserving their first occurrence in the offspring chromosome and removing subsequent duplicates.

3.5 Greedy construction baseline

To establish a comparative performance benchmark against the GA, a deterministic greedy construction algorithm was implemented (Algorithm 2). For each vehicle sequentially, the algorithm iteratively selects the unvisited PoI nearest to the current vehicle location and appends it to the route until no further PoI can be visited within $T_{\max}$. The constructed genotype is then passed to the decoding function $\mathcal{D}$ (Section 3.2) to insert required charging tasks and yield the final phenotypic schedule.

Algorithm 2 Greedy Baseline Algorithm

```

1:  $P \leftarrow \emptyset$ 
2:  $\text{UnvisitedPoIs} \leftarrow \text{PoIs} \setminus \{0\}$ 
3: for  $k = 1$  to  $|K|$  do
4:    $P \leftarrow P \cup \{0\}$ 
5:    $\text{currentPoI} \leftarrow 0$ 
6:    $t \leftarrow T_0$ 
7:   while  $t < T_{\max}$  and  $\text{UnvisitedPoIs} \neq \emptyset$  do
8:      $\text{nextPoI} \leftarrow \text{findNearestUnvisited}(\text{currentPoI}, \text{UnvisitedPoIs})$ 
9:     if  $\text{canVisitWithinHorizon}(\text{currentPoI}, \text{nextPoI}, t, T_{\max})$  then
10:       $P \leftarrow P \cup \{\text{nextPoI}\}$ 
11:       $\text{UnvisitedPoIs} \leftarrow \text{UnvisitedPoIs} \setminus \{\text{nextPoI}\}$ 
12:       $t \leftarrow t + \tau_{\text{currentPoI}, \text{nextPoI}}$ 
13:       $\text{currentPoI} \leftarrow \text{nextPoI}$ 
14:     end if
15:   end while
16: end for
17:  $\text{phenotype} \leftarrow \text{decodeGenotype}(P)$  ▷ Insert charge tasks and handle
    truncation

```

4 Numerical results

The algorithms presented in Section 3 were implemented in C++ and evaluated on an AMD Ryzen 7 8845HS processor (3.8 GHz) equipped with 32 GB RAM.

To systematically evaluate operator efficacy, two genetic algorithm configurations were benchmarked:

- **GA I:** Employs standard single-element mutation operators (*AddUnused*, *Reinsert*, and *SwapWithUnused*).
- **GA II:** Employs sequence-based and domain-tailored operators (*Reinsert*, *AddUnusedInverse*, *Modified 2-Opt*, and *BreakEnd*).

While *BreakMiddle* was implemented during initial testing, it yielded no statistically significant improvement over *BreakEnd* and was excluded from final runs. Similarly, Order Crossover (OX) was omitted due to poor performance on variable-length chromosomes with delimiter constraints.

4.1 Experimental setup and parameter tuning

Prior to benchmark evaluations, hyperparameter fine-tuning was performed across representative test instances. The GA configuration was set to a population size of $N = 100$, an elitism threshold of 2%, and a mutation probability of 20%. The *BreakEnd* operator was configured to excise 15–20% of PoIs from a target route before greedy re-filling.

Unless stated otherwise, test instances adopted the following physical parameterization:

- Battery capacity: $E_{\max} = 263,520$ J.
- Effective solar array area: $S = 0.15$ m².
- Planning horizon: $T_{\max} = 10,000$ s post-sunrise.
- Peak solar irradiance: $I_0 = 430$ W/m², corresponding to top-of-atmosphere Martian surface insolation Surampudi et al. (2001).
- Fleet size: $|K| = 3$

Experiments were conducted on synthetic instances comprising 10–1,000 PoIs randomly distributed, using uniform distribution, over deployment regions of 2.5×2.5 km² and 10×10 km² containing obstacles. Paths between PoIs were constructed using (PRM*) algorithm. Travel duration and energy cost matrices were constructed using a with the Floyd-Warshall all-pairs shortest path algorithm. An example solution for instance with 600 PoI is shown in Figure 1.

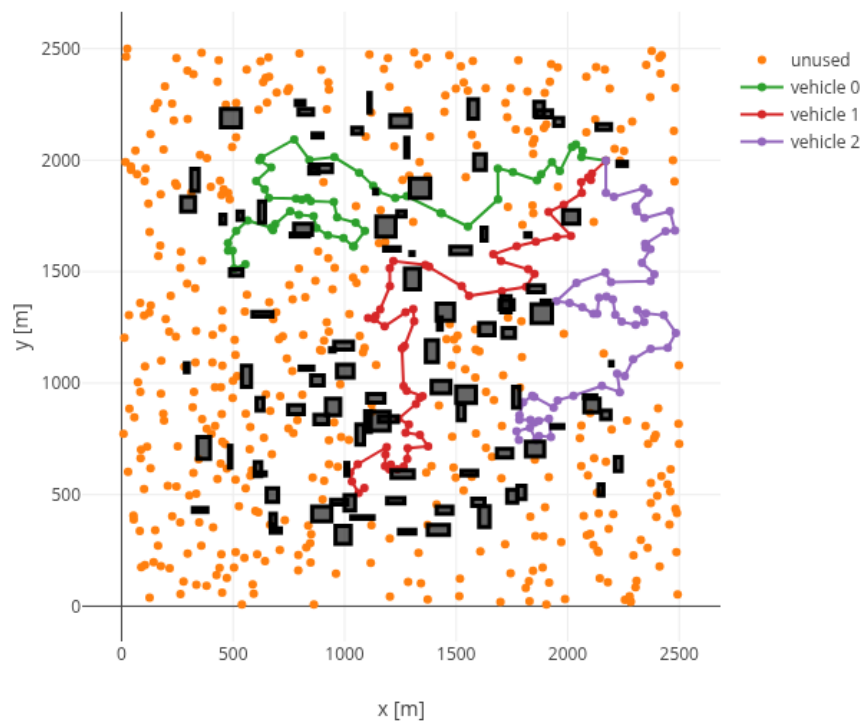


Figure 1: Example solution for a 600-PoI instance.

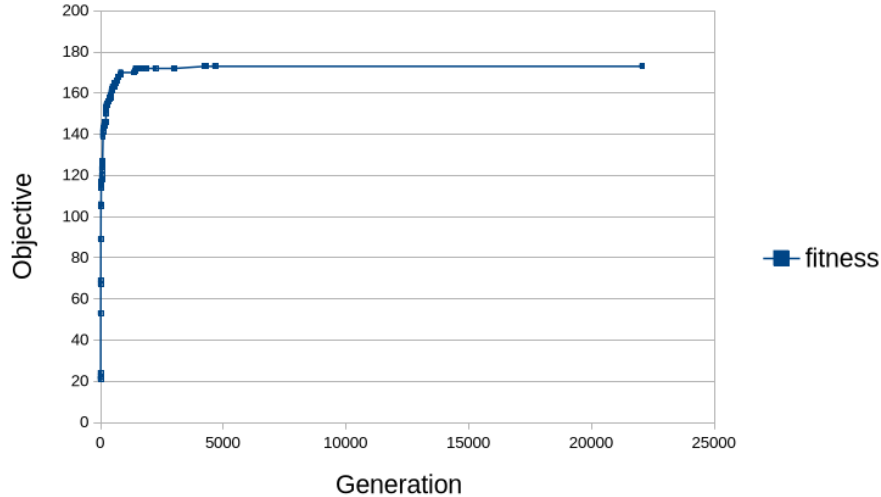


Figure 2: Fitness progress of one of the GA II runs for the dataset with 600 PoIs.

4.2 Operator benchmark evaluation

Table 3 presents comparative performance metrics for the Greedy baseline algorithm, GA I, and GA II over a runtime limit of 3 minutes per GA instance. This termination threshold provided a trade-off between computational efficiency and search convergence, as empirical convergence curves indicated that over 90% of objective gains occurred within the first 120 seconds. Figure 2 shows example fitness progress. Extended runs up to 4 hours yielded negligible solution improvements.

Except on small instances (100 PoIs), GA I was consistently outperformed by the simple Greedy heuristic. Because standard mutation operators perturb only single PoIs at a time, candidate solutions frequently became trapped in localized plateaus; escaping these required compound multi-gene mutations that individual generational steps rarely achieved.

In contrast, GA II significantly outperformed both the Greedy baseline and GA I across all benchmark instances. Sequence-manipulation operators (*Break-End*) effectively restructured paths, while *AddUnusedInverse* inserted candidate PoIs into locally optimal path slots. Furthermore, integrating *2-Opt* compactively reordered route geometry, freeing temporal headroom that enabled subsequent mutation steps to insert additional unvisited target PoIs.

4.3 Comparative evaluation against literature benchmarks

To contextualize algorithm efficacy against established standards, GA II was evaluated on public standard instances from Chao et al. (1996). The benchmark

PoIs	Area (km $\times$ km)	Greedy		GA I	GA II
		Time (s)	Obj	AVG Obj	AVG Obj
100	10 $\times$ 10	<0.01	15	15.6	18.0
100	2.5 $\times$ 2.5	<0.01	49	50.0	66.4
600	10 $\times$ 10	0.01	28	29.0	42.8
600	2.5 $\times$ 2.5	0.03	130	113.6	172.0
800	10 $\times$ 10	0.02	45	35.2	54.8
800	2.5 $\times$ 2.5	0.06	164	110.6	208.4
1000	10 $\times$ 10	0.03	51	41.6	65.2
1000	2.5 $\times$ 2.5	0.09	172	112.0	218.2

Table 3: Performance comparison between the Greedy Baseline, GA I (standard operators), and GA II (proposed domain operators) across datasets up to 1,000 PoIs (GA execution time limit: 180 s).

set consists of 353 distinct instances varying in target density and planning horizons.

Because existing Team Orienteering Problem (TOP) literature neglects non-linear energy replenishment and task-dependent charging constraints, the proposed model was transformed into a relaxed TOP instance by setting battery capacity $E_{\max} \rightarrow \infty$. This relaxation eliminates charging station insertion during decoding, allowing direct comparison against state-of-the-art TOP heuristics.

In the original TOP formulation each vehicle must end in a specific location. In the case of the RRP, vehicles may finish their paths in any node; therefore, the RRP with infinite battery capacities is a relaxation of TOP.

Table 4 compares the objective values obtained by GA II against published TOP results from Hammami et al. (Hammami et al., 2020). All instances in this table feature 102 PoIs and a fleet of $|K| = 4$ vehicles, differing only in the length of their planning horizons. While this table presents results exclusively for the largest instances in the dataset, consistent improvements were observed across all remaining instances. As anticipated, eliminating battery constraints enables GA II to achieve higher total rewards across all evaluated benchmark sets, demonstrating the strength of the underlying evolutionary search when mapped onto relaxed routing problems.

5 Summary

In this paper, we introduced the Rovers Routing Problem, a novel variant of the Team Orienteering Problem with non-linear, time-dependent battery dynamics and task-dependent energy replenishment. We presented a mixed-integer programming formulation alongside a tailored Genetic Algorithm featuring specialized sequence-based mutation and phenotype-decoding operators. Empirical evaluations across synthetic topological networks and relaxed public benchmarks demonstrated that the proposed evolutionary approach significantly out-

Instances	Planning horizon	Literature results	GA II
P7.4g	35	217	502.2
P7.4h	40	285	565.4
P7.4i	45	366	646.8
P7.4k	55	520	768.4
P7.4l	60	590	825.6
P7.4m	65	646	839.6
P7.4n	70	730	925.8
P7.4o	75	781	967.0
P7.4p	80	846	1017.4
P7.4r	90	970	1102.0
P7.4s	95	1022	1144.8
P7.4t	100	1077	1169.2

Table 4: Comparative objective results on Chao et al. Chao et al. (1996) benchmark sets. Literature results represent constrained TOP optima from Hammami et al. Hammami et al. (2020); GA II results represent the relaxed battery formulation.

performs standard operator variants and baseline greedy heuristics.

Several promising avenues remain for future research. Problem extensions could incorporate stochastic reward distributions, system reliability constraints, and heterogeneous vehicle fleets with varying power consumption profiles. Furthermore, while the current deterministic greedy charging heuristic is effective for constant-power sources such as Radioisotope Thermoelectric Generators (RTGs), photovoltaic generation varies significantly with the diurnal solar cycle. Explicitly scheduling charging activities around peak solar irradiance could minimize overall downtime and improve route compactness. Finally, extending the operational framework to multi-sol and multi-week horizons will enable long-term mission planning under seasonal energy fluctuations.

6 References

- J. Appelbaum and D. J. Flood. Solar radiation on mars. *Solar Energy*, 45(6):353–363, 1990. ISSN 0038-092X. doi: [https://doi.org/10.1016/0038-092X\(90\)90156-7](https://doi.org/10.1016/0038-092X(90)90156-7). URL <https://www.sciencedirect.com/science/article/pii/0038092X90901567>.
- W. Burzyński and M. Kaleta. A mixed integer programming approach to the rechargeable rover routing problem on mars. *Foundations of Computing and Decision Sciences*, 50(3):325–346, 2025. doi: 10.2478/fcds-2025-0012. URL <https://doi.org/10.2478/fcds-2025-0012>.
- I.-M. Chao, B. L. Golden, and E. A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464–474, 1996.

- ISSN 0377-2217. doi: [https://doi.org/10.1016/0377-2217\(94\)00289-4](https://doi.org/10.1016/0377-2217(94)00289-4). URL <https://www.sciencedirect.com/science/article/pii/0377221794002894>.
- L. Davis et al. Applying adaptive algorithms to epistatic domains. In *IJCAI*, volume 85, pages 162–164, 1985.
- F. Hammami, M. Rekik, and L. C. Coelho. A hybrid adaptive large neighborhood search heuristic for the team orienteering problem. *Computers & Operations Research*, 123:105034, 2020. ISSN 0305-0548. doi: <https://doi.org/10.1016/j.cor.2020.105034>. URL <https://www.sciencedirect.com/science/article/pii/S0305054820301519>.
- B. Karoum and B. Elbenani. Clonal selection algorithm for the team orienteering problem. In *2016 11th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pages 1–5, 2016. doi: 10.1109/SITA.2016.7772274.
- A. Mansfield, S. Manjanna, D. G. Macharet, and M. Ani Hsieh. Multi-robot scheduling for environmental monitoring as a team orienteering problem. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6398–6404, 2021. doi: 10.1109/IROS51168.2021.9636854.
- M. H. Mirzaei, K. Ziarati, and M.-T. Naghibi. Bi-objective version of team orienteering problem (btop). In *2017 7th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 1–7, 2017. doi: 10.1109/ICCKE.2017.8167930.
- R. Surampudi, B. Carpenter, M. El-Genk, L. Herrera, L. Mason, J. Mondt, B. Nesmitz, D. Rapp, and R. Wiley. Advanced radioisotope power systems report. Technical report, NASA, 2001.
- K. Sylejmani, J. Dorn, and N. Musliu. A tabu search approach for multi constrained team orienteering problem and its application in touristic trip planning. In *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*, pages 300–305, 2012. doi: 10.1109/HIS.2012.6421351.
- G. Zhou, J. Luo, J. Bian, and Y. Zhang. Coordinated observation and data transmission scheduling of heterogeneous space platforms: A time-dependent team orienteering problem with multiple time windows. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 56(1):415–427, 2026. doi: 10.1109/TSMC.2025.3624873.