New GA Applied Route Calculation for Multiple Robots with Energy Restrictions

Killdary Aguiar de Santana, Vandilberto Pereira Pinto, Darielson A. Souza, João Lucas Oliveira Torres and Igor Antônio Gomes Teles

July 9, 2020

New GA Applied Route Calculation for Multiple Robots with Energy Restrictions

Killdary A. Santana[#], Vandilberto P. Pinto[*], Darielson A. Souza[+], João L. O. Torres[^], Igor A. G. Teles[#]

[#] *Postgraduate Program in Electrical and Computer Engineering, Federal University of Ceará, Sobral, 62010-560, Brazil*
E-mail: killdary@alu.ufc.br

* *Postgraduate Program in Electrical and Computer Engineering, Federal University of Ceará, Sobral, 62010-560, Brazil*
Institute for Engineering and Sustainable Development – IEDS, University for the International Integration of the Afro-Brazilian Lusophony – UNILAB, Redenção, 62790-000, Brazil
E-mail: vandilberto@unilab.edu.br

+Department of Electrical Engineering, Federal University of Ceará, Fortaleza, 60440-554, Brazil
E-mail: darielson@dee.ufc.br

^Grendene, Sobral, 60440-554, Brazil
E-mail: eng.jlot@gmail.com

***Abstract*— In recent years, there has been a great advance in technology applied to mobile robots. However, route calculation has still been one of the biggest challenges and has received attention from researchers, both in industry and academia where the greater autonomy of robots is linked directly to the calculation of your routes. The complexity of the robot motion planning problem has motivated the development of several algorithms. This stems from the need to integrate robot navigation with sense, efficiency, and route planning, as well as the need to save important resources such as energy, and the participation of multiple robots. Most robot missions must meet multiple points. In addition to having a cost to complete and weights that determine your priority of service if they prove to be quite complex, there is still the possibility that several robots in different locations collaboratively participate in the mission, which increases the complexity of the problem. This paper presents a route calculation approach for multiple collaborative robots with energy constraints and base shifting. The proposed method was modelled using the team orienteering problem (TOP). It combined with the multiple knapsack problem (MKP). This combination allows each robot to have a unique's energy restriction different from the traditional TOP where all agents have the same restriction value, to solve the problem a genetic algorithm was developed. For verifying the method's effectiveness, a comparison between the results of proposed method[1] and this work was made. The results show a good efficiency in solving the problem.**

***Keywords*— Multi-robot, Path Planning, Metaheuristic Algorithm, Team Orienteering Problem.**

I. Introduction

In recent years, there has been a breakthrough in technology applied to mobile robots. However, route calculation is still one of the biggest challenges and has received attention from researchers, both in industry and academia, where the greater autonomy of robots is linked directly to the calculation of your routes. Numerous algorithms have been developed in order to solve the problem of route planning [2], [3]. Such complexity comes from the need to integrate robot navigation with sense, efficiency, and route planning, as well as the need to save significant resources and preserve its mechanical parts [4], [5]. The majority of the missions of robots must meet several points. In addition to having a cost for its completion and have weights that determine your attendance priority, also have the possibility of several robots, in different locations, can participate in the mission collaboratively, which ends up for increasing the complexity of the problem.

This research focuses on detecting missions in which energy restrictions need to be respected. This research focuses on detecting missions in which energy restrictions must be followed. The mission calls for a vehicle team to visit multiple points and collect prizes within the energy limitations of each agent. Its importance comes from research by several authors [6],[7] on the estimation of battery health and discharge, which shows that even similar robots can have different batteries discharged due to their frequency of use or age so that the robots would have distinct energy constraints. This phenomenon needs to be taken into account when calculating the multiple robot route.

The routing problem of several collaborative robots, due to their similarity, has the possibility of being modelled through the team orientation problem (TOP) [8]. The TOP is defined as a set of vertices, in which each vertex has a score or prize. The vertices must be visited by a set of identical agents, where each agent has a limitation of movement cost equal to Tmax. The TOP's goal is to collect the highest score possible within each agent's costs. The TOP is known to be NP-hard, and such problems can be solved either optimality or near-optimality. The Top has several solutions proposed in the literature, among them an ant colony optimization [9], guided local search [10], linear programming [11], evolutionary algorithms [1], [12], among others. But most of the solutions are mono-objective, which aims to maximize premium collection, where cost minimization is a consequence of the higher premium collected, except for [1], [11] and [17], where the last two applied to single-agent problems. In [1], a solution was proposed using an objective function with two objectives. One is to minimize the cost of travel, and second one is to maximize the collection of prizes. This solution was implemented using genetic algorithms, but this solution assumes that all agents have the same cost of locomotion, this resource would hardly be adequate in case of route planning for several collaborative robots.

The problem of the multiple backpacks (MKP) consists of a set of backpacks, in which each backpack has a capacity, and a set of items, where each item has a weight and a value, the objective of the problem is to place the items in the backpacks of in order to collect as much value as possible without exceeding the capacity of each backpack. [13].

This research aims to provide a solution for calculating routes for various energy-restricted robots using a genetic algorithm based on the solution [1]. Our method differs from [1] when combining TOP to MKP, this combination allows the insertion of different mobility restrictions for each robot, making the solution closer to reality. The objective function used in this research is a multi-objective solution equal to that of [11], which calculates the conditioning points by assigning weights to the cost and premium objective, making the solution more adaptable to various applications..

The proposed solution was validated using a test environment composed of symmetric maps, in which the number of vertices to be visited varies between forty and eighty; each vertex has an award. In the tests, scenarios were applied that had different numbers of robots with different energy restrictions. In the tests, the execution of the routes for multiple robots is performed using an algorithm similar to the one of [1] and with the proposed solution of this research, in which a comparison between the obtained results is performed.

The remaining sections of this paper are organized as follows. Section II presents the mathematical formulation of the problem to be considered, how the genetic algorithm works and the built test environment. Section III presents the results obtained in the numerical experiments and discusses the results obtained. Concluding remarks are presented in Section IV.

II. Material and Method

In the current section, the proposed techniques are explained in an outlined manner. Subsection II-A presents the formulation of the combined TOP with the MKP. In II-B, the details of the proposed GA will be presented. Subsection II-C presents the testing environment that the heuristic underwent.

A. Mathematical Formulation

It is given a graph $G(N,A)$ where N list the set $(|N|=N_c)$ and (A) the set of edges. For the $c_{ij}=\forall\, i \square N$ can represent a symmetric matrix with lower cost. It is given N_k robots as a solution, which k is the representation of the robots and C_{kmax} is the maximum energy restriction. In set N some waypoints have a price p_i . The TOP method uses Boolean variables x_{ik} True (1) and False (0) to know if a waypoint x_i has been visited or not. A robot's route is represented by the variable T_k , which is also a set of waypoints. A route taken by a robot k can be represented by an edge x_{ij} that was a $X[x_{ijk}]$ matrix of Boolean variables.

Thus, the mathematical formulation can be defined as [1],[8] and [11]:

$$min(EP-EC) \tag{1}$$

$$EP=\alpha\sum_{i=1}^{N_c}\sum_{j=1,i\neq j}^{N_c}\sum_{k=1}^{N_k}c_{ij}x_{ijk} \tag{2}$$

$$EC=\beta\sum_{i=1}^{N_c}\sum_{k=1}^{N_k}p_i x_{ik} \tag{3}$$

$$\sum_{i=1}^{N_c-1}x_{\mathrm{ni}}=0 \tag{4}$$

$$\sum_{i=1}^{N_c}x_{ij}\leq 1 \qquad j=2,3,\ldots,N_c \tag{5}$$

$$\sum_{j=1}^{N_c}x_{ij}\leq 1 \qquad i=2,3,\ldots,N_c \tag{6}$$

$$\sum_{i=1}^{N_c}\sum_{j=1}^{N_c}\sum_{k=1}^{N_k}c_{ij}x_{ijk}\leq C_{k_{max}} \tag{7}$$

$$1\leq u_{ik}\leq N_c \qquad \forall\, i \square T_k \tag{8}$$

$$i_i-u_j+1\leq(1+-x_{ijk})N_c \qquad 2\leq i\neq j\leq N_c \tag{9}$$

The objective function of equation (1) has two purposes; one is to minimize the cost of agents and to maximize premium collections. This work uses an adapted version of the objective function presented in [11]. Equation (2) presents the cost of locomotion of robots using an alpha quantity defined in [10], and equation (3) presents the collection of prizes with the Beta weight also presented in [10]. Equation (4) presents a constraint, where it prevents vertices from being visited after reaching the final objective, to help this, equations (5) and (6) limit the vertices to be visited. As a consequence of the agent cost restriction given by equation (7), equations (8) and (9) will not be used representing the generation of sub-routes. The optimization will have as its final objective an environment with vertices in an orderly way to facilitate the visit of agents.

B. Novel Genetic Algorithms

Genetic Algorithms (AG) are methods of optimization and search, which were inspired by the mechanisms of evolution of populations of living beings. They follow the principle of natural selection and survival of the fittest [14] and [15].

The AG representativeness in this article is based on the work of [1], where chromosomes are composed of genes that portray a robot path. The waypoint visit indexes are stored in the GA genes. During the AG implementation, a permutation occurs with a set of maximum costs represented by C, where N_k is the robot number, each robot represented by k and a maximum cost C_{kmax} using as an evaluation. Each robot has its cost individually represented by the battery discharge time. The operation of the proposed GA can be visualized in Fig. 1.

```
Algorithm 1 Genetic Algorithms
1: start population;
2: while Stop Test do
3:     while population is not complete do
4:         crossover
5:         mutation
6:         evaluation
7:         selection
8:     update
9: Finishing
```

Fig. 1 GA Algorithm

The steps of GA during chromosomal generation take place from the initialization of a random population, crossing and by the last mutation.

```
Algorithm 2 Random nearest neighbor
IN: CostList, ListDepositBegin, ListDepositEnd,
NumberRobots, AvailableWaypoints
OUT: Chromosome
1: for each R ∈ NumberRobots do
2:     Gene ← New
3:     Begin ← ListDepositBegin(R)
4:     End ← ListDepositEnd(R)
5:     W ← SelectRandom(AvailableWaypoints)
6:     TmpG ← Join(Begin, Gene, End)
7:     GeneCost ← MeasureCost(TmpGene)
8:     N ← AvailableWaypoints
9:     C ← CostList(R)
10:    while GeneCost < C e Size(N) > 0 do
11:        Remove(W, AvailableWaypoints)
12:        Insert(W,Gene)
13:        TmpG ← Join(Begin, Gene, End)
14:        W ← NearestNeighbor(W)
15:        GeneCost ← MeasureCost(TmpG)
16:    Gene ← TmpG
17:    Insert(Gene, Chromosome)
18: Finishing
```

Fig. 2 Random Nearest Neighbor Algorithm

The algorithm of the nearest neighbour is used during the population generation stage. When choosing the waypoint, the algorithm of the nearest neighbour is used for the selection. The process can be seen in Fig 2.

The stopping criterion is the maximum amount of chromosome creation; with that, the fitness is computed, the evaluation method can be seen in Fig. 3.

```
Algorithm 3 Fitness
IN: Chromosome, CostsMap, Prizes, WeightCost,
WeightPrizes
OUT: Fitness
1: Fitness ← 0
2: for each G ∈ Chromosome do
3:     A ← Prizes(G)*WeightPrizes
4:     C ← CostsMap(G)*WeightCost
5:     FitnessGene ← A - C
6:     Fitness ← Fitness + FitnessGene
7: Finishing
```

Fig. 3 Fitness Algorithm

The fitness of the AG using in the present work was used in [10], where the weight for the prize is given by the sum of the genes is the fitness calculation. After executing the first step, which was the population initialization at random, the next step is the generation process. The end of the step that follows the generation process occurs when a maximum quantity G_{size} is reached as a stopping criterion or when the best $BestC_{kmax}$ chromosome is found according to the conditions of the criterion.

```
Algorithm 4 Crossover
IN: Parent1, Parent2, ListDepositBegin, ListDe-
positEnd
OUT: Offspring1, Offspring2
1: idxGene ← SelectRandom(Pai1)
2: Start ← ListDepositBegin(indexGene)
3: End ← ListDepositBegin(indexGene)
4: P1 ← Parent1
5: P2 ← Parent2
6: Gene1, Gene2 ← P1(idxGene), P2(idxGene)
7: Gene1, Gene2 ← RemoveDeposits(Gene1,
   Gene2)
8: Slice1, Slice2 ← Cut(Gene1), Cut(Gene2)
9: pos1, pos2 ← SelectRandom(Gene1,Gene2)
10: Gene1 ← Remove(Gene1, Slice1)
11: Gene2 ← Remove(Gene2, Slice2)
12: Gene1 ← Join(Gene1, Slice2, pos1)
13: Gene2 ← Join(Gene2, Slice1, pos2)
14: Gene1, Gene2 ← InsertDeposits(Gene1,
    Gene2, Start, End, indexGene)
15: Offspring1, Offspring2 ← New
16: for each g ∈ Indexes(P1) do
17:     if g = idxGene then
18:         Insert(Gege1, Offspring1)
19:         Insert(Gege2, Offspring2)
20:     else
21:         Insert(SelectGene(P1, g), Offspring1)
22:         Insert(SelectGene(P2, g), Offspring2)
23: RemoveRepeatedElements(Offspring1)
24: RemoveRepeatedElements(Offspring2)
25: Finishing
```

Fig. 4 Crossover Algorithm

At the beginning of the generations, there is a process of selecting individuals with a population rate; there are several types of selection. However, the one chosen for this application was the Tournament. During this selection, n individuals are pre-selected at random, and the most suitable are chosen.

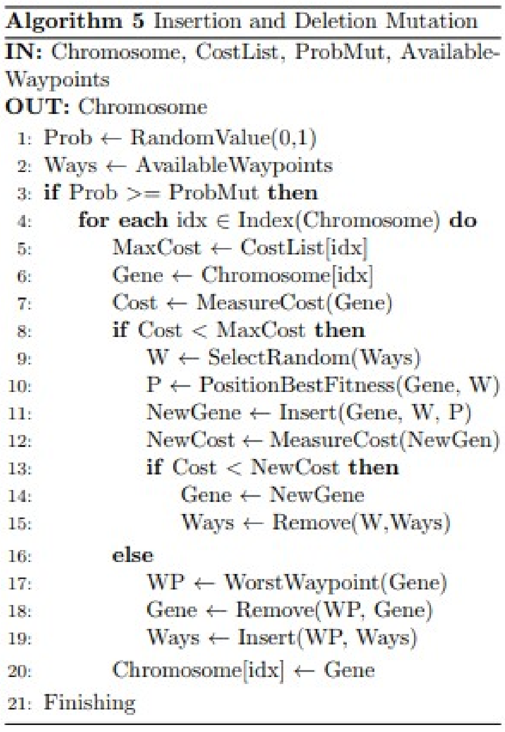
During the crossing operation, it has the objective of selecting two chromosome vectors to perform the operation and generate two new child chromosomes. Still during the crossing process for the interest of the work, in the end, only one gene is selected since each gene on the chromosome can represent a path of the robot. An operation called cutting is performed to choose the gene that will represent the robot path. The cutting operation aims to remove a part of the selected gene.

The slice operation can be different for each gene, where the slices can be of different sizes, this is justified because the gene is the representation of a route for each robot, which may have the number of different points. The final step is to exchange the slices of the selected genes with each other.

There is still the mutation process that is carried out after the crossing. There is a variable called the probability of mutation $ProbMut$ that serves as an evaluation basis for selecting individuals.

In the chromosome vector, the mutation has 3 procedures which are gene removal, gene insertion and exchange called (SWGLM) [16].

The insertion operation will be done on a single chromosome gene, selecting a waypoint that is not present on the chromosome. Then a permutation is performed at each position of the chosen gene until it finds the best position



```
Algorithm 5 Insertion and Deletion Mutation
IN: Chromosome, CostList, ProbMut, Available-
Waypoints
OUT: Chromosome
1:  Prob ← RandomValue(0,1)
2:  Ways ← AvailableWaypoints
3:  if Prob >= ProbMut then
4:      for each idx ∈ Index(Chromosome) do
5:          MaxCost ← CostList[idx]
6:          Gene ← Chromosome[idx]
7:          Cost ← MeasureCost(Gene)
8:          if Cost < MaxCost then
9:              W ← SelectRandom(Ways)
10:             P ← PositionBestFitness(Gene, W)
11:             NewGene ← Insert(Gene, W, P)
12:             NewCost ← MeasureCost(NewGen)
13:             if Cost < NewCost then
14:                 Gene ← NewGene
15:                 Ways ← Remove(W,Ways)
16:         else
17:             WP ← WorstWaypoint(Gene)
18:             Gene ← Remove(WP, Gene)
19:             Ways ← Insert(WP, Ways)
20:         Chromosome[idx] ← Gene
21: Finishing
```

Fig. 5 Crossover Algorithm

As for the mutation using the removal of a waypoint represented by a gene, the waypoint is removed, which is to obtain less fitness compared to the others. The exchange mutation is called SWGLM according to [16], to find the least suitable waypoint □ and obtain two exchanges between a neighbourhood, a neighbour located on his right and another on his left will be exchanged.

The insertion and deletion mutation process can be viewed in Fig. 5.

C. Test Environment

This section demonstrates the testing environment. In this testing stage, 20 symmetrical scenarios were proposed, with the numbers of robots and battery discharge times for each robot all different. Table 1 provides an example of the test environment. The distance between the points was used the Euclidean distance for the cost calculation. During tests, object measurements are provided using a time measurement in minutes, as well as the battery level until discharge, considering the robots speed will are constant.

Table 1 is detailed on how instance 1 is organized, in which the complete map can be viewed in Fig. 6. The first scenario has instance I1, where it has 40 waypoints and 5 deposits. The deposits serve as inputs and outputs for robots and have no punctuation. Still in that instance, they still have 218 points in prizes, as a rule, the points alternate from 2 to 9.

The results section is separated into 2 subsections. The first subsection is used to adjust GA parameters to determine the best parameterization in solving the proposed problem. The second section will return the results of the tests performed using the GA already parameterized.

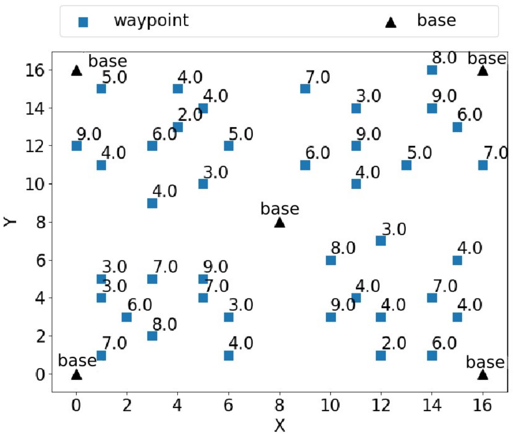


Fig. 6 Instance I1

The tests will be performed in 5 experiments:

- **Traditional TOP**: Normal condition for multi-robot route calculation modelled using the traditional TOP, where the locomotion restriction value will be the lowest battery level among robots;
- **Traditional TOP Robot Reduction**: Normal condition for multi-robot route calculation modelled using the traditional TOP, but excluding the robot with the lowest battery level, to increase TOP's travel range;
- **Modified TOP**: Normal condition for multi-robot route calculation modelled using the modified TOP, in which all robots will maintain their battery level for route calculation;

- **Modified TOP Robot Reduction**: Normal condition for multi-robot route calculation modelled using the modified TOP, but excluding the robot with the lowest battery level;
- **Modified Top Deposit Change**. Normal condition for multi-robot route calculation modelled using the modified TOP, but with the change of deposits.

TABLE I
INSTANCES

Inst.	Rob.	Wayp.	Prizes	Battery Time Robots(min)
I1	4	40	218	20, 23, 25, 30
I2	4	45	222	23, 25, 27, 30
I3	4	50	241	23, 25, 27, 30
I4	4	60	313	20, 23, 25, 30
I5	4	70	387	20, 23, 25, 30
I6	4	80	436	20, 23, 25, 30
I7	5	40	206	18, 22, 24, 25, 28
I8	5	50	236	19, 23, 25, 26, 29
I9	5	60	309	20, 22, 25, 27, 30
I10	5	70	383	21, 23, 26, 27, 30
I11	5	80	441	22, 24, 26, 28, 32
I12	6	50	239	18, 20, 22, 24, 25, 27
I13	6	60	310	19, 22, 23, 24, 25, 28
I14	6	70	379	20, 22, 24, 25, 27, 30
I15	6	80	440	21, 23, 24, 26, 29, 30
I16	7	60	310	15, 20, 23, 25. 25, 27, 30
I17	7	70	377	19, 20, 21, 23, 24 ,25, 29
I18	7	80	443	19, 21, 23, 24, 25 ,26, 30
I19	8	70	388	15, 20, 23, 25, 25, 27, 30, 30
I20	8	80	428	15, 20, 23, 25, 25, 27, 30, 30

The simulations were performed in the Python 3.6 language, using Numpy 1.14.5. The computer used had an Ubuntu 16.04 x64 system, 15-7600 processor and 8GB of RAM.

III. RESULT AND DISCUSSION

In this section, GA parameterization and test results will be presented, as well as the discussion of the results.

The proposed solution and the methods it consists of are presented in detail. Subsection III-A presents the results obtained from the experiments performed based on the environment presented in section II-C , 30 runs were performed to measure the performance and accuracy of the proposed method. In III-B, the test results will be presented. Subsection III-C will discuss the results

A. Parameterization of the proposed GA

The parameterization of a GA is one of the most important steps for its perfect execution, in which the use of parameters in the maximum values eventually generate premature convergence. For the parameterization of the proposed GA, a random parameterization was used, which will generate initial parameter values, and then it will be combined with an exhaustive search, to search for the data that generate the least premature convergence.

First, a random parameter set was generated and then tests were performed by changing each parameter from a minimum value to a maximum value. The parameterization was tested using a map with 4 robots and 40 points to be visited. Each parameter has been executed 10 times. Table II presents the proposed data, along with the result achieved. The ideal parameterization was chosen based on the results obtained where the parameters that obtained the values with the lowest standard deviation were chosen and finally an execution was performed between the random parameter and the ideal parameter. The convergence curve with the data specified as an ideal can be seen in Fig. 7.

TABLE II
PARAMETERIZATION OF GENETIC ALGORITHM

Parameters	Random	Range	Ideal
Population Size	30	50,…, 500	100
Prob. Crossover	.2	.1, …, .9	.6
Prob. Mutation	.4	.1, …, .9	.8
Gen. Maximum	35	5, …, 50	25

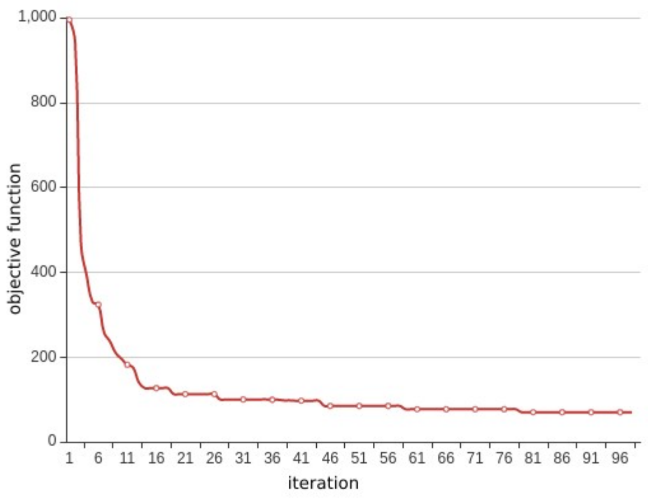


Fig. 7 Instance I1

B. Tests Performed and Discussion

The tests were performed based on the instances and experiments proposed in the section III-C.

The first proposed algorithm was based on [1], where it aims to calculate the best path for the robot, which proposed a genetic algorithm for traditional TOP resolution with a single value for locomotion cost, and a Genetic algorithm proposed by this work that combines the traditional TOP with MKP. Each instance was combined with all experiments and performed a total of 30 runs. Table III presents the results obtained, the calculated route performance is given by the total points collected by all robots involved, in which the displayed data show the average prize collection of the execution of each experiment.

All tests were run with a single source and target deposit, the robots should leave one deposit and return to the same deposit, except for experiment 5 which was performed with one source deposit and different target deposits, where each robot goes to different deposits.

For illustration purposes, the routes generated by running the instance I1 will be displayed for all experiments in Fig. 6.

Fig. 8 has 5 images, which shows the routes with the best scores obtained by running each experiment on the instance I1.

The performance of the solution presented in this paper is compared in 3 different ways with the model presented by [1]. Table III, specifically in columns 1 and 3, shows the comparison of the generated path of the models.

The model proposed in this paper obtained a better score; the improvement is quite significant considering the use of the mobility ability of each robot. It can be noted that the score obtained with the proposed model in certain scenarios was 31.1% better compared to the traditional one. In instance I20, a difference of 91.7% between the two models is justified by the low battery level of robot 1. In real scenarios, these robots can be removed from the route calculation to accomplish a mission. In this case, robot 1 will be removed from the route calculation. Thus, in the results of experiment 2, and making a new comparison with experiment 3, the score difference drops to 34.5%.

Other comparisons consider an environment with a decrease in the number of robots. Columns 2 and 4 of Table III show the results considering the second proposed model and the traditional one. The latter obtained an improvement in the collection of prizes. In some cases, the collection of prizes is close to its first execution comparing experiment 1 to experiment 2 due to the increase in travel distance of the remaining robots. However, the solution proposed in this research still presented a better performance, where in certain scenarios managed to overcome the traditional model by up to 30.1% in premium collection.

The proposal of the third comparison, where the target deposits are modified, forming different situations. Results of this execution are compared to results obtained in this research, performing the route calculation a single depot.

The results show that in some cases, depending on the mission map, their score does not show great divergences as can be observed in experiment 3 compared to experiment 5 of instance 1, but it can be noted that in other maps the gain reaches be 13% higher as seen in I2. These results demonstrate how important it is to make maximum use of the mobility of each robot independently.

TABLE III
RESULTS

Inst.	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**	**Exp. 5**
I1	140.86	139.36	180.23	153.13	207.5
I2	170.16	150.2	205.36	175	208
I3	176.3	164.5	207.5	177	223.4
I4	223.4	196.04	254.8	210	265.36
I5	261.1	228.6	299.3	262.53	329.5
I6	276.1	253.6	324.7	273.4	347.7
I7	120.5	157.5	193.6	186.1	199.3
I8	145.7	162.8	202.4	191.9	224.4
I9	202.9	197	265.3	246.4	283.7
I10	242.4	230.2	306	284.6	334.7
I11	266.4	253.2	340.8	312.4	379.2
I12	155.7	170.9	210.8	197.1	228.1
I13	204.6	225.2	270.9	255.6	294.2
I14	257.1	255.8	337.4	313.5	360
I15	288.5	282	365.8	344	396.6
I16	207.2	206	274.7	267	294.6
I17	270.8	258	336.4	318	361.1
I18	272.8	276	388.4	359	401.1
I19	188.2	227	329.2	317	363.7
I20	193.9	275	370.5	358	402.1

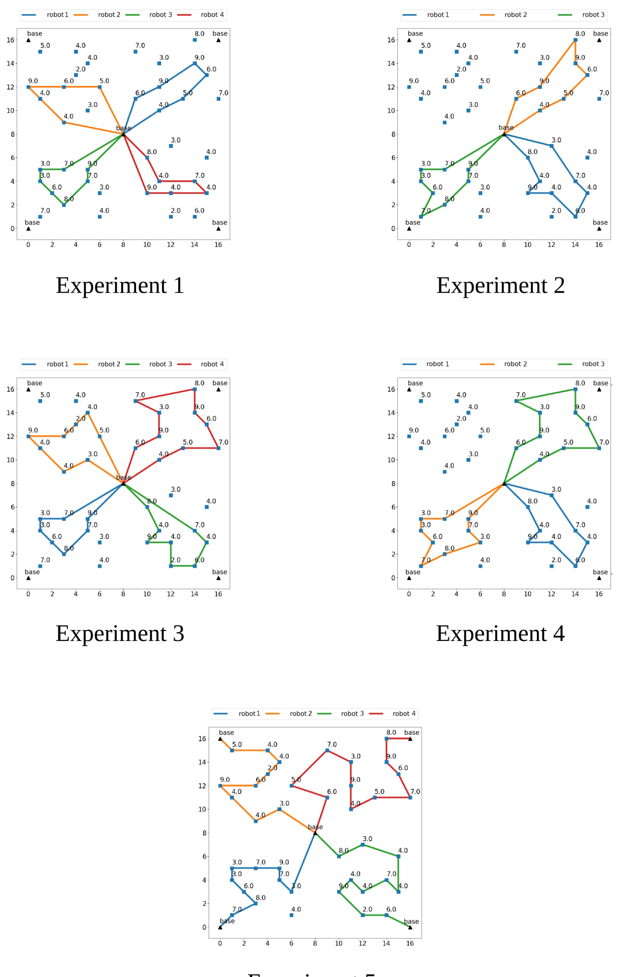


Fig. 8 Experiments Instance I1

IV. CONCLUSION

This paper studies route planning for multiple energy-constrained collaborative robots. Such a problem can be modelled using the team orienteering problem combined with the multiple knapsack problem. This paper presents a heuristic for solving the modified TOP that is based on a genetic algorithm. The GA is modelled based on the solution proposed by Bederina and then added the restriction of locomotion as energy restriction and the ability to change deposits for departure and arrival. The proposed heuristic is compared to the Bederin solution. The performance of the methods is measured by the total points collected using the same constraints. The proposal of this work was superior in all experiments performed, reaching in some cases to collect 30% more points than the solution of [1], this performance can only be possible due to the full use of the restrictions of all robots, characteristic absent in bederina. Depot change proved to be effective compared to using a single depot for all robots, where depot change point collection was shown to collect up to 13% of points compared to collecting this work using a single depot. Due to the symmetrical nature of the problem, this research is suitable for application in non-volatile environments, such as industrial environments that use AGV.

For future work, it would be necessary to test the performance of the solution on asymmetric maps to extend its use. The ability to handle multiple deposits proposed by this research makes it ideal for testing in conjunction with battery failure diagnostics systems as the cost of locomotion can be reduced rapidly, requiring route recalculation to get the most out of the mission. Finally, it is worth extending its future use to maps with task execution times and time window for completion.

Acknowledgement

The authors acknowledge the support of FUNCAP (BP3-0139-00241.01.00/18) BPI 03/2018.

References

[1] H. Bederina and M. Hifi, *A hybrid multi-objective evolutionary algorithm for the team orienteering problem,* 2017 4th International Conference on Control, Decision and Information Technologies (CoDIT), pp. 0898--0903, 2017.

[2] J. Liang and C. Lee, *Efficient collision-free path-planning of multiple mobile robots system using efficient artificial bee colony algorithm*, Advances in Engineering Software, vol. 79, pp. 47--56, 2015.

[3] A. Asma and B. Sadok, *Dynamic Distributed PSO joints elites in Multiple Robot Path Planning Systems: theoretical and practical review of new ideas,* Procedia Computer Science, vol. 112, pp. 1082--1091, 2017.

[4] A. Chen, J. Harwell and M. Gini, *Maximizing Energy Battery Efficiency in Swarm Robotics*. arXiv preprint arXiv:1906.01957, 2019.

[5] M. Tomy, B. Lacerda, N. Hawes and J. Wyatt, *Battery Charge Scheduling in Long-Life Autonomous Mobile Robots,* 2019 European Conference on Mobile Robots (ECMR), pp. 1–6, 2019.

[6] J. Lu, L. Wei, V. Pour, Y. Mekonnen and A. Sarwat, *Modeling discharge characteristics for predicting battery remaining life*, Transportation Electrification Conference and Expo (ITEC), pp. 468–473, 2017.

[7] D. A. Souza, V. P. Pinto, L. B. P. Nascimento, J. L. O. Torres, J. P. P Gomes, J. J. M. Sá, R. N. C. Almeida, *Battery Discharge forecast applied in Unmanned Aerial Vehicle,* Elektrotechzny, vol. 1, Feb. 2016

[8] I. Chao, B. Golden and E. Wasil, *The Team Orienteering Problem*, European Journal of Operational Research, pp. 464–474, 1996.

[9] L . Ke, C. Archetti and Z. Feng, *Ants can solve the team orienteering problem*. Computers & Industrial Engineering, vol. 54, pp. 648--665, 2008.

[10] P. Vansteenwegen, W. Souffriau, G. V. Berghe and D. V. Oudheusden, *A guided local search metaheuristic for the team orienteering problem*. European Journal of Operational Research, vol. 196, pp. 118--127, 2009.

[11] A. Candido, *Sistema de gerenciamento do voo de quadrirotores tolerante a falhas*. ITA, 2015.

[12] J. Ferreira, A. Quintas, J. A. Oliveira, G. A. B. Pereira and L. Dias. , *Solving the Team Orienteering Problem: Developing a Solution Tool Using a Genetic Algorithm Approach*. Snášel V., Krömer P., Köppen M., Schaefer G. (eds) Soft Computing in Industrial Applications. Advances in Intelligent Systems and Computing, vol 223, 2014.

[13] H. Kellerer, U. Pferschy, D. Pisinger, *Multiple Knapsack Problems*, Knapsack Problems, Heidelberg , Berlin, 2004.

[14] B. Coppin, *Artificial Intelligence Illuminated*. Jones and Bartlett Publishers, 1st edn, 2004.

[15] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional, 1st edn, 1989.

[16] A. Hassanat, E. Alkafaween, N. Alnawaiseh, M. Abbadi, M. Alkasassbeh and M. Alhasanat, *Enhancing Genetic Algorithms using Multi Mutations: Experimental Results on the Travelling Salesman Problem*. International Journal of Computer Science and Information Security, vol. 16, pp. 785--801, 2016.

[17] M. Schilde, K. F. Doerner, R. F. Hartl, and G. Kiechle, *Metaheuristics for the bi-objective orienteering problem*, Swarm Intelligence, vol. 3, pp. 179–201, Sep. 2009.