Engineering Applications of Artificial Intelligence 78 (2019) 260–272

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

TERRA: A path planning algorithm for cooperative UGV–UAV exploration

Fernando Ropero [*], Pablo Muñoz, María D. R-Moreno

Computer Engineering Department, Universidad de Alcalá, Campus Universitario, Ctra. Madrid-Barcelona, Km. 33, 600, 28871 Alcalá de Henares, Spain

ARTICLE INFO

Keywords:
Exploration
Cooperation
Routing
Heterogeneous robots

ABSTRACT

In this paper, we consider the scenario of exploring a planetary surface with a system formed by an Unmanned Aerial Vehicle (UAV) and an Unmanned Ground Vehicle (UGV). The goal is to reach a set of target points minimizing the travelling distance. Some expected key problems in planetary explorations are the UGVs functionality constraints to reach some target points as a single robot system and the UAVs energy constraints to reach all the target points on its own. We present an approach based on the coordination of a hybrid UGV–UAV system, in which both robots work together for reaching all the target points. Our strategy proposes the UGV as a moving charging station to solve the UAV energy constraint problem, and the UAV as the robotic system in charge of reaching the target points to solve the UGV functionality constraints. To overcome this problem, we formulate a strategy merging combinatorial classic techniques and modern evolutionary approaches aiming to optimize the travelling distance. Our solution has been tested in several simulation runs with different target points distributions. The results demonstrate that our approach is able to generate a coordinated plan for optimizing the hybrid UGV–UAV system in the exploration scenario.

1. Introduction

Over the last decades, there has been a strong scientific concern in planetary exploration with a single robot system (Unmanned Ground Vehicle or UGV). Nevertheless, the exponential complexity of exploration missions along with the technological growth of alternative robotic systems, e.g., Unmanned Aerial Vehicle (UAV) or Autonomous Underwater Vehicle (AUV), have conducted many researchers to shift their focus to use multiple robot systems. This is the case of the Mars 2020 mission, in which NASA plans to launch the Mars Helicopter Scout along with the 2020 Mars Rover. The Scout could help pinpoint interesting targets for studying and planning the best driving route. This is a clear evidence of the rising of the exploration research in coordinated multiple robot systems either with homogeneous multiple robot systems (Simmons et al., 2000; McLain et al., 2001; Burgard et al., 2005) or heterogeneous multiple robot systems (Sujit et al., 2009; Aghaeeyan et al., 2015).

The benefits of deploying small UAVs in coordinated multiple robot systems have been demonstrated in several current applications such as persistent surveillance missions (Leahy et al., 2016), data collection (Ergezer and Leblebiciolu, 2014), reconnaissance (Tian et al., 2006) or cooperative exploration (Chandler et al., 2001). However, a drawback of small UAVs in such applications is usually the energy constraint. Also, UGVs have functionality constraints, e.g., they cannot reach (or hardly can reach them by compromising their safety) targets located in a mountain ridge or take pictures from a polar view (or other angles which require a high altitude). This constraints makes a hybrid UGV–UAV system a demanded robotic system to address such issues.

The problem we consider in this article is motivated by a scenario where there is a set of target points distributed over an exploration area. The hybrid system is formed by a UGV and a UAV. The target points have to be visited by the UAV due to the UGV functionality constraints. Also, the UAV requires UGV charging stops across the exploration area due to UAV energy constraints. We assume that the UGV has enough energy resources to accomplish the mission. We have not considered some physical constraints inherent to ground and aerial vehicles (such as terrain slope, wind speed or atmospheric density) for two reasons: first, some of them imply an explosion in the problem complexity and, second, dynamic constraints (for instance, wind speed) cannot be easily modelled neither predicted to be exploited in off-line planning (they are better suitable for on-line adaptation of the plan during execution). Fig. 1 shows an instance of the problem where there is a set of target points which have to be reached through the placement of charging stops. In order to tackle this problem, we propose a strategy that computes a UGV–UAV cooperative routing to explore all the targets.

The strategy starts computing Voronoi Tessellations (Watson, 1981) as an interpolation method to find the suitable set of charging stops around the exploration area. The autonomous charging method used is out of the scope of this article, but for instance, we can mention two processes using battery swapping systems (Swieringa et al., 2010; Suzuki et al., 2012). Voronoi Tessellations is a well-studied solution on several backgrounds such as network coverage (Cortes et al., 2004) or

[*] Corresponding author.
E-mail address: fernando.ropero@uah.es (F. Ropero).

https://doi.org/10.1016/j.engappai.2018.11.008
Received 2 November 2017; Received in revised form 13 September 2018; Accepted 19 November 2018
Available online 14 December 2018
0952-1976/© 2018 Elsevier Ltd. All rights reserved.

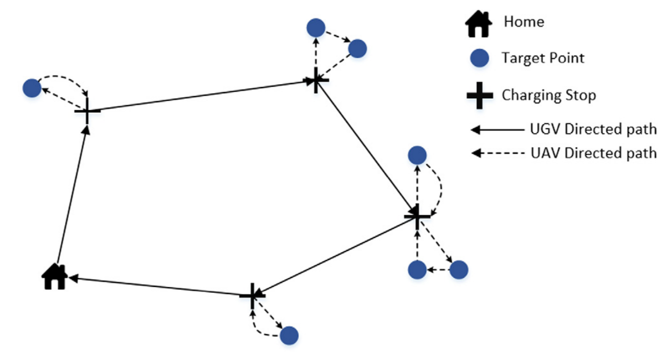


Fig. 1. A possible scenario and solution for the hybrid UGV–UAV system. Both systems start at the Home location. The black lines represent the UGV Directed Path. The dashed black lines are the UAV Directed Path.

path planning (McLain et al., 2001), where it is used to compute an initial path to the target. Once the set of charging stops is computed, we use the (Maini and Sujit, 2015) formulation to choose the optimal number of charging stops for reaching every target point. That is, the problem is modelled as a geometric Hitting Set Problem (Mustafa and Ray, 2010), and we solve it using the implementation developed by Gori et al. (2011). Then, an optimization algorithm is computed to reduce the UGV travelling distance among the charging stops. To the best of our knowledge, there is no previous work on this kind of optimization algorithms. The set of charging stops represents the Travelling Salesman Problem (TSP) for the UGV, and the set of target points represents multiple TSPs for the UAV. On the one hand, we have developed a genetic algorithm to solve the TSP for the UGV, inspired by the works of Larranaga et al. (1999) and Kirk (2007). On the other hand, we have developed a search algorithm to solve the TSPs for the UAV, which is based on the A* algorithm (Hart et al., 1968). Hence, our algorithm does not implement a multi-objective optimization algorithm, because multiple optimization goals are solved independently. Nevertheless, the sequential stages reveal a trade-off between the UGV and UAV travelling distance during the strategy, which is going to be the spotlight in the experimental evaluation.

The rest of the paper is organized as follows. Next section presents related works. Section 3 briefly formulates the problem of visiting a set of target points on cooperation, with the key definitions and nomenclature that are referred to throughout the paper. Section 4 describes the TERRA algorithm designed to solve the stated problem. Experimental results are presented and discussed in Section 5. Finally, conclusions and future research lines are outlined.

2. Related work

The cooperative multiple robot system has been extensively studied on different applications in the existing literature. For instance, the surveillance problem has been addressed by Tanner (2007), where a hybrid UGV–UAV system cooperates for the detection of a moving target. Also, it has been addressed by Ding et al. (2010), where a group of UAVs provide protection to convoy through coveraged and tracking techniques. Grocholsky et al. (2006) propose a team of UAVs to cover a target area identifying points of interest. Bellingham et al. (2002) study the problem of cooperative path planning and trajectory optimization for a team of UAVs. Richards et al. (2002) compare two methods based on the coordination and control of multiple UAVs. Nevertheless, our focus is on the exploration problem.

The exploration problem introduces difficulties when the area to cover is large enough that it requires the inclusion of charging stops into the problem. This is addressed in different ways in the literature. For instance, Mathew et al. (2015) formulate a cooperative strategy for a team of UAVs and a set of UGVs as mobile refuelling stations. This scenario differs from our problem because it consists of predefined routes and tasks for the UAVs, and the goal is to schedule multiple rendezvous with the mobile charging stations. Other solution is proposed by Funke et al. (2015), in which it is designed a method for the placement of charging stations around a road to avoid UGVs run out of fuel.

The problem considered in this work involves a hybrid UGV–UAV system in a planetary surface exploration, and it is referred as to find a path for both robotic systems optimizing the travelling distance. The problem of finding an optimal solution among a set of target points is the TSP (Dantzig et al., 1954), which is a well-known NP-Hard optimization combinatorial problem (Gutin and Punnen, 2006). Sundar and Rathinam (2014) formulate the Fuel Constrained, UAV Routing Problem (FCURP) which is a combinatorial generalization of the TSP based on finding an optimal path for the UAV among the target points, using intermediate depots for refuelling. Nevertheless, FCURP is only focused on the UAV routing through a set of static depots, so there is no cooperative routing as it does not consider neither the UGV path nor robot cooperation. Our problem uses intermediate depots for refuelling too, but considering the UGV path planning into the problem equation. A derived work is presented in Maini and Sujit (2015), who define the Fuel Constrained UAV Refuelling Problem with Mobile Refuelling Station (FCURP-MRS). In this case, the UAV routing will be through a mobile refuelling station instead of static depots. Also, the strategy proposed in this work introduces an initial combinatorial stage to select an optimal number of charging stops for the mobile refuelling station. However, the route of the mobile refuelling station is around a road network previously established, so, still there is no routing for the mobile refuelling station. Indeed, the strategy proposed by these works were our starting point of inspiration to guide the problem as a cooperative routing subject.

As we pointed out, our problem involves solving multiple TSP instances such as the work presented by Oberlin et al. (2009), which introduces the Heterogeneous, Multiple Depot, Multiple Travelling Salesman Problem (HMDMTSP). It is related to apply a generalized asymmetric TSP (ATSP) to teams of heterogeneous UAVs aiming to reach every target point only once. At this point, our problem formulates a symmetric TSP (STSP) for each robotic system. That is, if we have a distance d_{ij} between the city i and the city j, in a symmetric problem $d_{ij} = d_{ji}$ and in an asymmetric problem $d_{ij} \neq d_{ji}$. Also, Sundar and Rathinam (2016) formulate the Multiple Ground Vehicle Path Planning Problem (MGVPP) and the Multiple Around Vehicle Path Planning Problem (MAVPP) as mixed-integer linear programs (typical representation for such problems). The single ground vehicle variant of the MGVPP is a generalization of the STSP, meanwhile the single around vehicle variant of the MAVPP is the ATSP. Here, there are heterogeneous vehicles starting on its initial depot at different locations, and a set of target points are assigned to them based on its functionalities. The aim of each vehicle is to visit its target points ending in the initial depots. This work differs from our problem in that it does not consider a UGV refuelling station to do charging stops, so there is no cooperative routing. Manyam et al. (2016) define the cooperative air-ground vehicle routing problem (CAGVRP) in which a multiple UGV–UAV system is involved. The goal in CAGVRP is to visit a set of target points either by the UAV or the UGV, keeping alive the communication link between both vehicles. The initial state in CAGVRP is similar to our problem in which both UAV–UGV start at the same point. Also, the CAGVRP formulates the UGV as a transportation system for the UAV as our problem. However, in our problem, the UAV has to visit every target point and we do not consider communication problems. Besides, the CAGVRP does not introduce the charging stops variable into the problem equation.

3. Problem statement

Following the scenarios presented in the previous section, we propose a particular scenario where the Energy Constrained UAV and Charging Station UGV Routing Problem (ECU-CSURP) arises. It is formally defined as follows (see Fig. 2):

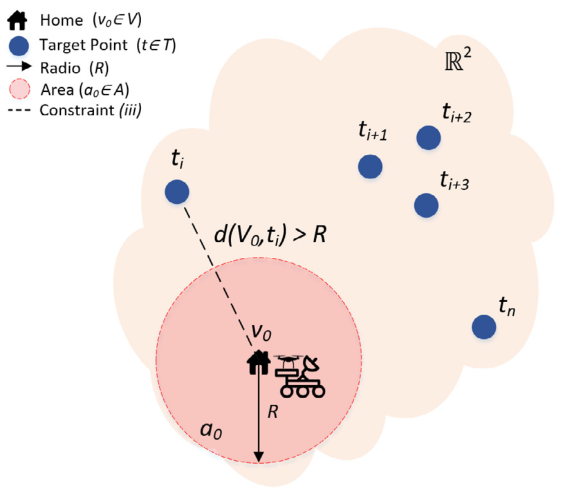


Fig. 2. An ECU-CSURP example where target points ($t_i, t_{i+1}, t_{i+2}, t_{i+3}, t_n \in T$) and a home location ($V_0 \in V$) are distributed around a $\mathbb{R}^2$ Euclidean space. R denotes the farthest distance the UAV can travel considering a return flight to the take-off point, and $a_0 \in A$ is the area initially covered by V_0. Note the distance constraint is satisfied with $d(V_0, t_i) > R$.

(i) An exploration area modelled as an $\mathbb{R}^2$ Euclidean space.
(ii) A set of target points which has to be visited.
(iii) A distance constraint: at least one target point is out of the boundaries of the initial UAV energy constraint.
(iv) A hybrid UGV–UAV system in which both have been modelled as Dubins vehicles (Dubins, 1957), which are vehicles with just a single constraint: forward movement at a constant speed.
(v) A UAV energy constraint.
(vi) A home location where both systems start the exploration.

The objective is to find a cooperative routing for the hybrid robot system to allow the UAV to visit every target point while trying to minimize the overall travelling distance. We model the $\mathbb{R}^2$ Euclidean space as an area where the distance travelled by both robotic systems is directly proportional to the time spent and the energy consumed in a trip. Therefore, the shorter the distance travelled, the shorter the time spent and the lower the energy consumed, and vice versa. The STSPs will focus on minimizing the UGV and UAV travelling distance.

The UAV energy constraint is modelled as the maximum distance that the UAV can travel with a fully charged battery. Let V_{uav} be the UAV constant velocity and t_{trip} the UAV flight time computed in a trip with a fully charged battery, where $d_{max} = (V_{uav} * t_{trip})$. Thus, the farthest distance the UAV can travel ensuring a return flight to the take off point is $R = d_{max}/2$ (see Fig. 2). Also, we assume the UGV does not have energy constraints, so it has enough energy resources to complete the exploration mission.

Let T denote the set of targets points $\{t_1, \dots, t_n\}$ where a target $t_i \in T$ for $i = 1, \dots, n$, and V denote the set of charging stops, called as vertices, $\{v_0, \dots, v_m\}$ where a vertex $v_j \in V$ for $j = 0, \dots, m$ (see Fig. 3). Let $v_0 \in V$ be the home location and let a UGV path be denoted as a tuple $\{v_0, \dots, v_{m-1}, v_m, v_0\}$. Then, let the distance constraint be represented as $\{t \in T : d(V_0, t) > R\} \neq \emptyset$. Let A denote the set of areas $\{a_0, \dots, a_m\}$ where an area $a_j \in A$ (taking R as the radius) represents the set of covered target points $\{t_1, \dots, t_n\}$ by $v_j \in V$ (see Fig. 3). Let f_{v_i,v_j} represent the travel cost as the distance travelled by the UGV to travel from $v_i \in V$ to $v_j \in V$ (see Fig. 3). Let S be the set of UAV sub-tours $S = \{s_1, \dots, s_k\}$. Let a UAV sub-tour $s_i \in S$ be denoted as a pattern $\{v_j, t_i, t_{i+1}, \dots, t_p, v_j\}$ where $v_j \in V$ and $t_i \in T$ for $i = 1, \dots, p$. Note that a UAV sub-tour $s_i \in S$ could have intermediate visits to the same $v_j \in V$, e.g. $\{v_j, t_i, v_j, t_{i+1}, v_j\}$, which means the UAV needs an intermediate charging stop to continue the exploration. Let g_{v_j,t_i} or g_{t_j,t_i} denote the travel cost as the distance travelled by the UAV to travel from the vertex $v_j \in V$ or the target $t_j \in T$ to $t_i \in T$ (see Fig. 3). Hence,

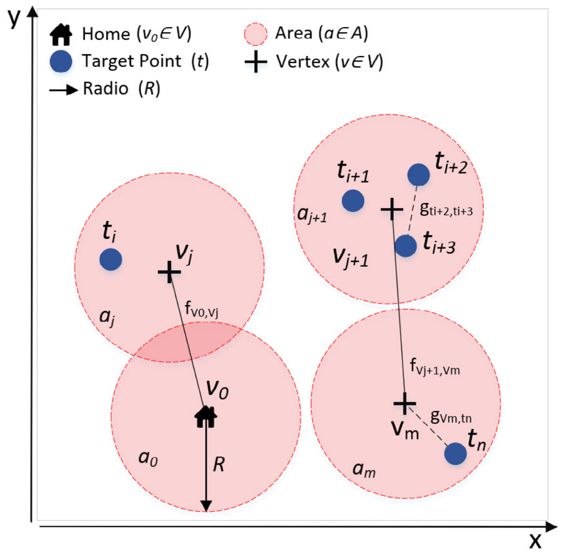


Fig. 3. A TERRA representation of the ECU-CSURP instance shown in Fig. 2. There are placed in a $\mathbb{R}^2$ Euclidean space (X,Y): a set of target points ($t_i, t_{i+1}, t_{i+2}, t_n \in T$), a possible set of vertices ($v_j, v_{j+1}, v_m \in V$), a set of surfaces ($a_0, a_j, a_{j+1}, a_m \in A$) for each $v \in V$ and a home location ($V_0 \in V$). The arrow is the farthest distance R the UAV can travel.

let F_{ugv} denote the UGV travelling distance and let F_{uav} denote the UAV travelling distance. Then, let F_{total} denote the total travelling distance of the hybrid robot system. The ECU-CSURP goals are summarized in the following *Objective Equations* (1)–(3):

$$F_{ugv} = \sum_{i=0}^{m-1} f_{v_i,v_{i+1}} + f_{v_m,v_0} \rightarrow min \tag{1}$$

$$F_{uav} = \sum_{j=0}^{m} \sum_{i=1} \left[g_{v_j,t_i} + \left(\sum_{i=1}^{p-1} g_{t_i,v_j} * x_i + g_{v_j,t_{i+1}} * x_i + g_{t_i,t_{i+1}} * \overline{x_i} \right) + g_{t_p,v_j} \right] \rightarrow min \tag{2}$$

$$F_{total} = F_{ugv} + F_{uav} \rightarrow min \tag{3}$$

where x_i is a binary variable which takes a value 1 if there is an intermediate charging stop from the target $t_i \in T$ to the vertex $v_j \in V$, and 0 otherwise. The *Objective Equation* (1) aims to minimize the UGV travelling distance to accomplish its path. The *Objective Equation* (2) aims to minimize the UAV travelling distance to accomplish every sub-tour. The *Objective Equation* (3) formulates the travelling distance of the hybrid UGV–UAV system.

4. The TERRA algorithm

The main contribution of this article is the cooperaTive ExploRation Routing Algorithm (TERRA). This algorithm has been designed to solve the ECU-CSURP and it has been split into five stages. Each stage attempts to solve a single issue of the ECU-CSURP:

1. Voronoi Tessellations are computed to find a new set of vertices V to minimize F_{ugv}.
2. The Hitting Set Problem is computed as a combinatorial optimization of the vertices in V to reduce its cardinality and minimize F_{ugv}.
3. The Gravitational Optimization Algorithm is implemented to reduce the $\mathbb{R}^2$ Euclidean distance among the vertices in V and minimize F_{ugv}.
4. The STSP is computed to the vertices in V to optimize F_{ugv}.
5. The STSP is computed for each UAV sub-tour in S to optimize F_{uav}.

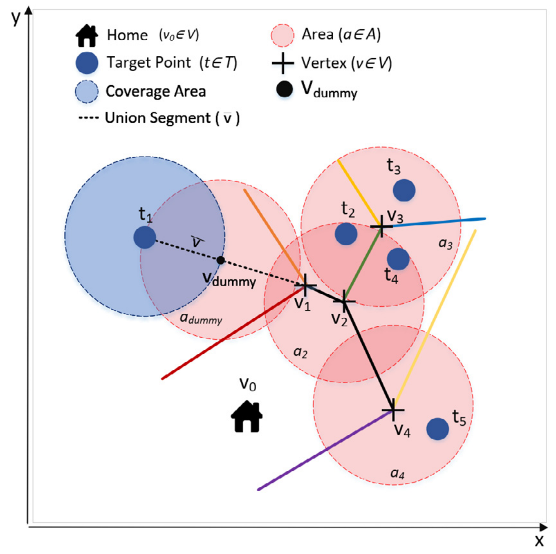


Fig. 4. First stage of TERRA for the ECU-CSURP instance shown in Fig. 2. Every target point t_i is covered by at least one vertex v_j. Note that the home location v_0 is considered as an input, but no area encompasses it because it does not cover any target point.

4.1. First stage

The first stage is similar to the work in Maini and Sujit (2015), placing intermediate UGV charging stops $v_j \in V$, so that $g_{v_0,t_i} = f_{v_0,v_j} + g_{v_j,t_i}$, allowing the UAV to reach t_i without running out of energy. However, our computational method is based on Voronoi Tessellations (see Fig. 4). The goal is to compute a set of vertices $v_j \in V$, whose areas $a_j \in A$ cover the whole set of target points.

Algorithm 1 shows the pseudo code implemented in TERRA. The **inputs** are the set of target points T and the farthest distance R the UAV can travel. The following functions help to understand it:

– $Voronoi$ (line 18): is the *voronoi function*[1] integrated in the MATLAB computational geometry package. It computes a Voronoi Tessellation with the Voronoi points ($V_{vpoints}$) given as input. The output is a set of vertices which represent the called Voronoi edges (V_{edges}) (v_1, v_2, v_3 and v_4 in Fig. 4).
– $NearestVertex$ (line 30): it computes the distance between each $t \in T_{nc}$ and every $v \in V_{edges}$. Then, it gives as output the nearest v of each t.
– $UnionSegment$ (line 15): it computes the union segment ($\bar{v}$ in Fig. 4) formed by the two vertices in $V_{vpoints}$ (only when it complies the equality in line 14). A vertex corresponds to the latest remaining target point in T_{nc} and the other to its nearest vertex in V_{nears} (union in line 32).
– $JunctionPoint$ (line 16): it computes the junction point between the Linear Equation traced by the segment ($\bar{v}$ in Fig. 4) and the Equation of the Circle of the latest target point (t_1 in Fig. 4) following the Pythagorean Theorem. Thus, it solves the following system of equations:

$$\left.\begin{aligned} \frac{X - x_1}{x_2 - x_1} &= \frac{Y - y_1}{y_2 - y_1} \\ (X - x_1)^2 - (Y - y_1)^2 &= R^2 \end{aligned}\right\} \tag{4}$$

where (x_1, y_1) and (x_2, y_2) are the Cartesian Coordinates of the latest remaining target point in T_{nc} and its nearest vertex in V_{nears} (t_1 and v_1 in Fig. 4). It returns the junction point (X, Y) as a false Voronoi edge (V_{dummy} in Fig. 4) required to cover the target point.

[1] https://es.mathworks.com/help/matlab/ref/voronoi.html.

Algorithm 1 Voronoi Diagram Algorithm

Input: T, R
Output: V, A

```
1  // $T_{nc}$: set of target points not covered yet
2  // $V_{nears}$: set of nearest vertices for every $t \in T_{nc}$
3  // $V_{edges}$: set of Voronoi edges
4  // $V_{vpoints}$: set of Voronoi points
5  // $v_{dummy}$: false Voronoi edge to cover an isolated target point
6  // $s_e$: union segment between a target point and a vertex
7  $T_{nc} \leftarrow T$
8  $V \leftarrow \emptyset$
9  $V_{vpoints} \leftarrow T_{nc}$
10 $V_{edges} \leftarrow \emptyset$
11 $V_{nears} \leftarrow \emptyset$
12 while $T_{nc} \neq \emptyset$ do
13     $v_{dummy} \leftarrow \emptyset$
14     if $|V_{vpoints}| == 2$ then
15         $s_e \leftarrow UnionSegment(V_{vpoints})$
16         $v_{dummy} \leftarrow JunctionPoint(s_e, R, T_{nc})$
17     else
18         $V_{edges} \leftarrow Voronoi(V_{vpoints})$
19     end if
20     $V_{edges} \leftarrow V_{edges} \bigcup v_{dummy}$
21     for $t \in T_{nc}$ do
22         for $v \in V_{edges}$ do
23             if $distance(t, v) < R$ then
24                 $T_{nc} \leftarrow T_{nc} \setminus t$
25                 $V \leftarrow V \bigcup v$
26             end if
27         end for
28     end for
29     for $t \in T_{nc}$ do
30         $V_{nears} \leftarrow V_{nears} \bigcup NearestVertex(t, V_{edges})$
31     end for
32     $V_{vpoints} = T_{nc} \bigcup V_{nears}$
33 end while
34 $A \leftarrow ComputeAreas(V, R)$
35 return $V, A$
```

– $ComputeAreas$ (line 34): it computes the set of areas $\{a_0, \ldots, a_m\} \in A$ where $a_j = \{t_1, \ldots, t_n\}$.

The algorithm performs iterative Voronoi Tessellations covering the target points in T_{nc}. Sometimes, the algorithm detects that there are not enough $V_{vpoints}$ (condition in line 14) to compute a Voronoi Tessellation, i.e., there is only one isolated target point remaining to be covered and its nearest vertex. Then, the algorithm generates a false Voronoi edge which guarantees the coverage of the isolated target point (union in line 20).

The **outputs** are a set of vertices V (Eq. (5)) and their areas A containing the set of target points (Eq. (6)). At this point, every target point in T has been covered by A.

$$VT(T, R) \rightarrow \{v_0, \ldots, v_m\} \in V,\ \{a_0, \ldots, a_m\} \in A \tag{5}$$

$$a_j \in A \leftarrow \{t_1, \ldots, t_n\} \tag{6}$$

4.2. Second stage

The second stage aims to find a minimum set of vertices whose areas guarantee full covering of the set of target points (see Fig. 5). In computer science this is called the Hitting Set Problem. It is modelled as a bipartite graph where the vertices $v_j \in V$ are represented on the left side, the universe is represented by the target points $t_i \in T$ on the right side, and the edges $a_j \in A$ representing the inclusion of elements

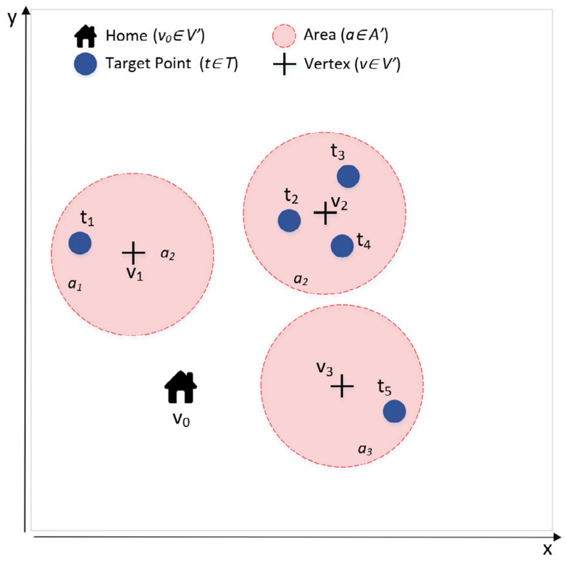


Fig. 5. Second stage of TERRA for the ECU-CSURP instance shown in Fig. 2.

in sets. The task is to find a minimum cardinality subset of $v_j \in V$ whose $a_j \in A$ cover every $t_i \in T$.

We tailored the implementation developed by Gori et al. (2011). The **inputs** are the vertices in V and the areas in A including the set of target points T. Fig. 4 shows how TERRA realizes that v_2 (whose area a_2 covers $\{t_2, t_4\}$) can be deleted from the set of vertices V because v_3 covers $\{t_2, t_3, t_4\}$, and thus, the cardinality is optimized, as Fig. 5 shows.

The **outputs** are a minimum set of vertices V' (Eq. (7)) and their respective areas A' (Eq. (8)).

$$HSP(V, A) \rightarrow \{v_0, \dots, v_m\} \in V', \ \{a_0, \dots, a_m\} \in A' \tag{7}$$

$$a_j \in A' \leftarrow \{t_1, \dots, t_n\} \tag{8}$$

4.3. Third stage

The third stage aims to minimize F_{ugv} through the F_{uav} increment due to the follow assumption: the UGV motion speed is expected to be much slower than the UAV motion speed in planetary explorations, i.e., $V_{ugv} \ll V_{uav}$. Consequently, F_{ugv} minimization will have higher impact than F_{uav} from a time perspective. Given the areas $a_j \in A'$, the farthest distance R the UAV can travel, the vertices $v_j \in V'$ and a gravity point G_p, the objective is to find a set of vertices $p_j \in P$ where $d(p_j, G_p) < d(v_j, G_p) \forall\ p_j \in P$ and the rule $g_{p_j,t_i} \leq R \forall\ t_i \in a_j \in A', p_j \in P$ is satisfied.

We developed the Gravitational Optimization Algorithm which places a gravity point G_p in the exploration area, and then, attracts the vertices $v_j \in V'$ to it by creating the new vertices $p_j \in P$. The **inputs** are the farthest distance R the UAV can travel, the gravity point G_p, and the set of vertices V' and areas A' solved in the *Second Stage*. On each execution, TERRA computes one solution without applying a gravity point and three gravity point solutions ($G_p = \{XC, HC, MC\}$), then, it chooses the best from all four. The three gravity points are defined as follows:

1. Mean centre (XC). It is the mean of the x-axis and y-axis of the target points.
2. Home location (HC).
3. Median centre (MC). It is the median of the x-axis and y-axis of the target points.

Algorithm 2 shows the pseudo code implemented in TERRA. The following functions summarize the algorithm behaviour:

Algorithm 2 Gravitational Optimization Algorithm

```
Input: $A'$, $R$, $V'$, $G_p$
Output: $V''$
1  // $a_v$: set of target points covered by $v$
2  // $s_e$: segment traced from $v$ to $G_p$
3  // $P$: set of candidate junction points
4  $V'' \leftarrow \emptyset$
5  for $v \in V'$ do
6      $a_v \leftarrow GetArea(A', v)$
7      $s_e \leftarrow UnionSegment(v \bigcup G_p)$
8      for $t_c \in a_v$ do
9          $P \leftarrow P \bigcup JunctionPoint(s_e, R, t_c)$
10     end for
11     $V'' \leftarrow V'' \bigcup GetMin(P, G_p)$
12 end for
13 return $V''$
```

– $GetArea$ (line 6): it obtains the target points covered by v.
– $UnionSegment$ (line 7): it computes the union segment ($\bar{v}_j \in \bar{V}$ in Fig. 6a) between the vertex ($v_j \in V'$ in Fig. 6a) and the G_p considered.
– $JunctionPoint$ (line 9): it builds the system of equations defined in the *First Stage* (Eq. (4)), where (x_1, y_1) and (x_2, y_2) are the Cartesian Coordinates of the target point t_c (coverage area $c \in C$ in Fig. 6a) and its covering vertex v. It returns a candidate junction point ($p_j \in P$ in Fig. 6a) to replace the vertex $v_j \in V$.
– $GetMin$ (line 11): it computes the distance between G_p and every candidate junction point $p \in P$. Then, it gives as output the nearest candidate to replace the vertex ($v_j \in V''$ in Fig. 6b).

The algorithm computes finite iterations until it finds a candidate junction point to replace every vertex. The **output** is a set of replaced vertices V'' (Eq. (9)).

$$GOA(A', R, V', G_p) \rightarrow \{v_0, \dots, v_m\} \in V'' \tag{9}$$

4.4. Fourth stage

The fourth stage aims to compute the shortest UGV travelling distance in order to minimize F_{total}. In computer science this is called the Travelling Salesman Problem (TSP). Given a set of vertices $V'' \leftarrow \{v_0, \dots, v_m\}$ and the distances between each pair of vertices f_{v_i,v_j} $\forall\ (v_i, v_j \in V'')$, the task is to find the shortest possible route that visits each vertex and returns to the home location v_0. Particularly, we model a symmetric TSP (STSP), where the distance f_{v_i,v_j} between the vertex v_i and the vertex v_j follows $f_{v_i,v_j} = f_{v_j,v_i}$.

We developed a genetic algorithm taking as inspiration the work done by Kirk (2007). The **input** is the set of vertices V'' computed in *Third Stage*. Our algorithm is built under three main steps. The first step is a selection mechanism where a portion of the existing population is selected to breed a new generation. It applies the tournament selection method (Goldberg and Deb, 1991) to repeatedly select the best individual of a randomly chosen subset. The second step implements an elitist selection process in which the best individuals from the current population are carried over to the next, unaltered. The tournament and elitist selections methods ensure a generational process which keeps the population size constant on each generation. The third step produces the new generation from those selected in the tournament selection through the combination of the genetic operators: mutation and crossover. The mutation operator uses flipping, swapping and sliding techniques to create new chromosomes. The crossover operators applied are: Order Crossover, Cycle Crossover and Order Base Crossover. In order to properly adjust the genetic algorithm to ECU-CSURP instances, we performed a tuning experiment to select an appropriate parameter configuration

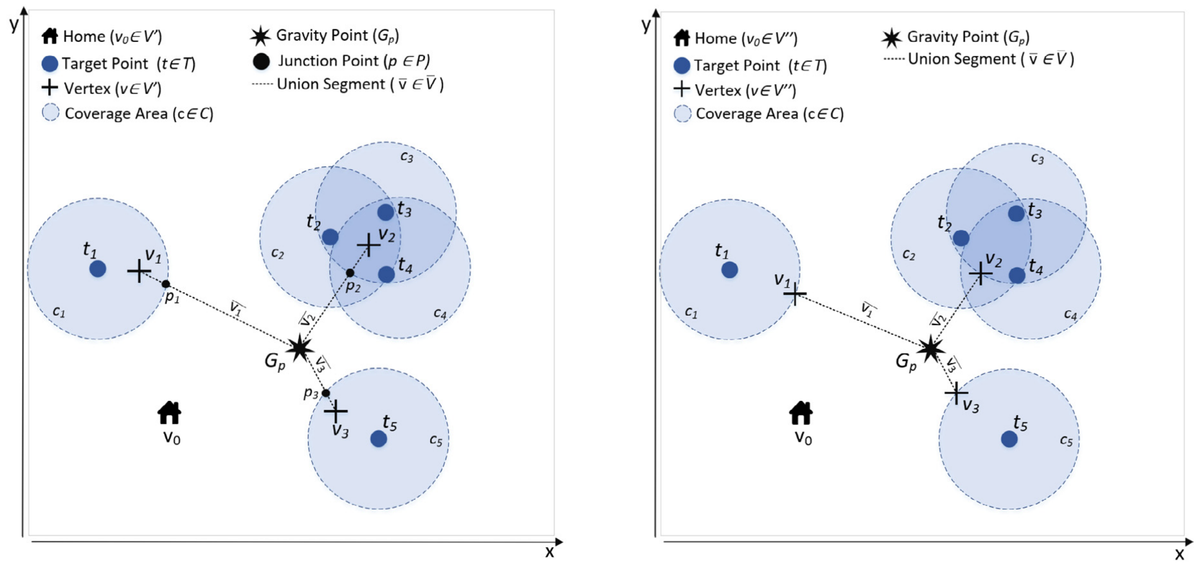


Fig. 6. Third stage of TERRA for the ECU-CSURP instance shown in Fig. 2. In this instance, the solution selected is the mean centre (MC). The coverage areas $c \in C$ represent the maximum distance to replace each vertex v_j with p_j so that, the following rule is satisfied: $g_{p_j,t_i} \leq R \,\forall\, t_i \in a_j \in A', p_j \in P$.

(see Section 5.2). The **output** is an ordered path (Eq. (10)) satisfying the *Objective Equation* (1) (see Fig. 7).

$$GA(V'') \rightarrow \{v_0, v_j, v_{j+1}, \dots, v_m, v_0\} \in V''' \tag{10}$$

4.5. Fifth stage

The fifth stage aims to compute the shortest UAV travelling distance among the multiple UAV sub-tours in order to minimize F_{total}. This problem envelopes multiple STSPs. Given a set of areas A', where each area $a_j \in A'$ is an unordered set of target points $\{t_1, \dots, t_n\}$, the farthest distance R the UAV can travel, a set of vertices V'', and the distances $g_{v_j,t_i}\ \forall\ v_j \in V'', t_i \in a_j \in A'$ and $g_{t_j,t_i}\ \forall\ t_j, t_i \in a_j \in A'$, the objective is to find the multiple shortest sub-tours $s_i \in S$ which visits each target point $t_i \in a_j \in A'$ and returns to the linked charging stop $v_j \in V''$.

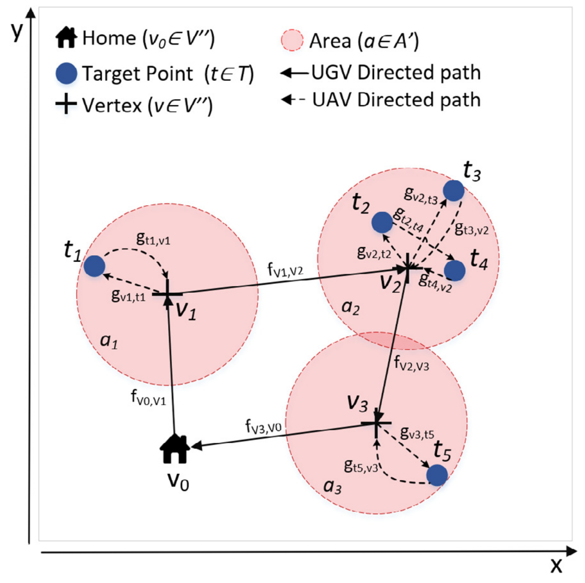


Fig. 7. *Fourth* and *Fifth Stage* of TERRA for the ECU-CSURP instance shown in Fig. 2. The *Fourth Stage* gives as output the UGV Directed Path, and the *Fifth Stage* gives the UAV Directed Path.

Algorithm 3 Optimal Operation Search Algorithm

Input: A',V'',R
Output: S

```
1  // $a_v$: set of target points covered by $v$
2  // $s$: a UAV sub-tour
3  // open: node list ordered by the evaluation function (f)
4  // closed: list of already visited nodes
5  // $p_{start}$: starting node or charging stop of each sub-tour
6  // $p_{curr}$: current node of the search graph
7  // $p_{neighbours}$: neighbour nodes list of $p_{curr}$
8  for $v \in V''$ do
9      $a_v \leftarrow GetArea(A', v)$
10     $a_v \leftarrow a_v \bigcup v$
11     $coord(p_{start}) \leftarrow v$
12     $d(p_{start}) \leftarrow 0$
13     $g(p_{start}) \leftarrow 0$
14     $h(p_{start}) \leftarrow |a_v| - 1$
15     $f(p_{start}) = g(p_{start}) + h(p_{start})$
16     $parent(p_{start}) \leftarrow p_{start}$
17     $closed \leftarrow \emptyset$
18     $open \leftarrow \emptyset$
19     $open.Push(p_{start})$
20     while $open \neq \emptyset$ do
21         $p_{curr} \leftarrow open.Pop()$
22         if $coord(p_{curr}) == v$ and $h(p_{curr}) == 0$ then
23             return $s \leftarrow GetSubTour(p_{curr})$
24         end if
25         $closed.Push(p_{curr})$
26         $p_{neighbours} \leftarrow ExpandGraph(p_{curr}, p_{start}, R, a_v)$
27         for $p_{nbr} \in p_{neighbours}$ do
28             if $p_{nbr} \notin closed$ then
29                 if $p_{nbr} \notin open$ then
30                     $g(p_{nbr}) \leftarrow Inf$
31                     $parent(p_{nbr}) \leftarrow \emptyset$
32                 end if
33                 $open \leftarrow UpdateNode(open, p_{curr}, p_{nbr})$
34             end if
35         end for
36     end while
37     return $S \leftarrow S \bigcup s$
38 end for
39 return $S$
```

Algorithm 4 ExpandGraph Function

```
Input: $p_{curr}, p_{start}, R, a_v$
Output: $p_{neighbours}$
1  // $p_{nbr}$: $p_{curr}$ neighbour node expanded in the graph
2  // $n$: candidate to neighbour node
3  $p_{neighbours} \leftarrow \emptyset$
4  for $n \in a_v$ do
5    if $coord(p_{curr}) \neq n$ and not $relative(p_{curr}, n)$ then
6      if $coord(p_{curr}) == coord(p_{start})$ then
7        $d \leftarrow distance(coord(p_{curr}), n)$
8      else
9        $d \leftarrow d(p_{curr}) + distance(coord(p_{curr}), n)$
10     end if
11     $d_{start} \leftarrow distance(coord(p_{start}), n)$
12     if $R * 2 >= d + d_{start}$ then
13       if $n == coord(p_{start})$ then
14         $h \leftarrow h(p_{curr})$
15       else
16         $h \leftarrow h(p_{curr}) - 1$
17       end if
18       $coord(p_{nbr}) \leftarrow n$
19       $parent(p_{nbr}) \leftarrow p_{curr}$
20       $d(p_{nbr}) \leftarrow d$
21       $h(p_{nbr}) \leftarrow h$
22       $g(p_{nbr}) \leftarrow p_{curr}.g + distance(coord(p_{curr}), n)$
23       $f(p_{nbr}) \leftarrow g(p_{nbr}) + h(p_{nbr})$
24       $p_{neighbours}.Push(p_{nbr})$
25     end if
26   end if
27 end for
28 return $p_{neighbours}$
```

Algorithm 5 UpdateNode Function

```
Input: $open, p_{curr}, p_{nbr}$
Output: $open$
1  // $p_{nbr}$: $p_{curr}$ neighbour node
2  $d \leftarrow distance(coord(p_{curr}), coord(p_{nbr}))$
3  if $g(p_{curr}) + d < g(p_{nbr})$ then
4    $g(p_{nbr}) \leftarrow g(p_{curr}) + d$
5    $parent(p_{nbr}) = p_{curr}$
6    if $p_{nbr} \in open$ then
7      $open.Remove(p_{nbr})$
8    end if
9    $f(p_{nbr}) \leftarrow g(p_{nbr}) + h(p_{nbr})$
10   $open.Push(p_{nbr})$
11 end if
12 return $open$
```

We developed the Optimal Operation Search Algorithm taking as inspiration the A* algorithm (Hart et al., 1968; Daniel et al., 2010). The **inputs** are the set of areas A' computed in *Second Stage*, R and the set of vertices V'' computed in *Third Stage*. As well as A*, it uses the evaluation function $f = g + h$, where the cost function g is the accumulated distance to reach a target point and the heuristic function h represents the remaining target points to visit.

Algorithm 3 shows the pseudo code implemented in TERRA. It computes finite sub-tours $s \in S$ for each $v \in V''$ looking for the shortest travelling distance. Each sub-tour denotes the starting charging stop v as the p_{start} node. A node is an object containing the following information: the Cartesian Coordinates (*coord*) of the target point in a_v or the vertex in V'', its accumulated distance from the last charging stop (d), its cost function (g), its heuristic function (h), its evaluation function (t) and its parent node (*parent*). The following functions help to understand this algorithm:

– *getSubTour* (line 23): it recursively obtains the parent nodes of the search graph starting from p_{curr}.

– *ExpandGraph* (line 26, detailed in Algorithm 4): it represents the novelty of this algorithm because it integrates the UAV energy constraint R into the searching process. Each $n \in a_v$ represents a candidate neighbour node for p_{curr} (p_{start} included). At first, it checks if $n \in a_v$ has already been visited (a node can be visited only once in a route) (line 5).
Then, if $coord(p_{curr})$ is equal to $coord(p_{start})$ (line 6), it means that the route does an intermediate charging stop and it is starting again to visit the remaining target points, i.e., the accumulated distance from the last charging stop (d) has to be equal to the distance required to visit the neighbour n. If not, it accumulates the travelling distance to the neighbour n. Then, it checks that the sum of the distance required to visit n plus the distance required from n to return to the charging station p_{start} ($d + d_{start}$) is less than the maximum distance the UAV can travel ($d_{max} = R * 2$) ensuring the return to the charging station (line 12). If not, n is not a valid neighbour and it is not included into the search graph because the UAV would run out of energy following that route. Otherwise, the node n is included into the neighbours list $p_{neighbours}$. Note that p_{start} does not compute as a real target point in h (line 13).

– *UpdateNode* (line 33, detailed in Algorithm 5): it checks if the new distance ($d + g(p_{nbr})$) computed to reach p_{nbr} node from the p_{curr} parent is shorter than the distance ($g(p_{nbr})$) previously computed to reach p_{nbr} from another parent node (line 3). Thereupon, it updates the travel cost g, *parent* and the evaluation function f, and updates the node in the *open* nodes list.

For each $v \in V''$, the Optimal Operation Search Algorithm returns an ordered set of locations $s \in S$ denoting a UAV sub-tour where the UAV travelling distance has been minimized, as in Eq. (11):

$$F_{s \in S}(v_j \in V'', t_i \in a_j \in A') = g_{v_j,t_i} + \left(\sum_{i=1}^{p-1} g_{t_i,v_j} * x_i + g_{v_j,t_{i+1}} * x_i + g_{t_i,t_{i+1}} * \overline{x_i} \right) + g_{t_p,v_j} \tag{11}$$

The **output** is a set of UAV sub-tours S (Eq. (12)) satisfying the *Objective Equation* (2) as in Eq. (13) (see Fig. 7).

$$O^2SA(A', V'', R) \rightarrow \{s_1, \dots, s_k\} \in S \tag{12}$$

$$F_{uav} = \sum_{i=1}^{k} F_{s_i \in S} \tag{13}$$

Finally, once F_{ugv} and F_{uav} have been minimized, the *Objective Equation* (3) is minimized and the ECU-CSURP is solved.

5. Experimental evaluation

In this section we present the TERRA assessment solving the ECU-CSURP. To the best of our knowledge, the ECU-CSURP has never been defined as such, and then, there are no algorithms that we can use for comparison. In this way, the goal is to test the overall TERRA performance to analyse its capabilities and to provide an accurate algorithm characterization. At first, a random map generator and an effective parameter tuning of the genetic algorithm are devised in order to perform a proper evaluation. Then, the experimental evaluation will address the trade-off between the UGV travelling distance F_{ugv} and the UAV travelling distance F_{uav}. The following three experiments were carried out:

1. The first experiment (Section 5.3) aims to evaluate the overall TERRA performance analysing the main parameter fluctuations of the random map generator.

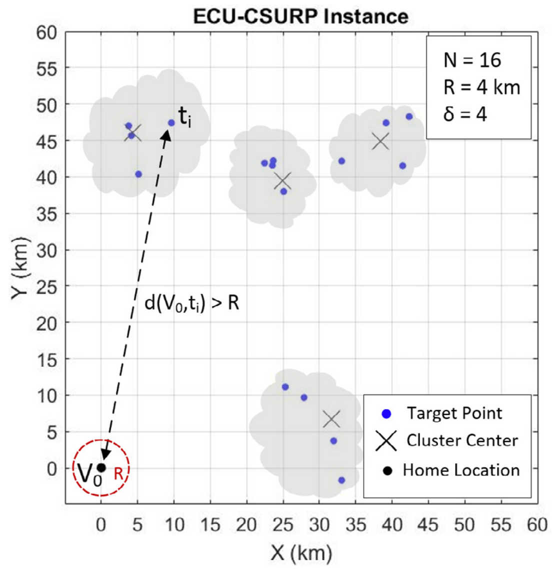


Fig. 8. A randomly generated ECU-CSURP instance. Note that the ECU-CSURP statements (i–vi) are satisfied.

2. The second experiment (Section 5.4) analyses the Gravitational Optimization Algorithm effects in the TERRA behaviour.
3. The third experiment (Section 5.5) analyses the Optimal Operation Search Algorithm effects in the TERRA behaviour.

The algorithm has been implemented in MATLAB, and all the experiments were carried out on a 2.6 GHz Intel Core i7 with 16 GB of RAM under Windows 10. The results of the three experiments are publicly available on GitHub.[2] Additionally, Appendix A shows the TERRA computational results in instances generated using the travelling salesman problem library TSPLIB (Reinelt, 1991). Also, Appendix B shows the results from the statistical tests carried out during the first experiment.

5.1. Random map generator

We developed a random map generator to build ECU-CSURP instances. A ECU-CSURP instance is a target point distribution which satisfies the problem statements (i–vi) described in Section 3. Each distribution controls the location of the target points by the number of groups of closely located targets, i.e., by clusters. The location of the target points belonging to the same cluster are generated using a 2D Gaussian Distribution[3] with a randomly cluster centre, and parameters $\mu = 0$ and $\sigma = a * R$, where $a \in (0.5, 2)$ and R is the farthest distance the UAV can travel. Thus, we introduce a standard deviation which goes from the half to the double of R, giving to TERRA a broad window to solve a single problem in different ways. The random number generation for the random centre and a is controlled by the Shuffle seed (based on the current time has been selected) and the Mersenne twister generator (Matsumoto and Nishimura, 1998). Therefore, our map generator relies on the following well defined parameters:

- N: number of target points.
- R: farthest distance the UAV can travel in km.
- δ: number of clusters.

The random map generator creates a map ensuring that N target points are distributed in δ clusters of N/δ target points following a 2D Gaussian Distribution inside an exploration area. Fig. 8 shows a feasible ECU-CSURP instance whose parameters satisfy the problem statements. The (i) ECU-CSURP statement is controlled by R. N and R represent the (ii) and (v) statements respectively. The (iv) and (vi) statements are controlled by placing a constant home location (V_0 in Fig. 8) for the whole experimentation. Also, the target points are distributed around an area satisfying the ECU-CSURP distance constraint $\{t \in T : d(V_0, t) > R\} \neq \emptyset$ ((iii) statement). Additionally, δ allows us to generate a broad random diversity of target points distributions. We assume N as a constant parameter during the experimental evaluation because there is a direct correlation with the *Objective Equation* (3). Therefore, we focused the TERRA assessment on the R and δ parameters.

5.2. Genetic parameters tuning

The genetic algorithm described in Section 4.4 requires a parameter tuning to allow us to characterize TERRA from a properly performance perspective. The objective is to select the best parameter setting minimizing the *Objective Equation* (1). A parameter setting is formed by the parameters: population size (P_s), tournament size (T_s), mutation operator (M_o), mutation rate (M_r), crossover operator (C_o), crossover rate (C_r), elitism size (E_s). The mutation operator included into the evaluation are: Flip, Swap and Slide. The crossover operators included are: Order Crossover (OX), Cycle Crossover (CX) and Order Base Crossover (CBX).

The tuning process was started by computing a fixed budget of solution evaluations ($Seval$) to set a fair racing (which means to allocate the same resources) among different parameters settings. For example, an unfair racing would be to compare a parameter setting A with $P_s = 500$ with a parameter setting B with $P_s = 100$, both running the same number of generations (denoted as N_G). In this case, A is given five times more solutions evaluations than B, and therefore, A can perform a much wider exploration, which is unfair. Thus, we denoted $Seval = N_G * P_s$ to act as a stopping criterion of the genetic algorithm in the racing procedure. In this way, given a fixed $Seval$, the parameter setting A and B will run a proportional $N_G = Seval/P_s$ according to its P_s. Once $Seval$ was fixed, the racing process could be launched to select the best parameter setting minimizing the *Objective Equation* (1). The results of the experiment are publicly available on GitHub.[4]

For the $Seval$ computing, we generated one hundred random maps with $N = 16$ and one hundred random maps with $N = 64$. We set a standard parameter setting following the De Jong and Spears (1989) guidelines: $P_s = 500$, $T_s = 4$, $M_o = \text{Flip}$, $M_r = 0.1$, $C_o = \text{CX}$, $C_r = 0.9$, $E_s = 1$. Then, we run the algorithm one time per random map, i.e., two hundred executions. Fig. 9 shows the results of this experiment. It plots the mean values (min, max, average) of the fitness function F_{ugv} on each generation, for random maps with $N = 16$ (black lines) and $N = 64$ (red lines). This convergence graph demonstrates that $N_G = 35$ is enough to compute the minimum fitness function. However, it may exist instances, with a similar parameter setting, whose convergence may be higher than 35 generations. In those cases, the algorithm would fail in finding the minimum fitness function. Consequently, we decided to set an offset until $N_G = 60$ generations enough to minimize the *Objective* Eq. (1). Then, the fixed budget computed was $Seval = 60 * 500 = 30{,}000$ solution evaluations.

For the racing process, we used the *irace* package (López-Ibáñez et al., 2016). *irace* performs iterated racing procedures to automatically configure the genetic algorithm by finding the most appropriate setting, given a set of random map instances and a specific parameters space. The parameters space defines the range of allowed (and not allowed) values of each parameter. Table 1 shows the chosen parameters space for this racing. Then, *irace* performed a training experiment where it selected the best four appropriate settings. Once the training was finished, it run a testing experiment where the best four settings were computed.

[2] https://github.com/FRopero/TERRA_Experiments.

[3] https://es.mathworks.com/help/stats/normrnd.html.

[4] https://github.com/FRopero/tuningGA.

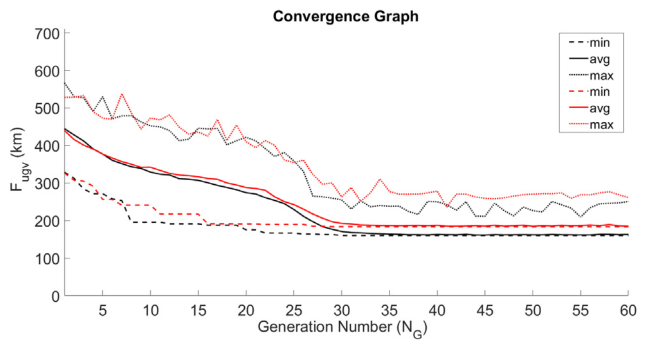


Fig. 9. Convergence graph plotting the mean values (min,avg,max) of F_{ugv} on each generation of the genetic algorithm. The black lines are random maps generated with $N = 16$, and the red lines with $N = 64$.

Table 1
Parameters space of the genetic algorithm used for the *irace* tuning software. The percentage is referred to P_s.

Parameter	Type	Values
Ps	Integer	[100, 500]
Ts	Integer	{3, 6, 9, 12}
Mo	Enum	{Flip, Swap, Slide}
Mr	Float	[1.0%, 10.0%]
Co	Enum	{OX, CX, OBX}
Cr	Float	[90.0%, 99.0%]
Es	Float	[1.0%, 5.0%]

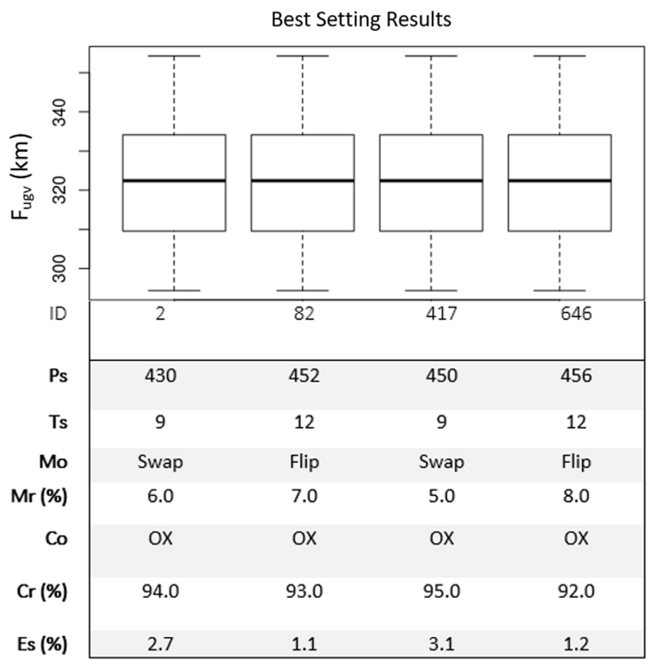
We generated one hundred random maps with $N = 16$ for training and one hundred more for testing. Fig. 10 shows at the bottom table, ordered from left to right, the best settings $ID = \{2, 82, 417, 646\}$ found by *irace*. At the top box-plot, Fig. 10 shows the results of executing each ID setting with each testing random map. We observe that the four settings have an equal performance and we can choose any of them to tune our genetic algorithm. In particular, all of them have a similar P_s, which means a similar N_G. However, the higher is P_s the higher are the selection pressure T_s and M_r. Also, we observe that a low selection pressure T_s is balanced with a high elitism E_s. Finally, we set up the genetic algorithm with the best setting $ID = 2$ for the following experimental evaluation.

5.3. First experiment. TERRA performance

The objective is to analyse the TERRA performance in ECU-CSURP instances with different R and δ values. As we mentioned in Section 5.1, N has been kept constant ($N = 16$) because of its correlation with the *Objective Equation* (3) of the ECU-CSURP, i.e., a constant N increment will approximately get a constant F_{total} increment.

The experiment setting is based on nine ECU-CSURP configurations combining three different values of $R = \{1, 3, 9\}$ and $\delta = \{2, 4, 8\}$. Each parameter configuration will be tested over five hundred random ECU-CSURP instances. On each instance, TERRA will select the best of the four Gravitational Optimization Algorithm solutions. Fig. 11 shows the experiment results in a box-plot matrix.

Firstly, we observe that the R increment generates a significant F_{uav} increment mainly due to the Gravitational Optimization Algorithm, as we will explain in Section 5.4. That is, the higher R, the higher is the attraction of the gravity point ($d(v_j, G_p)$ in *Third Stage*) and the higher is the reduction of the distance among the vertices and the gravity point ($d(p_j, G_p)$ in *Third Stage*), which conducts to a F_{uav} increment and a F_{ugv} decrement. Despite of this F_{ugv} decrement, the R increment building ECU-CSURP instances leads to a F_{ugv} increment due to the dispersion of the target points locations on each cluster (note that $\sigma = a * R$). That is, the higher R, the more scatter are the target points on each cluster, which in turns conducts to TERRA to require more vertices



ID	2	82	417	646
Ps	430	452	450	456
Ts	9	12	9	12
Mo	Swap	Flip	Swap	Flip
Mr (%)	6.0	7.0	5.0	8.0
Co	OX	OX	OX	OX
Cr (%)	94.0	93.0	95.0	92.0
Es (%)	2.7	1.1	3.1	1.2

Fig. 10. Results of *irace* for the training and testing experiment. At the bottom table, ordered from left to right, the best settings $ID = \{2, 82, 417, 646\}$ obtained in the training experiment. At the top box-plot, the *irace* results for computing the best settings in the testing experiment.

to cover the cluster and so, F_{ugv} is incremented. This F_{ugv} increment slightly dominates over the F_{ugv} decrement because of the Gravitational Optimization Algorithm. Due to $F_{total} = F_{ugv} + F_{uav}$, we can assert the following Lemma 5.1.

Lemma 5.1. *There is a direct correlation between* R *and* F_{total}.

In Appendix B we show the results of the One Way ANOVA tests performed to demonstrate the statistical significance of Lemma 5.1. Table B.3 shows that every p-value is lower than the significance level $\alpha = 0.05$, which demonstrates the statistical significance of Lemma 5.1.

Secondly, we observe that a δ decrement result in a F_{ugv} decrement for the same reason explained in the above paragraph. That is, the lower δ, the lower is the vertices number and so, F_{ugv}, which dominates over the Gravitational Optimization Algorithm effects and explains the F_{ugv} decrement. Also, we can observe that F_{uav} decreases with the δ decrement. This is because of the Optimal Operation Search Algorithm, as we will explain in Section 5.5. Due to $F_{total} = F_{ugv} + F_{uav}$, we can assert the following Lemma 5.2.

Lemma 5.2. *There is a direct correlation between* δ *and* F_{total}.

As in previous lemma, in Appendix B we show the results of the One Way ANOVA tests performed to demonstrate the statistical significance of Lemma 5.2. Table B.4 shows that every p-value is lower than the significance level $\alpha = 0.05$, which demonstrates the statistical significance of Lemma 5.2.

5.4. Second experiment. Analysing gravitational effects

The objective is to assess Lemma 5.1 by evaluating the Gravitational Optimization Algorithm effects in the four computed solutions, i.e., the three gravitational solutions with three different gravity points and the solution without applying a gravity point.

The experimental setting is based on four ECU-CSURP configurations in which $R = \{2, 4, 8, 16\}$. The rest of key parameters have been kept constant in the experiment ($\delta = 1$ and $N = 6$). Each configuration has

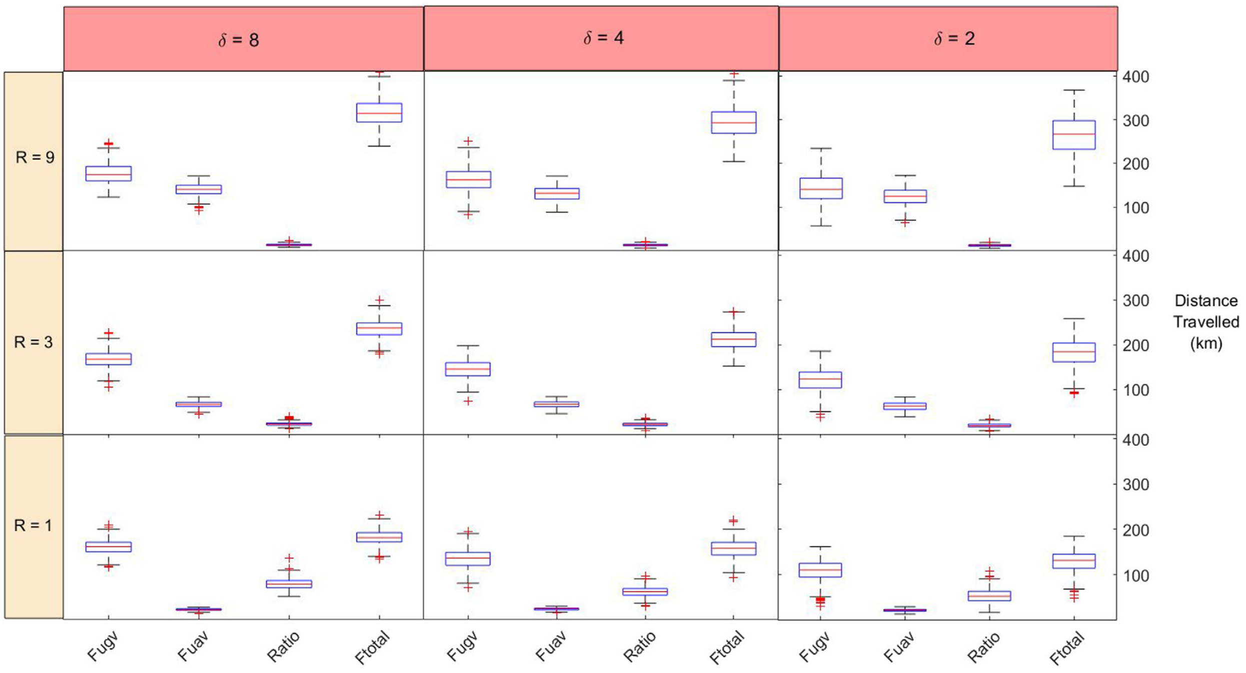


Fig. 11. Results for the TERRA execution in nine different δ and R ECU-CSURP configurations. Each configuration has been evaluated over five hundred randomly generated maps. δ (top x-axis) is the number of clusters. R (left y-axis) is the farthest distance R the UAV can travel in km. The parameters to evaluate (bottom x-axis) are: F_{ugv}, F_{uav}, $Ratio = F_{ugv}/F_{uav}$ and F_{total}. The parameter values (right y-axis) are displayed as the travel cost in km. We can assert that there is a direct correlation between R and F_{total}, and δ and F_{total}.

been tested over five hundred random ECU-CSURP instances. Fig. 12 shows the experiment results split up into six plots. On the one hand, the three top plots represent the results of the difference between the travelling distance of the solution without applying the Gravitational Optimization Algorithm f_{ugv}, f_{uav} and f_{total}, and the travelling distance of the three gravity solutions $f(g)_{ugv}$, $f(g)_{uav}$, $f(g)_{total}$ where $g \in G_p$ is the gravity point. Let $G_p = \{XC, HC, MC\}$ be the set of gravity points (see the definitions in Section 4.3). Then, the travelling distance deviation λ is computed as in Eqs. (14)–(16).

$$\lambda_{ugv} = f_{ugv} - f(g)_{ugv} \forall g \in G_p \tag{14}$$

$$\lambda_{uav} = f_{uav} - f(g)_{uav} \forall g \in G_p \tag{15}$$

$$\lambda_{total} = f_{total} - f(g)_{total} \forall g \in G_p \tag{16}$$

On the other hand, the three bottom plots represent a theoretical computation (there are not considered real-world variables such as the wind speed or terrain slope) of the time taken by the hybrid UGV–UAV system to cover the distances F_{ugv} and F_{uav}, respectively. Thus, we need to add to TERRA the motion speed parameters. We follow the assumption in Section 4.3 denoting $V_{ugv} = 0.13$ km/h (maximum speed of the Mars Science Laboratory) as the UGV motion speed, and $V_{uav} = 30$ km/h (standard value) as the UAV motion speed, i.e., $V_{ugv} \ll V_{uav}$. Also, let $t_{ugv} = f_{ugv}/V_{ugv}$, $t_{uav} = f_{uav}/V_{uav}$ and $t_{total} = t_{ugv} + t_{uav}$ denote the time taken to accomplish the mission without computing a gravity point. Let $t(g)_{ugv}$, $t(g)_{uav}$ and $t(g)_{total}$ denote the time taken to accomplish the mission computing a gravity point $g \in G_p$. Then, the total time deviation α is computed for each gravity point as stated in Eqs. (17)–(19).

$$\alpha_{ugv} = t_{ugv} - t(g)_{ugv} \forall g \in G_p \tag{17}$$

$$\alpha_{uav} = t_{uav} - t(g)_{uav} \forall g \in G_p \tag{18}$$

$$\alpha_{total} = t_{total} - t(g)_{total} \forall g \in G_p \tag{19}$$

From Fig. 12, we can highlight three evidences. Firstly, λ_{ugv} and α_{ugv} plots are in positive outcomes (top and bottom left plots). We can observe that the three gravity point solutions $f(g)_{ugv} \forall g \in G_p$ always improve the results against the solution without applying any gravity point f_{ugv}, i.e., $f(g)_{ugv} < f_{ugv}$. Then, the higher R, the lower is the distance among the junction points and the vertices ($d(p_j, G_p)$ in Fig. 6), and so, the greater is the Gravitational Optimization Algorithm efficiency.

Secondly, λ_{uav} and α_{uav} plots are in negative outcomes (top and bottom middle plots). Here, we can observe the main drawback in the trade-off between the UGV–UAV travelling distances. That is, when $\delta = 1$, the Gravitational Optimization Algorithm computes the maximum F_{ugv} reduction by maximizing F_{uav} (see Fig. 6). Notwithstanding the F_{uav} maximization in λ_{uav} plot (it almost reaches 30 km with $R = 16$ km), we observe in α_{uav} plot that the time difference among f_{uav} and $f(g)_{uav}$ solutions is below 1 h.

Thirdly, λ_{total} and α_{total} plots have different outcomes (top and bottom right plots). λ_{total} is always in negative outcomes for every $g \in G_p$. That is because the λ_{total} results only have taken into account the UGV–UAV travelling distances. Here, both $f(g)_{ugv}$ and $f(g)_{uav}$ are computed with equal importance from a distance perspective. Nevertheless, if we take the motion speed assumption stated in Section 4.3, we can see that the UGV motion speed ($V_{ugv} = 0.13$ km/h) is much slower than UAV motion speed ($V_{uav} = 30$ km/h), i.e., $V_{ugv} \ll V_{uav}$. Then, we can ensure that the UGV will need more time to cover the same distance than the UAV. Consequently, the $f(g)_{ugv}$ optimization will be more significant than the $f(g)_{uav}$ optimization from a time perspective. As we can see in α_{total} plot, the results are always in positive outcomes. Both plots show that the Gravitation Optimization Algorithm computes better solutions (in average) computing the MC gravity point until $R \approx 12$. These statements can be defined as follows:

$$\lambda_{total}(MC) > \lambda_{total}(g) \; \forall g \in \{XC, HC\}, \;\; R \leq 12 \tag{20}$$

$$\alpha_{total}(MC) < \alpha_{total}(g) \; \forall g \in \{XC, HC\}, \;\; R \leq 12 \tag{21}$$

In summary, we can assert that R has a key impact on the TERRA performance. In particular, the Gravitational Optimization Algorithm performance will always depend on R and the motion speed of the hybrid UGV–UAV system. From a distance perspective, the performance is lower as R increases, i.e., λ_{total} increases, because the UGV travelling distance minimization is less significant than the UAV travelling distance maximization. Nevertheless, from a time perspective, the performance is higher as R increases, i.e., α_{total} decreases, because the UGV

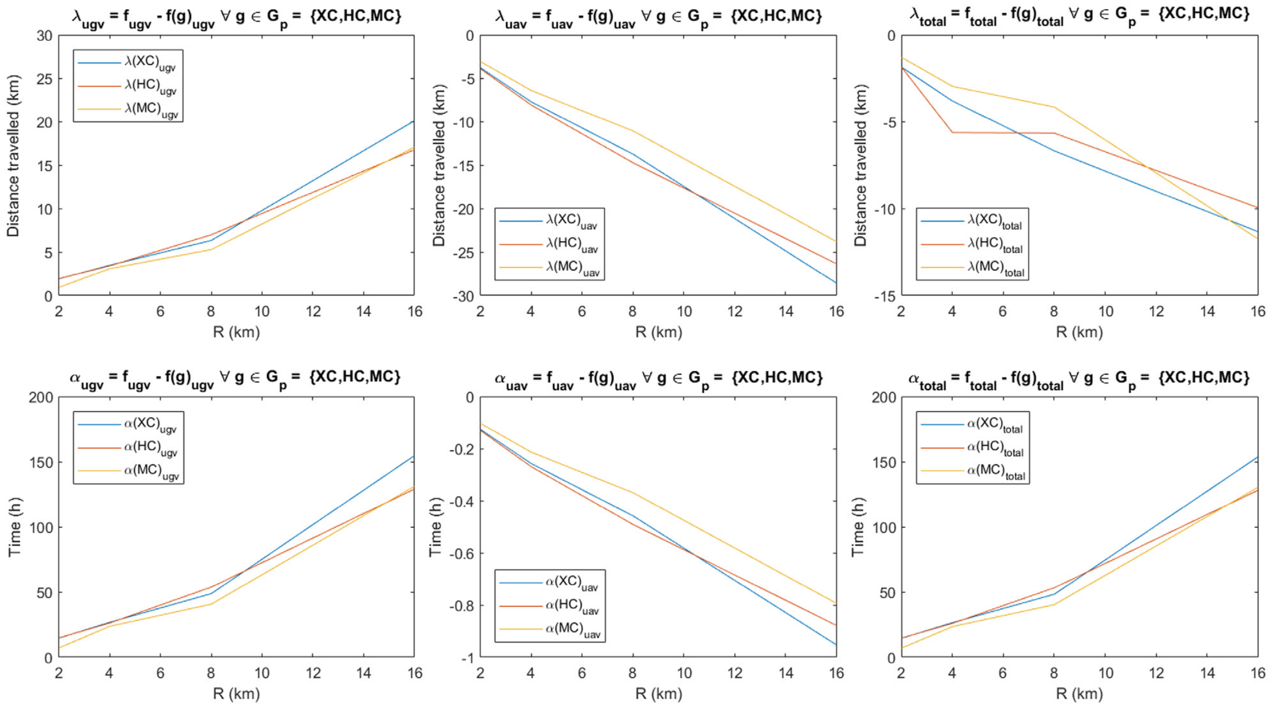


Fig. 12. Results for the TERRA execution in four different ECU-CSURP configurations with $R = \{2, 4, 8, 16\}$, $\delta = 1$ and $N = 6$. Each configuration has been evaluated over five hundred randomly generated ECU-CSURP instances. The top and bottom x-axis is the farthest distance R the UAV can travel in km. The top y-axis is the distance travelled in km. The bottom y-axis is the time taken to cover the distance in hours. Each plot shows the TERRA solutions in the three gravity point results and the results without a gravity point. We conclude that $f(g)_{ugv}$ optimization will be more significant than $f(g)_{uav}$ optimization from a time perspective.

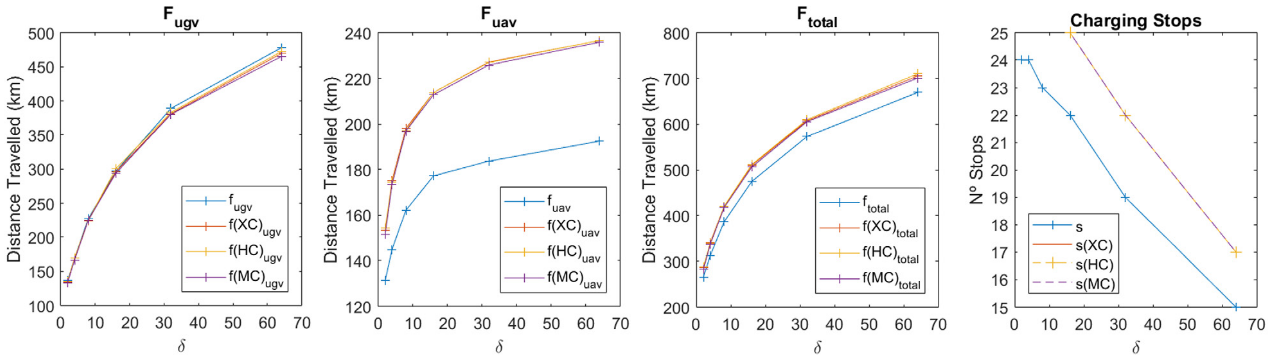


Fig. 13. Results for the TERRA execution in six different ECU-CSURP configurations with $\delta = \{2, 4, 8, 16, 32, 64\}$, $R = 2$ and $N = 64$. Each configuration has been evaluated over five hundred randomly generated maps. The x-axis is the number of clusters (δ). The y-axis for the three left plots is the distance travelled in km. The y-axis for the right plot is the mean of charging stops. Each plot shows the TERRA solutions in the three gravity point results and the results without gravity point. We demonstrate that between two maps with the same key parameters but different δ, TERRA generates a better solution to the map with lower δ.

travelling time minimization is more significant than the UAV travelling time maximization.

5.5. Third experiment. Analysing clustering effects

The objective is to analyse Lemma 5.2 by evaluating the Optimal Operation Search Algorithm over different clustering settings.

This experiment consists of six ECU-CSURP configurations in which $\delta = \{2, 4, 8, 16, 32, 64\}$. The rest of the key parameters have been kept constant during the experimentation ($R = 2$ and $N = 64$). Each configuration has been tested over five hundred random ECU-CSURP instances. Fig. 13 shows the experiment results. From left to right, the plots represent F_{ugv}, F_{uav}, F_{total} and $ChargingStops$ for the three gravity point solutions (XC, HC, MC) and the solution without computing a gravity point.

From Fig. 13, we can highlight three evidences. Firstly, the F_{ugv} plot shows that, the higher δ, the higher are f_{ugv} and $f(g)_{ugv} \forall g \in G_p$. In fact, the higher δ, the higher is the Gravitational Optimization Algorithm performance because it finds less constraints to compute a junction point near to the gravity point (see Section 4.3). This explains why the $f(g)_{ugv} \forall g \in G_p$ are slightly lower than f_{ugv} while δ increases. Nevertheless, the higher δ, the higher is the number of charging stops and so, the higher are f_{ugv} and $f(g)_{ugv} \forall g \in G_p$.

Secondly, the F_{uav} plot shows that, the higher δ, the higher are f_{uav} and $f(g)_{uav} \forall g \in G_p$. This is because of the Optimal Operation Search Algorithm performance, i.e., the higher δ, the lower the probability to schedule a path with intermediate charging stops. Also, we can appreciate two differences among $f(g)_{uav} \forall g \in G_p$ and f_{uav}. On the one hand, the Optimal Operation Search Algorithm computes the maximum increment of $f(g)_{uav} \forall g \in G_p$, i.e., $d(p_j, G_p) \forall\ p_j \in P$ in Fig. 6a. Then, it needs to place the maximum number of charging stops in the UAV sub-tours to accomplish them. As we can see in the *Charging Stops* plot, the number of charging stops in $f(g)_{uav} \forall g \in G_p$ ($s(XC), s(HC), s(MC)$) is always higher than f_{uav} (s). Also, we can see that this effect decreases as δ increases. On the other hand, if any gravity point is applied, there is no f_{uav} aggravation, then the Optimal Operation Search Algorithm has a greater performance.

The third evidence is related to the intermediate charging stops number in the *Charging Stops* plot. Here, we can observe the $\Delta = \delta/R$ relation. We can assert that the Optimal Operation Search Algorithm has a turning point in $\Delta = 2$. In $\Delta \leqslant 2$, the algorithm performance is high because δ is too low compared with the area covered by R. Then, the Optimal Operation Search Algorithm requires a large number of intermediate charging stops to find a UAV sub-tour. In $\Delta \geqslant 2$, the performance decreases because δ is too high compared with R, and so, the algorithm has less probability to find a UAV sub-tour with a minimum set of intermediate charging stops.

In summary, we can assert that δ has a key impact on the TERRA performance. As F_{total} plot shows, between two ECU-CSURP instances with the same key parameters but different δ, TERRA generates a better solution in the instance with lower δ. This parameter can be very useful to take it into account in the scientific goal's planning task.

6. Conclusions

In this article we formulated the ECU-CSURP, which is based on an exploration with a hybrid UGV–UAV system where the goal is to achieve a set of target points distributed around an exploration area. There are several problems in the existing literature defining the same goal as ECU-CSURP, but no one considers the same inputs, e.g., some approaches do not consider the UGV into the problem and use a set of UAVs and a set of properly located depots to do refuelling among target points. Instead, we use the UGV as a moving charging station to carry the UAV through the set of target points. Furthermore, unlike other approaches, the ECU-CSURP considers an exploration area and both the UGV and UAV need to find the finest route. Also, other approaches set specific constraints to the target points and are assigned to the robotic system which satisfies these constraints. In the ECU-CSURP, the target points need to be achieved by the UAV.

Then, we proposed TERRA as a solution to solve the ECU-CSURP. TERRA is a five-steps strategy to achieve the ECU-CSURP goals. Firstly, it computes Voronoi Tessellations to place the charging stops where the UGV is going to carry the UAV. Secondly, it applies the Hitting Set Problem to select the optimal set of charging stops covering the full set of target points. Thirdly, it reduces the UGV travelling distance by reducing the distance among the charging stops and a common point with a novel gravitational algorithm. Here, it generates four different solutions of charging stops distributions. Fourthly, it computes the Travelling Salesman Problem for the UGV on every solution. Fifthly, it computes the Travelling Salesman Problem for the UAV on every solution. Finally, TERRA selects the best of the four solutions.

The experimentation process aimed to characterize TERRA in a wide range of ECU-CSURP instances. It was split up in three experiments. The first experiment aimed to evaluate the TERRA performance on ECU-CSURP instances. The former results in the first experiment revealed that the UGV and UAV travelling distances lay on key parameters used to build ECU-CSURP instances. The second experiment evaluated the performance of the Gravitational Optimization Algorithm to optimize the overall mission distance from a time perspective. We demonstrated the algorithm efficiency in time terms. The third experiment focused on testing the performance of the Optimal Operation Search Algorithm designed to solve the Travelling Salesman Problem as a classic search algorithm. We proved that the algorithm efficiency depends on specific key parameters. The experimentation demonstrates that it is essential to take into account these results in order to design ECU-CSURP instances where TERRA will have a strong performance.

The results obtained in our experiments encourages to enhance our algorithm in different ways. For instance, a future research is to introduce environmental constrained path planning for the hybrid UGV–UAV system, considering variables such as terrain slope, wind air or weather conditions, but also to introduce online path planning for the UAV. Another direction is to update the First and Third Stages to improve TERRA results in a wider range of ECU-CSURP instances. We are also interested in studying the Optimal Operation Search Algorithm as a classical search against an evolutionary approach and to analyse the genetic algorithm with several settings and operators such as the Inver-Over Operator (Tao and Michalewicz, 1998).

Table A.2
TERRA computational results.

Name	Type	N	R (km)	F_{ugv} (km)	F_{uav} (km)	F_{total} (km)	#Stops	Time (s)
bays29	GEO	29	4	9535.1	224.0	9759.1	0	40.4
bays29	GEO	29	16	9679.8	656.1	10 335.9	0	36.9
bays29	GEO	29	64	8927.0	3522.5	12 439.5	0	37.3
bays29	GEO	29	128	8191.0	5708.4	13 899.4	5	42.2
eil51	EUC_2D	51	4	419.4	324.9	744.3	13	81.0
eil51	EUC_2D	51	16	220.3	790.4	1010.7	20	335.7
eil51	EUC_2D	51	64	0	729.3	729.3	5	9542.8
eil51	EUC_2D	51	128	0	545.8	545.8	2	10 279.0
eil76	EUC_2D	76	4	510.1	542.7	1052.8	30	145.3
eil76	EUC_2D	76	16	164.8	1143.5	1308.3	30	933.5
eil76	EUC_2D	76	64	0	1279.6	1279.6	10	33 333.0
eil76	EUC_2D	76	128	0	746.9	746.9	2	40 664.0
berlin52	EUC_2D	52	4	10 989.0	293.9	11 282.9	0	84.8
berlin52	EUC_2D	52	16	10 393.0	1595.5	11 988.5	3	83.4
berlin52	EUC_2D	52	64	7757.9	4579.2	12 337.1	14	110.6
berlin52	EUC_2D	52	128	5758.8	8225.7	13 984.5	20	178.0

Table B.3
One Way ANOVA tests results (p-value) for F_{ugv}, F_{uav} and F_{total} with the three groups of samples $R = \{1,3,9\}$ and a fixed δ value on each column.

	$\delta = 8$	$\delta = 4$	$\delta = 2$
	$R = \{1,3,9\}$	$R = \{1,3,9\}$	$R = \{1,3,9\}$
F_{ugv}	1.07×10^{-17}	1.59×10^{-39}	1.88×10^{-34}
F_{uav}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
F_{total}	7.38×10^{-307}	6.90×10^{-272}	8.76×10^{-205}

Acknowledgements

The authors want to thank Dr. David F. Barrero for the fruitful discussions and his help to carry out the experiments. The work is supported by the Universidad de Alcalá project 2016/00351/001.

Appendix A. Computational results

In this appendix we show the TERRA computational results (see Table A.2) for the TSBLIB instances (Reinelt, 1991): *bays29*, *eil51*, *eil76* and *berlin52*. These instances have $N = 29{,}51{,}76$ and 52 respectively. We performed a computational study with $R = [4, 16, 64, 128]$ in km. The home location selected was the first target point displayed on each instance. The column headings are: **Name** (instance name), **Type** (weight type), **N** (number of target points) **R** (farthest distance the UAV can travel in km), F_{ugv} (UGV's distance travelled), F_{uav} (UAV's distance travelled), F_{total} (total distance travelled), **#Stops** (number of charging stops) and **Time** (computational time in seconds). The distance travelled in EUC_2D instances is dimensionless, but the rest of them is in km.

Appendix B. Statistical results

In this appendix we show the statistical results of the One Way ANOVA tests performed to demonstrate the statistical significance of Lemma 5.1 (see Table B.3) and Lemma 5.2 (see Table B.4) taken from Fig. 11 (see Section 5.3). For the analysis, we took the results of TERRA shown in Fig. 11 and we applied the *anova1 function*[5] integrated in the

[5] https://es.mathworks.com/help/stats/anova1.html#bulav5i-group.

Table B.4
One Way ANOVA tests results (p-value) for F_{ugv}, F_{uav} and F_{total} with the three groups of samples $\delta = \{2,4,8\}$ and a fixed R value on each column.

	$R = 9$	$R = 3$	$R = 1$
	$\delta = \{2,4,8\}$	$\delta = \{2,4,8\}$	$\delta = \{2,4,8\}$
F_{ugv}	1.08×10^{-35}	9.19×10^{-92}	2.69×10^{-112}
F_{uav}	7.83×10^{-22}	2.67×10^{-13}	7.45×10^{-10}
F_{total}	2.87×10^{-43}	2.65×10^{-93}	6.91×10^{-110}

MATLAB Statistics and Machine Learning Toolbox. Both tables show the probability value (p-value) obtained on each test. As a typical analysis, we used the standard significance level $\alpha = 0.05$ (Nuzzo, 2014) to determine the statistical significance of the samples. For instance, a p-value $= 1.07 \times 10^{-17}$ in F_{ugv} and $\delta = 8$, in Table B.3, shows the One Way ANOVA test result for F_{ugv} with the three groups of samples $R = \{1,3,9\}$. More details are available on GitHub.[6]

Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.engappai.2018.11.008.

References

Aghaeeyan, A., Abdollahi, F., Talebi, H.A., 2015. UAV–UGVs cooperation: With a moving center based trajectory. Robot. Auton. Syst. 63, 1–9.

Bellingham, J.S., Tillerson, M., Alighanbari, M., How, J.P., 2002. Cooperative path planning for multiple UAVs in dynamic and uncertain environments. In: Decision and Control, 2002, Proceedings of the 41st IEEE Conference on. IEEE, pp. 2816–2822.

Burgard, W., Moors, M., Stachniss, C., Schneider, F.E., 2005. Coordinated multi-robot exploration. IEEE Trans. Robot. 21, 376–386.

Chandler, P.R., Pachter, M., Rasmussen, S., 2001. UAV cooperative control. In: American Control Conference, 2001. Proceedings of the 2001. IEEE, pp. 50–55.

Cortes, J., Martinez, S., Karatas, T., Bullo, F., 2004. Coverage control for mobile sensing networks. IEEE Trans. Robot. Autom. 20, 243–255.

Daniel, K., Nash, A., Koenig, S., Felner, A., 2010. Theta*: Any-angle path planning on grids. J. Artificial Intelligence Res. 39, 533–579.

Dantzig, G., Fulkerson, R., Johnson, S., 1954. Solution of a large-scale traveling-salesman problem. J. Oper. Res. Soc. Amer. 2, 393–410.

De Jong, K.A., Spears, W.M., 1989. Using genetic algorithms to solve NP-complete problems. In: ICGA. pp. 124–132.

Ding, X.C., Rahmani, A.R., Egerstedt, M., 2010. Multi-UAV convoy protection: An optimal approach to path planning and coordination. IEEE Trans. Robot. 26, 256–268.

Dubins, L.E., 1957. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. Amer. J. Math. 79, 497–516.

Ergezer, H., Leblebiciolu, K., 2014. 3D path planning for multiple UAVs for maximum information collection. J. Intell. Robot. Syst. 73, 737.

Funke, S., Nusser, A., Storandt, S., 2015. Placement of loading stations for electric vehicles: No detours necessary!. J. Artificial Intelligence Res. 53, 633–658.

Goldberg, D.E., Deb, K., 1991. A comparative analysis of selection schemes used in genetic algorithms. In: Foundations of Genetic Algorithms, Vol. 1. Elsevier, pp. 69–93.

Gori, F., Folino, G., Jetten, M.S., Marchiori, E., 2011. MTR: taxonomic annotation of short metagenomic reads using clustering at multiple taxonomic ranks. Bioinformatics 27, 196–203.

Grocholsky, B., Keller, J., Kumar, V., Pappas, G., 2006. Cooperative air and ground surveillance. IEEE Robot. Autom. Mag. 13, 16–25.

Gutin, G., Punnen, A.P., 2006. The Traveling Salesman Problem and its Variations, Vol. 12. Springer Science & Business Media.

Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4, 100–107.

Kirk, J., 2007. Traveling salesman problem-genetic algorithm. Retrieved from the MATLAB File Exchange website: www.mathworks.com/matlabcentral/fileexchange/13680-travelingsalesman-problem-genetic-algorithm.

Larranaga, P., Kuijpers, C.M., Murga, R.H., Inza, I., Dizdarevic, S., 1999. Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artif. Intell. Rev. 13, 129–170.

Leahy, K., Zhou, D., Vasile, C.I., Oikonomopoulos, K., Schwager, M., Belta, C., 2016. Persistent surveillance for unmanned aerial vehicles subject to charging and temporal logic constraints. Auton. Robots 40, 1363–1378.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., Birattari, M., 2016. The 1irace package: Iterated racing for automatic algorithm configuration. Oper. Res. Perspect. 3, 43–58. http://dx.doi.org/10.1016/j.orp.2016.09.002.

Maini, P., Sujit, P., 2015. On cooperation between a fuel constrained UAV and a refueling UGV for large scale mapping applications. In: Unmanned Aircraft Systems (ICUAS), 2015 International Conference on. IEEE, pp. 1370–1377.

Manyam, S.G., Casbeer, D.W., Sundar, K., 2016. Path planning for cooperative routing of air-ground vehicles. In: American Control Conference (ACC), 2016. IEEE, pp. 4630–4635.

Mathew, N., Smith, S.L., Waslander, S.L., 2015. Multirobot rendezvous planning for recharging in persistent tasks. IEEE Trans. Robot. 31, 128–142.

Matsumoto, M., Nishimura, T., 1998. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. (TOMACS) 8, 3–30.

McLain, T.W., Chandler, P.R., Rasmussen, S., Pachter, M., 2001. Cooperative control of UAV rendezvous. In: American Control Conference, 2001. Proceedings of the 2001. IEEE, pp. 2309–2314.

Mustafa, N.H., Ray, S., 2010. Improved results on geometric hitting set problems. Discrete Comput. Geom. 44, 883–895.

Nuzzo, R., 2014. Scientific method: statistical errors. Nat. News 506, 150.

Oberlin, P., Rathinam, S., Darbha, S., 2009. A transformation for a heterogeneous, multiple depot, multiple traveling salesman problem. In: American Control Conference, 2009. ACC'09. IEEE, pp. 1292–1297.

Reinelt, G., 1991. TSPLIB—A traveling salesman problem library. ORSA J. Comput. 3, 376–384.

Richards, A., Bellingham, J., Tillerson, M., How, J., 2002. Coordination and control of multiple UAVs. In: AIAA Guidance, Navigation, and Control Conference, Monterey, CA.

Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., Younes, H., 2000. Coordination for multi-robot exploration and mapping. In: AAAI/IAAI. pp. 852–858.

Sujit, P., Sousa, J., Pereira, F.L., 2009. UAV and AUVs coordination for ocean exploration. In: Oceans 2009-Europe. IEEE, pp. 1–7.

Sundar, K., Rathinam, S., 2014. Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots. IEEE Trans. Autom. Sci. Eng. 11, 287–294.

Sundar, K., Rathinam, S., 2016. Algorithms for heterogeneous, multiple depot, multiple unmanned vehicle path planning problems. J. Intell. Robot. Syst. 1–14.

Suzuki, K.A., Kemper Filho, P., Morrison, J.R., 2012. Automatic battery replacement system for UAVs: Analysis and design. J. Intell. Robot. Syst. 65, 563–586.

Swieringa, K.A., Hanson, C.B., Richardson, J.R., White, J.D., Hasan, Z., Qian, E., Girard, A., 2010. Autonomous battery swapping system for small-scale helicopters. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, pp. 3335–3340.

Tanner, H.G., 2007. Switched UAV-UGV cooperation scheme for target detection. In: Robotics and Automation, 2007 IEEE International Conference on. IEEE, pp. 3457–3462.

Tao, G., Michalewicz, Z., 1998. Inver-over operator for the tsp. In: International Conference on Parallel Problem Solving from Nature. Springer, pp. 803–812.

Tian, J., Shen, L., Zheng, Y., 2006. Genetic algorithm based approach for multi-UAV cooperative reconnaissance mission planning problem. In: International Symposium on Methodologies for Intelligent Systems. Springer, pp. 101–110.

Watson, D.F., 1981. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. Comput. J. 24, 167–172.

[6] https://github.com/FRopero/TERRA_Experiments.