Soft Comput (2017) 21:4883–4900
DOI 10.1007/s00500-016-2376-7

FOCUS

Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms

Cristian Ramirez-Atencia[1] · **Gema Bello-Orgaz**[1] · **María D. R-Moreno**[2] · **David Camacho**[1]

Published online: 3 October 2016

Abstract Due to recent booming of unmanned air vehicles (UAVs) technologies, these are being used in many fields involving complex tasks. Some of them involve a high risk to the vehicle driver, such as fire monitoring and rescue tasks, which make UAVs excellent for avoiding human risks. Mission planning for UAVs is the process of planning the locations and actions (loading/dropping a load, taking videos/pictures, acquiring information) for the vehicles, typically over a time period. These vehicles are controlled from ground control stations (GCSs) where human operators use rudimentary systems. This paper presents a new multi-objective genetic algorithm for solving complex mission planning problems involving a team of UAVs and a set of GCSs. A hybrid fitness function has been designed using a constraint satisfaction problem to check whether solutions are valid and Pareto-based measures to look for optimal solutions. The algorithm has been tested on several datasets, optimizing different variables of the mission, such as the makespan, the fuel consumption, and distance. Experimental results show that the new algorithm is able to obtain good solutions; however, as the problem becomes more complex, the optimal solutions also become harder to find.

Communicated by C. Analide.

✉ Cristian Ramirez-Atencia
cristian.ramirez@inv.uam.es

Gema Bello-Orgaz
gema.bello@uam.es

María D. R-Moreno
mdolores@aut.uah.es

David Camacho
david.camacho@uam.es

[1] Universidad Autonónoma de Madrid, Madrid, Spain

[2] Universidad de Alcalá, Madrid, Spain

Keywords Unmanned air vehicles · Mission planning · Multi-objective optimization · Genetic algorithms · Constraint satisfaction problems

1 Introduction

Nowadays, unmanned air vehicles (UAVs) or drones have become very popular in many potential applications including surveillance (Pereira et al. 2009), disaster and crisis management (Wu and Zhou 2006), and agriculture or forestry (Merino et al. 2006) among others. For this reason, many research works related to this field have been developed over the past 20 years (Kendoul 2012; Lee and Kim 2008; Rodríguez-Fernández et al. 2015a).

The rapid development of the UAV capabilities has caused their incorporation into many areas to perform complex tasks which involve a high risk to the vehicle driver, such as detecting forest fires or rescue tasks. So using UAVs avoids risking human lives while their manageability permits to reach areas of hard access.

The process of mission planning for a team of UAVs involves generating tactical goals, commanding structure, coordination, and timing. Currently, UAVs are controlled remotely by human operators from ground control stations (GCSs), using rudimentary planning systems, such as following preplanned or manually provided plans. In order to perform more complex tasks and coordinated missions, these systems require more advanced capabilities.

Mission planning problems (MPPs) are a big challenge in actual NP-hard optimization problems. Classic planners are based on graph search or use a logic engine. But this kind of planners have several limitations, probably the most important is the high computational cost that their algorithms need to solve these missions. These missions have a lot of require-

ments that have to be considered, and it is also necessary to coordinate all the UAVs. These requirements generate search graphs that require a huge process capabilities to find a solution. In addition, multi-UAV missions usually require the use of several GCSs for controlling all the UAVs involved. This generates a new multi-GCS approach that makes this problem even harder to solve.

Another critical issue in MPP is that there are several parameters which can be used to define the quality of a solution, such as the fuel consumption, the makespan, and the cost of the mission. In these cases, a Pareto-optimal frontier (POF) can be computed in order to get the best solutions optimizing different objectives at the same time. Due to mission planning is based on search problems, an option to solve this type of problems could be using multi-objective evolutionary algorithms (MOEAs). In this work, we extend a previous work (Bello-Orgaz et al. 2015) in order to design and implement a multi-objective genetic algorithm (MOGA) to solve this problem. For this purpose, a fitness function consisting of two phases has been designed. Firstly, modeling the MPP as a CSP, the fitness function checks that the solution plans fulfill all the constraints given by the different capabilities of the UAVs and the GCSs involved. Afterward, using the validated plans, a Pareto-based function is calculated to optimize different quality parameters of the solutions.

The rest of the paper is structured as follows. Section 2 describes the related work concerning mission planning, CSPs, and GAs. Section 3 presents the mission planning problem, while Sect. 4 presents the CSP approach used to model it. Section 5 presents the MOGA-CSP approach, the encoding designed and the fitness function implemented to solve multi-GCS MPP. Section 6 provides a description of the dataset employed, the setup employed in the MOGA-CSP, and a complete experimental evaluation of it. Finally, in Sect. 7, the conclusions and some future research lines of the work are presented.

2 Related work

This section starts with a general introduction to mission planning techniques. After this brief introduction, an overview of constraint satisfaction problems is presented showing the different methods used in the literature to solve them. Finally, a description of genetic algorithms (GAs) and their applications to optimization problems has been carried out.

2.1 Mission planning

Planning has been an area of research in artificial intelligence (AI) for over three decades. A variety of tasks including robotics (Diaz et al. 2013), Web-based information gathering (Kuter et al. 2005), autonomous agents (Camacho et al. 2006), and mission control (Vachtsevanos et al. 2005) have benefited from planning techniques. Moreover, mission planning is a common problem in AI. A mission can be described as a set of goals that must be achieved by performing some task with a group of resources over a period of time. The whole problem can be summed up in finding the correct schedule of resource–task assignments that satisfies the proposed constraints.

In the literature, there are some attempts to implement mission planning systems. Doherty et al. (2009) presents an architectural framework for Mission Planning and execution monitoring, using temporal action logic (TAL). Fabiani et al. (2007) modeled the problem for search and rescue scenarios using Markov decision process (MDP) and solve it with dynamic programming algorithms. German Aerospace Centre (DLR) also developed a mission management system based on the behavior paradigm (Adolf and Andert 2010) which has been integrated onboard the ARTIS helicopter and validated in different scenarios, including waypoints following and search and track missions.

An essential concept in mission planning is cooperation or collaboration, which occurs at a higher level when various UAVs work together in a common mission sharing data and controlling actions together. There are few contributions that deal with multi-UAV problems in a deliberative paradigm (cooperative task assignment and mission planning). Bethke et al. (2008) proposed an algorithm for cooperative task assignment that extends the receding horizon task assignment (RHTA) algorithm to select the optimal sequence of tasks for each UAVs. Another approach by Kvarnström and Doherty (2010) proposes a new mission planning algorithm for collaborative UAVs based on combining ideas from forward-chaining planning with partial-order planning. This approach led to a new hybrid partial-order forward-chaining (POFC) framework that meets the requirements on centralization, abstraction, and distribution found in realistic emergency services settings.

Other works focus on distributed approaches for solving mission planning. Pascarella et al. (2015) proposed a core paths graph (CPG) algorithm for trajectory planning.

Finally, other approaches formulate the mission planning problem as a constraint satisfaction problem (CSP), where the tactic mission is modeled and solved using constraint satisfaction techniques (Ramirez-Atencia et al. 2015b).

2.2 Constraint satisfaction problems

The MPP can be summed up in finding the correct schedule of resource–task assignments which satisfies the proposed constraints, similarly to CSPs. It can be defined as follows (Barták 1999):

- A set of variables $V = v_1, ..., v_n$.
- For each variable, a finite set of possible values D_i (its domain).
- A set of constraints C_i restricting the values that variables can simultaneously take.

A CSP is usually represented as a graph where the pairs <Variable,Value> are the nodes and the constraints are the edges. Although there are other representations as those presented in Gonzalez-Pardo et al. (2014) for ant colony optimization and videogames. These methods usually have a propagation phase where the different constraints of the problem are checked. In the literature, there are many proposed methods to search the space of solutions for CSPs, such as backtracking (BT), backjumping (BJ) or look-ahead techniques (i.e., forward checking (FC) Bessière et al. 1999) among others. These algorithms are usually combined with other techniques like consistency techniques (Bessière 2006) (domain consistency, arc consistency, or path consistency) to modify the CSP and ensure its local consistency conditions.

In many real-life applications, it is necessary to find a good solution, and not the complete space of possible solutions. For this purpose, combining a CSP with an optimization function results in a constraint satisfaction optimization problem (CSOP). In these approaches, the optimization function maps every solution (complete labeling of variables) to a numerical value measuring the quality of the solution. The most widely used algorithm for finding optimal solutions is called branch and bound (B&B) (Rasmussen and Shima 2006). This algorithm searches for solutions in a depth first manner pruning the sub-tree under the current partial labeling when it exceeds the bound of the best value so far. In the case of multi-objective optimization, an extension of this method, known as multi-objective branch and bound (MOBB) (Rodríguez-Fernández et al. 2015a) is used to find the Pareto-optimal frontier (POF) composed of all non-dominated solutions of the problem. Other methods for solving CSOP include Russian doll search (Rollon and Larrosa 2007), bucket elimination (Rollon and Larrosa 2006), genetic algorithms (Fonseca and Fleming 1998), and swarm intelligence (Gonzalez-Pardo and Camacho 2013).

A TCSP is a particular class of CSP where variables represent times (time points, time intervals, or durations) and constraints represent sets of allowed temporal relations between them (Schwalb and Vila 1998). Different classes of constraints are characterized by the underlying set of basic temporal relations (BTR). Most types of TCSPs can be represented using point algebra (PA), with BTR = $\{\varnothing, <, =, >, \leq, \geq, ?\}$. A commonly used approach is Allen's interval algebra (Allen 1983), which defines several relations between time intervals, with BTR = $\{<, >, m, mi, o, oi, s, si, d, di, f, fi, =\}$.

In the related literature, Mouhoub (2002) proved that on real-time or Maximal TCSPs (MTCSPs), the best methods for solving TCSPs are Min-Conflict-Random-Walk (MCRW) for under and middle-constrained problems, and Tabu search and Steepest-Descent-Random-Walk (SDRW) in the over-constrained case. In this work, the author also developed a temporal model (TemPro Mouhoub 2004) which was based on interval algebra, to translate an application involving temporal information into a CSP. A TCSP can perfectly represent a multi-UAV mission as a set of temporal constraints over the time the tasks in the mission start and end.

2.3 Genetic algorithms

Genetic algorithms (GAs) have been traditionally used in a large number of different domains, mainly related to optimization problems (Holland 1992). These stochastic methods are inspired by natural evolution and genetics, and the complexity of the algorithm depends on the codification and the operations used to reproduce, cross, mutate, and select the different individuals of the population. There is a wide range of applications where GAs have been successful, from optimization (Bin et al. 2010) to data mining (Bello-Orgaz and Camacho 2014; Menendez et al. 2014). GAs have demonstrated to be robust, able to find satisfactory solutions in highly multi-dimensional problems with complex relationships between the variables. In recent works (Hao and Liu 2015; Ramirez-Atencia et al. 2015a), GAs have been used to represent CSPs.

There exists several research studies regarding the application of GAs to solve MPPs, but most of them are focused on just one UAV or in a single type of task. The Soliday et al.'s (1999) approach developed a GA to solve UAV missions under complex constraints. The GA was constructed using a representation based on the nearest neighbor search, being each allele an Nth nearest neighbor, and uses a qualitative fitness function based on the number of mission objectives and the time permitted. Tang et al. (2011) created a nested GA for military planning (resource allocation and task scheduling) based on the robustness measure (RM) and test it with different probabilities and durations. In Geng et al. (2013) work, the authors designed a graph based representation for mission planning of UAVs to carry out a series of tasks. The flying space for these tasks was constrained with the presence of flight-prohibited zones (EPZs) and enemy radar sites. Finally, in Savuran and Karakaya (2015), authors presented a GA for the Capacity Mobile Depot Vehicle Routing Problem, improving the GA process using insertion local search (ILS) and 2-opt local search.

Several criteria can be taken into account in MPPs for multi-UAVs to measure the quality of a solution, such as the fuel consumption, the makespan, or the cost of the mission,

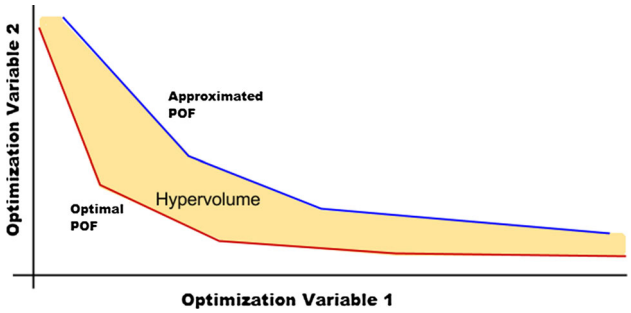

Fig. 1 Hypervolume for two optimization variables. The optimal POF is represented in *red*, the solutions obtained using a specific algorithm are represented in *blue*, and the hypervolume comprised between them is represented in *yellow* (color figure online)

among others. Therefore, it can be interesting to optimize simultaneously different objectives in order to get the best solutions. This type of problems can be solved using multi-objective genetic algorithms (MOGAs) (Zhou et al. 2011; Zitzler et al. 2004) based on Pareto-optimization techniques. The most known approaches are SPEA2 (Deb et al. 2002) and NSGA-II (Zitzler et al. 2001).

Finally, there exist some metrics to evaluate the performance of the algorithm, such as the hypervolume (Zitzler et al. 2007) or the generational distance (Van Veldhuizen et al. 2000). The hypervolume value of a set of solutions with n objective variables consists of the n-dimensional domain comprised between these solutions (the approximated POF) and the optimal POF of the problem (see Fig. 1). When the optimal POF is obtained, the volume comprised between the obtained solutions and the optimal POF is 0, so is the hypervolume. Otherwise, the higher the hypervolume, the worse the approximated POF.

On the other hand, it is also necessary to decide when the algorithm has reached a good POF and stop its execution. There exist several stopping criteria (Wagner et al. 2011) in the literature. One of the most used consists of a comparison function which will stop the execution if the POF remains changeless for a number of generations.

3 Description of the multi-UAV mission planning problem

A UAV mission is typically defined as a number n of ***tasks***, $T = \{t_1, t_2, ...t_n\}$, performed by a team of m ***UAVs***, $U = \{u_1, u_2, ...u_m\}$, at a specific time interval. Each mission should be performed in a specific geographic zone. In addition, in this approach, there exist a number l of ***GCSs***, $G = \{g_1, g_2, ..., g_l\}$, controlling these UAVs. A solution for a mission planning problem should be the assignment of each task to a specific UAV, and each UAV to a specific GCS, ensuring that the mission can be successfully performed.

In Fig. 2, a mission scenario with 7 tasks (represented in green), 5 UAVs, and 3 GCSs is presented. As shown in figure, the zone of the mission could contain some No Flight Zones (NFZs), represented in red. These zones must be avoided in the trajectories of the UAVs during the mission.

In this section, we define the different components of a mission and the computations that must be achieved to obtained the different times related to the assignments of tasks. First, we will define the types and characteristics of tasks, UAVs, and GCSs. Then, we will describe the computations that are performed in the process of task assignments.

3.1 Task description

There exists different kinds of task, such as monitoring a zone or photographing a target in a specific point. These

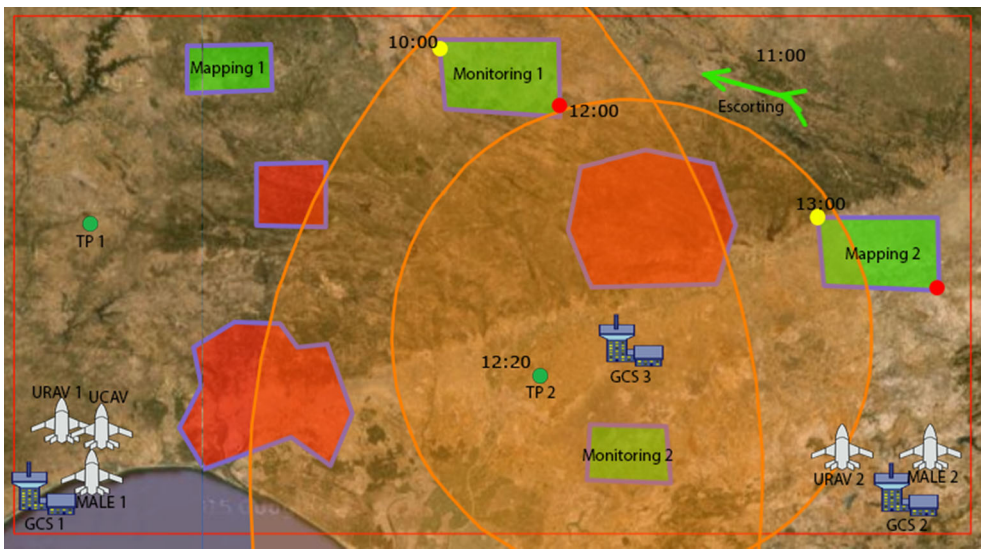

Fig. 2 Mission with 7 tasks (2 of them multi-UAV), 5 UAVs, and 3 GCSs

Table 1 Type of tasks

ID	Name	Description	Multi-UAV	Sensors needed
MON	Monitoring a zone	Fly circling in a zone during a specific time	No	Videotracking EO/IR sensor ISAR radar
ES	Escorting a path	Follow a path	No	Thermal EO/IR sensor SAR radar
TP	Target photographing	Go to a point and take a photograph	No	EO/IR sensor
MAP	Mapping a zone	Travel a zone performing a step & stare pattern	Yes	SAR radar ISAR radar MPR radar

For each task, a description is given, as well as the sensors that could be used to perform it and whether this task can be performed by several UAVs or not

Table 2 Different UAV features considered

Feature		Description	Symbol
Initial position		The position of the UAV at the beginning of the mission	pos_u
Initial fuel		The fuel of the UAV at the beginning of the mission	$fuel(u)$
Available sensors		The sensors contained in the UAV	$sensors(u)$
Range		The maximum distance that the UAV can traverse in the mission	$range(u)$
Autonomy		The maximum time that the UAV can stay in fly	$autonomy(u)$
Cost		The cost per hour of use of the UAV	$cost(u)$
Max. speed		The maximum speed attainable by the UAV	$max_{speed}(u)$
Max. altitude		The highest altitude that the UAV can reach	$max_{alt}(u)$
Max. fuel		The maximum fuel capacity of the UAV tank	$max_{fuel}(u)$
Flight profiles (FP)		One or more profiles that specify at each moment the fly features of the UAV	$fps(u)$
FP	Speed	Speed of the UAV for a flight profile	$speed(fp_u)$
FP	Fuel consumption ratio	Fuel consumption by hour of the UAV for a flight profile	$fuelRatio(fp_u)$
FP	Altitude	Altitude of the UAV when using a route flight profile	$altitude(fp_u)$
FP	Angle	Angle of the UAV when using a climb/descent flight profile	$angle(fp_u)$

tasks are performed using the ***sensors*** available by the UAVs of the mission: electro-optical and infrared sensors (EO/IR sensors), synthetic aperture radars (SARs), inverse synthetic aperture radars (ISARs), and maritime patrol radars (MPRs).

Definition 1 Given a task $t \in T$, the set of sensors that can be used to perform a task is represented as $sensors(t)$.

The different tasks considered in this approach and the sensor or sensors required to perform each task are represented in Table 1.

In addition, each task has a ***time interval***, which could be specified with a start and end time for the task, or just with the task duration. In the last case, the start and end times will be obtained at the planning process.

On the other hand, a mission can have some task dependencies. There exist two types of task dependencies: vehicle dependencies, which impose if two tasks must be performed by the same or by different UAVs, and time dependencies, which constraint the relation of the time intervals of two tasks. These time dependencies are represented using Allen's interval algebra (Allen 1983).

Definition 2 Given two tasks $t_1, t_2 \in T$, vehicle dependency $sameUAV(t_1, t_2)$ constraints both tasks to be performed by the same UAV.

Definition 3 Given two tasks $t_1, t_2 \in T$, vehicle dependency $diffUAV(t_1, t_2)$ constraints both tasks to be performed by different UAVs.

3.2 UAV description

The UAVs of a mission, $u \in U$, have some ***features*** that must be considered when checking whether a plan is correct. These features are presented in Table 2.

On the other hand, during the mission, the UAV will be positioned in different points at each moment.

Definition 4 Given a UAV $u \in U$, the position of u at any time $t \in \mathbb{R}$ is represented as $pos(u, t)$.

Table 3 Different types of UAVs considered and their features

Name	Range (NM)	Autonomy (h)	Cost/h	Max. speed (kt)	Max. altitude (ft)	Max. fuel (kg)	Available sensors
URAV	1000	20	5	120	20000	500	Videotracking and thermal EO/IR sensor
MALE	5000	30	10	250	40000	2500	EO/IR sensor
							MPR radar
HALE	15000	40	15	400	65000	6000	Videotracking EO/IR sensor
							ISAR radar
UCAV	1500	15	25	450	35000	9000	EO/IR sensor
							SAR radar

Table 4 Basic features considered for a GCS

Feature	Description	Symbol
Position	The position of the GCS	pos_g
Max. number of UAVs	The maximum number of UAVs that the GCS can control	$maxNum(g)$
Permitted types	The permitted types of UAVs that the GCS can control	$types(g)$
Coverage	The within range of the GCS	$coverage(g)$

Each type of UAV, $type(u)$, will have different values for these features. In this approach, four basic ***types*** of UAVs have been considered. These are described in Table 3.

3.3 GCS description

To solve multi-UAV missions, it is necessary to use several GCSs controlling the UAVs. Therefore, the problem is multi-GCS, and it should be checked that each UAV is controlled by an appropriate GCS. Every GCS $g \in G$ has some features to be considered that are represented in Table 4.

In Fig. 2, the coverage is represented for each GCS in translucent orange. It can be appreciated that GCS3 has a low coverage, while GCS1 and GCS2 have a higher range.

3.4 Task assignment processes

In Fig. 3, an assignment of a UAV u to two tasks i and j is represented, and the different times computed in the process can be observed. In this assignment process, it is necessary to compute several variables related to time, fuel consumption, and distance traversed, in order to validate that the task can be fulfilled at its time interval using the assigned UAV.

The variables related to time that must be computed in this task assignment process are:

- The departure time when the vehicle starts moving to the task zone. In Fig. 3, it is represented as t_{di} for task i, and t_{dj} for task j.
- The duration of the path between the departure of the UAV and the start of the task. In order to compute this duration, the path flight profile used by the UAV in this path must be set. With the speed (v_i) provided by this profile and the distance from the UAV departure position to the task zone, it is possible to compute the duration of the path ($d_{u \to i}$).
- The start time of the task. This time could be fixed in the definition of the task. If not, it is computed during the assignment process. It is represented as s_i in Fig. 3 for task i.
- The duration of the task (τ_i). This time could be given (e.g., in monitoring tasks) or must be computed (e.g., in mapping tasks). In the second case, it is necessary to know the speed ($\overline{v_i}$) of the UAV in the task performance. This is given by the sensor used by the UAV to perform the task, which provides the optimum speed and altitude for its use.
- The end time of the task. This time could be fixed in the definition of the task. If not, it is computed during the assignment process. It is represented as e_i in Fig. 3 for task i.
- The duration of the loiter. When start and end times of tasks are fixed, it may happen that the time when a UAV finishes a task does not meet the time when the UAV departs for the next task. The difference between these two times is known as the loiter duration for the second task.
- The duration of the return. In order to compute this duration, the return flight profile used by the UAV in this return must be set. With the speed (v_u) provided by this profile and the distance from the zone of the last task of

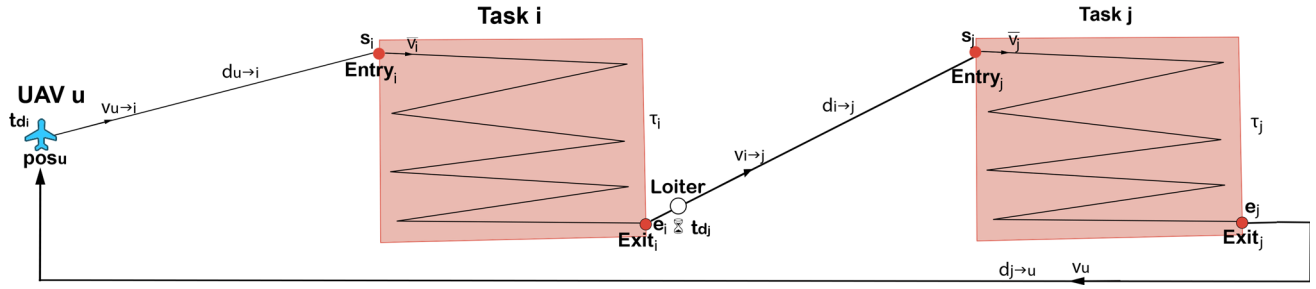

Fig. 3 Example of assignment of a UAV to two tasks. The path to each task, the task performance, the loiter and the return phases are represented, as well as every time point and duration related

the UAV to its initial position, it is possible to compute the duration of the return ($d_{j\to u}$).
- The return time when the UAV has returned to its initial position. It is computed as the sum of the end time of the last task performed by the UAV and the duration of the return.

On the other hand, there are some variables related to fuel consumption that must be computed in this task assignment process. These variables are computed using the previous durations and the fuel consumption ratio given by the flight profile used in each case. Specifically, these variables are: the fuel consumption of the path; the fuel consumption of the task; the fuel consumption of the loiter; and the fuel consumption of the return.

Finally, the variables related to the distance traversed are computed as the sums of distances between the points of the path employed in each case. These variables are: the distance of the path; the distance of the task; the distance of the loiter; and the distance of the return.

Definition 5 Given two points $p_1, p_2 \in \mathbb{R}^3$ in 3D geographic coordinates (longitude, latitude, altitude), we define the distance function $distance(p_1, p_2)$ between them as the 3D distance in WGS84 system.

4 Modeling the MPP as a CSP

In this section, we define how the MPP can be modeled using a CSP. First, we define which are the variables of the CSP and their domain. Then, we explain the different constraints considered for the MPP.

4.1 CSP variables

Looking at the assumptions explained so far in the previous section, the variables of the CSP that we have considered are as follows:

- Assignments ($assign$) of tasks to UAVs. As some tasks could be multi-UAV, these variables are represented as a binary array of size $n \times m$. An assignment $assign[t, u] = 1$ means that task t is assigned to UAV u.
- Orders ($order$), which define the order in which each UAV performs the tasks assigned to it. These variables are necessary when start and end times of tasks are not fixed, and they are represented as an array of size $n \times m$. Their domain is $[-1..n-1]$, where -1 is only assigned when the UAV does not perform the task.
- Assignments of UAVs to GCSs ($gcss$). There are m variables of this type, and their domain is $[-1..l-1]$, where -1 is only assigned when the UAV is not assigned to any task.
- Path Flight Profiles ($fpPath$), setting the flight profile that the vehicle must take for the path performance. These variables are represented as a $n \times m$ array, and their domains are the flight profiles of the UAV in the column: $fpPath[t, u] \in fps(u)$.
- Return Flight Profiles ($fpReturn$), similar to the previous set of variables but for the return path of each UAV. There are m variables of this type, and their domain is the same as the previous variables: $fpReturn[u] \in fps(u)$.
- Sensor used in the task performance ($sensTask$). These variables set the sensor of the vehicle that will be used during the task performance. It will be necessary to consider these variables just in the case that the vehicle performing the task has several sensors that could perform that task. These variables are represented as a $n \times m$ array, and their domains are the sensors of the task and UAV available for that assignment: $sensTask[t, u] \in sensors(t) \cap sensors(u)$.

On the other hand, there are some extra variables that will be computed during the propagation phase of the CSP. These variables are directly related to the variables presented in Sect. 3.4: *departure, durPath, start, durTask, end, durLoiter, durReturn, returnTime, fuelPath, fuelTask, fuelLoiter, fuelReturn, distancePath, distanceTask, distanceLoiter*, and *distanceReturn*.

4.2 CSP constraints

Now, we define the different constraints of the CSP, which consider all the specifications explained so far:

1. Sensor constraints: they check whether a UAV has the sensor needed to perform its assigned tasks. Let $sensors(u)$ denote the sensors available for UAV u and $sensors(t)$ the sensors that could perform the task t then:

$$\forall t \in T \quad \forall u \in U \;\; assign[t,u] = 1 \Rightarrow |sensors(t) \cap sensors(u)| > 0 \qquad (1)$$

2. Order constraints: they assure that the values of the order variables are less than the number of tasks assigned to the UAV performing that task:

$$\forall t \in T \quad \forall u \in U \quad assign[t,u] = 1 \Rightarrow order[t,u] < \sharp\{\tau \in T | assign[\tau,u] = 1\} \qquad (2)$$

and if two tasks are assigned to the same UAV, they have different orders:

$$\forall i,j \in T \quad \forall u \in U \quad assign[i,u] = assign[j,u] = 1 \Rightarrow order[i,u] \neq order[j,u] \qquad (3)$$

3. GCS constraints: they assure that the GCSs assignments are correct. First, it is necessary to assure that the UAVs assigned to the GCS are of a type supported by that GCS (both in initial assignment and during tasks performance):

$$\forall u \in U \quad \forall g \in G \quad gcss[u] = g \Rightarrow type(u) \subset types(g) \qquad (4)$$

Then, a constraint assures that the maximum number of UAVs that a GCS can handle is not overpassed at any moment:

$$\forall g \in G \quad \sharp\{u \in U | gcss[u] = g\} < maxNum(g) \qquad (5)$$

Finally, it is necessary to check that GCS can cover the UAV during the mission:

$$\forall u \in U \quad \forall g \in G \quad gcss[u] = g \Rightarrow \quad \forall t \in \mathbb{R} \quad distance(pos(u,t), pos_g) \leq coverage(g) \qquad (6)$$

4. Temporal constraints: they assure the consistency of all the time variables considered. First, it is necessary to assure that the start time of the task equals the sum of the departure time and the duration for the path:

$$\forall t \in T \quad \forall u \in U \quad assign[t,u] = 1 \Rightarrow departure[t,u] + durPath[t,u] = start[t,u] \qquad (7)$$

and that end time is the sum of the start time and the duration of the task:

$$\forall t \in T \quad \forall u \in U \quad assign[t,u] = 1 \Rightarrow start[t,u] + durTask[t,u] = end[t,u] \qquad (8)$$

Then, the duration of the path is computed as the distance traversed in the path divided by the speed given by the path flight profile:

$$\forall t \in T \quad \forall u \in U \quad assign[t,u] = 1 \Rightarrow durPath[t,u] = \frac{distancePath[t,u]}{speed(fpPath[t,u])} \qquad (9)$$

If tasks have fixed start and end times, then it is necessary to compute the duration of the loiter as the difference between the end of a task and the departure for its consecutive task:

$$\forall i,j \in T \quad \forall u \in U \quad assign[i,u] = assign[j,u] = 1 \quad \wedge \quad order[i,u] = order[j,u] - 1 \Rightarrow durLoiter[j,u] = departure[j,u] - end[i,u] \qquad (10)$$

On the other hand, the duration of the return is computed as the distance traversed in the return path divided by the speed given by the return flight profile:

$$\forall u \in U \quad durReturn[u] = \frac{distanceReturn[u]}{speed(fpReturn[u])} \qquad (11)$$

Once we have computed the return path duration, we can compute the return time as the sum of the end of the last task performed by the UAV and this return duration:

$$\forall t \in T \quad \forall u \in U \quad assign[t,u] = 1 \wedge order[t,u] = \sharp\{\tau \in T | assign[\tau,u] = 1\} - 1 \Rightarrow returnTime[u] = end[t,u] - durReturn[u] \qquad (12)$$

Finally, it is necessary to assure that two tasks that collide in time are never assigned to the same UAV:

$$\forall i,j \in T \quad \forall u \in U \quad assign[i,u] = assign[j,u] = 1 \wedge order[i,u] < order[j,u] \Rightarrow end[i,u] \leq departure[j,u] \qquad (13)$$

5. Dependency Constraints: these constraints are related to the time and vehicle dependencies mentioned before. The time dependency constraints, based on Allen's interval algebra (Allen 1983), for each pair of tasks i and j, assuming $\forall i, j \in T \quad \forall u \in U \quad assign[i,u] = assign[j,u] = 1$, are as follow:

$$i < j \Rightarrow \quad end[i,u] \leq start[j,u] \tag{14}$$

$$i \quad m \quad j \Rightarrow end[i,u] = start[j,u] \tag{15}$$

$$i \quad o \quad j \Rightarrow \begin{cases} start[i,u] \leq start[j,u] \\ end[i,u] \geq start[j,u] \\ end[i,u] \leq end[j,u] \end{cases} \tag{16}$$

$$i \quad s \quad j \Rightarrow \begin{cases} start[i,u] = start[j,u] \\ end[i,u] \leq end[j,u] \end{cases} \tag{17}$$

$$i \quad d \quad j \Rightarrow \begin{cases} start[i,u] \geq start[j,u] \\ end[i,u] \leq end[j,u] \end{cases} \tag{18}$$

$$i \quad f \quad j \Rightarrow \begin{cases} start[i,u] \geq start[j,u] \\ end[i,u] = end[j,u] \end{cases} \tag{19}$$

$$i = j \Rightarrow \begin{cases} start[i,u] = start[j,u] \\ end[i,u] = end[j,u] \end{cases} \tag{20}$$

On the other hand, vehicle dependencies imply the following constraints:

$$\forall i, j \in T \quad sameUAV(i,j) \Rightarrow \forall u \in U \quad assign[i,u] = assign[j,u] \tag{21}$$

$$\forall i, j \in T \quad diffUAV(i,j) \Rightarrow \forall u \in U \quad assign[i,u] \neq assign[j,u] \tag{22}$$

6. Autonomy constraints: they assure that the total flight time for each vehicle is less than its autonomy:

$$\forall u \in U \quad flightTime[u] = \sum_{\substack{t \in T \\ assign[t]=u}} (durPath[t] + durTask[t] + durLoiter[u]) + durReturn[u] < autonomy(u) \tag{23}$$

7. Distance constraints: they assure that the distance traversed by each vehicle is less than its range:

$$\forall u \in U \quad distance[u] = \sum_{\substack{t \in T \\ assign[t]=u}} (distancePath[t] + distanceTask[t] + distanceLoiter[u]) + distanceReturn[u] < range(u) \tag{24}$$

To compute these distances, we have used GeographicLb[1] for the computation of distance and points in geographic coordinates; and Theta* (Nash et al. 2007) to perform a path between these points avoiding No Flight Zones (NFZ) and terrain obstacles. The elevation of the terrain has been read from DTED maps using GDAL.[2]

8. Fuel constraints: they assure that the fuel consumed by each vehicle is less than its initial fuel $fuel_u$:

$$\forall u \in U fuel[u] = \sum_{\substack{t \in T \\ assign[t]=u}} (fuelPath[t] + fuelTask[t] + fuelLoiter[t]) + fuelReturn[u] < fuel(u) \tag{25}$$

Each one of these fuel consumptions is computed as the product of its associated duration and fuel consumption ratio. For example, the fuel consumption for the path is computed multiplying the fuel consumption ratio given by the path flight profile and the duration of the path:

$$\forall t \in T \quad \forall u \in U \quad assign[t,u] = 1 \Rightarrow fuelPath[t,u] = durPath[t,u] \times fuelRatio(fpPath[t,u]) \tag{26}$$

5 MOGA-CSP algorithm for multi-UAV mission planning problems

Given the large amount of solutions that the problem can generate and the huge amount of constraints involved in the search of solutions, we have decided to use a hybrid approach based on MOGAs and CSPs to solve MPPs. In this new approach, the constraints of the problem have been applied as penalty function in the evaluation phase of the genetic algorithm. This section describes this algorithm, including the encoding, the fitness function designed, and the genetic operators implemented.

5.1 Encoding

To encode the multi-UAV MPP, a representation based on six different alleles has been designed (see Fig. 4). Each allele

[1] http://geographiclib.sourceforge.net/.

[2] http://www.gdal.org/.

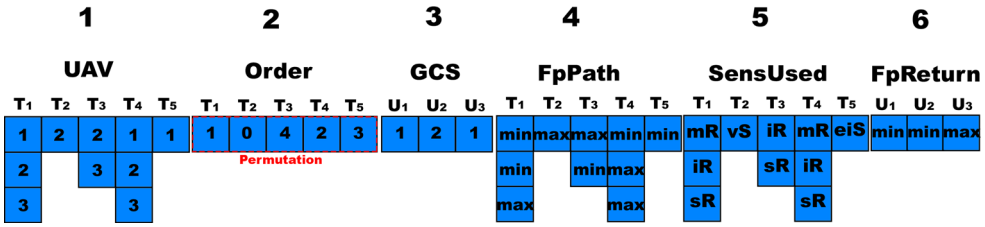

Fig. 4 Example of an individual that represents a possible solution for a problem with 5 tasks, 3 UAVs, and 2 GCSs

Table 5 Optimization variables used in the fitness function by MOGA-CSP algorithm

Variable	Description	Formula
N_{uavs}	The number of UAVs used in the mission	$\sharp\{u \in U \mid \exists t \in T \quad assign[t] = u\}$
$totalFlightTime$	The total flight time of all the UAVs during the mission	$\sum_{u \in U} flightTime[u]$
$totalFuel$	The total fuel consumed by all the UAVs during the mission	$\sum_{u \in U} fuelConsumed[u]$
$totalDistance$	The total distance traversed by all the UAVs in the mission	$\sum_{u \in U} distance[u]$
$totalCost$	The total cost of the mission, computed as the sum of the individual cost of each UAV	$\sum_{u \in U} cost(u) \times flightTime[u]$
$makespan$	The time when the mission ends (all UAVs have returned)	$\max_{u \in U} returnTime[u]$

is used to encode the features that have been described in previous sections representing a complete solution that will be optimized by MOGA algorithm. Next, a short description for each allele is given:

1. UAVs assigned to each task. If the T_i task is multi-UAV, then this cell contains a vector representing the different UAVs assigned to this task.
2. Permutation of the task orders. These values indicate the absolute order of the tasks. It is only used if there are several tasks assigned to the same UAV and some of them do not have the start and end times fixed.
3. GCSs controlling each UAV.
4. Flight profiles used for each UAV to each assigned task. As in the first allele, some of the cells could contain a vector if the corresponding task is performed by several UAVs.
5. Sensors used for the task performance by each UAV.
6. Flight Profiles used by each UAV to return to the base.

An example of this representation is shown in Fig. 4. Firstly, this figure shows a mission with 5 tasks. Assuming that there is not any task with start and end time fixed, it is necessary to use the permutation allele for the task orders (2). Using together, this allele and the allele of UAV assignments (1), we have that UAV 1 performs tasks 1, 4, and 5 in this order; UAV 2 performs tasks 2, 1, 4, and 3; and UAV 3 performs tasks 1, 4, and 3. On the other hand, according to allele of GCCs information (3), we have that UAVs 1 and 3 are controlled by GCS 1, while UAV2 is controlled by GCS 2. Furthermore, in the allele of Flight profiles per task (4), we can see that UAV 1 uses minimum consumption flight profile for all its assigned tasks; UAV 2 uses minimum consumption profile for task 1, and maximum speed profile for the rest of tasks, and UAV 3 uses minimum consumption profile for task 3, while maximum speed profile for the rest of tasks. Regarding the sensors used (5), it can be seen that task 1 is performed by UAV 1 using MPR (mR) sensor, while UAV 2 uses an ISAR (iR), and UAV 3 uses a SAR (sR); task 2 is performed using EO/IR sensor (eiS), etc. Finally, the last allele (6) represents that UAVs 1 and 2 use minimum consumption profile for their return path, while UAV 3 uses maximum speed profile.

A key point in this representation is that only a valid sensor to perform the task assigned could be used for the allele of sensors used per task. With this, the algorithm is avoiding some invalid solutions due to sensor constraints.

5.2 Fitness function

Evaluation is computed in terms of a fitness function composed by two check steps. First, for a given solution, it handles that all constraints are fulfilled. If not, it acts as a penalty function, giving the solution the worst possible value so it would not be evolved in future generations. If all constraints are fulfilled, the fitness function works as a multi-objective function minimizing the objectives of the problem. For this purpose, we have considered the optimization variables described in Table 5.

The multi-objective fitness function compares the solution evaluated with the stored solutions in order to obtain

the Pareto-optimality frontier (POF) based on the NSGA-II approach (Deb et al. 2002).

5.3 Algorithm

In this new approach, as shown in Algorithm 1, after evaluating the individuals of the population with the fitness previously explained (Line 8), a N elitist selection is performed. It means that a number N of best individuals in the population is retained (Line 9). Then, a roulette wheel selection over these N individuals (Line 12) selects those that will be applied the genetic operators.

Algorithm 1: Hybrid MOGA-CSP algorithm for Mission Planning Problems

Input: A mission $M = (T, U, G)$ where T is a set of tasks to perform denoted by $\{t_1, \ldots, t_n\}$, U is a set of UAVs denoted by $\{u_1, \ldots, u_m\}$ and G is a set of GCSs denoted by $\{g_1, \ldots, g_l\}$. And positive numbers $generations$, $population$, μ, λ, $mutprobability$ and $stopGeneration$

Output: POF obtained with best solutions

1 $S \leftarrow$ randomly generated set of $population$ of p chromosomes with 6 alleles representing the tasks assignments to UAVs, the orders, the UAVs assignments to GCSs, the path flight profiles, the sensors used and the return flight profiles
2 $i \leftarrow 1$
3 $convergence \leftarrow 0$
4 $pof \leftarrow createPOF(S)$
5 **while** $i \leq generations \wedge convergence < stopGenerations$ **do**
6 $F \leftarrow \emptyset$
7 **for** $j \leftarrow 1$ ***to*** p **do**
8 $F \leftarrow Fitness(S_j)$
9 $Sbest \leftarrow SelectNBest(\mu, F)$
10 $newS \leftarrow Sbest$
11 **for** $j \leftarrow \mu$ ***to*** λ **do**
12 $p1, p2 \leftarrow RouleWheelSelection(Sbest)$
13 $i1, i2 \leftarrow Crossover(p1, p2)$
14 $i1 \leftarrow Mutation(i1, mutprobability)$
15 $i2 \leftarrow Mutation(i2, mutprobability)$
16 $newS \leftarrow newS \cup \{i1, i2\}$
17 $NSGA2_{UpdatePopulation}(S, newS)$
18 $newpof \leftarrow createPOF(S)$
19 **if** $newpof = pof$ **then**
20 $convergence \leftarrow convergence + 1$
21 $pof = newpof$
22 $i \leftarrow i + 1$
23 **return** pof

Next, we use a proper crossover operator (Line 13) to combine the chromosomes of each pair of parents to generate a new pair of children. This operator consists of a specific crossover operation for each of the alleles of the representation. The first allele performs a 2-point crossover, and the same cross points used for this allele are reused for the fourth and fifth allele in order to maintain the size for multi-UAV tasks and the consistency of the sensors used. On the other hand, in the second allele, as it is a permutation, is applied a partially matched crossover (PMX). This passes a chunk of values from one parent to the other and then performs a replacement of the invalid values of the new child based on its previous parent. Finally, in the third and sixth alleles are applied another 2-point crossover (with different points than the previous). Figure 5 shows an example of this crossover operation, where the first, fourth, and fifth allele have selected points 2 and 4 for the 2-point crossover. In the second allele, a chunk composed of tasks $T_2..T_3$ has been selected for the PMX crossover, and finally, the third and sixth allele have selected points 1 and 2 for the 2-point crossover.

Once the new pair of individuals has been generated from crossover operation, a mutation operator (Line 14) will be applied to them depending on a probability P_m (usually low, $\sim 5\%$). This genetic operator helps to avoid that the obtained solutions stagnate at local minimums. This mutation operator is designed to perform a uniform mutation over the same genes for the first, fourth, and fifth allele in order to maintain the size of multi-UAV tasks and avoid invalid solutions accomplishing sensor constraints. On the other hand, the second allele is applied an insert mutation, which will select two random positions from the permutation and move the second one next to the first one. Finally, the third and sixth allele are updated using another uniform mutation. Figure 6 presents an example of this mutation, where T_4 has been mutated for the first, fourth, and fifth allele, the insert mutation has moved the value of T_4 next to T_1, and the third and sixth allele have mutated the value of U_1.

Finally, after the population is updated by NSGA-II (Line 16), the stopping criteria designed for this algorithm compare the POF obtained so far in each generation with the POF from the previous generation (Line 18). If this POF remains unchangeable for a number of generations, then the algorithm will stop and return this POF.

6 Experiments

In this section, we explain the experiments carried out to test the functionality of the new MOGA-CSP approach for MPP. For this purpose, we have designed several missions with different configurations of tasks, UAVs, GCSs, and NFZs in order to check the different characteristics of the model. These datasets are described in Table 6 which shows the characteristics of the model that are checked for each one.

The first experiment shows the results obtained when the different objectives are optimized individually and by pairs, and compare it with their optimization all together. From this experiment, we will obtain which variables are the most appropriate to use for this problem for the MOGA-CSP.

Finally, all datasets are tested using the objective variables obtained in the previous experiment. In order to evaluate

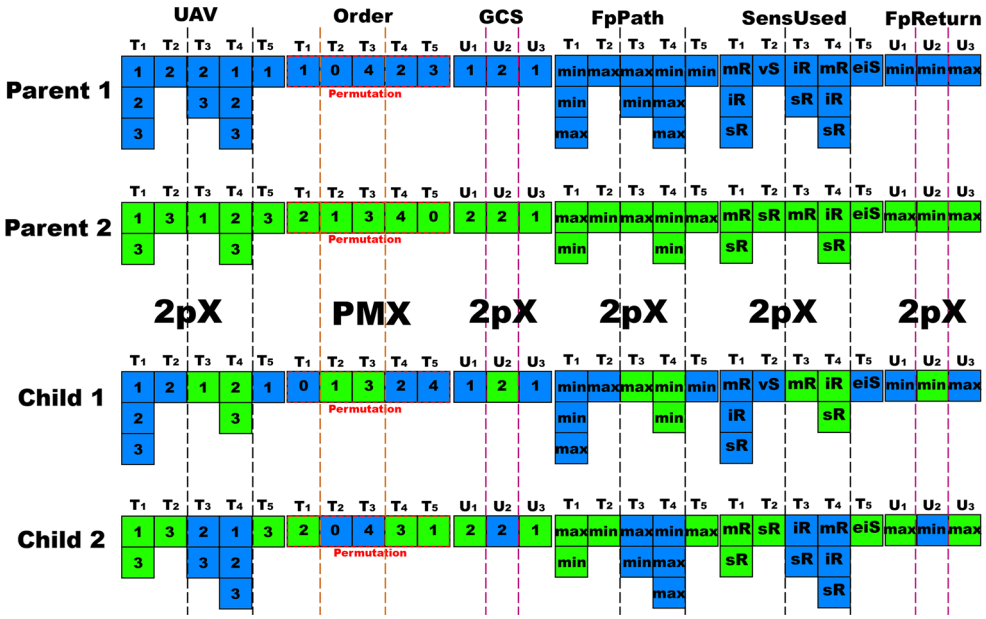

Fig. 5 Example of crossover of two parents with 5 tasks, 3 UAVs, and 2 GCSs. Each allele is performed a different type of crossover: UAV, FpPath, and SensUsed are performed a 2-point crossover; Order is applied a PMX crossover, and GCS and FpReturn are applied another 2-point crossover

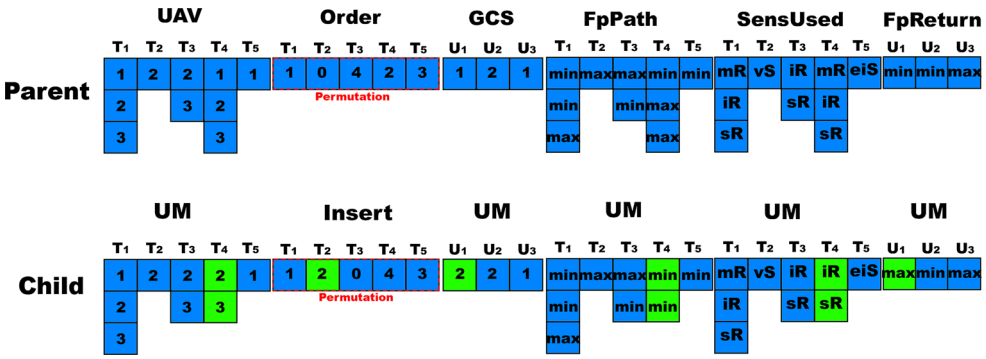

Fig. 6 Example of mutation for an individual with 5 tasks, 3 UAVs, and 2 GCSs. Each allele is performed a different type of mutation: UAV, FpPath, and SensUsed are performed an uniform; Order is applied an insert mutation, and GCS and FpReturn are applied another uniform mutation

the performance of the algorithm, the hypervolume metric is calculated. To apply this metric, it is necessary to compute the optimal POF using the MOBB algorithm for each dataset. For this purpose, MOBB algorithm provided from Rodriguez-Fernandez et. al. approach (2015a) is applied. Then, the solutions returned by the MOGA-CSP are compared with the MOBB results to analyze their optimality.

6.1 Experimental setup

Table 7 shows the parameters used throughout the experimental phase. $\mu + \lambda$ represents the selection criteria used, where λ is the number of offspring (population size), and μ the elitism size (i.e., the number of the best parents that survive from current generation to the next). Each problem is run 10 times, and the best of these 10 executions is selected.

6.2 Comparative assessment of objective variables

There are several parameters which can be used to measure the quality of a solution, such as the fuel consumption, the makespan, and the cost of the mission. As shown in section 5.2, this new algorithm considers 6 different optimization variables: number of UAVs, total flight time, total fuel consumption, total distance traversed, total cost, and the makespan. A comparative assessment of these variables is carried out in order to tune up the fitness function designed for the new algorithm. For this purpose, the mission from dataset 1 has been chosen to study the behavior of the algorithm according to the variables which are being optimized.

In these experiments, the average of each optimization variable is computed when obtaining several solutions in an execution. Then, in order to compare different executions (of

Table 6 Features of the different datasets designed

Dataset		Tasks	UAVs	GCSs	NFZs	Task times		
Id.	Description					Fixed	Unfixed	Deps.
1	Simple mission with fixed times	2 MON	1 HALE	1	0	6	0	0
			1 MALE					
		2 ES	1 UCAV					
		2 TP	1 URAV					
2	Path avoidance	2 MON	1 HALE	1	1	6	0	0
			1 MALE					
		2 ES	1 UCAV					
		2 TP	1 URAV					
3	Multi-UAV tasks	3 MAP	1 HALE	1	0	0	3	0
			1 MALE					
4a	Multi-GCS with fixed times	2 MON	1 HALE	2	2	6	0	0
			1 MALE					
		2 ES	1 UCAV					
		2 TP	2 URAV					
4b	Multi-GCS with half fixed times	2 MON	1 HALE	2	2	3	3	0
			1 MALE					
		2 ES	1 UCAV					
		2 TP	2 URAV					
4c	Multi-GCS with half fixed times and dependencies	2 MON	1 HALE	2	2	3	3	1
			1 MALE					
		2 ES	1 UCAV					
		2 TP	2 URAV					
4d	Multi-GCS with unfixed times	2 MON	1 HALE	2	2	0	6	0
			1 MALE					
		2 ES	1 UCAV					
		2 TP	2 URAV					
4e	Multi-GCS with unfixed times and dependencies	2 MON	1 HALE	2	2	0	6	3
			1 MALE					
		2 ES	1 UCAV					
		2 TP	2 URAV					
5	Complex mission with all assets at a time	2 MON		3	3	4	3	1
			2 MALE					
		1 ES	1 UCAV					
		2 TP	2 URAV					
		2 MAP						

different optimization variables), a weighted average over the values of the optimization variables is employed as rating value:

$$Rating(sol) = \sum_{v \in OptVar} \frac{v(sol) - min(v)}{max(v) - min(v)} \tag{27}$$

First, each variable is optimized individually. The results obtained for each optimization variable are shown in Table 8. Analyzing the results, it can be noticed that there are many different optimal solutions for the variables number of UAVs and makespan. In fact, none of them got to converge because new solutions were still being obtained at generation 300.

Table 7 Experimental setup for the MOGA-CSP

Mutation probability	0.1
Generations	300
Population size	1000
Selection criteria ($\mu + \lambda$)	$100 + 1000$
Stopping criteria generations	10

Table 8 Comparative assessment of optimization variables using each variable individually

Variable	N. Sol.	N. Gen.	UAVs	Fuel (L)	F. Time (h)	Dist. (NM)	Cost	Mak. (h)	Rating
Distance	1	**13**	4	754.12	3.02	**939.09**	**47.66**	3.64	**1.611**
F. Time	4	14	4	771.08	**3.02**	949.74	**47.66**	3.64	2.031
Cost	4	14	4	771.08	**3.02**	949.74	**47.66**	3.64	2.031
Fuel	1	**13**	4	**751.57**	3.24	939.17	49.69	3.67	2.828
Makespan	1000	*300*	**3**	810.06	3.21	1021.35	52.71	**3.59**	3.355
UAVs	1000	*300*	**3**	798.44	3.52	1018.49	52.84	3.64	4.332

The values of the optimization variables presented here are the average of their values in all the solutions obtained. The best result for each optimization variable is marked in bold

Regarding the rest of variables, it can be seen that optimizing the **cost** or flight time gave the same results. However, the fuel consumption and the distance traversed gave different results. In the case of the distance, it can be appreciated that it also got the best optimization value for cost, and nearly optimal value for flight time (with a difference of 10^{-6} respect to the best value). In fact, optimizing the distance obtained the best rating, so it is a potential candidate to use in the fitness function of the MOGA-CSP approach.

Afterward, the MOGA-CSP algorithm has been run optimizing each pair of the previous variables. The results obtained are shown in Table 9.

In these results, it is appreciable that the two best combinations obtained according to the rating (i.e., the optimization of the distance and the flight time, and the distance and the cost) obtained good results for four of the variables (the cost, the distance, the fuel, and the flight time), but poor results for the rest (the makespan and the number of UAVs). On the other hand, the third and fourth best combinations according to the rating (i.e., the optimization of the distance and the makespan, and the distance and the number of UAVs) obtained medium results for all the variables.

So, in order to find good solutions optimizing all the variables, the variables selected to optimize are the distance and the makespan, which gave medium values for all the optimization variables. Other possibility would have been using the number of UAVs instead of the makespan, but as the makespan is a float value, it will be better for optimizing problems with very similar solutions (e.g., all the best solutions when optimizing any variable uses 1 UAV because the mission can be performed with just one and other available UAVs are far away from the tasks of the mission).

Finally, the MOGA-CSP algorithm is executed with this problem trying to optimize all the six objective variables, and the results obtained are shown in Table 10. As can be seen, the average obtained here for all objective variables are worst than the ones obtained in the previously proposed combination of distance and makespan, as well as the rating value. This corroborates the assumption of selecting this combination, which will be used in the fitness of the MOGA-CSP in the next experiment when solving the different datasets.

6.3 Evaluation of the algorithm results

Once the fitness function of the algorithm has been tuned up, and the better optimization variables (distance and makespan) have been selected, the MOGA-CSP algorithm is tested using them for each dataset described in Table 6. To evaluate the results obtained, the real POF of each dataset is computed using MOBB algorithm. Then, it is compared with the solutions provided by the new MOGA-CSP approach using the hypervolume metric. As was mentioned in Sect. 2.3, when the hypervolume is 0, the obtained solutions are optimal. On the other hand, as this value increases, the result obtained distances from the optimal result. The tests have been run in a Haswell 2.3 GHz with 20 cores and 256GB DDR4 RAM. Table 11 shows the results obtained from this experiment. This table presents the hypervolumes obtained, the number of generations needed to converge for each dataset, and the runtime spent for each execution.

As can be appreciated, the datasets with 1 GCS (from 1 to 3) converge very fast, independently of the NFZs needed to avoid or the multi-UAV tasks. On the other hand, the datasets with 2 GCS (from 4 to 5) converge near generation 50. We observed that it is easier to obtain convergence for the problems with more fixed times tasks and harder for the problems with more unfixed tasks. The dataset 4d did not get a hypervolume so good as the others datasets. Figure 7 represents the distance vs. makespan POFs for the solutions obtained with both algorithms (MOGA-CSP and MOBB) in this problem. There, it is appreciable that the MOGA-CSP approach did not get to obtain the best solution optimizing the distance (left of the POF), and this made the hypervolume (represented in yellow) higher in this problem.

Finally, the complex solution with 7 tasks, 5 UAVs, and 3 GCSs, i.e., mission 5, got to converge at generation 122, and its hypervolume resulted quite good, very close to the optimum POF.

Table 9 Comparative assessment of optimization variables using each pair of variables

Variables	N. Sol.	N. Gen.	UAVs	Fuel (L)	F. Time (h)	Dist. (NM)	Cost	Mak. (h)	Rating
Distance F. Time	3	16	4	757.57	3.02	939.32	47.66	3.64	**1.396**
Distance Cost	4	15	4	759.39	3.02	941.50	47.66	3.64	1.450
Distance Makespan	2	15	3.5	769.57	3.10	965.57	49.48	3.61	1.681
Distance UAVs	2	17	3.5	769.57	3.10	965.57	49.48	3.61	1.681
F. Time Cost	4	15	4	771.08	**3.02**	949.74	**47.66**	3.64	1.729
Cost Fuel	13	14	4	753.50	3.11	939.17	48.47	3.66	1.741
Distance Fuel	20	14	4	**752.67**	3.12	**939.12**	48.50	3.66	1.745
F. Time Fuel	12	15	4	753.48	3.11	939.18	48.58	3.66	1.784
Fuel Makespan	4	15	3.5	767.52	3.25	965.62	50.76	3.63	2.179
Cost Makespan	9	18	3.11	805.22	3.16	1004.30	50.89	3.59	2.529
Cost UAVs	9	16	3.11	805.22	3.16	1004.30	50.89	3.59	2.529
Makespan UAVs	*1000*	*300*	**3**	808.92	3.21	1020.69	52.65	**3.58**	3.046
F. Time Makespan	18	16	3.11	815.64	3.12	1027.05	51.97	3.59	3.121
F. Time UAVs	18	20	3.11	815.64	3.12	1027.05	51.97	3.59	3.121
Fuel UAVs	2	14	3.5	757.53	3.87	977.82	51.96	3.75	3.879

The values of the optimization variables presented here are the average of their values in all the solutions obtained. The best result for each optimization variable is marked in bold

Table 10 Comparative assessment of optimization variables using all variables

Variable	N. Sol.	N. Gen.	UAVs	Fuel (L)	F. Time (h)	Dist. (NM)	Cost	Mak. (h)	Rating
All	46	20	3.52	772.76	3.15	970.13	50.13	3.62	2.077

The values of the optimization variables presented here are the average of their values in all the solutions obtained

In conclusion, the MOGA-CSP approach with the distance and the makespan used as optimization variables approximates quite well the POF in most cases. As the problem becomes more complex, specially in number of GCSs, the algorithm needs more generations to converge.

7 Discussion

In this paper, the multi-UAV mission planning problem has been presented, with a special consideration as a multi-GCS problem. This problem involves a complex characteristic management during the process of task assignment, such as task dependencies, NFZ avoidance, and time computations.

The problem has been modeled as a temporal constraint satisfaction problem, where six sets of variables have been considered for the CSP: the task assignments, the orders, the GCS assignments, the path flight profiles, the return flight profiles, and the sensors used. On the other hand, a wide range of constraints have been presented, including GCS constraints, temporal constraints, dependency constraints, autonomy constraints, or distance constraints among others.

Table 11 Hypervolume, number of generations needed for convergence and runtime of the MOGA-CSP solver for the 9 MPP datasets provided

Problem	Hypervolume	Generations	Runtime
1	0	15	3 min 5 s
2	0	14	4 min 33 s
3	0	12	4 min 12 s
4a	0	39	11 min 56 s
4b	0	43	13 min 27 s
4c	0	56	15 min 42 s
4d	0.99	57	16 min 32 s
4e	0	62	18 min 13 s
5	0.01	122	26 min 43 s

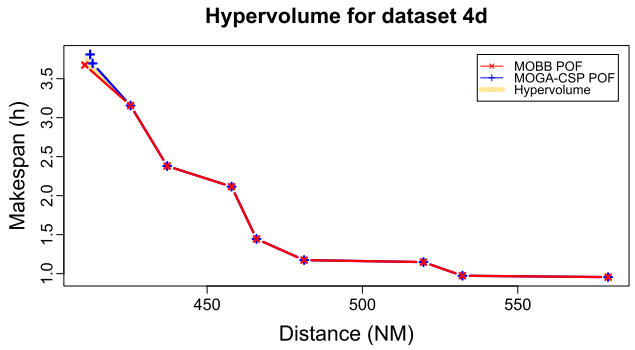

Fig. 7 Hypervolume for the solutions obtained with MOGA-CSP optimizing the distance and the makespan in dataset 4d

Finally, a hybrid MOGA-CSP approach has been presented for solving the MPP problem. This approach uses a fitness function divided in two phases. Firstly, a penalty function uses the CSP to check whether the solutions are valid. Then, a multi-objective function tries to approach the Pareto-optimal frontier of the problem minimizing the optimization variables (number of UAVs employed, makespan of the mission, total fuel consumption, etc.). In addition, the crossover and mutation operators, and also the stopping criteria have been specifically designed and implemented for this approach.

The experiments presented have been performed using varied datasets of different complexity. First, a comparative assessment of the optimization variables has been performed in order to tune up the fitness function designed. The results show that the best combination in order to obtain good results for all the variables is optimizing the distance and the makespan.

Afterward, the MOGA-CSP approach using the previous combination of variables in the fitness function has been tested with all the datasets designed. Analyzing the experimental results, it can be seen that the MOGA-CSP algorithm obtains good results for all the proposed datasets, converging to the optimal POF in most of them. Nevertheless, as the problems become more complex, the MOGA-CSP approach needs more generations to reach an optimal or near-optimal solution. In order to outperform these results, it can be interesting to extend the new approach applying some constraints in the operators of the GA in order to avoid some invalid solutions before the CSP check.

In future works, the approach will be compared using other multi-objective algorithms, such as SPEA2, in order to find the best performing combination for this approach. In order to find an optimum configuration, a meta-evolutionary algorithm will be implemented and used to optimize the different parameters of the approach. On the other hand, this problem will be extended adding a decision-making layer that will interact with a UAV mission operator. This new feature will allow the operator to decide which variables must be optimized and which of the obtained solutions are the most suitable. Finally, this work will be used as a starting point for a further real-time approach, which will be developed in order to support onboard replanning.

Acknowledgements The authors would like to acknowledge the support obtained from Airbus Defence & Space, specially from Savier Open Innovation project members: José Insenser, César Castro, Gemma Blasco, and Inés Moreno. This study was funded by Spanish Ministry of Science and Education and Competitivity and European Regional Development Fund FEDER (TIN2014-56494-C4-4-P), Comunidad Autónoma de Madrid (CIBERDINE S2013/ICE-3095), and Airbus Defence & Space under Savier Project (FUAM-076915).

Compliance with ethical standards

Conflict of interest The Authors: Cristian Ramirez-Atencia, Gema Bello-Orgaz, Maria D. R-Moreno, and David Camacho declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Adolf F, Andert F (2010) Onboard mission management for a VTOL UAV using sequence and supervisory control, chap. 19, InTech, pp 301–316

Allen JF (1983) Maintaining knowledge about temporal intervals. Commun ACM 26(11):832–843

Barták R (1999) Constraint programming: In pursuit of the holy grail. In: Week of Doctoral Students, pp 555–564

Bello-Orgaz G, Camacho D (2014) Evolutionary clustering algorithm for community detection using graph-based information. In: Evolutionary Computation (CEC), 2014 IEEE congress on, pp 930–937

Bello-Orgaz G, Ramirez-Atencia C, Fradera-Gil J, Camacho D (2015) Gampp: genetic algorithm for uav mission planning problems. In: Intelligent distributed computing IX. Springer International Publishing, pp 167–176

Bessière C (2006) Constraint propagation. Found Artif Intell 2:29–83

Bessière C, Meseguer P, Freuder E, Larrosa J (1999) On forward checking for non-binary constraint satisfaction. In: Jaffar J (ed) International conference on principles and practice of constraint programming, lecture notes in computer science, vol 1713. Springer, Berlin, pp 88–102

Bethke B, Valenti M, How JP (2008) UAV task assignment. IEEE Robot Autom Mag 15(1):39–44

Bin X, Min W, Yanming L, Yu F (2010) Improved genetic algorithm research for route optimization of logistic distribution. In: Proceedings of the 2010 international conference on computational and information sciences, ICCIS '10, IEEE Computer Society, Washington, pp 1087–1090

Camacho D, Fernandez F, Rodelgo MA (2006) Roboskeleton: an architecture for coordinating robot soccer agents. Eng Appl Artif Intell 19(2):179–188

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. Evol Comput 6(2):182–197

Diaz D, Cesta A, Oddi A, Rasconi R, R-Moreno MD (2013) Efficient energy management for autonomous control in rover missions. IEEE Comput Intell Mag 8(4):12–24 Special Issue on Computational Intelligence for Space Systems and Operations

Doherty P, Kvarnström J, Heintz F (2009) A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems. Auton Agent Multi-agent Syst 19(3):332–377

Fabiani P, Fuertes V, Piquereau A, Mampey R, Teichteil-Konigsbuch F (2007) Autonomous flight and navigation of VTOL UAVs: from autonomy demonstrations to out-of-sight flights. Aerosp Sci Technol 11(2–3):183–193

Fonseca C, Fleming P (1998) Multiobjective optimization and multiple constraint handling with evolutionary algorithms. A unified formulation. IEEE Trans Syst Man Cybern- A: Syst Hum 28(1):26–37

Geng L, Zhang Y, Wang J, Fuh J, Teo S (2013) Cooperative task planning for multiple autonomous uavs with graph representation and genetic algorithm. In: Control and automation (ICCA), 10th IEEE international conference on, pp 394–399. IEEE

Gonzalez-Pardo A, Camacho D (2013) A new CSP graph-based representation for ant colony optimization. In: IEEE conference on evolutionary computation (CEC 2013), vol 1, pp 689–696. IEEE

Gonzalez-Pardo A, Palero F, Camacho D (2014) An empirical study on collective intelligence algorithms for video games problem-solving. Comput Inform 34(1):233–253

Hao X, Liu J (2015) A multiagent evolutionary algorithm with direct and indirect combined representation for constraint satisfaction problems. Soft Comput 1–13. doi:10.1007/s00500-015-1815-1

Holland JH (1992) Adaptation in natural and artificial systems. MIT Press, Cambridge

Kendoul F (2012) Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. J Field Robot 29(2):315–378

Kuter U, Sirin E, Parsia B, Nau D, Hendlerb J (2005) Information gathering during planning for web service composition. Web semantics: science, services and agents on the World Wide Web 3(2):183–205. Selected papers from the international semantic web conference, 2004 ISWC, 2004 3rd international semantic web conference, 2004

Kvarnström J, Doherty P (2010) Automated planning for collaborative UAV systems. In: Control automation robotics and vision, pp 1078–1085. IEEE

Lee Yi, Kim YG (2008) Comparison of fuzzy implication operators by means of fuzzy relational products used for intelligent local path-planning of auvs. Soft Comput 13(6):535–549

Menendez HD, Barrero DF, Camacho D (2014) A co-evolutionary multi-objective approach for a k-adaptive graph-based clustering algorithm. In: Evolutionary computation (CEC), 2014 IEEE congress on, pp 2724–2731. IEEE

Merino L, Caballero F, Martínez-de Dios JR, Ferruz J, Ollero A (2006) A cooperative perception system for multiple uavs: application to automatic detection of forest fires. J Field Robot 23(3–4):165–184

Mouhoub M (2002) Solving temporal constraints in real time and in a dynamic environment. Tech. Rep. WS-02-17, AAAI

Mouhoub M (2004) Reasoning with numeric and symbolic time information. Artif Intell Rev 21(1):25–56

Nash A, Daniel K, Koenig S, Felner A (2007) Theta*: any-angle path planning on grids. In: Proceedings of the National Conference on Artificial Intelligence, vol 22. AAAI Press, MIT Press, Menlo Park, Cambridge, London, pp 1177–1183

Pascarella D, Venticinque S, Aversa R, Mattei M, Blasi L (2015) Parallel and distributed computing for uavs trajectory planning. J Ambient Intell Humaniz Comput 6(6):773–782

Pereira E, Bencatel R, Correia J, Félix L, Gonçalves G, Morgado J, Sousa J (2009) Unmanned air vehicles for coastal and environmental research. J Coast Res II(56):1557–1561

Ramirez-Atencia C, Bello-Orgaz G, R-Moreno MD, Camacho D (2015a) A hybrid MOGA-CSP for multi-UAV mission planning. In: Proceedings of the companion publication of the 2015 on genetic and evolutionary computation conference. ACM, pp 1205–1208

Ramirez-Atencia C, Bello-Orgaz G, R-Moreno MD, Camacho D (2015b) Performance evaluation of multi-UAV cooperative mission planning models. In: Computational collective intelligence. Springer International Publishing, pp 203–212

Rasmussen S, Shima T (2006) Branch and bound tree search for assigning cooperating uavs to multiple tasks. In: American control conference. IEEE, pp 6–14

Rodríguez-Fernández V, Menéndez HD, Camacho D (2015a) Automatic profile generation for uav operators using a simulation-based training environment. Prog Artif Intell 5(1):37–46

Rodriguez-Fernandez V, Ramirez-Atencia C, Camacho D (2015b) A multi-uav mission planning videogame-based framework for player analysis. In: Evolutionary computation (CEC), 2015 IEEE congress on. IEEE, pp 1490–1497

Rollon E, Larrosa J (2006) Bucket elimination for multiobjective optimization problems. J Heuristics 12(4–5):307–328

Rollon E, Larrosa J (2007) Multi-objective Russian doll search. In: Proceedings of the national conference on artificial intelligence, vol 22. AAAI Press, MIT Press, Menlo Park, Cambridge, London, pp 249–254

Savuran H, Karakaya M (2015) Efficient route planning for an unmanned air vehicle deployed on a moving carrier. Soft Comput 20(7):2905–2920

Schwalb E, Vila L (1998) Temporal constraints: a survey. Constraints 3(2–3):129–149

Soliday SW, et al. (1999) A genetic algorithm model for mission planning and dynamic resource allocation of airborne sensors. In: Proceedings, 1999 IRIS National Symposium on Sensor and Data Fusion. Citeseer

Tang L, Zhu C, Zhang W, Liu Z (2011) Robust mission planning based on nested genetic algorithm. In: Advanced computational intelligence (IWACI), 2011 fourth international workshop on, pp 45–49. IEEE. doi:10.1109/IWACI.2011.6159972

Vachtsevanos G, Tang L, Drozeski G, Gutierrez L (2005) From mission planning to flight control of unmanned aerial vehicles: strategies and implementation tools. Ann Rev Control 29(1):101–115

Van Veldhuizen D, Lamont GB, et al. (2000) On measuring multiobjective evolutionary algorithm performance. In: Evolutionary computation, 2000. Proceedings of the 2000 congress on, vol 1. IEEE, pp 204–211

Wagner T, Trautmann H, Martí L (2011) A taxonomy of online stopping criteria for multi-objective evolutionary algorithms. In: Evolutionary multi-criterion optimization. Springer, pp 16–30

Wu J, Zhou G (2006) High-resolution planimetric mapping from uav video for quick-response to natural disaster. In: Geoscience and remote sensing symposium, 2006. IGARSS 2006. IEEE international conference on. IEEE, pp 3333–3336

Zhou A, Qu BY, Li H, Zhao SZ, Suganthan PN, Zhang Q (2011) Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm Evolut Comput 1(1):32–49

Zitzler E, Laumanns M, Thiele L (2001) SPEA2: Improving the strength pareto evolutionary algorithm. In: Eurogen, vol. 3242, pp. 95–100

Zitzler E, Laumanns M, Bleuler S (2004) A tutorial on evolutionary multiobjective optimization. In: Metaheuristics for multiobjective optimisation. Springer, pp 3–37

Zitzler E, Brockhoff D, Thiele L (2007) The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In: Evolutionary multi-criterion optimization. Springer, pp 862–876