A Hybrid MOGA-CSP for Multi-UAV Mission Planning

Cristian Ramirez-Atencia
Departamento de Ingeniería Informática, Universidad Autónoma de Madrid
Madrid, Spain
cristian.ramirez@inv.uam.es

Gema Bello-Orgaz
Departamento de Ingeniería Informática, Universidad Autónoma de Madrid
Madrid, Spain
gema.bello@uam.es

Maria D. R-Moreno
Departamento de Automática, Universidad de Alcalá
Alcalá de Henares, Spain
mdolores@aut.uah.es

David Camacho
Departamento de Ingeniería Informática, Universidad Autónoma de Madrid
Madrid, Spain
david.camacho@uam.es

ABSTRACT

Mission Planning Problem for a large number of Unmanned Air Vehicles (UAV) consists of a set of locations to visit in different time windows, and the actions that the vehicle can perform based on its features such as the sensors, speed or fuel capacity. After formulating this problem as a Constraint Satisfaction Problem (CSP), we try to search the set of Non dominated solutions which minimize the fuel consumption and the makespan of the mission. To solve it, we will use a Multi-Objective Genetic Algorithm (MOGA), that will match the model constraints and use a multi-objective function in order to optimize these objective variables.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*heuristic methods*

Keywords

Unmanned Aircraft Systems, Mission Planning, Multi-Objective, Genetic Algorithms, Constraint Satisfaction Problems

1. INTRODUCTION

Mission planning for Unmanned Aircraft Vehicles (UAVs) [4] involves planning the actions that the vehicle must perform (loading/dropping a load, taking videos/pictures, etc.), over a time period. These planning problems can be solved using different methods to find optimal solutions but, as the number of restrictions increases, the complexity grows exponentially because it is a NP-hard problem. Some modern approaches formulate mission planning as a Constraint Satisfaction Problem (CSP) [1], where the tactic mission is modelled and solved using constraint satisfaction techniques.

This work deals with multiple UAVs that must perform one or more tasks in different locations. The solution plans obtained should fulfill all the constraints given by the different components and capabilities of the UAVs involved. In previous works [5, 6] the mission planning problem was modelled as a Constraint Satisfaction Optimization Problem (CSOP) along with and optimization function designed to minimize the fuel cost, the flight time and the number of UAVs needed, and solved with Branch and Bound (B&B).

The main goal of this work is to present a Multi-Objective Genetic Algorithm (MOGA) approach to solve this model using a multi-objective fitness function. We will present two different representations for this problem and compare them in order to check which one gives better results.

This paper is structured as follows: section 2 describes how a Misison is defined and the modelization of the problem as a CSP. Section 3 explains the development of the MOGA-based solver for the model and the different representations considered. Section 4 shows some experiments performed in order to validate this algorithm. Finally, last section presents conclusions about this work.

2. UAV MISSION PLANNING BASED ON CSP

UAV missions consists of a number n of tasks performed by a team of m UAVs. Each task must be performed in a specific *area*, *time interval* and needs several *sensors*. This approach considers three different types of tasks: **Taking pictures of a zone**, which requires a EO/IR Camera; **Taking real-time pictures of a zone**, which requires a EO/IR Camera and a Communications Equipment, and **Tracking a zone**, which requires a SAR radar.

On the other hand, the vehicles performing the mission have some features that must be taken into account in order to check if a mission plan is correct. These features include the **initial position** (Latitude, Longitude), the **initial fuel** (L), the **available sensors** and one or more **flight profiles**. A vehicle's flight profile specifies at each moment its **speed** (Km/h) and **fuel consumption rate** (L/Km). The different flight profiles of a UAV are used depending on the situation of the mission: climb or descent, normal flight, ...

GECCO'15 Companion, July 11–15, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739482.2768481

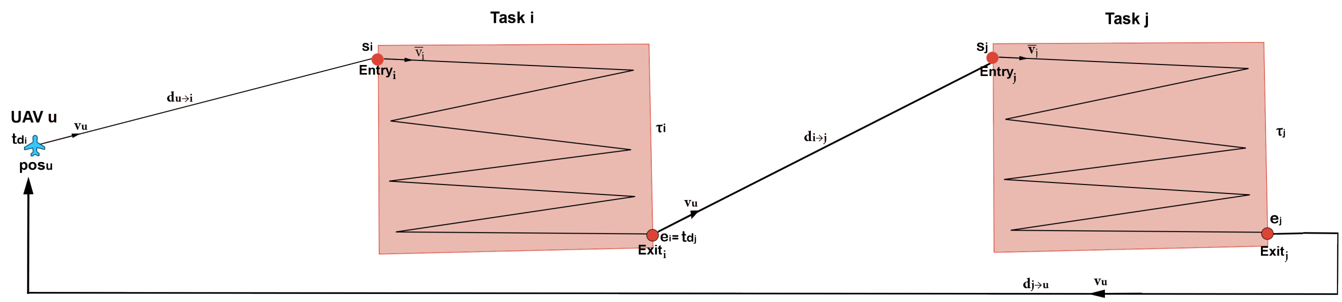

Figure 1: Example of assignment of a UAV u to tasks i and j.

Figure 1 shows an assignment of a UAV u to two tasks i and j. In this example, we define the **departure** of the task as the time when the UAV leaves its actual position to reach the task, the **start** of the task as the exact moment when the task is performed and the **end** of the task as the moment when it is finished. In this assignment, it is necessary to compute the distances from u to each one of the tasks and between the tasks, and then take the fuel consumption rate and speed from the flight profiles in order to compute **path** and **return durations** and **fuel consumptions**.

In this approach, each task has a specific duration, and the time windows will be obtained when a UAV is assigned all the tasks it must perform and the order in which it performs these tasks. So the variables of the CSP are the assignments of each task to the UAV performing it, and the orders in which tasks are performed. There exists additional variables that are computed in the propagation phase of the constraint solving, such as the times, durations and fuel consumptions of the tasks. The main constraints of this model are:

1. Order constraints assuring two tasks assigned to the same UAV have different order values, and that those orders values are less than the number of tasks assigned to the UAV performing the considered tasks:

$$order[t] < \sharp\{\tau \in T | assign[\tau] = assign[t]\} \quad (1)$$

2. Temporal constraints assuring the consistency of the different time variables. We must assure that the start time of the task is the sum of the departure time and the duration for the path, and that the end time is the sum of the start time and the duration of the task:

$$departure[t] + durPath[t] = start[t] \quad (2)$$
$$start[t] + durTask[t] = end[t] \quad (3)$$

Then, when two tasks are assigned the same UAV, given their orders, we must assure that the departure time for the second task is less or equal than the end time for the first one:

$$assign[i] = assign[j] \land order[i] < order[j] \Rightarrow end[i] \leq departure[j] \quad (4)$$

Now, we compute the duration of the path for the first task performed by the UAV u. Given the flight profile used by the UAV in the path to the task, we compute the duration of the path as the distance $d_{u \to i}$ from the UAV to the task divided by the speed v_u given by the flight profile: $durPath[t] = \frac{d_{u \to t}}{v_u}$.

Moreover, for each pair of consecutive tasks performed by the same UAV, the duration of the path between them is computed as the distance $d_{i \to j}$ from the first to the second divided by the speed v_u given by the path flight profile of the vehicle, $durPath[j] = \frac{d_{i \to j}}{v_u}$.

Finally, we compute the return duration as the distance $d_{t \to u}$ from the last task performed by the UAV u divided by the speed v_u given by the return flight profile: $durReturn[u] = \frac{d_{t \to u}}{v_u}$.

3. Sensor constraints assuring UAVs carries the corresponding sensors to perform a task. Let S_u denote the sensors available for the UAV u and S_t the sensor required by the task t (performed by u), then:

$$S_t \subseteq S_u \quad (5)$$

4. Fuel constraints, in order to check the fuel cost for each UAV. For each task t, we compute the fuel consumed in the task performance by multiplying the duration of the task, the speed $\overline{v_t}$ and the fuel consumption rate $\overline{fuelRate_t}$ given by the sensors required flight profile: $fuelTask[t] = durTask[t] \times \overline{v_t} \times \overline{fuelRate_t}$.

On the other hand, similarly to the temporal constraints used to compute the durations, we use the path flight profile to compute the fuel consumed in the path to the first task by multiplying distance and the fuel consumption rate $fuelRate_u$ given in the flight profile: $fuelPath[t] = d_{u \to t} \times fuelRate_u$.

Similarly, for each pair of consecutive tasks, we have: $fuelPath[j] = d_{i \to j} \times fuelRate_u$.

And finally, for the last task t performed by the UAV u, we have: $fuelReturn[u] = d_{t \to u} \times fuelRate_u$.

Now, we just constraint the sum of all fuel consumption values to be less than the UAV's initial fuel $fuel_u$:

$$\sum_{\substack{t \in T \\ assign[t]=u}} (fuelPath[t] + fuelTask[t]) + fuelReturn[u] < fuel_u \quad (6)$$

3. PROPOSED MOGA-CSP FOR MULTI-UAV MISSION PLANNING

Genetic Algorithms (GAs) are stochastic methods inspired by natural evolution and genetics. 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.

Given the big 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 Multi-Objective GA to solve the CSP modelled Mission Planning problem. In this approach, we will develop a hybrid MOGA-CSP, where the constraints of the problem will be applied as penalty functions in the evaluation phase of the MOGA.

3.1 Encoding

Now we consider two possible representations for the UAV Mission Planning Problem: the ordering representation and the permutation representation. In the first one, an individual of the MOGA will be formed by two rows of gene strings $T_1 T_2 T_3 ... T_n$, i.e. its chromosome. An example of this representation is shown in Figure 2. The first row of the chromosome represents the UAVs assigned to each task T_i, while the second represents the orders of the tasks according to their UAV assignment. With this representation, the gene values in the first row are integers in the interval $[1, m]$, i.e. the UAV identifier; and the gene values in the second row are integers in the interval $[0, m-1]$. In the example, we can see that tasks 1, 2 and 5 are performed first (order 0), then task 4 (order 1) and finally task 3 (order 2); so: UAV1 performs tasks 1, 4 and 3 in that order; UAV2 performs task 2, and UAV3 performs task 5.

	T_1	T_2	T_3	T_4	T_5
UAV	1	2	1	1	3
Order	0	0	2	1	0

Figure 2: Example of ordering representation with 5 tasks and 3 UAVs.

The second representation considered involves the second row being a permutation, i.e. its values indicating an absolute order of the tasks, independently of their UAVs assignment. This way, we reduce the space of possible individuals and avoid many invalid solutions. An example of this representation can be shown in Figure 3. This example shows that tasks have this order: first task 2, then 1, 4, 5 and finally 3. With this and the values from the first row, we can obtain the same order values from Figure 2.

	T_1	T_2	T_3	T_4	T_5
UAV	1	2	1	1	3
Order	1	0	4	2	3

Permutation

Figure 3: Example of permutation representation with 5 tasks and 3 UAVs.

3.2 Fitness Function

Evaluation is computed in terms of a fitness function composed by two check steps. First, for the 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 for the parameters of the model:

- The total *fuel consumption* of all the UAVs employed, computed as: $fuelCost = \sum_{u \in U} fuelConsumed[u]$, where the fuel consumed by each UAV is:

$$fuelConsumed[u] = \sum_{\substack{t \in T \\ assign[t]=u}} (fuelPath[t] + fuelTask[t]) + fuelReturn[u] \quad (7)$$

- The makespan of the mission, computed as the maximum of the flight times of each UAV: $makespan = \max_{u \in U} flightTime[u]$, where the flight time of each UAV is:

$$flightTime[u] = \sum_{\substack{t \in T \\ assign[t]=u}} (durPath[t] + durTask[t]) + durReturn[u] \quad (8)$$

The multi-objective fitness function will compare the solution tested with the stored solutions in order to obtain the Pareto-Optimality Frontier (POF) [7]. In this approach we have used NSGA-II [3], but there are other approaches that could be used here, such as SPEA2 [8] or PESA-II [2].

3.3 Algorithm

In our approach, the selection consists of two steps: first, a N elitist selection is performed for retaining a number N of best individuals in the population; then, a roulette wheel selection over these N individuals is performed to select those that will be applied the crossover and mutation operators.

In this approach, a 2-point crossover is used to combine the chromosomes of each pair of parents to generate a pair of children. Finally, a uniform mutation operator will mutate these chromosomes depending on a probability P_m (usually low, $\sim 5\%$). This operator will help to avoid that the obtained solutions stagnate at local minimums.

4. EXPERIMENTAL RESULTS

The Mission Scenario used in this experiment consists of 8 tasks to be assigned to 5 UAVs scattered throughout the map. HALE has a EO/IR camera and a Communications Equipment; MALE, all the three types of sensor; URAV, a EO/IR Camera; UCAV, a EO/IR Camera and a SAR radar, and TACT, a SAR radar.

In this approach, we have set up all the UAVs with the same flight profile, consisting of a speed of $100 Km/h$ and a fuel consumption rate of $0.15L/Km$. Besides, all UAVs have the same initial amount of fuel: $100L$.

Now, in order to check the performance of the developed algorithm, we first use a Multi-Objective B&B in order to obtain the POF of this problem. We got that this POF is composed of 6 different solutions. In this first approach, we want to compare the two chromosome representations explained in the previous section. For that, we will use two measures of quality very used in evolutionary algorithms comparison: hypervolume and generational distance [9].

First of all, we set the parameters of the MOGA to the values shown in Table 1.

Table 1: MOGA setup.

Initial Population	1000
Elitism population	50
Mutation probability	5 %
Number of generations	100

With this, we execute the solver with each of the chromosome representations a total of 50 times, then compute these quality measures for each run, and finally return the minimum, maximum and median of these measures for each representation (see Table 2).

Table 2: Min, Median and Max Hypervolume and generational distance of the 50 runs of the MOGA solver for the two different encodings.

Representations		Ordering	Permutation
Hypervolume	Min.	0.5367	0.4037
	Med.	∞	0.5281
	Max.	∞	0.5378
Generational distance	Min.	0.0534	0.0088
	Med.	∞	0.0175
	Max.	∞	0.0648

In the execution of the solver for the first representation, we obtained that most runs could not obtain even a valid solution, so the Hypervolume and generational distance were Infinity, as can be seen in the Median and Maximum row of both measures in Table 2. It is then clear that the first approach is not suitable for this kind of problem. On the other hand, it can be seen that for the permutation representation, we see that the minimum generational distance obtained is quite good (although not the optimum), and in general the range of generational distance obtained in the 50 runs are good. Similarly, the hypervolume measures gives promising results for this problem.

5. DISCUSSION AND FUTURE WORKS

In this paper, we propose a hybrid GA-CSP approach to search feasible solutions for a UAV Mission Planning model. The presented model defines missions as a set of tasks to be performed by several UAVs. The CSP model defines several constraints, including order and temporal constraints assuring that each UAV only performs one task at a time, the needed sensors or the fuel consumption.

The MOGA proposal counts with a multi-objective fitness function that penalizes the unfulfilled constraints and minimize two objectives: the fuel consumption and the makespan of the mission. We proposed two chromosome representations, and in the experimental phase we proved that the order permutation is better for this kind of problem.

In future works, several selection, crossover and mutation operators, as well as different Multiobjective Algorithms, such as SPEA2, should be tested and compared to find the best performing combination. On the other hand, we will also compare this approach with other from the state-of-art in terms of optimality of the solutions and runtime spent.

6. ACKNOWLEDGMENTS

This work is supported by: Spanish Ministry of Science and Education under project TIN2014-56494-C4-4-P, Junta de Comunidades de Castilla-La Mancha under project PEII-2014-015-A, Comunidad Autónoma de Madrid under project CIBERDINE S2013/ICE-3095 and Savier Project (Airbus Defence & Space, FUAM-076915). The authors would like to acknowledge the support obtained from Airbus Defence & Space, specially from Savier Open Innovation project members: José Insenser, César Castro and Gemma Blasco.

7. REFERENCES

[1] R. Barták. Constraint programming: In pursuit of the holy grail. In *Week of Doctoral Students*, pages 555–564, 1999.

[2] D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, and M. J. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Genetic and Evolutionary Computation Conference*, pages 283–290, 2001.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation*, 6(2):182–197, 2002.

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

[5] C. Ramirez-Atencia, G. Bello-Orgaz, M. D. R-Moreno, and D. Camacho. A simple CSP-based model for Unmanned Air Vehicle Mission Planning. In *IEEE International Symposium on INnovations in Intelligent SysTems and Application*, pages 146–153, 2014.

[6] C. Ramirez-Atencia, G. Bello-Orgaz, M. D. R-Moreno, and D. Camacho. Branching to find feasible solutions in Unmanned Air Vehicle Mission Planning. In *Intelligent Data Engineering and Automated Learning*, pages 286–294, 2014.

[7] M. Torrens and B. Faltings. Using Soft CSPs for Approximating Pareto-Optimal Solution Sets. In *AAAI Workshop Preferences in AI and CP: Symbolic Approaches*. AAAI Press, 2002.

[8] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. In *Evolutionary Methods for Design, Optimisation, and Control*, pages 95–100, 2002.

[9] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *Evolutionary Computation*, 7(2):117–132, 2003.