Performance Evaluation of Multi-UAV Cooperative Mission Planning Models

Cristian Ramirez-Atencia[1](✉), Gema Bello-Orgaz[1], Maria D. R-Moreno[2], and David Camacho[1]

[1] Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, C/Francisco Tomás Y Valiente 11, 28049 Madrid, Spain
cristian.ramirez@inv.uam.es, {gema.bello,david.camacho}@uam.es
http://aida.ii.uam.es/

[2] Departamento de Automática, Universidad de Alcalá, Carretera Madrid Barcelona, Km 33 600, 28871 Madrid, Spain
mdolores@aut.uah.es

Abstract. The Multi-UAV Cooperative Mission Planning Problem (MCMPP) is a complex problem which can be represented with a lower or higher level of complexity. In this paper we present a MCMPP which is modelled as a Constraint Satisfaction Problem (CSP) with 5 increasing levels of complexity. Each level adds additional variables and constraints to the problem. Using previous models, we solve the problem using a Branch and Bound search designed to minimize the fuel consumption and number of UAVs employed in the mission, and the results show how runtime increases as the level of complexity increases in most cases, as expected, but there are some cases where the opposite happens.

Keywords: Unmanned aircraft systems · Mission planning · Constraint satisfaction problems · Branch and bound

1 Introduction

Unmanned Aircraft Systems (UAS) have been used in many potential applications including monitoring coastal frontiers, road traffic, disaster management, agriculture, etc [6]. Nowadays, Unmanned Air Vehicles(UAVs) are controlled remotely from Ground Control Stations(GCSs) by humans operators who use legacy mission planning systems.

Mission planning for UAS deals with the actions that the vehicle must perform (loading/dropping a load, taking videos/pictures, etc.), typically over a time period. These planning problems can be solved using different methods such as Mixed-Integer Linear Programming (MILP) [14], Simulated Annealing (SA) [3], Auction Algorithms (AA) [8], etc. Usually, these methods are the best way to find the optimal solutions but, as the number of restrictions increases, the complexity grows exponentially because it is a NP-hard problem. Therefore, when the complexity of these models increases, it is quite usual to employ Heuristic Methods [16].

In the literature there are some attempts to implement UAS that achieve mission planning and decision making using Temporal Action Logic (TAL) for reasoning about actions and changes [4], Markov Decision Process (MDP) and dynamic programming algorithms [5], or hybrid Partial-Order Forward-Chaining (POFC) [7], among others. Other modern approaches formulate the mission planning problem as a Constraint Satisfaction Problem (CSP) [2], 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 [11,12] a simple approach of the Multi-UAV Cooperative Mission Planning Problem (MCMPP) 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 & Bound (B&B). In this new work, we will redefine this model using 5 different levels of complexity to analyse the computational performance of these models when the complexity increases.

The rest of the paper is structured as follows: Section 2 shows basic concepts and definitions about CSPs. Section 3 describes how a UAV Mission is defined. Section 4 shows how the MCMPP is modelled as a CSP and the different levels of complexity to deal with. Section 5 explains the experiments carried out and the results obtained. Finally, the last section presents the final analysis and conclusions of this work.

2 Some Basis in Constraint Satisfaction Problems

Any CSP can be defined as a tuple $< V, D, C >$ where:

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

There are many studied methods to search for all the solutions in a CSP problem, such as Backtracking (BT), Backjumping (BJ) or Forward Checking (FC). These algorithms are usually combined with other techniques like consistency techniques (Node Consistency (NC), Arc Consistency (AC) or Path Consistency (PC)) to modify the CSP ensuring its local consistency conditions.

In most of cases, we have a goal test function that returns a numerical value (how good the solution is), and this special CSP in which we try to optimize the solutions is referred as CSOP. The most widely used method for finding optimal solutions is B&B.

A Temporal Constraint Satisfaction Problem (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 [15]. Different classes of constraints are characterized by the underlying set of

Basic Temporal Relations (BTR). Most types of TCSPs can be represented with Point Algebra (PA), with BTR $= \{\varnothing, <, =, >, \leq, \geq, \neq, ?\}$.

In the related literature, Mouhoub [9] proved that on real-time or Maximal Temporal Constraint Satisfaction Problems(MTCSPs), the best methods for solving them are Min-Conflicts-Random-Walk (MCRW) in the case of under-constrained and middle-constrained problems, and Tabu Search (TS) and Steepest-Descent-Random-Walk (SDRW) in the over-constrained case. He also developed a temporal model, TemPro [10], which is based on interval algebra, to translate an application involving temporal information into a CSP.

3 Modelling a UAV Mission Planning

The MCMPP can be defined as a number n of tasks to accomplish for a team of m UAVs. There exists different kind of tasks, such as exploring a specific area or search for an object in a zone. These tasks can be carried out thanks to different sensors available by the UAVs performing the mission. In this approach, we consider several types of sensors: Electro-optical or Infra-red (EO/IR) cameras, Synthetic Aperture Radar (SAR), Signal Intelligence (SIGINT), etc. In addition, each task must be performed in a specific geographic *area* and in a specific *time interval*. This time interval could be specified with a start and end time for the task, or just with the duration of the task, so the start and end of the task must be computed in the planning.

On the other hand, the vehicles performing the mission has some features that must be taken into account in order to check if a mission plan is correct: their **initial positions**, their initial fuels, their **available sensors**, and one or more **flight profiles**. A vehicle's flight profile specifies at each moment its **speed**, its **fuel consumption ratio** and its **altitude** (unless it is a climb or descend profile).

When a task is assigned a vehicle, it is necessary to compute the **departure** time when the vehicle starts moving to the task. In addition, it is necessary to compute the **duration of the path** between the departure of the UAV and the start of the task. Finally, if a task is the last of the tasks assigned to a vehicle, then we must compute the **duration of the return** from this last task to the base (the initial position of the UAV).

In order to compute these durations, it is necessary to know which of the flight profiles (if there are more than one) of the UAV will be used. A UAV usually has, at least, three flight profiles: a **climb profile**, a **descend profile** and a **route profile**. Moreover, given that the flight profile provides the fuel consumption ratio, it will be possible to compute the fuel consumptions both in the path and the return. On the other hand, to compute the fuel consumption in the performance of the task, we must know the flight profile required by the sensor used in the task performance.

When start and end time of tasks are fixed, it is possible that the time when a vehicle finishes a task does not meet the time when the vehicle departs for the next task. In this case, we define this interval of time as the **loiter duration** for the second task. In this situation, the UAV must fly using the **minimum cost flight profile**, so the fuel consumption is minimum.

Finally, a mission could have some time and vehicle dependencies between different tasks. **Vehicle dependencies** consider if two tasks must be assigned the **same UAV** or **different UAVs**. Moreover, we consider **time dependencies** given by **Allen's Interval Algebra** [1].

4 Proposed CSP Model

Given the assumptions described in Section 3, we can consider several sets of variables in this kind of problem:

- Assignments ($assign$) of tasks to UAVs of the MCMPP.
- 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.
- Path Flight Profiles ($fpPath$). These variables set the flight profile that the vehicle must take for the path performance. It is necessary to consider these variables just in the case that there are several route profiles.
- Return Flight Profiles ($fpReturn$), similar to the previous set of variables but for the return performance of each UAV.
- Task Flight Profile ($fpTask$). These variables set the flight profile that the vehicle must take to complete the task, which is required by the sensors used. It is necessary to consider these variables just in the case that the vehicle performing the task has several sensors that could perform that task.

In addition, it is necessary to define some extra variables that are computed in the propagation phase of the CSP solver. These variables are the time points ($departure$, $start$ and end) and durations ($durPath$, $durTask$, $durLoiter$, $durReturn$) of the mission, as well as fuel consumptions ($fuelPath$, $fuelTask$, $fuelLoiter$, $fuelReturn$);

Now, we define the constraints of the CSP according to the level of complexity used. In this work, we have defined 5 levels of complexity, each one with a higher level of constraints.

4.1 CSP Model-1

This first level only uses sensor constraints, which are used to check if a UAV carries the required sensors to perform a task. Let $sensors(u)$ denote the sensors available for UAV u, and $sensors(t)$ the sensors that could perform the task t. The constraint to check it can be modelled as follows:

$$assign[i] = u \Rightarrow |sensors(t) \cap sensors(u)| > 0 \quad (1)$$

In this level, the only variables that are considered are the assignments.

4.2 CSP Model-2

Here, we add order and temporal constraints that will check the time consistency of the model, transforming it into a TCSP. Consequently, this level will consider the order variables, and also all the secondary variables related to the time points and durations. We consider that each UAV has just one flight profile that will be used for the duration computations.

First of all, it is necessary to assure that the value of the order variable is less than the number of tasks assigned to the UAV performing that task. This constraint is modelled as Equation 2 shows:

$$assign[t] = u \Rightarrow order[t] < \sharp\left\{t \in T | assign[t] = u\right\} \tag{2}$$

and if two tasks are assigned the same UAV, they have different order values:

$$assign[i] = assign[j] \Rightarrow order[i] \neq order[j] \tag{3}$$

Then, 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:

$$departure[t] + durPath[t] = start[t] \tag{4}$$

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

$$start[t] + durTask[t] = end[t] \tag{5}$$

If tasks have fixed starts 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:

$$assign[i] = assign[j] \wedge order[i] = order[j] - 1 \Rightarrow durLoiter[j] = departure[j] - end[i] \tag{6}$$

Then, when two tasks are assigned to the same UAV, given their orders, the end time of the first task must be less or equal than the departure of the second:

$$assign[i] = assign[j] \wedge order[i] < order[j] \Rightarrow end[i] \leq departure[j] \tag{7}$$

Now, to compute the durations of the paths to the tasks, it is necessary to compute the distances from the UAV to the task ($d_{u \to t}$), between each pair of consecutive tasks ($d_{i \to j}$) and from the last task performed by a UAV to its initial position ($d_{t \to u}$). Given that these points are represented in geodesic coordinates, the distance between two points is computed using the Haversine formula with the latitude and longitude of the points.

With this and the speed v_u given by the flight profile of the vehicle, we can compute the duration of the path for the first task t performed by the vehicle u as $durPath[t] = \frac{d_{u \to t}}{v_u}$. On the other hand, for each pair of consecutive tasks i and j assigned to the same UAV, the duration of their path is computed as $durPath[j] = \frac{d_{i \to j}}{v_u}$. Finally, we compute the return duration for a UAV u, given its last planned task t, as $durReturn[u] = \frac{d_{t \to u}}{v_u}$.

4.3 CSP Model-3

In this level, we will add the fuel constraints, which check the fuel consumption for each UAV. The variables in this level remain the same as in Level 2.

First of all, in order to compute the fuel consumption in the performance of each task, it is necessary to know the flight profile required by the sensors to perform the task. For this, we will consider that all sensors require the same flight profile, which give us the speed $\overline{v_t}$ and the fuel consumption rate $\overline{fuelRate_t}$ for the performance of a task t.

With this, we can compute the fuel consumed by the UAV performing the task as $fuelTask[t] = durTask[t] \times \overline{v_t} \times \overline{fuelRate_t}$

On the other hand, the fuel consumed in the path for the first task t performed by a vehicle u is computed as $fuelPath[t] = d_{u \to t} \times fuelRate_u$. Moreover, the fuel consumed in the path between two consecutive tasks i and j is computed as $fuelPath[j] = d_{i \to j} \times fuelRate_u$.

Then, the fuel consumption for the loiter of a task, given that each vehicle has just one flight profile, is $fuelLoiter[t] = durLoiter[t] \times v_u \times fuelRate_u$.

Finally, we compute the return fuel consumption given the last task t performed by a UAV u as $fuelReturn[u] = d_{t \to u} \times fuelRate_u$.

With all this, we just have to sum all these fuel consumption values of a UAV and constraint it to be less than its initial fuel $fuel_u$:

$$\sum_{\substack{t \in T \\ assign[t]=u}} (fuelPath[t] + fuelTask[t]) + fuelReturn[u] < fuel_u \tag{8}$$

4.4 CSP Model-4

In this level, we will consider that each UAV has four flight profiles: a climb profile, a descend profile, a minimum consumption profile and a maximum speed profile. In addition, each of the sensors will have a specific required flight profile, and not necessarily the same as assumed previously.

With this, all the duration and fuel consumption computations from previous levels must be recomputed considering the values given by the flight profiles selected in variables $fpPath$, $fpTask$ and $fpReturn$. On the computation of loiter fuel consumption, we must use the minimum consumption flight profile.

Following this, for the first task t performed by a UAV u, we check if its path profile must be climb or descend according to its increase or decrease of altitude. The altitude of the task is given by its task profile, so we have that:

$$\begin{cases} altitude(fpTask[t]) - altitude(u) > 0 \Leftrightarrow fpPath[t] = u.CLIMB \\ altitude(fpTask[t]) - altitude(u) < 0 \Leftrightarrow fpPath[t] = u.DESCEND \end{cases} \tag{9}$$

On the other hand, for each pair of consecutive tasks i and j, a similar situation must be accomplished:

$$\begin{cases} altitude(fpTask[j]) - altitude(fpTask[i]) > 0 \Leftrightarrow fpPath[j] = u.CLIMB \\ altitude(fpTask[j]) - altitude(fpTask[i]) < 0 \Leftrightarrow fpPath[j] = u.DESCEND \end{cases} \tag{10}$$

On the other hand, for the return flight profile, given the last task t assigned to vehicle u, we must check if the return profile of u is climb or descend:

$$\begin{cases} altitude(u) - altitude(fpTask[t]) > 0 \Leftrightarrow fpReturn[u] = u.CLIMB \\ altitude(u) - altitude(fpTask[t]) < 0 \Leftrightarrow fpReturn[u] = u.DESCEND \end{cases} \tag{11}$$

4.5 CSP Model-5

Finally, in this level, we will add all the time and vehicle dependencies mentioned in Section 3. The time dependency constraints, for each pair of tasks i and j and according to Allen's Interval Algebra, are as follow:

$$i \quad precedes \quad j \Rightarrow \quad end[i] \leq start[j] \tag{12}$$

$$i \quad meets \quad j \Rightarrow end[i] = start[j] \tag{13}$$

$$i \quad overlaps \quad j \Rightarrow \begin{cases} start[i] \leq start[j] \\ end[i] \geq start[j] \\ end[i] \leq end[j] \end{cases} \tag{14}$$

$$i \quad starts \quad j \Rightarrow \begin{cases} start[i] = start[j] \\ end[i] \leq end[j] \end{cases} \tag{15}$$

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

$$i \quad finishes \quad j \Rightarrow \begin{cases} start[i] \geq start[j] \\ end[i] = end[j] \end{cases} \tag{17}$$

$$i \quad equals \quad j \Rightarrow \begin{cases} start[i] = start[j] \\ end[i] = end[j] \end{cases} \tag{18}$$

On the other hand, the vehicle dependencies implies that some task must be performed by the same vehicle or by different vehicles:

$$sameUAV(i, j) \Rightarrow assign[i] = assign[j] \tag{19}$$

$$differentUAV(i, j) \Rightarrow assign[i] \neq assign[j] \tag{20}$$

4.6 Optimization Variables

Now, in order to solve the CSOP with B&B, we must define the optimization function that this algorithm will use. Following a previous work [12], we will consider three variables: the number of vehicles, the fuel consumption and the flight time. To reckon them, first we compute the fuel consumed by each vehicle:

$$fuelConsumed[u] = \sum_{\substack{t \in T \\ assign[t]=u}} (fuelPath[t] + fuelTask[t] + fuelLoiter[t]) + fuelReturn[u] \tag{21}$$

and the flight time of each vehicle:

$$flightTime[u] = \sum_{\substack{t \in T \\ assign[t]=u}} (durPath[t] + durTask[t] + durLoiter[t]) + durReturn[u] \tag{22}$$

And then just compute the objective variables as:

- The number of UAVs used in the mission: $N_{uavs} = \sharp\{u \in U | \exists t \in T \quad assign[t] = u\}$.
- The total flight time of the mission: $flightTimeCost = \sum_{u \in U} flightTime[u]$.
- The total fuel consumption of the mission: $fuelCost = \sum_{u \in U} fuelConsumed[u]$.

Now, according to the previous work [12], we will use the function $0.9 \times N_{uavs} + 0.1 \times fuelCost$ with these problems. Nevertheless, in levels 1 and 2 it is not possible to use this function because $fuelCost$ is not defined. Instead, we will use N_{uavs} for Level 1, and $0.9 \times N_{uavs} + 0.1 \times flightTimeCost$ for Level 2.

5 Experimental Results

The Mission Scenario used in this experiment consists of 8 tasks and 5 UAVs scattered throughout the map. Each UAV has different sensors and each task can be performed by different sets of sensors, so there are several possible solutions.

On the other hand, we have fixed four time dependencies (of type $<$, m, f, and $=$), and a same UAV vehicle dependency. Each UAV has been set with a different amount of initial fuel and all departs from ground (altitude 0).

Now, we run B&B with this MCMPP modelled as a CSP using Gecode [13] on a AMD FX-8350 8-Core 4.00GHz and 8GB RAM. For each level of complexity, we run the problem twice: first, with start and end times of all tasks fixed, and then with these unfixed, obtaining the run times shown in Table 1.

From the first three levels of complexity, we can see that adding complexity implies adding some secondary variables (times and fuel) that are not iterated, but computed in the propagation phase of the CSP. This is, the propagation phase takes more time due to the variables that must be computed. On the other hand, we can see that when times are unfixed, B&B takes much more time to find the optimal solution, because of the order variables that must be iterated.

Table 1. Runtime of the MCMPP of 8 tasks and 5 UAVs, with each of the 5 levels of complexity and task times fixed and unfixed.

Level of complexity	Fixed Times	Unfixed Times
Level 1	20ms	-
Level 2	830ms	11.34s
Level 3	1.03s	1min 50s
Level 4	55.32s	6h 52min 45s
Level 5	56.05s	4h 29min 23s

Looking now at levels 4 and 5, we can see that they take much more time than the three first levels, because of the flight profile variables to iterate. The most interesting fact here is that, for unfixed times, level 4 spends more time than level 5. This is because adding the dependency constraints makes a little more propagation time but highly reduce the space of solutions, making it easier to the algorithm to found the optimal solution.

6 Conclusions

In this paper, we have presented a CSP model for Multi-UAV Mission Planning. The presented approach defines missions as a set of tasks to be performed by several UAVs with some capabilities. The problem is modelled according to 5 levels of complexity: Level 1 considers sensor constraints, Level 2 adds temporal constraints, Level 3 adds fuel constraints, Level 4 considers flight profiles and Level 5 add temporal and vehicle dependencies.

Using B&B, we have performed an experiment in order to measure the scalability of the problem in the different levels of complexity. We have considered two possible approaches: with fixed times and with unfixed times. Results showed that adding constraints that imply considering new variables always increase the runtime; but constraints that do not add new variables, as the dependency constraint added in Level 5, do not increase the runtime, but may decrease it.

In order to make these results more trustful, in further works we will consider different scenarios and topologies, so a more general conclusion would be obtained. Furthermore, we will use a Multiobjective model, such as the Multi-Objective Evolutionary Algorithm (MOEA), to find the Pareto Optimal Frontier (POF) of the optimization variables, instead of using a percentages function.

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.

References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM, 832–843 (1983)
2. Barták, R.: Constraint programming: in pursuit of the holy grail. In: Week of Doctoral Students, pp. 555–564 (1999)
3. Chiang, W.C., Russell, R.A.: Simulated annealing metaheuristics for the vehicle routing problem with time windows. Annals of Operations Research **63**, 3–27 (1996)
4. Doherty, P., Kvarnström, J., Heintz, F.: A temporal logic-based planning and execution monitoring framework for Unmanned Aircraft Systems. Autonomous Agents and Multi-Agent Systems **19**(3), 332–377 (2009)
5. Fabiani, P., Fuertes, V., Piquereau, A., Mampey, R., Teichteil-Konigsbuch, F.: Autonomous flight and navigation of VTOL UAVs: from autonomy demonstrations to out-of-sight flights. Aerospace Science and Technology **11**(2–3), 183–193 (2007)
6. Kendoul, F.: Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. J. Field Robot. **29**(2), 315–378 (2012)
7. Kvarnström, J., Doherty, P.: Automated planning for collaborative UAV systems. In: Control Automation Robotics & Vision, pp. 1078–1085, December 2010
8. Leary, S., Deittert, M., Bookless, J.: Constrained UAV mission planning: a comparison of approaches. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 2002–2009, November 2011
9. Mouhoub, M.: Solving temporal constraints in real time and in a dynamic environment. Tech. Rep. WS-02-17. AAAI (2002)
10. Mouhoub, M.: Reasoning with numeric and symbolic time information. Artif. Intell. Rev. **21**(1), 25–56 (2004)
11. Ramirez-Atencia, C., Bello-Orgaz, G., R-Moreno, M.D., Camacho, D.: A simple CSP-based model for unmanned air vehicle mission planning. In: IEEE International Symposium on INnovations in Intelligent SysTems and Application, pp. 146–153 (2014)
12. Ramírez-Atencia, C., Bello-Orgaz, G., R-Moreno, M.D., Camacho, D.: Branching to find feasible solutions in unmanned air vehicle mission planning. In: Corchado, E., Lozano, J.A., Quintián, H., Yin, H. (eds.) IDEAL 2014. LNCS, vol. 8669, pp. 286–294. Springer, Heidelberg (2014)
13. Schulte, C., Tack, G., Lagerkvist, M.Z.: Modeling and Programming with Gecode (2010). http://www.gecode.org/
14. Schumacher, C., Chandler, P., Pachter, M., Pachter, L.: UAV Task Assignment with Timing Constraints via Mixed-Integer Linear Programming. Tech. rep, DTIC Document (2004)
15. Schwalb, E., Vila, L.: Temporal constraints: A survey. Constraints **3**(2–3), 129–149 (1998)
16. Tang, L., Zhu, C., Zhang, W., Liu, Z.: Robust mission planning based on nested genetic algorithm. In: 2011 Fourth International Workshop on Advanced Computational Intelligence (IWACI), pp. 45–49, October 2011