Modelling Unmanned Vehicles Mission Planning problems as Constraint Satisfaction Problems

Author: Cristian Oliver Ramírez Atencia

Advisor: David Camacho Fernández

A thesis submitted in partial fulfillment for the Master Degree on Research and Innovation in Information and Communication Technologies

in the
Escuela Politécnica Superior
Departamento de Ingeniería Informática

September 2014

"Planning is a process of choosing among those many options. If we do not choose to plan, then we choose to have others plan for us."

Richard I. Winwood

Abstract

Escuela Politécnica Superior
Departamento de Ingeniería Informática

Master on Research and Innovation in Information and Communication Technologies

by Cristian Oliver Ramírez Atencia

This Master Thesis [1] provides a first analysis of mission planning for Unmanned Air Vehicles (UAVs), dealing with multiple UAVs that must perform one or more tasks in a set of waypoints and specific time windows. The solution plans obtained should fulfill all the constraints given by the different components and capabilities of the UAVs involved over the time periods given. Therefore a Temporal Constraint Satisfaction Problem (TCSP) representation is needed.

In a first approach, a temporal constraint model is implemented and tested by performing Backtracking (BT) search in several missions. In this model, a set of resources and temporal constraints are designed to represent the main characteristics (task time, fuel consumption, ...) of this kind of aircrafts. On the other hand, BT algorithm is used to look through the whole solutions space to measure the scalability of the problem.

In a second approach, we consider a Constraint Satisfaction Optimization Problem (CSOP) with an optimization function to minimize the fuel cost, the flight time and the number of UAVs needed; and Branch & Bound (B&B) search is employed for solving this CSOP model. Finally, some experiments will be carried out to validate both the quality of the solutions found and the runtime spent to found them.

Keywords

Unmanned Aircraft Systems, Mission Planning, Temporal Constraint Satisfaction Problems, Backtracking, Branch & Bound

[1]This work has been funded by Airbus Defence & Space (Savier Project: FUAM-076915), and partially by Spanish Ministry of Science and Education (Project Code TIN2010-19872).

Resumen

Escuela Politécnica Superior

Departamento de Ingeniería Informática

Máster en Investigación e Innovación de las Tecnologías de la Información y las Comunicaciones

por Cristian Oliver Ramírez Atencia

El presente proyecto final de máster [2] muestra un primer análisis sobre planificación de misiones para Vehículos Aéreos no tripulados (UAVs), donde se trata con multiples UAVs que deben realizar una o más tareas en un conjunto de puntos o waypoints y en una ventana temporal específica. Los planes obtenidos como solución deben cumplir todas las restricciones dadas por los diferentes componentes y capacidades de los UAVs involucrados en un periodo de tiempo dado. Por tanto, se precisa de una representación del problema como un Problema de Satisfacción de Restricciones Temporales (TCSP).

En una primera aproximación, se implementa un modelo de restricciones temporales y se testea ejecutando una búsqueda Backtracking (BT) cronológico en varias misiones. En este modelo, se diseñan un conjunto de restricciones temporales y de recursos para representar las principales características (tiempo de la tarea, consumo de combustible, ...) de este tipo de aviones. Por otro lado, el algoritmo BT es usado para examinar todo el espacio de soluciones para medir la escalabilidad del problema.

En una segunda aproximación, consideramos un Problema de Optimización de Satisfacción de Restricciones (CSOP) con una función de optimización que minimice el coste de combustible, el tiempo de vuelo y el número de UAVs necesarios; y se utiliza Branch & Bound (B&B) para resolver este modelo de CSOP. Finalmente, se realizarán algunos experimentos para validar tanto la calidad de las soluciones encontradas como el tiempo de ejecución gastado en su búsqueda.

Palabras Clave

Sistemas Aéreos no tripulados, Planificación de Misiones, Problemas de Satisfacción de Restricciones Temporales, Backtracking, Branch & Bound

[2]Este trabajo ha sido financiado por Airbus Defence & Space (Savier Project: FUAM-076915), y parcialmente por el Ministerio Español de Educación y Ciencia (Código de Proyecto TIN2010-19872).

Acknowledgements

First of all, I want to express my deepest gratitude to my family for the effort they have put in to give me the best education and always supporting me in everything I have done. Thanks to my parents for giving me the possibility to study the degree and master degree that I wanted, even it was at a University long from home. Thanks to my sister for being so kind, and to my grandparents for having always believed in and been proud of me.

Secondly, I want to thank my friend Víctor R.F. for making my university life easier and funnier, for helping me and for the great moments lived together. There have been 6 years sharing life experiences that I will never forget.

I would like to thank Professor David Camacho, for giving me the opportunity to work in this project, and Professor María D. Rodríguez Moreno for her tutorship and time dedicated in its achievement. Thanks to all my workmates for their help, Fernando Palero, Héctor Menéndez, and specially to Gema Bello Orgaz, my mentor, without her guidance and persistent help this dissertation would not have been possible.

Finally, I would like to acknowledge the financial support given by Airbus Defence & Space under the Savier project (FUAM-076915), as well as all the information provided from Savier Open Innovation project members: José Insenser, César Castro and Gemma Blasco.

Contents

List of Figures

List of Tables

Abbreviations

AA	**A**uction **A**lgorithms
AC	**A**rc **C**onsistency
AI	**A**rtificial **I**ntelligence
B&B	**B**ranch **&** **B**ound
BC	**B**ack**C**hecking
BE	**B**ucket **E**limination
BJ	**B**ack**J**umping
BM	**B**ack**M**arking
BPA	**B**asic **P**oint **A**lgebra
BT	**B**ack**T**racking
BTR	**B**asic **T**emporal **R**elations
CPA	**C**onvex **P**oint **A**lgebra
CSOP	**C**onstraint **S**atisfaction **O**ptimization **P**roblem
CSP	**C**onstraint **S**atisfaction **P**roblem
DAC	**D**irectional **A**rc **C**onsistency
DPC	**D**irectional **P**ath **C**onsistency
FC	**F**orward **C**hecking
GA	**G**enetic **A**lgorithm
GCS	**G**round **C**ontrol **S**tation
GLS	**G**reedy **L**ocal **S**earch
GT	**G**enerate-and-**T**est
HC	**H**ill **C**limbing
HCI	**H**uman **C**omputer **I**nterface
IA	**I**nterval **A**lgebra
LAS	**L**ook **A**head **S**chema

LBS	**L**ook **B**ack **S**chema
MAC	**M**aintaining **A**rc **C**onsistency
MC	**M**in-**C**onflic
MCRW	**M**in-**C**onflict-**R**andom-**W**alk
MDP	**M**arkov **D**ecision **P**rocess
MILP	**M**ixed-**I**nteger **L**ineal **P**rogramming
MOEA	**M**ulti-**O**bjective **E**volutionary **A**lgorithm
MTCSP	**M**aximal **T**emporal **C**onstraint **S**atisfaction **P**roblem
NC	**N**ode **C**onsistency
NSGA-II	**N**on-dominated **S**orting **G**enetic **A**lgorithm-**II**
PA	**P**oint **A**lgebra
PC	**P**ath **C**onsistency
PCSP	**P**artial **C**onstraint **S**atisfaction **P**roblem
PDL	**P**rocedure **D**efinition **L**anguage
PLA	**P**artial **L**ook **A**head
POF	**P**areto **O**ptimal **F**rontier
POFC	**P**artial-**O**rder **F**orward-**C**haining
RDS	**R**ussian **D**oll **S**earch
RHTA	**R**eceding-**H**orizon **T**ask **A**ssignment
RPC	**R**estricted **P**ath **C**onsistency
SA	**S**imulated **A**nnealing
SDRW	**S**teepest-**D**escent-**R**andom-**W**alk
SI	**S**warm **I**ntelligence
SPEA2	**S**trength **P**areto **E**volutionary **A**lgorithm **2**
TAL	**T**emporal **A**ction **L**ogic
TCSP	**T**emporal **C**onstraint **S**atisfaction **P**roblem
TS	**T**abu **S**earch
UAS	**U**nmanned **A**ircraft **S**ystem
UAV	**U**nmanned **A**ir **V**ehicle
WCOP	**W**eight **C**onstraint **O**ptimization **P**roblem

Chapter 1

Introduction

1.1 Motivation

A Unmanned Aircraft System (UAS) ground control station hosts one or several operators who could take different roles along the course of a mission. Typically, navigation modes of the UAS are based on automatic and semi-automatic mechanisms and it does not allow operators to manually control the Unmanned Air Vehicle (UAV) surfaces. Mission plans are pre-loaded into the UAV pre-takeoff, so it is conceivable that the UAS operator can just sit back and monitor the mission and/or exploit sensor data. However, as no mission can be completely predicted, likely mission re-planning and semi-automatic commands will be required during the mission execution.

Nowadays, each Ground Control Station (GCS) has been designed to control only a specific UAV. All the waypoints to follow and their associated tasks defined in the mission have to be manually inserted by the UAS operators. In addition, the tactical scenarios for the missions are on real-time and dynamic. Many changes can affect the pre-loaded plan during its execution, and the operators have to manually re-plan the mission again. The complexity and effort necessary to perform all these manual activities by the operators is very high.

Moreover, future GCSs are likely to manage missions involving multiple UAVs, so it is required that those manual tasks are automatized or simplified in order to reduce operator's workload. For this purpose, Automated Planning techniques can help to

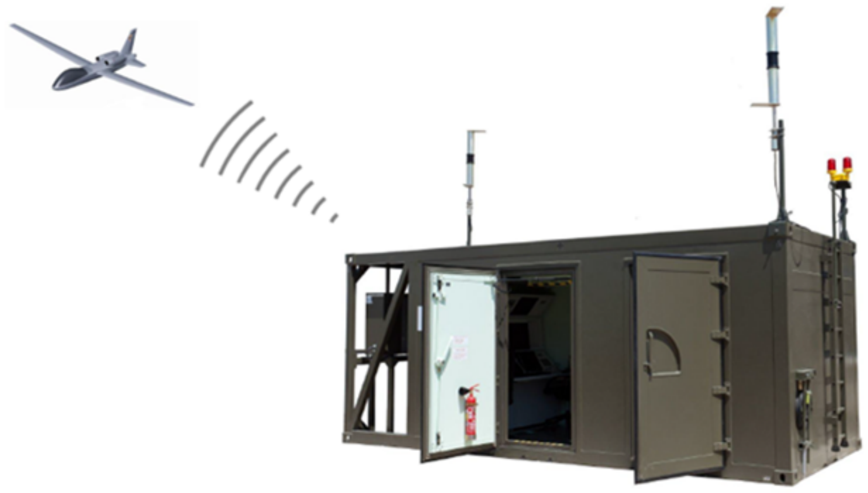


FIGURE 1.1: GCS controlling a UAV.

build a planner able to generate several plans by interpreting the model of the mission through a standard planning engine.

In general, in planning and scheduling problems, a set of activities and available resources, temporal constraints and a performance measurement are given as an input. Then as an output, the system will find the best assignment between the resources and the activities satisfying the time constraints, and maximizing the performance. This is common to many different engineering domains such as workflow problems, production scheduling, or planning space missions.

These planning problems can be solved using different optimization methods such as Mixed-Integer Linear Programming (MILP) [1], Simulated Annealing (SA) [2] or Auction Algorithms (AA) [3], among others. Usually, these methods are the best way to find the optimal solutions but, as the number of restrictions increase, the complexity of the problem grows exponentially because it is a NP-hard problem.

Other modern approaches formulate the mission planning problem as a Constraint Satisfaction Problem (CSP) [4], where the tactic mission is modelled and solved using constraint satisfaction techniques.

The main goal of this work is to develop a mission planner which can deal with multiple UAVs that must perform several tasks in different zones and in specific time windows. The different components and capabilities of the UAVs involved in the mission entail several constraints that must be fulfilled by the solution plans generated. Therefore, it is necessary to model the problem as a Temporal Constraint Satisfaction Problem (TCSP).

We will study several actual CSP solvers in order to select the best one in terms of quality and runtime, and then use it to model and solve our mission planning problem. The resolution of the problem will be carried out in two experiments. The first one will focus on the search of the complete space of solutions of the problem, using chronological Backtracking (BT). The second one will look for "optimal" solutions that minimize some objective variables (the flight time, the fuel consumption and the number of vehicles used) using Branch & Bound (B&B).

1.2 Objectives

The aim of this project is to model a UAVs Mission Planning problem as a TCSP, and its subsequent resolution by a search algorithm. To do this, the following milestones have been carried out:

- A review and analysis of existing tools to solve CSPs has been performed. In this study, the best tool in terms of quality, optimal runtime and documentation to facilitate the learning of the same is selected.
- Then, the modelling of the mission planning problem for UAS as a TCSP using the selected tool has been performed. Thanks to the knowledge acquired during studies conducted in the course Introduction to Research and Innovation, a simple model was made, but rather close to the actual models used today in mission planning.
- Then we proceeded to solve TCSP using some search algorithm. For a first simple approach, chronological BT algorithm is used.
- On the other hand, the search for optimal solutions (minimizing the fuel consumption, flight time and number of vehicles) is carried out considering the problem as a Constraint Satisfaction Optimization Problem (CSOP). The modelling is the same but adding a cost function and choosing B&B algorithm for the resolution.
- Finally, we have studied the temporal scalability and the number of solutions obtained in the search of solutions, both for complete and optimal approaches.

1.3 Document structure

This document is structured as follows: chapter 2 shows the state of the art in the aforementioned topics of Mission Planing and CSPs. Chapter 3 describes the architecture model implemented for the UAV mission planning problem, including the modelling of the problem as a TCSP. Chapter 4 describes the implementation and experimental setup of the problem, and chapter 5 explains the experimental results obtained for the search of the whole space of solutions and the search of some optimal solutions. Finally, the last chapter presents the final analysis and conclusions of this work and future lines of research.

Chapter 2

Related Work

In this chapter, we introduce a state of the art on Mission Planning problems, focusing on collaborative missions. Then, in a second section, we provide a basic background on CSPs and their temporal (TCSP) and optimization (CSOP) approaches.

2.1 Mission Planning

Planning has been an area of research in Artificial Intelligence (AI) for over three decades [5]. A variety of tasks including robotics [6], web-based information gathering[7][8], autonomous agents [9] and mission control [10][11] have benefited from planning techniques. Therefore, mission planning is a common problem in AI.

Sometimes, these planning problems involve considering dynamic environments [12][13] and/or cooperation between interacting agents.

Now, in a first subsection, we will talk about mission planning for UAS and the main state of the art approaches. Secondly, we will focus on collaborative mission planning and talk about some approaches in the field.

2.1.1 Mission Planning for UAS

Mission planning for UAS can be defined as the planning process of the locations to visit (waypoints) and the vehicle actions to do (loading/dropping a load, taking videos/pictures, acquiring information), typically over a time period. Functionally, mission planning resides above the process of path planning, where the mission planner generates a desired mission plan, and then the path planner generates the flight plan (trajectories) between the waypoints.

In the literature there are some attempts to implement UAS guidance systems that achieve mission planning. Doherty et al.[14] presented an architectural framework for mission planning and execution monitoring and its integration into a fully deployed unmanned helicopter. The knowledge gathered from the sensors during plan execution is used to create state structures, incrementally building a partial logical model representing the actual development of the system and its environment over time. Then planning and monitoring modules use Temporal Action Logic (TAL) for reasoning about actions and changes.

NASA/Army autonomous rotorcraft project developed a guidance system for the autonomous surveillance planning problem for multiple and varying targets [15], which generates mission plans using a decision theoretic approach. High-level autonomous control is provided by Apex framework [16], a reactive, procedure based planner/scheduler used for mission-level task execution, navigation. Apex synthesizes a course of action mainly by linking together elemental procedures expressed in Procedure Definition Language (PDL), a notation developed specifically for the Apex reactive planner. This guidance system has been integrated into a robotic helicopter and flight tested in more than 240 scenarios.

A similar project, called ReSSAC (Search and Rescue by Cooperative Autonomous System), was carried out by the French Aerospace Lab (ONERA) for a search and rescue scenarios [17]. This architecture for an exploration mission has been developed based on the idea of decomposing the mission into a sequence of tasks or macro-actions associated with rewards. The problem has been modeled using a Markov Decision Process (MDP) framework and dynamic programming algorithms for the mission planning. Konigsbuch [18] extends the Guidance System and integrates it in a robotic helicopter.

Finally, German Aerospace Centre (DLR) also developed a mission management system based on the behavior paradigm [19], which has been integrated onboard the ARTIS helicopter and validated in different scenarios, including waypoint following and search and track missions.

2.1.2 Collaborative Mission Planning

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. Besides, techniques and algorithms for cooperative missions can be divided into two main categories: cooperative perception and cooperative mission planning and decision-making [20].

The COMETS25 project [21] is one of the main projects for cooperative perception that implements a system for cooperative activities using heterogeneous UAS such as unmanned helicopters and blimps. This cooperative system processes data from the different vehicles for fire detection/alarm confirmation, localization, and monitoring. When a fire alarm is detected and localized, the mission is replanned to send more UAVs to confirm the alarm. In the cooperative perception area also the Aerospace Control Lab (ACL) at MIT has been studying and testing UAS cooperation using the RAVEN platform [22]. This research work is addressed to the problem of persistent vision-based search and track using multiple UAVs.

Regarding cooperative mission planning, there are few contributions that deal with multi UAS problems in a deliberative paradigm (cooperative task assignment and mission planning). A mission planner should provide a list of tasks assignment, where each task is assigned to an available vehicle that should perform this task. This assignment is based on information about the tasks and the capabilities of the vehicles.

Bethke et al. [23] proposes an algorithm for cooperative task assignment that extends the Receding-Horizon Task Assignment (RHTA) algorithm [24] developed at MIT. This algorithm solves an optimization problem to select the optimal sequence of tasks for each UAS by breaking it down to smaller problems and iteratively solving them using Petal algorithm [25].

Finallly, Kvarnstrom et al. [26] proposed a new mission-planning algorithm for collaborative UAS based on combining ideas from forward-chaining planning with partial-order planning, leading to a new hybrid Partial-Order Forward-Chaining (POFC) framework. This framework meets certain degree of centralization and abstraction for understanding and eventually signing off on potential plans, which is necessary in realistic environments such as natural and man-made catastrophes where emergency services personnel are involved.

2.2 Constraint Satisfaction Problems

A mission can be described as a set of goals that are 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, like a CSP does. A CSP can be defined as [27]:

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

In a CSP, the environment of the problem is represented by a state space. A path through the state space from the initial state to a goal state is a solution.

In the initial space all the variables are unassigned, and using some operators, they will be assigned a value from their domains. Then, the goal test function will check if all variables are assigned and all constraints satisfied. The goal test is decomposed into a set of constraints on variables rather than being a "black box.".

The domain D of each variable V can be discrete or continuous. In discrete CSPs, where the domains are finite, constraints can be represented simply by enumerating the allowable combinations of values.

Constraints come in several varieties. Unary constraints concern the value of a single variable, Binary constraints relate pairs of variables, Higher-order constraints involve three or more variables, and Global Constraints apply to all the variables. Finally, constraints can be absolute constraints, violation of which rules out a potential solution, or preference constraints that say which solutions are preferred.

Theoretically, solving a CSP is trivial using systematic exploration of the solution space, but this is not efficient practically. An example is the generate-and-test (GT) constraint satisfaction algorithm, which generates a random complete labelling of variables and test its constraint satisfaction.

Consistency techniques [28] are methods to solve CSPs based on removing inconsistent values from the variables' domains. These techniques are deterministic, but most of them are not complete. The main techniques are:

- Node Consistency (NC), which removes values from variables domains that are inconsistent with unary constraints on the respective variable.
- Arc Consistency (AC), which removes values from variables domains that are inconsistent with binary constraints. The arc (V_i, V_j) is arc consistent if and only if $\forall x \in D_i$ current domain of $V_i, \exists y \in D_j$ current domain of V_j such that $V_i = x$ and $V_j = y$ is permitted by the binary constraint between V_i and V_j. There are several AC algorithms named from AC-1 to AC-7. The AC-3 algorithm performs re-revisions only for those arcs that are possibly affected by a previous revision. The AC-4 works with individual pairs of values to remove potential inefficiency of checking pairs of values again and again.
- Path Consistency (PC), which requires for every pair of two variables X, Y satisfying the respective binary constraint that there exists a value for each variable along some path between X and Y such that all binary constraints in the path are satisfied.
- K-consistency, which involves that if for every system of values for K-1 variables satisfying all the constraints among these variables, there exists a value for arbitrary K-th variable such that the constraints among all K variables are satisfied. Strongly K-consistency is J-consistency $\forall JK$.

Restricted forms of these techniques, as Directional Arc Consistency (DAC) (revises each arc only once), Directional Path Consistency (DPC) (revises each path only once) or

Restricted Path Consistency (RPC) (extends AC-4 to some form of PC), remove similar amount of inconsistencies but they are more efficient.

Both systematic search and constraint techniques are used simultaneously to solve CSPs (Constraint Propagation). There are two main schemas for this approach: the Look Back Schema (LBS), which uses consistency checks among already instantiated variables; and the Look Ahead Schema (LAS), which avoid late detection of conflicts.

The most known method on LBS is BT [29] which incrementally extends a partial solution towards a complete solution by repeatedly choosing a value for another variable consistent with the values in the current partial solution. If a partial solution violates any of the constraints, backtracking is performed to the most recently instantiated variable that still has alternatives available. BT is strictly better than random generate-and-test algorithm, however, its running complexity for most nontrivial problems is exponential. This method has three principal problems: thrashing (avoided in Backjumping (BJ)), redundant work (avoided in Backchecking (BC) and Backmarking (BM)) and late detection of conflicts (which is avoided in LASs).

The main strategies on LAS are Forward Checking (FC) (which performs AC between pairs of not yet instantiated variable and instantiated variable) [30], Partial Look Ahead (PLA) (similar to FC but using DAC) and Full Look Ahead or Maintaining Arc Consistency (MAC) (which uses full AC after each labelling step). Figure 2.1 shows where each method makes consistency checks.

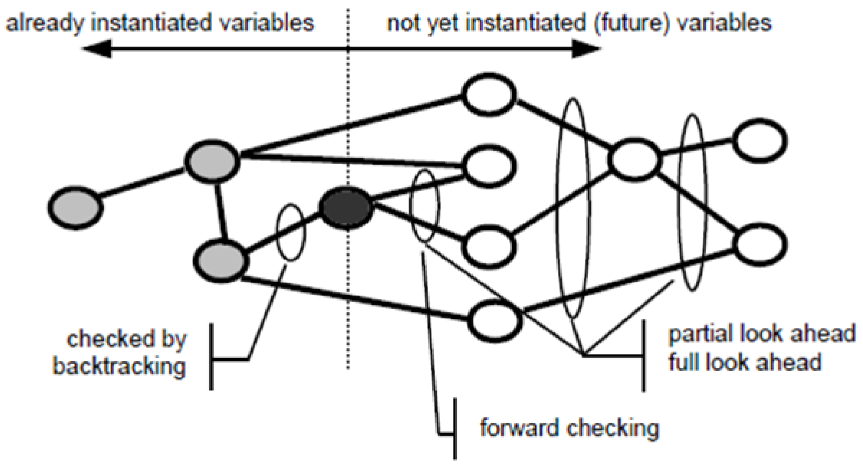


Figure 2.1: Comparison of propagation techniques.

On the other hand, Greedy Local Search (GLS) strategies used in Stochastic and Heuristic Algorithms have become popular over the last decade. Most known stochastic algorithms are:

- Hill Climbing (HC). It starts with an initial random labelling of variables and, at each step, it changes a value of some variable in such a way that the resulting labelling satisfies more constraints. If a strict local minimum is reached then the algorithm restarts at other randomly generated state. The algorithm stops as soon as a global minimum is found, i.e., all constraints are satisfied, or some resource is exhausted.
- Min-Conflicts (MC). It avoids exploring the whole state's neighbourhood like HC. This heuristic chooses randomly any conflicting variable and then picks a value which minimises the number of violated constraints. If no such value exists, it picks randomly one value that does not increase the number of violated constraints.
- Tabu Search (TS). It avoids cycling and getting trapped in local minimum by preventing using the configurations of a tabu list, i.e. a special short term memory that maintains a selective history, composed of previously encountered configurations or more generally pertinent attributes of such configurations.

Sometimes they used the Random Walk strategy to avoid local-minimum, as in MC (Min-Conflicts-Random-Walk (MCRW)) or HC (Steepest-Descent-Random-Walk (SDRW)).

2.2.1 Temporal Constraint Satisfaction Problems

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 [31]. Different classes of constraints are characterized by the underlying set of Basic Temporal Relationss (BTRs). The main classes of TCSPs and their corresponding BTRs can be shown in Table 2.1.

In the related literature, Mouhoub [32] proved that on real-time or Maximal Temporal Constraint Satisfaction Problem (MTCSP), the best methods for solving them were MCRW in the case of under-constrained and middle-constrained problems. In the over-constrained case, TS and SDRW would be the best choice. He also design an algorithm

Table 2.1: TCSP approaches and their BTRs.

TCSP	Variable types	Algebras	BTR
Qualitative Point	Time points	Basic Point Algebra (BPA)	$<, =, >, ?$
		Convex Point Algebra (CPA)	$\varnothing, <, =, >, \leq, \geq, ?$
		Point Algebra (PA)	$\varnothing, <, =, >, \leq, \geq, ?, \neq$
Qualitative Interval	Time intervals	Interval Algebra (IA)	before, after, meets, meetBy, overlaps, overlapsBy, during, contains, equal, starts, startedBy, finishes, finishedBy
Metric Point	Time points		

(AC-3.1—DC) based on the AC-3 algorithm implemented for dynamic environments that gave efficient time and space results.

In other works, Mouhoub developed a temporal model, TemPro [33], which was based on interval algebra, to translate an application involving temporal information into a CSP.

Ragni [34] used Allen's IA and Franks cardinal direction calculus (CD) to create the temporalized cardinal direction calculus (TCD), which allows to encode temporalized spatial constraint satisfaction problems as deterministic planning problems.

A TCSP can perfectly represent an UAS mission as a set of temporal constraints over the time the tasks in the mission start and end. Besides the temporal constraints, the problem has various constraints imposed by the proficiency of the UAVs to perform the tasks.

2.2.2 Constraint Satisfaction Optimization Problems

In many real-life applications it is necessary to find a good solution, and not the complete space of possible solutions. CSOP consists of a standard CSP and an optimization function (objective function) that maps every solution (complete labelling of variables) to a numerical value measuring the quality of the solution.

There are several methods for solving CSOP such as Russian Doll Search (RDS) [35], Bucket Elimination (BE) [36], Genetic Algorithm (GA) [37] and Swarm Intelligence (SI) [38]. The most widely used algorithm for finding optimal solutions in CSOP is called B&B [39][40]. This algorithm searches for solutions in a depth first manner and behaves like BT except that as soon as a value is assigned to the variable, the value of heuristic function for the labelling is computed. If this value exceeds the bound (initially set to minus or plus infinity given it is a minimization or maximization problem), then the sub-tree under the current partial labelling is pruned immediately. The efficiency of B&B is determined by two factors: the quality of the heuristic function and whether a good bound is found early.

In many problems, it is necessary to optimize several variables all together, and the optimization function becomes a Multi-Objective function. In most of these cases, the optimality of the solutions is analysed looking at the Pareto Optimal Frontier (POF).

Other approaches avoid computing the POF by using Soft Constraints and/or mapping the objectives into a single weighted cost function. Torrens called this approach Weight Constraint Optimization Problem (WCOP) [41].

A common approach for computing an approximation of the POF are Multi-Objective Evolutionary Algorithms (MOEAs). These algorithms has been used in Mission Planning problems in recent researches [42]. The most known MOEA methods are Strength Pareto Evolutionary Algorithm 2 (SPEA2) [43] and Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [44]. SPEA2 uses a regular population and an archive (external set which will contain the POF), which is initially empty. After the fitness assignment phase, the archive is updated with the nondominated individuals from both the population and the archive (environmental phase). This method performs a mating selection, which consists on binary tournament selection with replacement on the archive obtained from the environmental phase in order to fill the mating pool. On the other hand, NSGA-II performs a nondominated sorting of the individuals of the population, followed by a crowding-distance sorting. That is, between two solutions with differing nondomination ranks, the crowding-distance sorting prioritizes the solution with the lower (better) rank; otherwise, if both solutions belong to the same front, then it prioritizes the solution that is located in a lesser crowded region.

Chapter 3

Architecture Model for UAV Mission Planning

UAV missions consists of a number n of tasks performed by a team of m UAVs. The main goal to solve the Misison Planning problem is to assign each task with an UAV that is able to perform it in a departure time sufficient to reach the task area in time. Note that the UAV could be parked at an airport or in flight after performing a previous task. Figure 3.1 shows an overview of the mission planning process.

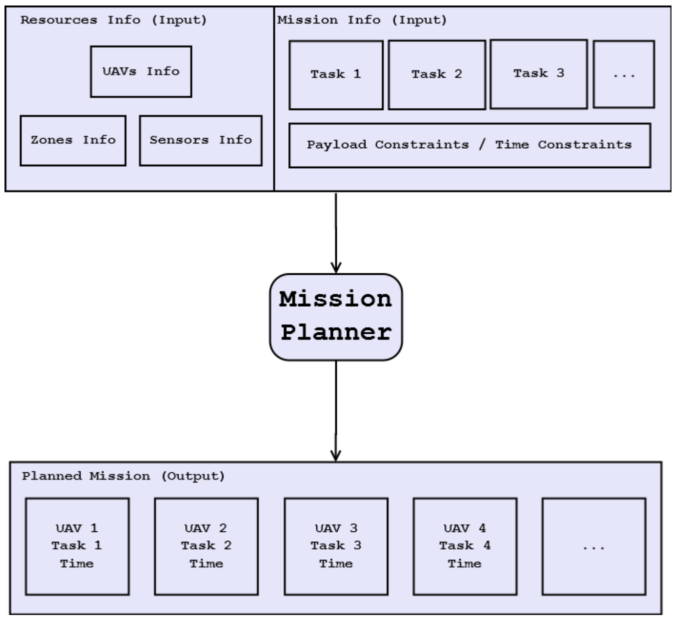


FIGURE 3.1: Mission Planning overview.

In this process, the Mission Planner receives a big amount of data about the environment, the available vehicles and its sensors and the mission for the purpose of using it for planning. The mission planner uses this information to compute the plans and then returns a set of tuples $< Task, Vehicle, Time >$ that specifies what tasks must do a UAV in a determinate moment.

Aiming to reproduce this functionality, we have developed a Mission Planning framework, that will be shown in next section.

3.1 Framework Architecture

The architecture of the framework developed is shown in Figure 3.2. In this architecture, the Mission Planner, which is placed in the Mission Planning Module, uses a CSP solver, in this case Gecode [45], to model and solve the TCSP model that will be explained in section 3.3. The planner receives the resource information (i.e. the information about the zones, sensors and UAVs involved in the mission), which is a static information stored in the system. On the other hand, the operator of the mission, through a Human Computer Interface (HCI), provides the information about the mission (i.e. its tasks). After the execution of the Mission Planner, it returns a set of plans or solutions, which contain the tuples $< Task, Vehicle, Time >$ and some extra information about the estimated parameters of the mission, such as the fuel consumption, the speed of the vehicles, the total flight time, etc.

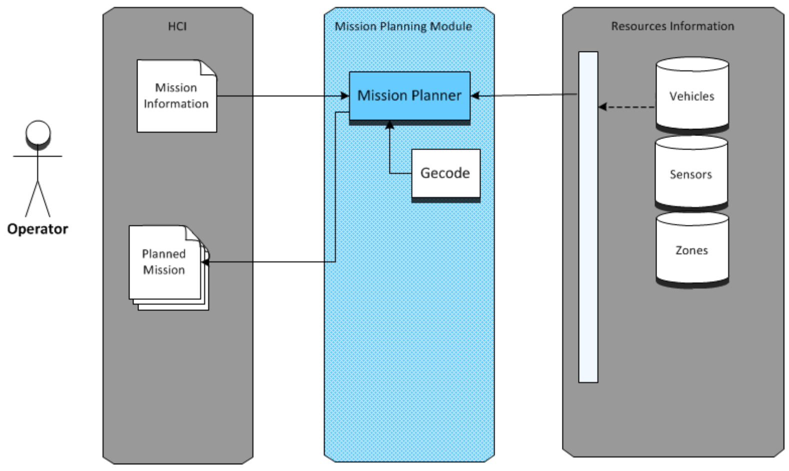


Figure 3.2: Mission Planning Framework Architecture.

The following sections will describe in detail the data model used in the Mission Planner and the TCSP modelling of the mission planning problem.

3.2 Mission Data Model

The data model used in the Mission Planner can be divided in four components related to the Sensors (or Payloads), Zones, UAVs and Mission (or Tasks) information. The next subsections explain each one of these components in detail.

3.2.1 Sensors

Sensors or payloads are attached to vehicles and they permit developing the tasks of the mission, such as taking photos or tracking a zone. Figure 3.3 shows a UML data model of class sensor and its main subclasses.

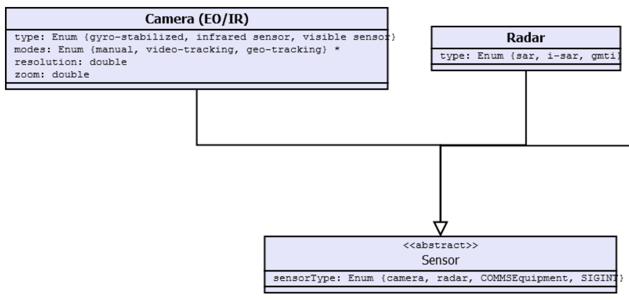

Camera (EO/IR)
type: Enum {gyro-stabilized, infrared sensor, visible sensor}
modes: Enum {manual, video-tracking, geo-tracking} *
resolution: double
zoom: double
Radar
type: Enum {sar, i-sar, gmti}
COMMS Equipment
<<abstract>>
Sensor
sensorType: Enum {camera, radar, COMMSEquipment, SIGINT}

Figure 3.3: Sensor UML Data Model.

In this work, we have considered three different sensors:

- Camera Electro-Optical/Infra-red (EO/IR): This sensor allow the UAV to take photos. It has some internal features, such as the type of the camera, its zoom, resolution and its modes.
- Radar: It allows to track the elements in a zone near the vehicle. Its main feature is the type of the radar (SAR, I-SAR or GMTI).
- Communications Equipment: This equipment allows the UAV to communicate and send real-time pictures to the GCS.

3.2.2 Zones

Zones or areas are used to represent the place where the tasks of the mission are developed. Figure 3.4 shows a UML data model of class Zone and its associate classes.

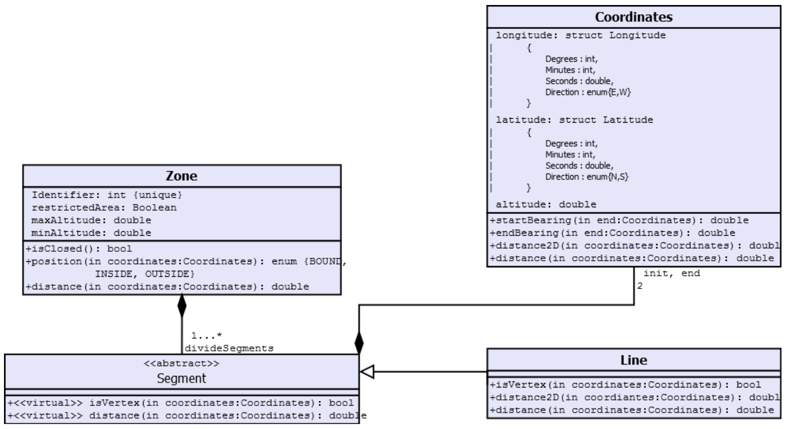

Coordinates
longitude: struct Longitude
{
Degrees : int,
Minutes : int,
Seconds : double,
Direction : enum{E,W}
}
latitude: struct Latitude
{
Degrees : int,
Minutes : int,
Seconds : double,
Direction : enum{N,S}
}
altitude: double
+startBearing(in end:Coordinates): double
+endBearing(in end:Coordinates): double
+distance2D(in coordinates:Coordinates): doubl
+distance(in coordinates:Coordinates): double
Zone
Identifier: int {unique}
restrictedArea: Boolean
maxAltitude: double
minAltitude: double
+isClosed(): bool
+position(in coordinates:Coordinates): enum {BOUND,
INSIDE, OUTSIDE}
+distance(in coordinates:Coordinates): double
init, end
2
1...*
divideSegments
<<abstract>>
Segment
+<<virtual>> isVertex(in coordinates:Coordinates): bool
+<<virtual>> distance(in coordinates:Coordinates): double
Line
+isVertex(in coordinates:Coordinates): bool
+distance2D(in coordiantes:Coordinates): doubl
+distance(in coordinates:Coordinates): double

FIGURE 3.4: Zone UML Data Model.

3.2.2.1 Coordinates

The class *Coordinates* is used to represent a geographic point. It is composed by its Longitude, Latitude and Altitude. In this work, the distance between two points (name them, $x_1[long_1, lat_1, alt_1]$ and $x_2[long_2, lat_2, alt_2]$) is computed using the Haversine formula with the latitude and longitude:

$$d_{2D}(x_1, x_2) = 2r_{EARTH} \arcsin \sqrt{\sin^2(\frac{lat_2 - lat_1}{2}) + \cos(lat_1)\cos(lat_2)\sin^2(\frac{long_2 - long_1}{2})} \tag{3.1}$$

and then the Euclidean distance with the resulting and the altitude

$$d_{3D}(x_1, x_2) = \sqrt{d_{2D}^2(x_1, x_2) + (alt_2 - alt_1)^2}. \tag{3.2}$$

Besides, the bearing between two points is computed as:

$$\theta_{12} = arctan2(\sin(long_2 - long_1)\cos(lat_2), \cos(lat_1)\sin(lat_2) - \sin(lat_1)\cos(lat_2)\cos(long_2 - long_1)) \tag{3.3}$$

3.2.2.2 Line

On the other hand, the class *Line*, which extends from class *Segment*, is composed by two points. The 2D distance from a line (name it, l_1 with points x_1 and x_2) to a external point (name it, $x_3[long_3, lat_3, alt_3]$) is computed using the cross-track distance:

$$d_{2D}(l_1, x_3) = \arcsin(\sin(\delta_{13})\sin(\theta_{13} - \theta_{12}))r_{EARTH} \tag{3.4}$$

where δ_{13} is the distance between the point and the first vertex of the line, θ_{13} is the starting bearing between the first vertex of the line and the point and θ_{12} is the starting bearing between the first vertex and the second vertex of the line.

Then, the 3D distance is computed using Euclidean distance with the 2D distance and the altitude difference of the point to the closest point in the line:

$$d_{3D}(l_1, x_3) = \sqrt{d_{2D}^2(l_1, x_3) + (alt_{closest} - alt_3)^2}. \tag{3.5}$$

The altitude of the closest point is directly known from:

$$alt_{closest} = alt_1 + (alt_2 - alt_1) * \frac{\arccos(\frac{\cos(\delta_{13}/r_{EARTH})}{\cos(d_{2D}(l_1,x_3)/r_{EARTH})})r_{EARTH}}{\delta_{12}}. \tag{3.6}$$

3.2.2.3 Zone

The class *Zone* represents an area where a task is developed. An zone is composed by

- Several segments. As the only type of segment implemented is the line, a zone is a polygon (or a polygonal prism).
- An altitude window $[h_{min}, h_{max}]$ defined by the minimum and maximum altitude.
- A flag indicating whether the zone is restricted or not.

The class Zone determines whether a zone is closed or not if all the points of every segment is repeated at least twice. The position of a point respect to a zone is determined using the winding number algorithm:

Algorithm 1 Calculate the position of a point respect to a zone.

```
$cumulated = 0$
for each segment of the zone do
    if $(initLong - pointLong) \cdot (pointLong - endLong) \geq 0$ AND $(initLat - pointLat) \cdot (pointLat - endLat) \geq 0$ AND $(pointLongitude - initLongitude) \cdot (endLatitude - initLat) = (pointLat - initLat) \cdot (endLong - initLong))$ then
        if $pointAlt < minAlt$ OR $pointAlt > maxAlt$ then
            return OUTSIDE
        end if
        return BOUND
    end if
    $angle = point.endBearing(init)point.endBearing(end)$
    if $angle \geq PI$ then
        $angle- = 2 * PI$
    else
        if $angle \leq -PI$ then
            $angle+ = 2 * PI$
        end if
    end if
    $cumulated+ = angle$
end for
if $cumulated/PI = 0$ OR $pointAlt < minAlt$ OR $pointAlt > maxAlt$ then
    return OUTSIDE
else
    return INSIDE
end if
```

Finally, the distance from a point to a zone is computed as the minimum distance to any of the segments of the zone.

3.2.3 UAVs

A mission counts with a number m of available UAVs for its development. Each UAV (named it, UAV k) has some specific characteristics:

- Position and fuel at the beginning of the mission.
- Fuel consumption rate
- The maximum reachable speed $v_{k,max}$
- The minimum cruise speed $v_{k,min}$
- Maximum and minimum flight altitude $[h_{min}, h_{max}]$

- Permission to go to restricted zones
- Available sensors P_k (cameras, radars, communication equipments, ...)

Moreover, in each point in time, each UAV is positioned at some specific *coordinates*, flies at some specific *cruise speed* $v_{k \to i}$ and is filled with a specific amount of *fuel*.

Figure 3.5 shows the class UAV and its attributes. Some additional attributes, such as the maxFlightTime and the withinRange are not consider in the modelling, but they have been added for future works.

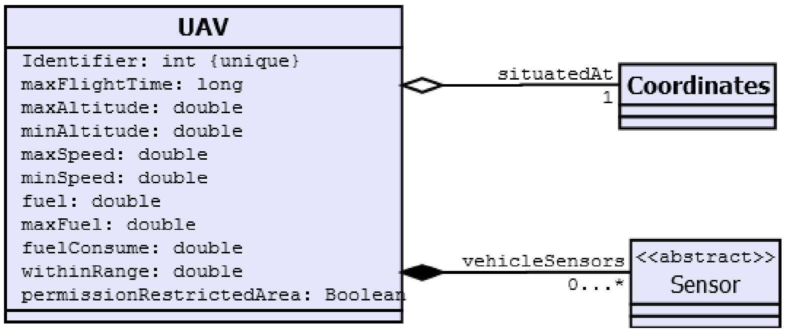


FIGURE 3.5: UAV UML Data Model.

3.2.4 Tasks

A mission consists of a set of n tasks to be performed. A task consists of performing an action in a specific zone, such as exploring the area or search for an object. Therefore, each task (name it, task i) consist of:

- An action, which can be carried out thanks to the sensors or payloads P_i belonging to a particular UAV. Table 3.1 shows the relation between actions and sensor needs.

TABLE 3.1: Different task actions considered

Action ID	Action	Sensors Needed
A0	Taking pictures of a zone	– Camera EO/IR
A1	Taking real-time pictures of a zone	– Camera EO/IR – Communications Equipment
A2	Tracking a zone	– Radar SAR

- Geographic area with altitude window $[h_{min}, h_{max}]$, which could be restricted.
- Time interval with duration τ_i and end time t_i
- Mean speed $\bar{v}_i$ at performing the task

Figure 3.6 shows the class *Task* and its attributes.

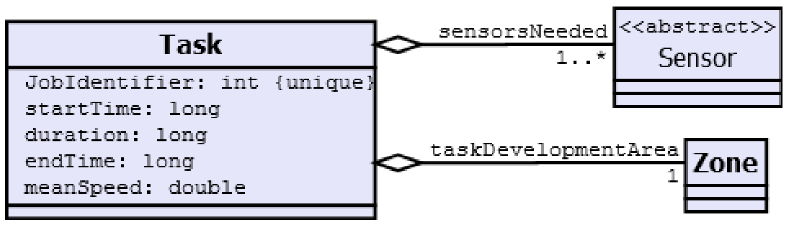


FIGURE 3.6: Task UML Data Model.

3.3 TCSP Mission Modelling

Nowadays there are several functionally CSP solvers developed. Our purpose is to use one of these already developed solvers to model and solve our Mission Planning problem.

For this purpose, different CSP solver technologies have been studied (see Appendix A) in order to choose the better one to be improved, not only the fastest but the most suitable to our aim.

From this study, Gecode [45] has been selected as the best tool for CSP solving in terms of efficiency. This tool will be used in the following section to model the mission planning problem.

3.3.1 TCSP Modelling using Gecode

One of the main advantages of using Gecode for TCSP modelling is that, in its most recent versions, it provides float variables, which can be used for defining all the real variables of the problem: times, speeds, distances, ...These float variables and the constraints involving them are internally solved through Allen's IA (see Chapter 2.2.1).

Now, the problem domain is modelled as a TCSP. The main variables are the *tasks* and their values will be the *UAVs* that perform each task and their respective *departure* *times.* Moreover, there are some additional variables: the cruise speed to reach the area of the task $v_{k\to i}$, the fuel cost, the distance travelled for each task; which can be deduced from tasks assignment and UAV characteristics.

Figure 3.7 shows an assignment of a UAV k to a task i. In this representation, it must be considered that:

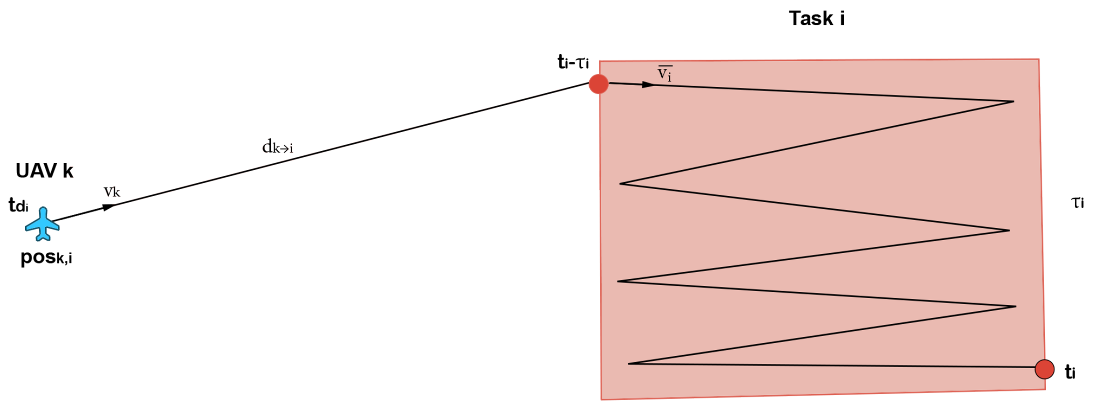


FIGURE 3.7: Scenario for performance of task i by UAV k.

- The vehicle is positioned at $pos_{k,i}$ at departure time t_{d_i}
- The distance travelled to reach the task area in time $d_{k\to i}$, is computed using the formulas from section 3.2.2:

$$d_{k\to i} = i.area.distance(pos_{k,i}) \tag{3.7}$$

- The flight time of the vehicle is

$$flightTime_i = \frac{d_{k\to i}}{v_{k\to i}} + \tau_i \tag{3.8}$$

- The fuel consumed by the vehicle is

$$f_i = k.fuelConsume * (d_{k\to i} + \tau_i \bar{v}_i) \tag{3.9}$$

The main constraints defined in this model are as follows:

1. Temporal constraints assuring a UAV does not perform two tasks at the same time. Let k be a UAV that executes two tasks i and j, where i takes place before j, then t_i must precede the departure time t_{d_j} (see Figure 3.8):

$$t_i \leqslant t_{d_j} = t_j - flightTime_i \tag{3.10}$$

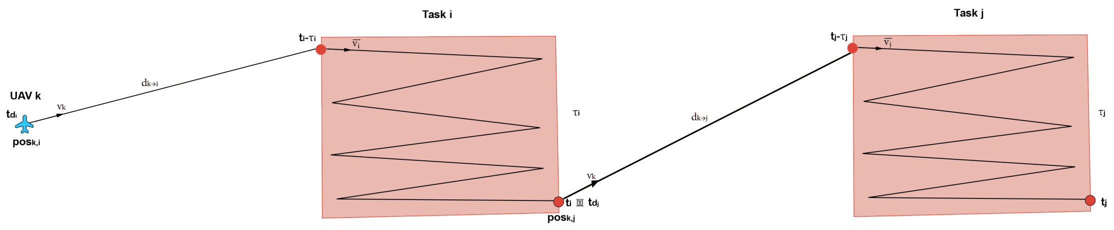


FIGURE 3.8: Scenario for performance of tasks i and j by UAV k.

2. Logical constraints:

   (a) Speed window constraints: the mean speed necessary to perform the task i, $\bar{v}_i$, must be contained in the speed window $v_{k,max}$ and $v_{k,min}$:

   $$v_{k,min} \leq \bar{v}_i \leq v_{k,max} \tag{3.11}$$

   (b) Altitude window constraints: a UAV k, with an altitude window $k_{h_{max}}$ and $k_{h_{min}}$, performing a task i developed in an area with an altitude window h_{max} and h_{min}, must obey:

   $$k.h_{max} \geq i.area.h_{max} \tag{3.12}$$

   $$k.h_{min} \leq i.area.h_{min} \tag{3.13}$$

   (c) Zone permission constraints: another constraint is the implication that a restricted area has in the tasks to perform. Just UAVs with permissions in those areas shall perform the tasks.

3. Resource constraints:

   (a) Sensor constraints: another constraint is whether a UAV carries the corresponding sensor to perform a task. Let P_k denote the sensors available for UAV k and P_i the sensors needed for the task i (performed by k), then:

   $$P_i \subseteq P_k, \tag{3.14}$$

(b) Fuel constraints: finally, we must constraint the fuel cost for each UAV. The fuel cost for a UAV k performing a task i is

$$f_i = k.fuelConsumeRate * (d_{k \to i} + \tau_i \bar{v}_i) \tag{3.15}$$

So the following inequality must be obeyed:

$$\sum_{i \in T_k} f_i \leqslant k.fuel \tag{3.16}$$

To compute the distance (needed for the compute of flight time, see Equation 3.8), it is necessary to know where the vehicle is located before the start of the task, i.e. its position $pos_{k,i}$. Therefore, we have created a $m \times n$ matrix of tasks to UAV position. This matrix is initialized with every row, i.e. the positions of a specific vehicle, to the initial position of that vehicle. Each time a task assignment is considered in the constraint propagation process, this matrix is updated with the computed position of each vehicle at the end of the task.

All the aforementioned variables and constraints have been computed in Gecode to represent the Mission Planning Model. In following chapters, it will be shown how to solve this model with some methods provided by the solver.

3.3.2 Optimization Function and Constraint Optimization Problem

As in many real-life applications, we just want to find some good solutions, what can be achieved considering a CSOP. In order to apply a method for solving CSOP, a new optimization function has been designed. This new function is looking to optimize (minimize) 3 objectives:

- The total fuel consumed, computed as the sum of the fuel consumptions for each task using equation 3.9.
- The number of UAVs used in the mission. A mission performed with a lower number of vehicles is usually better because the remaining vehicles can perform other missions at the same time.

- The total flight time, which is computed as the sum of the flight times for each task using equation 3.8.

As Gecode does not provide a method for computing the POF, our model uses weights to map these three objectives into a single cost function, as the similar approach WCOP [41]. This function is computed as the sum of percentage values of these three objectives, as shown in Equation 3.17. In this sense, in the second experimental phase, a comparative assessment of weights for finding feasible solutions of the problem will be carried out.

$$f_{cost}(i) = K_F \frac{Fuel(i)}{\max_j Fuel(j)} + K_U \frac{N^{\circ}UAVs(i)}{\max_j N^{\circ}UAVs(j)} + K_T \frac{FlightTime(i)}{\max_j FlightTime(j)}$$

$$K_F, K_U, K_T \in [0,1], \qquad K_F + K_U + K_T = 1 \tag{3.17}$$

Chapter 4

Experimental Setup

Since the Mission Planning problem is so complex and recent in the state of the art, there does not exist datasets or benchmarks available. For this reason, some simple datasets have been developed.

In the following sections, we explain the implemented datasets for missions (including the topology of the mission scenario) and teams of UAVs, and the different schemas considered according to the temporal preferences between tasks.

4.1 Missions datasets

There has been designed 10 missions, each one composed by an increasing number of tasks from 1 to 10, i.e the first mission has one task; the second, two tasks; and so on. Table 4.1 shows the 10 considered tasks, where the first mission will execute task with ID T1; the second will execute tasks with IDs T1 and T2; and so on. This table shows the duration of the tasks instead of the start and end times. These times will be fixed on the experimental phase depending on the number of dependencies between the tasks. The action IDs come from Table 3.1.

In this approach, we consider the simple topology specified in Figure 4.1, where coloured areas represent the areas where tasks are performed and helicopters represent the airports where UAVs are situated at the beginning of the mission. In this scenario, there are four areas and four airports.

Table 4.1: UAS mission with 10 tasks

Task ID	Action ID	Duration(min)	Zone altitude window (km)	Mean speed (km/h)	Restricted zone?
T1	A0	25	[1.5 – 5]	100	NO
T2	A2	20	[1.5 – 5]	100	NO
T3	A1	30	[2.5 – 6.15]	100	YES
T4	A0	25	[0.5 – 3.75]	100	NO
T5	A2	35	[0.5 – 3.75]	100	NO
T6	A1	30	[3.85 – 5]	100	NO
T7	A1	25	[3.85 – 5]	100	NO
T8	A1	12	[1.5 – 5]	100	NO
T9	A0	20	[1.5 – 5]	100	NO
T10	A2	25	[2.5 – 6.15]	100	YES

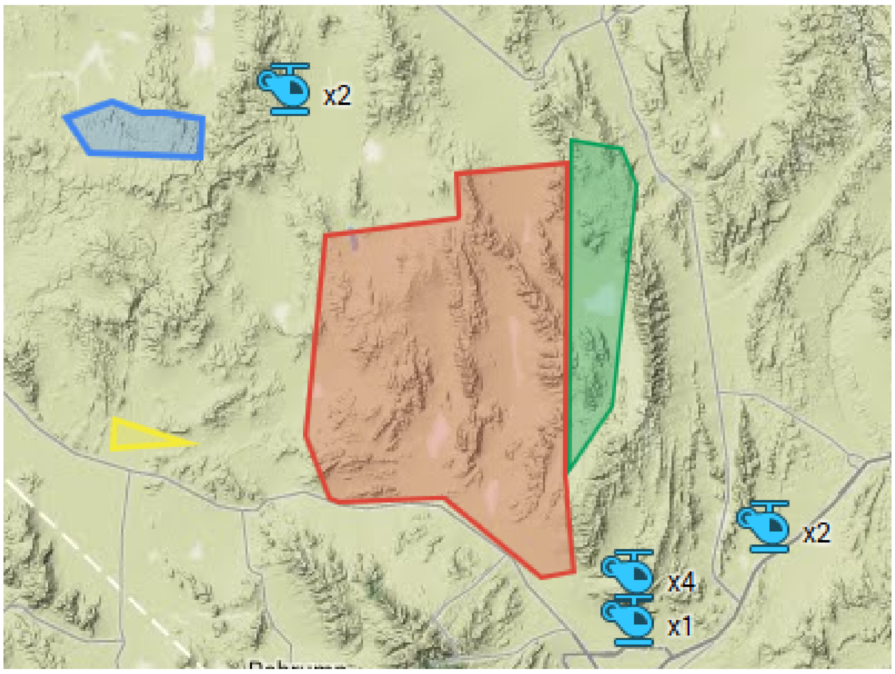


Figure 4.1: Topology of the scenario where missions are performed.

4.2 UAVs datasets

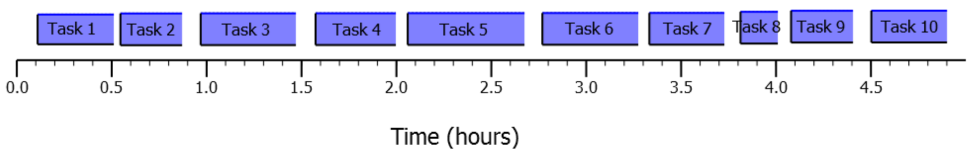
Different scenarios for solving the missions have been prepared with an increasing number of UAVs able to perform the tasks. The tasks contain several constraints, so when the number of tasks is very high, a high number of UAVs is also needed, mainly because of the fuel constraints. There has been considered groups of 1 to 9 vehicles available to perform the tasks (see Table 4.2). For a scenario with 1 vehicle, we use UAV with ID U1; for a scenario with 2 vehicles, UAVs with IDs U1 and U2; and so on.

Table 4.2: Team of 9 available UAVs

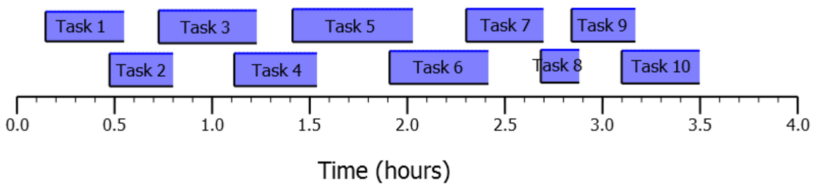
UAV ID	Cruise speed window (km/h)	Altitude window (km)	Restricted zone permission	Fuel consume (L/km)	Initial Fuel (L)	Sensors Available
U1	[90 – 110]	[0.3 – 6.5]	YES	0.159	97.52	• Camera EO/IR • Radar SAR • Communications Equipment
U2	[90 – 110]	[0.3 – 6]	NO	0.159	58.48	• Camera EO/IR
U3	[110 – 190]	[0.8 – 10]	YES	0.2	140.23	• Camera EO/IR • Radar SAR
U4	[90 – 110]	[0.3 – 6]	YES	0.159	47.12	• Camera EO/IR
U5	[90 – 110]	[0.3 – 6]	NO	0.159	101.48	• Camera EO/IR • Radar SAR • Communications Equipment
U6	[90 – 110]	[0.3 – 6]	NO	0.159	101.37	• Camera EO/IR • Radar SAR • Communications Equipment
U7	[90 – 110]	[0.3 – 6]	NO	0.159	58.15	• Camera EO/IR
U8	[110 – 190]	[0.8 – 10]	YES	0.2	140.23	• Camera EO/IR • Radar SAR
U9	[90 – 110]	[0.3 – 6]	YES	0.159	47.12	• Camera EO/IR

4.3 Temporal schemas

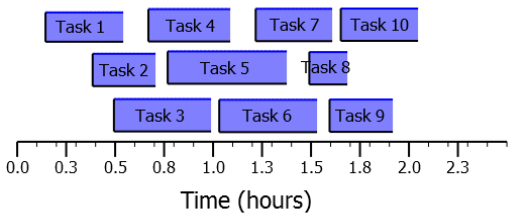
Three scenarios have been generated with different temporal schemas based on the time dependencies between the tasks. Figure 4.2a shows an scenario with no time dependencies between tasks, i.e. the tasks do not collide in time. Figure 4.2b shows an scenario where each task collides in time with the previous task, i.e. there are $n-1$ temporal dependencies, being n the number of tasks. Finally, when each task collides in time with the two previous tasks, i.e. there are $2(n-1)-1$ temporal dependencies, we have the scenario shown in Figure 4.2c.



(A) No dependencies



(B) Dependency of each task with the previous task.



(C) Dependency of each task with the two previous tasks.

FIGURE 4.2: Three schemas of the scenarios based on the number of temporal dependencies between the tasks.

These schemas will be compared in the experimental phase (see Chapter 5) in order to observe the scalability of the problem as the number of temporal dependencies increase.

Chapter 5

Experimental Results

5.1 Experiment 1: Search of the Complete Space of solutions with Backtracking

BT search implemented by Gecode solver has been used to solve the missions explained in the previous section, analysing the runtime spent in the process. This search algorithm performs constraint propagation with different consistency levels depending on the type of the constraint. For all the developed constraints in our problem, domain (or node) consistency is applied.

Due to some fuel and flight time constraints, with only 2 or 3 UAVs there is no solution for a high number of tasks (> 6), so the solver has been run in different scenarios with 4 to 7 vehicles to test the scalability of the problem.

In the following sections, the scalability of the runtime and number of solutions obtained is studied based on the three different schemas from section 4.3. First each schema is tested individually, and finally the three are compared between them.

5.1.1 Study with temporally independent tasks

Figures 5.1 and 5.2 shows the number of solutions and runtime obtained when the tasks do not collide in time (see Figure 4.2a). As we can see, the growth of the number of solutions is nearly exponential as the number of tasks increase. Indeed, the exponentiality is higher and more appreciable as the number of UAVs increase. For the runtime, the situation is similar, and the exponentiality growth is much higher. So it is clear that the scalability of the problem, as the number of variables increase, is exponential.

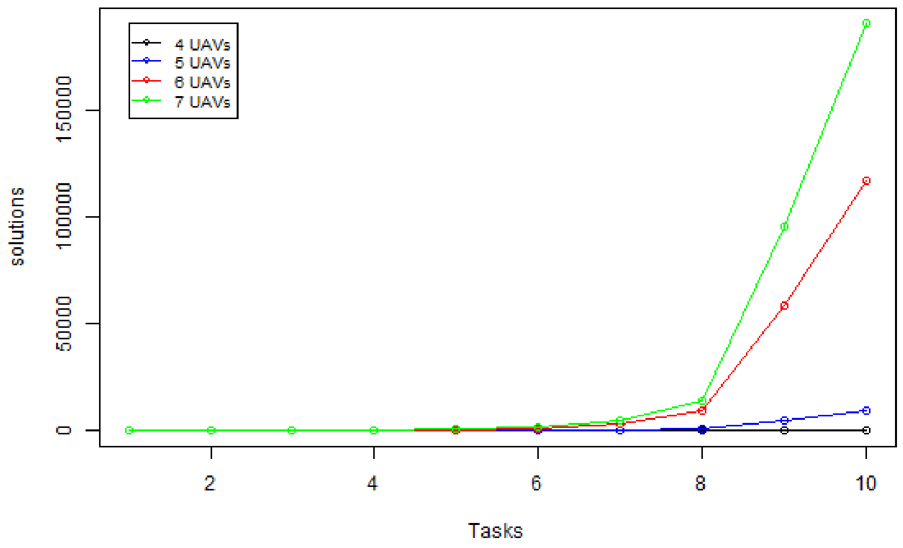


FIGURE 5.1: Number of solutions for missions with the No-Temporal Dependency Schema.

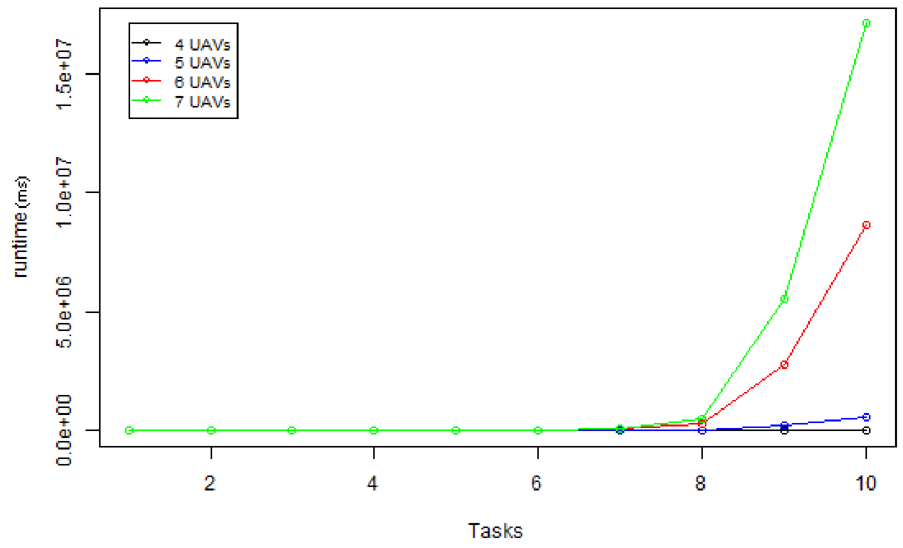


FIGURE 5.2: Runtime for missions with the No-Temporal Dependency Schema

5.1.2 Study with 1-temporal dependency tasks

On the other hand, Figures 5.3 and 5.4 shows what happens when each task collides in time with the previous task (see Figure 4.2b). As it can be seen, the growth is still pretty exponential for both the number of solutions and the runtime, but much smaller than with no dependencies. We also note that for less UAVs to perform the tasks, the exponentiality of the number of solutions disappears. This is because of the high number of constraints (increased with the new temporal constraints) that reduces the space search and, for a high number of tasks, makes the problem highly complex.

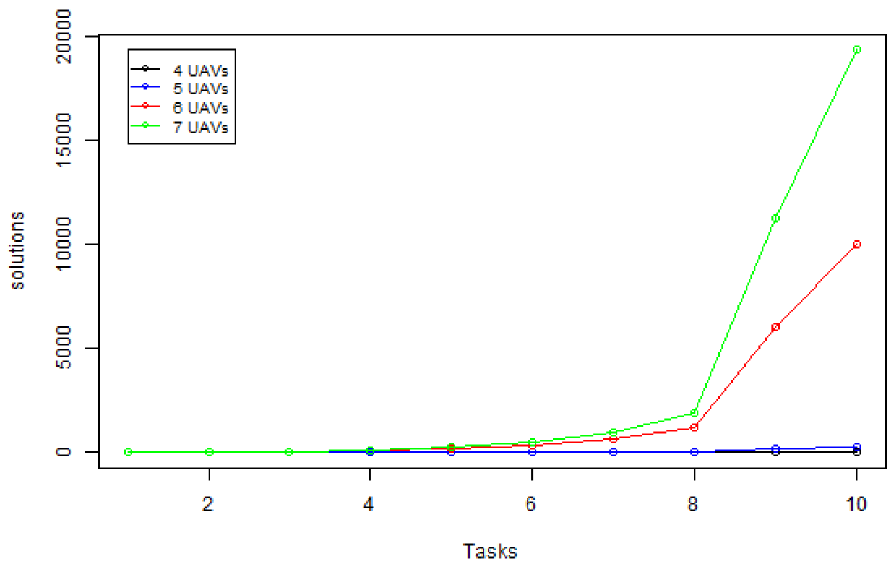


FIGURE 5.3: Number of solutions for mission with the 1-Temporal Dependency Schema.

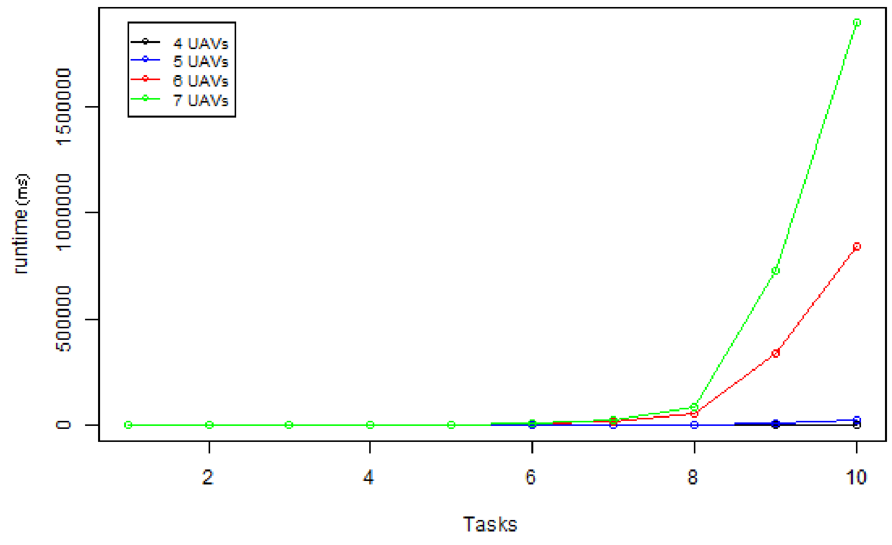


FIGURE 5.4: Runtime for mission with the 1-Temporal Dependency Schema.

5.1.3 Study with 2-temporal dependencies tasks

When each task collides in time with the two previous tasks (see Figure 4.2c), the results in Figures 5.5 and 5.6 show that the growth of the runtime is still exponential, but much smaller than in the two previous cases. On the other hand, the growth of the number of solutions has a more polynomial likely behaviour. We can notice how a great number of constraints affect the scalability of the solutions of the problem.

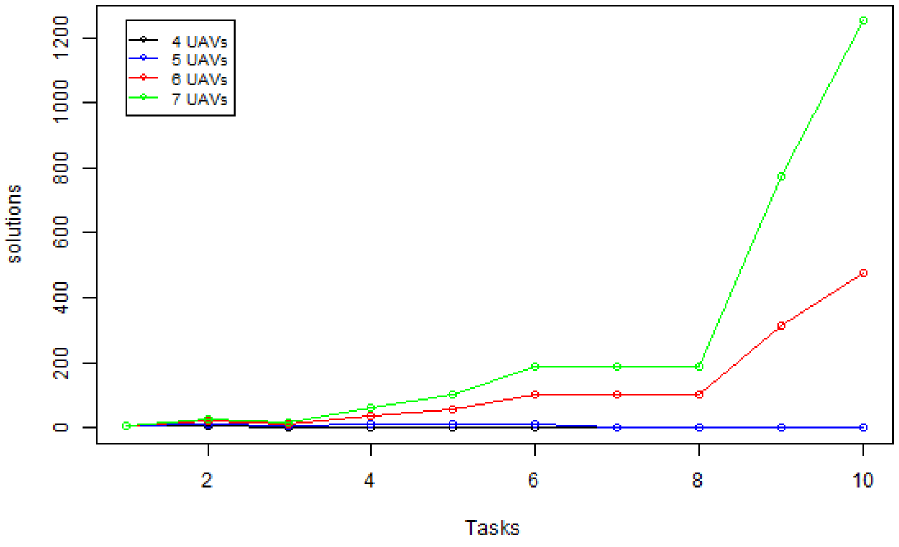


FIGURE 5.5: Number of solutions for missions with the 2-Temporal Dependencies Schema.

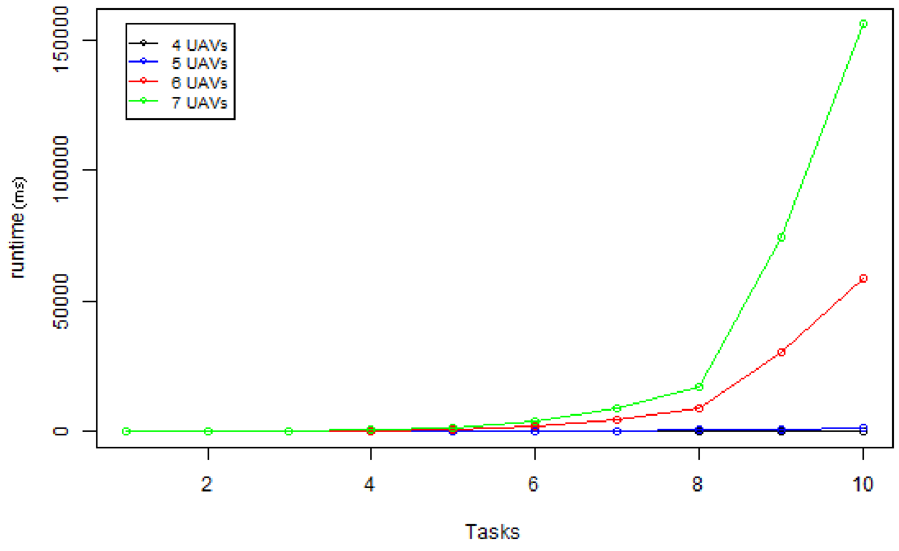


FIGURE 5.6: Runtime for missions with the 2-Temporal Dependencies Schema.

5.1.4 Interdependency comparison

Finally, in Figures 5.7 and 5.8 we can see for a group of 6 UAVs, a comparison of the results obtained according to the number of existing dependencies explained in the three previous experiments. We can see how the temporal constraints highly affect the space of solutions of the problem, but also the runtime necessary to find this new space of solutions.

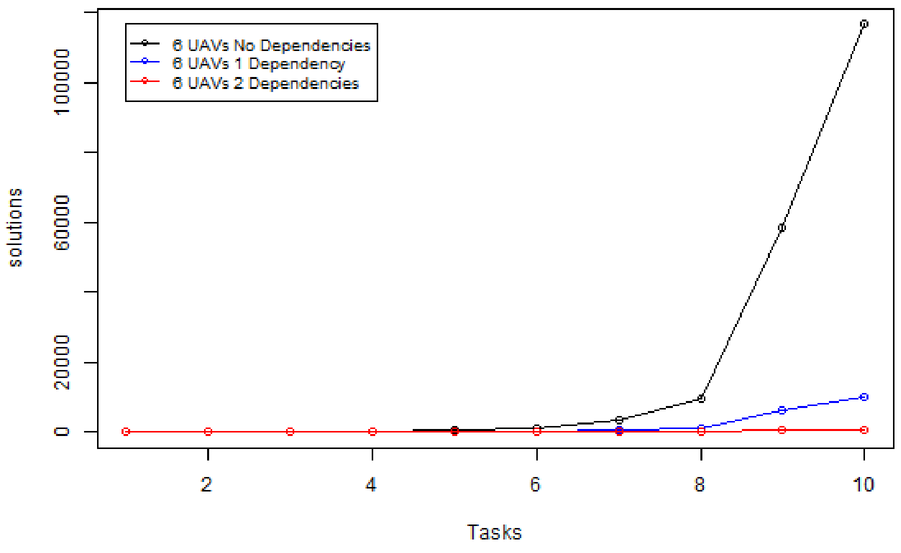


FIGURE 5.7: Number of solutions for missions with the three temporal dependency schemas.

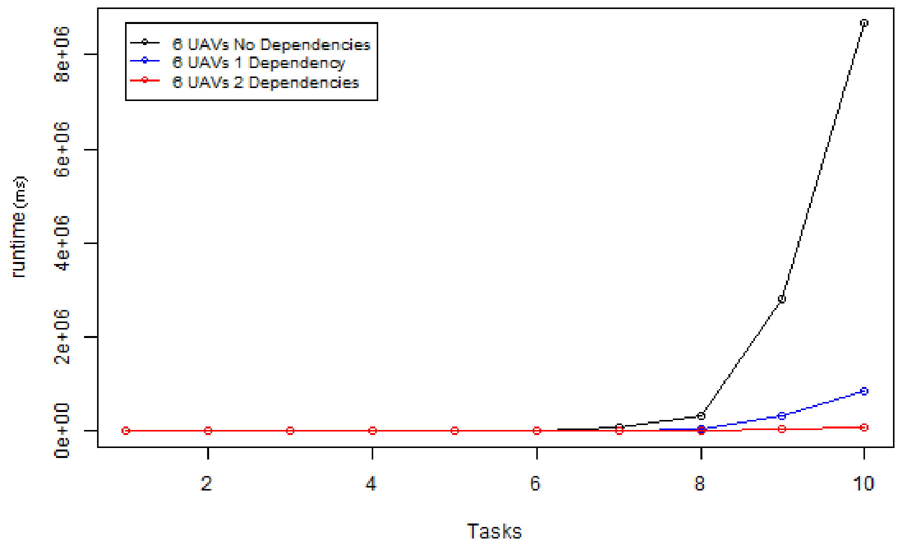


FIGURE 5.8: Runtime for missions with the three temporal dependency schemas.

On the other hand, we have computed the runtimes expended in these missions but with 9 UAVs (see table 5.1), which will be compared to the runtimes expended at finding optimal solutions in the next experiment.

Table 5.1: Runtime for missions with 1 to 10 tasks for a group of 9 UAVs, with the three temporal dependency schemas.

No. of tasks	No Temp. Dep. Schema Runtime	1 Temp. Dep. Schema Runtime	2 Temp. Dep. Schema Runtime
1 task	9.877ms	69.072ms	10.014ms
2 tasks	182.606ms	199.297ms	173.222ms
3 tasks	300.091ms	253.523ms	197.731ms
4 tasks	3.002687s	1.896258s	1.517302s
5 tasks	13.007988s	7.490989s	4.074907s
6 tasks	51.789774s	24.119561s	11.333178s
7 tasks	3m55s	1m10s	23.701752s
8 tasks	18m55s	3m51s	47.584619s
9 tasks	5h0m41s	47m45s	5m10s
10 tasks	22h20m57s	3h15m44s	17m14s

5.1.5 Conclusions

In this experiment, we show that the model is easily computable using a known solver, and the entire space of solutions can be found provided that the mission is resolvable. From the obtained results, we have observed that the runtime necessary to search the entire space of solutions by BT search is exponential as reported in literature. However, as the number of constraints increases (in this case the dependency constraints making tasks collide in time), the runtime decreases highly, but this scalability still resembles exponential. On the other hand, the number of solutions resembles exponential, but as the number of dependency constraints increases, the scalability loses its exponential behaviour and resembles more polynomial. This is due to the power of a dependency temporal constraint, which highly reduces the search space of solutions.

Although the runtime needed for exploring the space of solutions is exponential, we have seen that when there are too many constraints, as the number of tasks increase, there is a point where the resources of the available UAVs needed to supply all the tasks of the mission begin to decrease. In this situation, the number of solutions begins to decrease despite the increase of possible assignments due to a higher number of tasks.

5.2 Experiment 2: Search of optimal solution with Branch & Bound

This second experiment treats with the scenario with a group of 9 UAVs to perform a mission of 10 tasks, and where each task collides in time with its two previous tasks, i.e. the 2-Temporal Dependency Schema.

Gecode provides a B&B search method for optimization problems, but does not automatically compute the POF, so the cost function for the CSOP will be the one explained in section 3.3.2.

This experiment starts comparing the different results obtained optimizing the different objectives individually and then take some of them to optimize altogether. Finally, the runtime results will be compared with the results from the previous BT experiment in order to determine the order of the temporal gain from optimization.

5.2.1 Individual Optimization

B&B returns the best solution found based on the cost function used. So, firstly, an analysis of the optimal solution found considering as cost function each one of the objectives individually is carried out. It can be seen in Table 5.2.

TABLE 5.2: Objective values and runtime spent in the search of the optimal solution using cost functions considering individually each objective.

Cost function	Flight Time	No. of UAVs	Fuel	Runtime
100% Fuel	22h 8min 13s	**4**	**269.561L**	4min 9s
100% No. of UAVs	23h 22min 23s	**4**	282.003L	**8.87s**
100% Flight Time	**18h 0min 8s**	8	284.875L	7min 32s

It can be appreciated when considering cost function 100% Flight Time that, besides the high runtime needed, the optimal solution found has a high number of UAVs and fuel consumption. This could be due to shorter flight times are obtained using UAVs that reach higher speeds but consuming more fuel, i.e. the flight time and the fuel consumption (or the number of UAVs too) have some kind of inverse relation. On the other hand, the number of UAVs and the fuel consumption are highly related.

Respect to the runtime, when considering the number of UAVs the optimization search finishes very soon, because this variable is computed directly from the assignments. The fuel consumption lasts a little more to be computed in each iteration, and the flight time is nearly the last variable being computed.

In the following experiment, we will try to optimize multiple variables at the same time. With this purpose, we will try to find a combination of weights that gets a optimal solution reducing the runtime as much as possible.

5.2.2 Balanced cost function

Attempting to optimize multiple objectives, there has been considered to use balanced cost function. Table 5.3 shows solutions obtained when considering two objectives, while Table 5.4 shows the ones when considering three objectives.

Table 5.3: Objective values and runtime spent in the search of the optimal solution using binary balanced cost functions.

Cost function	Flight Time	No. of UAVs	Fuel	Runtime
50% Fuel + 50% No. of UAVs	22h 8min 13s	**4**	**269.561L**	**54.67s**
50% Fuel + 50% Flight Time	**18h 29min 20s**	7	279.353L	8m11s
50% No. of UAVs + 50% Flight Time	19h 37min 58s	**4**	278.436L	2min 51s

Table 5.4: Objective values and runtime spent in the search of the optimal solution using ternary balanced cost functions.

Cost function	Flight Time	No. of UAVs	Fuel	Runtime
33% Fuel + 33% No. of UAVs + 33% Flight Time	20h 23min 33s	4	269.561L	3m56s

Now it can be seen that combining weighted objectives reduce the runtime spent searching the solution compared to the previous individual optimization experiment in some cases. Specifically, we can see that combining the number of UAVs with the fuel consumption, gets an optimal solution for both variables. On the other hand, considering the flight time involves finding some suboptimal solutions for all the variables. In table 5.4, we can clearly see that the flight time is not optimized while the number of UAVs and the fuel consumption are.

Considering this aspect, we have decided to put the flight time variable aside and only consider the fuel consumption and the number of UAVs. So, in the next section, a simple experiment will be considered in which the fuel consumption and the number of UAVs are considered for a comparative assessment of optimization function weights.

5.2.3 Optimizing the runtime with weighted cost functions

Table 5.5 show the comparative assessment mentioned in the previously. For simplicity of the process, we have considered a weight step of 10% between each instance tested.

TABLE 5.5: Objective values and runtime spent in the search of the optimal solution using cost functions considering fuel and number of UAVs with different percentages.

Cost function	Flight Time	No. of UAVs	Fuel	Runtime
100% Fuel	22h 8min 13s	4	269.561L	4min 9s
90% Fuel + 10% No. of UAVs	22h 8min 13s	4	269.561L	3min 22s
80% Fuel + 20% No. of UAVs	22h 8min 13s	4	269.561L	2min 7s
70% Fuel + 30% No. of UAVs	22h 8min 13s	4	269.561L	1min 39s
60% Fuel + 40% No. of UAVs	22h 8min 13s	4	269.561L	1min 23s
50% Fuel + 50% No. of UAVs	22h 8min 13s	4	269.561L	54.67s
40% Fuel + 60% No. of UAVs	22h 8min 13s	4	269.561L	46.03s
30% Fuel + 70% No. of UAVs	22h 8min 13s	4	269.561L	35.02s
20% Fuel + 80% No. of UAVs	22h 8min 13s	4	269.561L	**33.99s**
10% Fuel + 90% No. of UAVs	22h 8min 13s	4	269.561L	34.13s
100% No. of UAVs	23h 22min 23s	4	282.003L	8.87s

Analysing results shown in Table 5.5, it can be appreciated that only considering the fuel consumption in a low percentage, an optimal solution both for the fuel and number of UAVs minimization is reached. Additionally, it is clearly appreciable that as the weight of the fuel consumption variable decreases, so it does the runtime spent in the search. Nevertheless, it can also be seen that the cost function 20% fuel + 80% No. of UAVs spends less runtime that the cost function 10% fuel + 90% No. of UAVs, breaking this linearity. This could be caused by some “noise” in the execution of the program and the little difference of runtime between these two functions.

For this reason, it can be considered that a cost function of 10% fuel + 90% No. of UAVs is pretty good for searching feasible solutions in low runtime for this kind of problems.

Finally, in the next section, we will compare the runtime obtained with this cost function with the one obtained in the BT experiment. In addition, we will compute the runtimes of this same problem with this cost function but considering the No-Temporal Dependencies Schema and the 1-Temporal Dependency Schema. Then, we will also compare these runtimes with the ones obtained in the BT experiment.

5.2.4 BT vs B&B

In this experiment, we have first calculated the runtime spent in the search of optimal solutions for the mission planning problem composed of 10 tasks and a group of 9 UAVs with the No-Temporal Dependencies Schema and 1-Temporal Dependency Schema (the 2-Temporal Dependencies Schema case was computed in the previous experiment) using B&B with the cost function 10% fuel + 90% No. of UAVs.

Then, the runtime spent in the search of feasible solutions and the runtime spent in the search of the entire space of solutions using BT are compared in Table 5.6.

Table 5.6: Runtime for missions with 10 tasks for a group of 9 UAVs, with the three temporal dependency schemas, using BT and B&B.

Algorithm	No Dependencies Schema Runtime	1-Dependency Schema Runtime	2-Dependencies Schema Runtime
BT	22h20m57s	3h15m44s	17m14s
B&B (10% Fuel + 90% No. of UAVs)	11.33s	26.74s	34.13s

The time difference observed is high, as expected. A surprising fact is that, unlike it happened in BT search, as the number of temporal constraints given by the temporal dependency schemas decrease, the runtime decreases. For instance, with the No-Temporal Dependency Schema, the runtime obtained for the B&B search is 11.33s; while the time obtained in the 2-Temporal Dependencies Schema is 34.13s. On the other hand, the runtime for BT in the No-Temporal Dependency Schema is 22h 20min 57s, being higher than B&B in an order of $8 \cdot 10^3$; while in the 2-Temporal Dependencies Schema the runtime for BT is 17min 14s, higher than B&B in an order of 30 units.

5.2.5 Conclusions

In this second experiment, we have designed an optimization function to minimize four objectives: the fuel consumption, the number of UAVs used in the mission and the total flight time of all the UAVs. From the obtained results, we have observed that the flight time is the most difficult variable to compute, while the number of vehicles is the easiest.

Studying the solutions found by several cost functions with different weights for fuel and number of UAVs, we have observed how the runtime spent in the search decrease as the percentage of fuel decreases. Finally, we have compared the runtime from the

B&B search obtained using the proposed weighted cost function 10% fuel + 90% No. of UAVs with the runtime obtained using BT. As shown in the literature, this second is much higher; concretely we have observed that for this problem with the No-Temporal Dependency Schema it is $3 \cdot 10^3$ times higher. The most interesting fact observed is that the runtime spent in the B&B search decreases as the number of temporal constraints given by the temporal dependency schemas decreases.

It is important to remark that the results obtained are highly dependant on the proposed scenarios and on the topology of the areas the missions are developed in. So further works should consider different scenarios and topologies, so a more general conclusion would be obtained.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

In this work, we try to search feasible solutions for a UAV Mission Planning model based on TCSP. The presented approach defines missions as a set of tasks to be performed by several UAVs with some capabilities. The problem is modelled using: (1) temporal constraints to assure that each UAV only performs one task at a time; (2) logical constraints such as the maximum and minimum altitude reachable or restricted zone permissions, and (3) resource constraints, such as the sensors and equipment needed or the fuel consumption, among others. This simple approach is quite close to real UAV missions, with less conditions treated.

Concretely, we have designed an optimization function to minimize three objectives: the fuel consumption, the number of UAVs used in the mission and the total flight time of all the UAVs.

We have shown that the model is easily computable using a known solver, i.e. Gecode, and both the entire space of solutions (using BT) and the optimal solution (using B&B) can be found provided that the mission is resolvable.

From the obtained results, we have observed that the runtime necessary to search the entire space of solutions using BT search is exponential, as reported in literature, but decreases as the number of constraints increase (because of the decrease of the number of possible solutions).

On a second experiment, we have shown that the WCOP approach is very useful to find a optimal solution, but not for computing the entire POF. We have also observed that is very important to consider bigger weights in variables that are computed faster in order to improve the runtime. An interesting fact observed is that the runtime spent in the B&B search decreases as the number of temporal constraints given by the temporal dependency schemas decreases.

6.2 Future works

As future lines of work, this developed UAV Mission Planning model will be improved in order to consider a model as close as possible to real missions. We will consider the GCS as a new scheduling part of the model, in order to decide the GCS for each UAV. We will also consider refuelling tasks, which will allow the planner to obtain a higher number of solutions in missions with teams of UAVs with low fuel capacity.

It is important to remark that the results obtained are highly dependant on the proposed scenarios and on the topology of the areas the missions are developed in. So further works should consider different scenarios and topologies, so a more general conclusion would be obtained. In this sense, we will try to developed some robust datasets in order to have reliable benchmarks for the comparison of our results.

In addition, we will developed a new approach for solving our problem based on the hybridization of CSP techniques with GAs. This approach will be compared against the B&B approach in order to compare the quality of the solutions and the runtime spent in the search.

Furthermore, we will use a Multiobjective model, i.e. MOEA, such as SPEA2 or NSGA-II algorithms; to find the POF. Using these new algorithms, new heuristics to reduce the complexity of the problem and adapting our current model, we expect to be able to simulate problems near to real scenarios.

Appendix A

CSP solvers comparison

There exists several tools for solving CSPs with good results. Here there are those studied in this work:

- AIspace [46] is a simple but complete Web Java Applet able to solve several CSPs. It is developed for educational purpose, and it only returns one solution to the problems. For this reason, it has been **discarded**.
- Choco 3 [47] is an open source Java library for solving CSPs. It is said to be flexible, efficient and reliable. It is specially developed for research and academic use. Although, documentation is still work in progress.
- Gecode [45] is a C++ open source library for solving CSPs. It is said to be comprehensive, portable, well documented, efficient, allows parallelism and well tested.
- Gurobi [48] is a commercial optimization tool. It is said to be very powerful, well documented, intuitive and with lightweight interfaces. Although, as we did not get an academic license in time, it was **discarded**.
- ILOG CPLEX CP Optimizer [49] is a commercial optimization tool. It is said to be very powerful and well documented, and it has been used in several projects. Although, as we did not get an academic license in time, it was **discarded**.
- JaCoP [50] is an open source Java library for solving CSPs. It is very simple to use and model with it, and well documented.

- Minion [51] is a C++ open source library for solving CSPs. It is fast and scalable, and does not need a program but just a text file with the program. Although, it is poor documented.
- Mistral [52] is a C++ open source library for solving CSPs. It is loosely documented, pretty bugged and need some good C++ knowledge.
- NumberJack [53] is a python open source library for solbing CSPs. It is well documented, fast to develop and easy to use.
- Opturion-CPX [54] is a commercial lazy clause generation CSP solver. It is very recent but it has already get very good results at some CSP contests. Although, as we did not get an academic license in time, it was **discarded**.
- OR-Tools [55] is a C++ open source library for solving CSPs. Developed by Google, it is alive, portable, well documented, pretty efficient and well tested. In spite of its recent appearance, it has obtained very good results at some CSP contests.
- Picat [56] is a B-Prolog open source library for solving CSPs. It works with pattern-machine, imperative, constraints, actors and tabling. It is well documented.

Table A.1 shows the performance of the CSP solvers through a test with a classical problem: the N-Queens problem, which consist of finding an order for the N queens on a NxN board such that no queen menace any other. In this case, we prove with 15-Queens to compare the different solvers. Some CSP solvers with low functionalities were discarded.

Table A.1: Comparison of different CSP solver technologies.

Solver	Language	Avg. Memory (MB)	Avg. CPU (%)	Runtime (s)
Choco 3	Java	46,95	100	142,8
Gecode	C++	6,5	99,8	45,15
JaCoP(*)	Java	*446,57*	*100*	*86,03*
Minion	C++	**0,33**	**99,3**	163,5
Mistral	C++	1,61	100	156,52
NumberJack	Python	24,1	99,6	170,05
OR-Tools	C++	49,18	99,8	161,87
Picat	B-Prolog	172,26	99,8	**37,16**

(*) These results were computed for the 14-Queens problem because the 15-Queens resulted in an out of memory error in Java.

The best performance was made by Gecode, which also has several implementations in different programming languages and is very well documented. Some tools that made not very good results with this problem, such as OR-Tools, are not mean to be bad-efficient with the planning problem, so they must not be entirely discarded for future works.

In conclusion, the best tool for CSP solving in terms of efficiency obtained from this tests is Gecode, which would be the first choice to improve. Other nice tools are OR-Tools (although it has not very good results in the tests done, it has good references and may be better in the planning problem) and Choco3 (which is easier to understand in terms of coding and has pretty good results).

Appendix B

Publications

1. Ramirez-Atencia, Cristian; Bello-Orgaz, Gema; R-Moreno, Maria D; Camacho, David. A simple CSP-based model for Unmanned Air Vehicle Mission Planning. 2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings, pp. 146-153. June 2014.

2. Ramrez-Atencia, Cristian; Bello-Orgaz, Gema; R-Moreno, Maria D.; Camacho, David. Branching to find feasible solutions in unmanned air vehicle mission planning. Proceeding of the 15th International Conference on Intelligent Data Engineering and Automated Learning IDEAL 2014, vol. 8669, pp. 286-294. September 2014.

Bibliography

[1] Corey Schumacher, Phillip Chandler, Meir Pachter, and Lior Pachter. UAV Task Assignment with Timing Constraints via Mixed-Integer Linear Programming. In *AIAA 3rd Unmanned Unlimited Systems Conference*, pages 238–252, 2004.

[2] Wen-Chyuan Chiang and Robert A Russell. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63:3–27, 1996.

[3] S. Leary, M. Deittert, and J. Bookless. Constrained UAV mission planning: A comparison of approaches. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 2002–2009, November 2011.

[4] Christophe Guettier, Bertrand Allo, Vincent Legendre, Jean-Clair Poncet, and Nelly Strady-Lecubin. Constraint model-based planning and scheduling with multiple resources and complex collaboration schema. In *Procedings of the Sixth International Conference on Artificial Intelligence Planning Systems (AIPS)*, pages 284–292, 2002.

[5] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, December 2010. ISBN 978-0-13-207148-2.

[6] Angelo Oddi Riccardo Rasconi Daniel Diaz, Amedeo Cesta and Maria D. R-Moreno. Efficient Energy Management for Autonomous Control in Rover Missions. *IEEE Computational Intelligence Magazine*, 8(4):12–24, 2013. Special Issue on Computational Intelligence for Space Systems and Operations.

[7] David Camacho, Ricardo Aler, Daniel Borrajo, and Jose Manuel Molina. A Multi-Agent architecture for intelligent gathering systems. *AI Communications*, 18(1): 15–32, 2005.

[8] David Camacho, Ricardo Aler, Daniel Borrajo, and Jose Manuel Molina. Multi-agent plan based information gathering. *Applied Intelligence*, 25(1):59–71, 2006.

[9] David Camacho, Cesar Hernandez, and Jose Manuel Molina. Information classification using fuzzy knowledge based agents. In *Proceedings of the IEEE Systems, Man, and Cybernectics Conference (SMC-2001)*, volume 4, pages 2575–2580, USA, 2001. IEEE.

[10] George Vachtsevanos, Liang Tang, Graham Drozeski, and Luis Gutierrez. From mission planning to flight control of unmanned aerial vehicles: Strategies and implementation tools. *Annual Reviews in Control*, 29(1):101 – 115, 2005. ISSN 1367-5788.

[11] Kanna Rajan, Frederic Py, Conor McGann, John Ryan, Tom OReilly, Thom Maughan, and Brent Roman. Onboard adaptive control of AUVs using automated planning and execution. In *International Symposium on Unmanned Untethered Submersible Technology (UUST)*, pages 1–13, 2009.

[12] David Camacho, Daniel Borrajo, Jos M. Molina, and Ricardo Aler. Abstract Planning in Dynamic Environments. In *Proceedings of the IEEE Systems, Man, and Cybernectics Conference (SMC-2001)*, volume 4, pages 2331–2336, Tucson (AZ), USA, 2001. IEEE.

[13] Mubbasir Kapadia, Alejandro Beacco, Francisco Garcia, Vivek Reddy, Nuria Pelechano, and Norman I Badler. Multi-domain real-time planning in dynamic environments. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 115–124. ACM, 2013.

[14] Patrick Doherty, Jonas Kvarnström, and Fredrik Heintz. A temporal logic-based planning and execution monitoring framework for Unmanned Aircraft Systems. *Autonomous Agents and Multi-Agent Systems*, 19(3):332–377, December 2009. ISSN 1387-2532.

[15] Matthew Whalley, M Freed, R Harris, Ma Takahashi, G Schulein, and Jason Howlett. Design, integration, and flight test results for an autonomous surveillance helicopter. In *Proceedings of the AHS International Specialists Meeting on Unmanned Rotorcraft*, 2005.

[16] NASA Dryden. NASA's environmental research aircraft and sensor technology program Apex Project completes critical design review, April 1998.

[17] P. Fabiani, V. Fuertes, A. Piquereau, R. Mampey, and F. Teichteil-Konigsbuch. 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. ISSN 1270-9638.

[18] F. Teichteil-Konigsbuch and P. Fabiani. A multi-thread decisional architecture for real-time planning under uncertainty. In *3rd ICAPS Workshop on Planning and Plan Execution for Real-World Systems, Providence, RI*, 2007.

[19] Florian Adolf and Franz Andert. *Onboard mission management for a VTOL UAV using sequence and supervisory control*, chapter 19, pages 301–316. InTech, October 2010. ISBN 978-953-307-062-9.

[20] Farid Kendoul. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *J. Field Robot.*, 29(2):315–378, March/April 2012. ISSN 1556-4959.

[21] Luis Merino, Fernando Caballero, J Ramiro Martínez-de Dios, Joaquin Ferruz, and Aníbal Ollero. A cooperative perception system for multiple uavs: Application to automatic detection of forest fires. *Journal of Field Robotics*, 23(3-4):165–184, 2006.

[22] Brett Bethke, Mario Valenti, and Jonathan How. Cooperative vision based estimation and tracking using multiple uavs. In *Advances in Cooperative Control and Optimization*, pages 179–189. Springer, 2007.

[23] B. Bethke, M. Valenti, and J. P. How. UAV Task Assignment. *IEEE Robotics and Automation Magazine*, 15(1):39–44, March 2008.

[24] Mehdi Alighanbari. Task Assignment Algorithms for Teams of UAVs in Dynamic Environments. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, June 2004.

[25] David M Ryan, Curt Hjorring, and Fred Glover. Extensions of the petal method for vehicle routeing. *Journal of the Operational Research Society*, pages 289–296, 1993.

[26] Jonas Kvarnström and Patrick Doherty. Automated planning for collaborative UAV systems. In *Control Automation Robotics & Vision*, pages 1078–1085, December 2010.

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

[28] Christian Bessière. Constraint propagation. *Handbook of constraint programming*, pages 29–83, 2006.

[29] Peter Van Beek. Backtracking search algorithms. *Handbook of constraint programming*, pages 85–134, 2006.

[30] Christian Bessière, Pedro Meseguer, EugeneC. Freuder, and Javier Larrosa. On forward checking for non-binary constraint satisfaction. In Joxan Jaffar, editor, *Principles and Practice of Constraint Programming CP99*, volume 1713 of *Lecture Notes in Computer Science*, pages 88–102. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-66626-4.

[31] Eddie Schwalb and Lluís Vila. Temporal constraints: A survey. *Constraints*, 3(2-3): 129–149, 1998. ISSN 1383-7133.

[32] Malek Mouhoub. Solving temporal constraints in real time and in a dynamic environment. Technical Report WS-02-17, American Asociation for Artificial Intelligence (AAAI), 2002.

[33] Malek Mouhoub. Reasoning with numeric and symbolic time information. *Artificial Intelligence Review*, 21(1):25–56, 2004.

[34] Marco Ragni and Stefan Wlfl. Temporalizing cardinal directions: From constraint satisfaction to planning. In *KR'06*, pages 472–480, 2006.

[35] Emma Rollon and Javier Larrosa. Multi-objective Russian doll search. In *Proceedings Of The National Conference On Artificial Intelligence*, volume 22, pages 249–254. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[36] Emma Rollon and Javier Larrosa. Bucket Elimination for Multiobjective optimization problems. *Journal of Heuristics*, 12(4-5):307–328, 2006. ISSN 1381-1231.

[37] C.M. Fonseca and P.J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation. *Systems, Man and Cybernetics, IEEE Transactions on*, 28(1):26–37, Jan 1998.

[38] Antonio Gonzalez-Pardo and David Camacho. A new CSP graph-based representation for ant colony optimization. In *2013 IEEE Conference on Evolutionary Computation (CEC 2013)*, volume 1, pages 689–696, 2013.

[39] Héctor Palacios and Héctor Geffner. Planning as Branch and Bound: A constraint programming implementation. In *Proceedings of Conferencia Lationamericana en Informatica (CLEI)*, volume 2, pages 239–251, 2002.

[40] S.J. Rasmussen and T. Shima. Branch and bound tree search for assigning cooperating uavs to multiple tasks. In *American Control Conference*, pages 6–14, 2006.

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

[42] A.J. Pohl and G.B. Lamont. Multi-objective uav mission planning using evolutionary computation. In *Winter Simulation Conference (WSC 2008)*, pages 1268–1279, December 2008.

[43] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. In *Proceedings of Evolutionary Methords for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100. Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK), 2001.

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

[45] Christian Schulte, Guido Tack, and Mikael Z. Lagerkvist. Modeling and Programming with Gecode, 2010. URL http://www.gecode.org/.

[46] Byron Knoll, Kisyński, Giuseppe Carenini, Cristina Conati, Alan Mackworth, and David Poole. Aispace: Interactive tools for learning artificial intelligence. In *Proceedings of the AAAI 2008 AI Education Workshop*, Chicago, IL, July 2008. URL http://www.aispace.org/.

[47] Narendra Jussien, Guillaume Rochart, Xavier Lorca, et al. Choco: an open source Java constraint programming library. In *CPAIOR'08 Workshop on Open-Source Software for Integer and Contraint Programming (OSSICP'08)*, pages 1–10, 2008. URL http://www.emn.fr/z-info/choco-solver/.

[48] Gurobi. URL http://www.gurobi.com/.

[49] Ilog cplex cp optimizer. URL http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/.

[50] Krzysztof Kuchcinski and Radosaw Szymanek. *JaCoP Library user guide*, 4.0 edition, May 2014. URL http://www.osolpro.com/jacop/index.php.

[51] Ian P. Gent, Chris Jefferson, and Ian Miguel. Minion: A fast, scalable, constraint solver. In *Proceedings of the 2006 Conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29 – September 1, 2006, Riva Del Garda, Italy*, pages 98–102, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press. ISBN 1-58603-642-4. URL http://minion.sourceforge.net/.

[52] Emmanuel Hebrard. Mistral, a constraint satisfaction library. *Proceedings of the Third International CSP Solver Competition*, pages 31–39, 2008. URL http://4c.ucc.ie/~ehebrard/mistral/doxygen/html/main.html.

[53] Emmanuel Hebrard, Eoin Mahony, and Barry Sullivan. Constraint programming and combinatorial optimisation in numberjack. In Andrea Lodi, Michela Milano, and Paolo Toth, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 6140 of *Lecture Notes in Computer Science*, pages 181–185. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13519-4. URL http://numberjack.ucc.ie/.

[54] Peter Stuckey and Mark Wallace. Opturion-CPX, 2011-2014. URL http://www.opturion.com/.

[55] Nikolaj van Omme, Laurent Perron, and Vincent Furnon. *Google or-tools open source library user's manual*. Google, 0.2.11 edition, September 2014. URL https://code.google.com/p/or-tools/.

[56] Neng-Fa Zhou and Jonathan Fruhman. *A Users Guide to Picat*, 0.6 edition, September 2014. URL http://www.picat-lang.org/.