HAL
archives-ouvertes.fr

Constraint programming for air traffic management : a survey

Cyril Allignol, Nicolas Barnier, Pierre Flener, Justin Pearson

▶ To cite this version:

Cyril Allignol, Nicolas Barnier, Pierre Flener, Justin Pearson. Constraint programming for air traffic management : a survey. Knowledge Engineering Review, Cambridge University Press (CUP), 2012, 27 (3), pp 361-392. 10.1017/S0269888912000215 . hal-00934670

HAL Id: hal-00934670
https://hal-enac.archives-ouvertes.fr/hal-00934670

Submitted on 1 Apr 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

The Knowledge Engineering Review, Vol. 00:0, 1–2. © 2012, Cambridge University Press
DOI: 10.1017/S000000000000000 Printed in the United Kingdom

Constraint programming for air traffic management: A survey[1]

In memory of Pascal Brisset

CYRIL ALLIGNOL,[1] NICOLAS BARNIER,[1]
PIERRE FLENER,[2] and JUSTIN PEARSON[2]

[1] *École Nationale de l'Aviation Civile, 7 Avenue Édouard Belin, F – 31055 Toulouse, France*
E-mail: {allignol, barnier}@recherche.enac.fr
[2] *Uppsala University, Department of Information Technology, Box 337, SE – 751 05 Uppsala, Sweden*
E-mail: {Pierre.Flener, Justin.Pearson}@it.uu.se

Abstract

Air traffic management (ATM) under its current paradigm is reaching its structural limits considering the continuously growing demand. The need for a decrease in traffic workload opens numerous problems for optimisation, from capacity balancing to conflict solving, using many different degrees of freedom, such as re-routing, flight-level changes, or ground-holding schemes. These problems are usually of a large dimension (there are 30, 000 daily flights in Europe in the year 2012) and highly combinatorial, hence challenging for current problem solving technologies. We give brief tutorials on ATM and constraint programming (CP), and survey the literature on deploying CP technology for modelling and solving combinatorial problems that occur in an ATM context.

Introduction

For more than forty years, global air traffic has never ceased to increase, except for a few rare and recent events like the September 11 attacks (of 2001) or the 2008 economic crisis. But this growth has not been followed by an equivalent increase in the capacity of air traffic control (ATC), which leads to severe congestion both in Europe and in the USA. Due to the distinct nature of air traffic on these two continents, this saturation especially affects *airports* in the USA, whereas Europe is more concerned with en-route portions of *airspace* where the main routes intersect.

One way of increasing capacity is to utilise present resources more effectively. Toward this, many computational problems arise within the realm of resource allocation. In this paper, we first give a tutorial on air traffic management (see Section 1) and then give another tutorial on constraint programming, which is a technology that is used for solving resource allocation problems (see Section 2). Next, we give an annotated bibliography on the state of the art of deploying constraint programming for air traffic problems (see Section 3). Finally, we assess the obtained results and state directions for future work (see Section 4).

1 Air Traffic Management

Air traffic control (ATC) is a set of ground-based services provided by air traffic controllers to aircraft in order to prevent any collisions by separating aircraft, supply relevant information and advisories for the safe operation of flights, as well as alert and assist search and rescue organisms

[1]The third and fourth authors are financed by the European Organisation for the Safety or Air Navigation (EUROCONTROL) under its innovative research grant scheme (grant 08-121447-C). The content of this paper issue does not necessarily reflect the official position of EUROCONTROL on the matter.

in case of emergency. With the steady increase of air traffic for several decades, air traffic control centres (ATCCs) and airports have come to saturation and their capacity would be exceeded if flights were not regulated in advance. This is one of the tasks of air traffic management (ATM), which, among other long-term airspace considerations, aims at preventing any capacity overflow while optimising the whole system to use the airspace in the most efficient possible way; ATC is therefore one of the tasks of ATM, though the most crucial one.

To expedite and maintain a safe, ordered, and seamless flow of traffic, the European Organisation for the Safety of Air Navigation (EUROCONTROL), which brings together the 38 states of the European Civil Aviation Conference (ECAC), and the Federal Aviation Administration (FAA) of the USA coordinate and plan the various air navigation services over their respective continents. Short of designing new ATC paradigms to overcome the saturation, both organisations attempt to optimise their air traffic management (ATM) practice so as to reduce the impact of the regulation measures (like ground holding and re-routing) required to guarantee the safety of air traffic. On either side of the Atlantic, major research and development programmes that involve the main stakeholders of civil aviation, namely Single European Sky ATM Research (SESAR) (SESAR Consortium, 2007) in Europe and Next Generation Air Transportation System (NextGen) (FAA, 2011) in the USA, have been launched to enhance ATM and meet the future demand and environmental requirements that will strain the system. Unless otherwise mentioned, the rest of this survey focuses on ATM in Europe, where the optimisation of air traffic is more difficult because of the fragmented structure of the airspace and the entanglement of air routes.

Air traffic management in Europe is structured in several layers of filters working at different time horizons. The *strategic* filter is concerned with long-term air traffic management and collects all activities related to the management of *airspace* (see Section 1.1), such as the definition of the airspace volumes, the geometry of air routes, and the climb and descent procedures around airports. The *pre-tactical* filter is activated a few days to a few hours before the scheduled traffic, implementing regulation measures (see Section 1.2) in order to adapt the traffic load to the airspace capacity defined during the strategic phase. Next, the *tactical* filter corresponds to the management of traffic as performed by air traffic controllers (see Section 1.3) and consists of monitoring the traffic, coordinating the traffic between control sectors, and resolving air conflicts. Last, the *emergency* filter (or safety nets) is triggered in case of a failure of all the preceding filters: if a collision is imminent (between two aircraft or between an aircraft and a ground obstacle), ground-based and airborne systems warn controllers or provide manoeuvres to pilots so as to prevent an accident. The following sections detail these various filter levels, eventually explaining the limits of the current ATM system in the context of the foreseen traffic growth.

1.1 Airspace Management

Aircraft that are due to fly within the European airspace have to choose between visual flight rules (VFR) and instrument flight rules (IFR). VFR are based on the see-and-avoid principle and therefore require good visibility conditions; they are more flexible than IFR and mostly suitable for private aircraft. On the contrary, IFR rely on navigation instruments in the cockpit and allow flying in poor weather conditions. It is the flight mode used by commercial aircraft in Europe.

The diverse performance and mission types between VFR and IFR flights have led to a partition of civil airspace into several classes defined by the International Civil Aviation Organisation (ICAO) to reduce the risk of collision. These classes can be roughly distinguished as follows:

- *Controlled* airspace, where separation between aircraft is ensured by an air navigation service provider (ANSP).
- *Uncontrolled* airspace, where only information and alert services are provided.

More specifically, the upper airspace (above flight level[2] 180) and crowded areas around the largest airports are not authorised to VFR flights in order to segregate the two types of traffic. Furthermore, some volumes of airspace are military zones subject to strong usage restrictions or even totally forbidden to general air traffic.

To be authorised to fly with IFR through controlled airspace, a *flight plan* specifying all the characteristics of the flight must be filed with the ANSP, which has to acknowledge it and grant a clearance, possibly with modifications. The flight plan must indicate, among other data, the route to be followed, consisting of a sequence of *beacons* that structure the route network, as explained in Section 1.1.1.

To ensure trajectory monitoring and separation between aircraft, the controlled airspace classes are further divided into *control sectors*, which are 3D volumes whose shape and capacity may vary over time and which are allocated to specific teams of controllers, as described in Section 1.1.2. The design of *air routes*, *control sectors*, and *control centres* is called airspace management (ASM) and is discussed in the two following sections.

1.1.1 Air Routes

An *air route* is a sequence of *legs* (segments) in the horizontal plane, whose extremities, named *waypoints*, are positioned above ground-based radio beacons. As they are expensive to install and cannot be arbitrarily placed (depending on surrounding terrain), it is costly in terms of time and fuel to adhere to the corresponding route network. However, the accuracy of recent airborne radio navigation systems, like the global positioning system[3] (GPS), which are integrated into the flight management system (FMS) responsible for the following of the planned trajectory, has allowed the definition of abstract waypoints that are not associated with any real ground-based devices and has enabled a more efficient usage of airspace. Such a navigation scheme is called *area navigation* (RNAV). Figure 1 presents a small part of the current route network over France and illustrates these two types of waypoints.

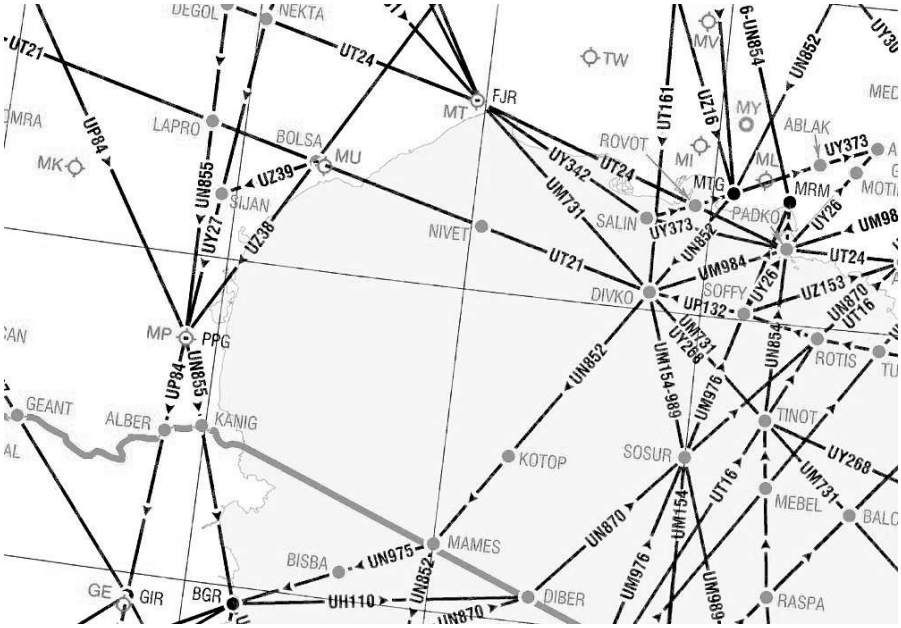
RNAV offers much flexibility to the management of the air route network. Abstract waypoints can be placed so as to shorten significantly the routes, or even design *direct routes*, i.e., shortest routes between two points (e.g., between origin and destination airports, or, more realistically, between entry and exit points of a given controlled airspace volume).

This direct route concept is operationally experimented since March 2011 in the Maastricht control centre (The Netherlands), in the context of the Free Route Airspace Maastricht (FRAM) project (EUROCONTROL, 2011), offering 142 direct routes to cross the concerned airspace. EUROCONTROL claims saving 624,000 NM (nautical miles) over a year, just by authorising these direct routes off-peak (on weekends and from 00:00 to 08:00 on weekdays). From the year 2013, these routes should be permanently opened, which would require the devising of new control procedures, as traffic following direct routes is more entangled than with standard ones.

Whichever navigation mode is selected, the flight plan must state the origin and destination airports, the time of departure and estimated total duration of the flight, the intended route as a sequence of waypoints, the requested flight level, the type of aircraft, and the air speed (plus other secondary data). Flight plans must be filed at least three hours before takeoff, which allows control and regulation organisations to know the intents of each controlled flight sufficiently in advance so as to compute the crossed control sectors. For regular flights operated by airlines, the filing of flight plans can be automated by the use of *repetitive flight plans*.

[2]Vertical distances are expressed in flight levels (FL), one FL corresponding to the nominal altitude of 100 ft = 30.48 m above the international standard pressure datum of 1013.25 hPa (the average sea-level pressure). Authorised cruise flight levels are usually positioned every 1000 ft, i.e. at multiples of 10 (e.g., FL 290, FL 300, FL 310, etc).

[3]Current avionic systems based on a global navigation satellite system (GNSS) like GPS make use of *augmentation* services, like the European geostationary navigation overlay service (EGNOS), which has been operational for aviation since March 2011 (in particular for the precise vertical positioning required by IFR precision approaches before landing), in order to improve signal accuracy and reliability.



Source: EUROCONTROL

Figure 1 Excerpt of a map of the route network in the south of France. Black dots correspond to real waypoints associated with radio navigation beacons, whereas grey dots are abstract ones.

1.1.2 Control Sectors

An airspace with the size and traffic volume of the European one cannot be globally managed by a single team of controllers. The European airspace is therefore divided into functional units, called *elementary sectors*, which allow the distribution of the control workload over several control stations but requires the coordination of the flows between adjacent sectors. An elementary sector is a volume of airspace bounded by a geographical contour in the horizontal plane, a floor flight level and a ceiling flight level.[4] Depending on traffic demand, elementary sectors can be merged or singled out to form a *control sector* monitored by a pair of controllers (see Section 1.3.2).

Control Centres. The management of control sectors is distributed among 75 European *air traffic control centres* (ATCC), which have the daily responsibility of defining open control sectors and allocating them to controllers. Figure 2 shows the shape of the European ATCC at FL 300 (i.e., around the cruise altitude of most commercial jets). Each control sector is associated with an hourly *capacity*, which is a number of entering flights that must never be exceeded and is determined by the relevant ATCC.

The capacity of a control sector is determined empirically, taking into account the volume of the sector; the type of flows – descending or climbing aircraft are more difficult to separate than levelled ones; the number of flow intersections; the presence of restricted (e.g., military) areas; and the experience of past traffic.

Opening Scheme. The shape of control sectors can be dynamically modified during the day to match the traffic demand. During peak hours, sectors can be split into smaller ones to provide more capacity, requiring more controllers. But when the traffic is low, it would be wasteful to keep too many open sectors, so they are grouped together and managed by fewer controllers. Only some predefined groups of elementary sectors (connected and manageable from a controller standpoint) can be used to partition each control centre, and the opened sectors must entirely

[4]Elementary sectors may actually have a more complex shape than simple cylinders and be made up of several vertically stacked adjacent cylinders.

Figure 2 ECAC control sectors at FL 300. Each control centre is depicted with a distinct shade of grey.

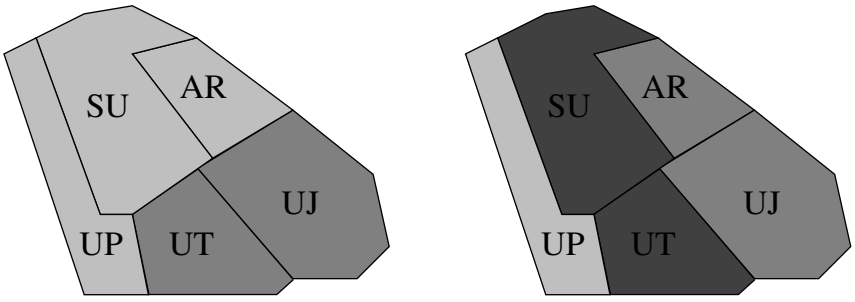


Figure 3 Splitting and merging of control sectors, with each shade of grey corresponding to a distinct control sector. Configuration (a) is adapted to SW-NE flows whereas configuration (b) corresponds to SE-NW ones.

cover the concerned airspace 24 hours a day. Figure 3 illustrates these configuration change operations on a fictitious (small) control centre.

As a control centre can be dynamically reconfigured during the day, the global capacity of the corresponding airspace may change accordingly (even if the capacity of a given sector is constant) to match the traffic demand. To anticipate possible congestion, each control centre issues a forecast of its reconfiguration sequence, called an *opening scheme*, specifying which control sectors will be opened when during the day of operation, taking the filed flight plans and the number of available controllers into account.

Once the opening scheme has been defined with the capacity of each open sector, regulation measures are required to control the traffic while preventing overloads, as explained in the next Section 1.2.

1.2 Regulations

In Europe, the Central Flow Management Unit (CFMU) is in charge of coordinating the various national control organisations of the ECAC area, so as to ensure flight safety while expediting the flow of traffic. In particular, one of its missions is to regulate traffic flows to prevent congestion and to make the most efficient use of available airspace. This section presents the regulation means, called Air Traffic Flow and Capacity Management (ATFCM), operated by the CFMU to carry its missions through, and the costs of these regulations in terms of delays.

1.2.1 Computer-Aided Pre-tactical Regulation

The Enhanced Tactical Flow Management System (ETFMS) operated by the CFMU (CFMU, 2011) allows the display of traffic data as well as various representations of the current situation and of the traffic demand. It also offers an interactive regulation tool called Computer Assisted Slot Allocation (CASA). This software helps the CFMU experts to detect control sector overloads and to propose and assess regulations with two possible degrees of freedom: postponing takeoff and re-routing.

ETFMS is fed with the flight plan demand for the next 48 hours and the current positions of all flights within the European airspace provided by the various national ANSPs. Based on these data, 4D profiles are generated for each flight, which allows the computation of traffic loads for all ECAC control sectors. These traffic load forecasts are then dispatched to the various air traffic control stakeholders through visualisation software. Whenever flights (or flows of flights) are rerouted or delayed to prevent the traffic from exceeding sector capacities, the modifications are automatically reported in the system network and the various views are updated accordingly.

1.2.2 CASA Regulation Algorithm

The CASA software is designed to allocate *takeoff slots* to the flights that have to cross saturated control sectors, which are determined during the prior traffic load computation phase described in the previous section. A takeoff slot is a time that must be respected with a tolerance of -5 to $+10$ minutes.

The CFMU and ANSPs jointly determine the zones (described by their geographical boundaries – most often control sectors – and their activation periods) for which regulations are required. For each zone, the CASA algorithm manages a slot allocation list, empty at initialisation, whose length is proportional[5] to the capacity of the zone.

The CASA algorithm attempts to enforce the first-scheduled first-served principle, which means that flights should enter the regulated zone in the same order as if no regulation had been placed. A flight whose plan has already been filed is attributed a tentative slot from the allocation list according to its estimated entry time into the regulated zone. During this pre-allocation phase, the slots are internal to the system (i.e., not dispatched) and may be updated upon reception of new flight plans or cancellations.

When a new flight plan is filed, the pre-allocation phase occurs as previously described, unless the slot has already been allocated; in this case, the slot is attributed to the earliest scheduled flight, the second one being allocated the next slot. If the latter has already been taken as well, the process is repeated, which can lead to the successive reallocation of many slots. When the slot is fixed, it is sent to the aircraft operator and to air traffic control (ATC) a few minutes (typically 15 to 20 minutes) before takeoff.

Whenever a flight is subject to multiple regulations because it crosses several regulated zones, the greatest delay is imposed: e.g., if a flight crosses two regulated zones and gets a 10 minute delay for the first one and a 15 minute delay for the second one, then its final delay will be 15 minutes, forcing the corresponding slot in the first zone even if it is not available.

[5]For instance, a sector regulated during 2 hours with a capacity of 30 flights per hour would have 60 slots, one every 2 minutes.

This algorithm can therefore violate both the capacity constraint and the first-scheduled first-served rule. Its greedy behaviour cannot either provide any bound on the optimality of the solution. Nevertheless, CASA computes slot allocation in a dynamic operational context with many side constraints, and, generally speaking, the CFMU has had a significant positive impact on ATFCM practice since its putting into service in 1995.

1.2.3 Regulation and Delays

The annual reports published by EUROCONTROL (CODA, 2009) provide a detailed analysis of the amount of delay and the causes of delays in the ECAC area. In these reports, only the flights that have been delayed by more than five minutes are considered.

In 2009, a total of 38% of European flights have been delayed by a 28 minute mean. The main causes of these delays are the airlines (technical problems on aircraft, boarding of passengers or cargo, strike, etc) for 49% of the flights, ATFCM regulations for 25% of the flights, airport congestion for 18% of the flights, and adverse weather conditions for the remaining 8% of the flights. These regulations mainly affect commercial flights between flight levels 330 and 390. Delays due to ATFCM regulations therefore represent a significant part of the overall amount of delay: they affect 7.5% of all flights with a 21 minute mean for each delayed flight. The related cost, discussed in the next section, is mainly borne by the airlines.

1.2.4 Cost of Delays

Many factors can be taken into account to estimate the cost of delays: personnel, fuel, maintenance, passenger management and compensation, etc. Cook et al. (2004) as well as Cook and Tanner (2009) provide a detailed analysis of the different types of delay and their respective costs. The outcome of this study assesses that the cost of ATFCM delay is €0.30 per passenger, per minute of delay and per delayed flight; the average cost for all aircraft at the time of the study is then €72 per minute of delay and per flight. Bontemps and Guittet (2004) questions some of the assumptions of these studies, in particular the loss of market share to the benefit of other transport modes instead of considering that passengers will simply change airline, which globally does not impact the system. The revised cost of the minute of delay then agrees with ITA (2000), i.e., between €35 and €51 per minute.

Whichever methodology is considered, the cost of ATFCM delay is quite high, estimated between €840 million and €1,200 million for 2002 in the ECAC area. Reducing these regulation delays is therefore a critical issue for airlines as well as a challenge for ANSPs and regulation authorities, which have the safety of flights as their primary objective.

1.3 Air Traffic Control

The two main objectives of air control services are the safety and fluidity of air traffic. In order to achieve these objectives, five missions have to be carried out successfully, namely preventing collisions between aircraft; preventing collisions between aircraft and potential obstacles; speeding up and organising air traffic; providing information for safer and more efficient flights; and alerting rescue units when needed.

1.3.1 Different Types of Control

Aircraft speed and manoeuvre capacity are very dependent on the flight stage, thus leading to different methods for controlling the traffic. For example, aircraft speeds are limited to 250 knots near airports, while cruise speeds reach 450 knots for most commercial jets. There are three different types of air traffic control, described hereafter.

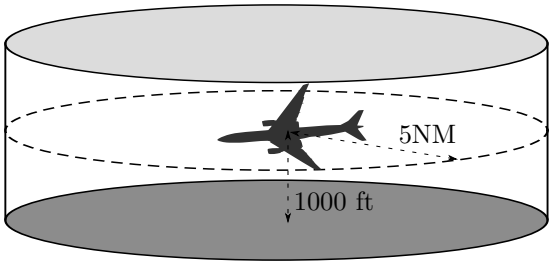


Figure 4 Separation norm for en-route traffic. No other aircraft must enter this volume.

Airport control is in charge of a limited area around an airport. Its main duty is runway management for both arriving and departing traffic. It is also in charge of ground movements[6] between parking areas and the runway. This control task is performed from the control tower.

Approach control manages climbing and descending stages. These stages are particularly complicated as they follow very precise procedures that may vary according to weather conditions (particularly wind force and direction). The related airspace is located around airports (approximately 10 nautical niles around the runway) and is particularly dense due to the convergence of all traffic flows toward a unique point near the runway.

En-route control manages the cruise phase of the flights. It is performed from en-route control centres (currently 75 in the ECAC area).

1.3.2 Methods for Air Traffic Control

Each control sector is handled by a pair of air traffic controllers: a coordinator and a radar controller.

The *coordinator* is in charge of medium-term traffic organisation. The main duty consists in the transfer of aircraft from the sector to the next sector or from the previous sector. The exchanged information is transmitted to the radar controller in the form of a *strip*, which is a small piece of paper that sums up important characteristics of each flight: altitude, speed, route, etc.

The *radar controller* ensures flight safety and a fluid traffic. The main duty is the prevention of collisions by indicating (by radio) manoeuvres to aircraft in order to respect a *separation norm*, which is a minimum safety distance, defined for en-route traffic (ICAO, 1996) as 5 nautical miles horizontally and 1 000 feet vertically (see Figure 4). A radar visualisation provides for each flight its identification number, position, and speed. Different types of manoeuvres might be proposed, such as heading change, speed adjustment, flight level change, or anticipated descent.

1.4 Limits and Perspectives

The current ATM system, as described in the previous sections, is efficient and reactive. However, some areas in the European airspace are regularly saturated, thus showing structural limits of the model. As for air traffic, predictions by EUROCONTROL indicate a continuous growth for the next decades (see Section 1.4.1). This growth is likely to saturate the current system (see Section 1.4.2). We state the impact of trajectory prediction accuracy on the performance of ATM systems (see Section 1.4.3) and present some perspectives for ATM, in terms of both operational concepts and research studies (see Section 1.4.4).

1.4.1 Traffic Evolution in Europe

Traffic forecasts provided by the Statistics and Forecasts (STATFOR) unit of EUROCONTROL (STATFOR, 2010) indicate a growth tendency in spite of the economic crisis taking place these

[6]At the biggest airports, ground movements are managed by an independent unit in order to reduce the workload of controllers.

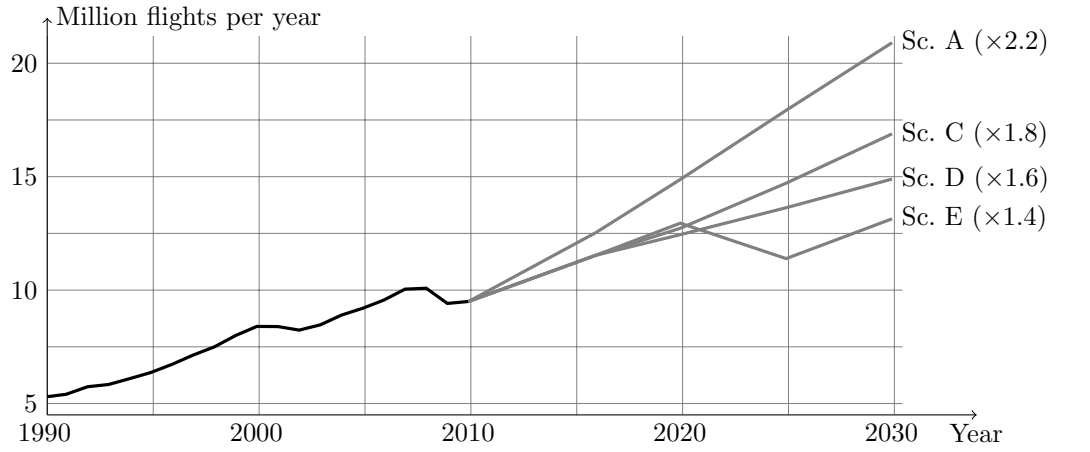


Figure 5 Air traffic predictions in Europe for the year 2030: Annual number of flights (millions) for several growth scenarios.

last years. Several growth scenarios are considered for the year 2030, taking into account economic, technological, and environmental parameters:

- Scenario A – Global Growth: Strong economic growth, increasing globalisation, resource availability enhanced by technology.
- Scenario C – Regulated Growth: Moderate economic growth, regulations to take into account the stakes of sustainable development.
- Scenario D – Fragmenting World: Increasing tensions between regions, leading to reduced trade and weaker economies.
- Scenario E – Resource Limits (Peak Oil): Oil supply deficiency after a production peak in the year 2020, economies unable to react.

In a previous version of the forecast, Scenario B was based on current growth indicators. It was discarded in favour of Scenario E, which was considered to be more relevant.

Regardless of the selected scenario, STATFOR predicts a significant growth of European air traffic, as depicted in Figure 5, namely 40% to 120% more flights in the year 2030 compared to the situation in the year 2009.

1.4.2 Airspace Overload

Based on these traffic growth scenarios, the airspace is likely to be saturated in a few years if managed according to current methods. Two main challenges have to be taken up in order to accommodate the growing demand: augmenting airport capacity and organising flows of traffic in control sectors where the current capacity will be exceeded.

Dividing the airspace into numerous control sectors made it possible to reduce the workload of the controllers by sharing the traffic among them. However, control sectors located within the *core area*[7] are already saturated during long periods each day. Splitting the airspace into smaller sectors would theoretically enable us to increase capacity, as the radar controller's workload would be reduced. Yet the time needed for coordination would increase in a prohibitive manner, thus implying a lower bound on the size for control sectors. Furthermore, too small sectors would not leave enough space and time for controllers to manoeuvre aircraft when resolving conflicts.

However, sector capacities might not be the most suitable measurement for the workload of controllers. Some authors like Degrand and Mercier (2000) advocate the use of the instantaneous

[7]The *core area* is the part of airspace located on the London-Frankfurt-Rome axis, where traffic density is the highest in Europe.

load of a control sector, defined as the number of aircraft simultaneously present in the sector, arguing that this measure better reflects the workload of controllers and is much less tight than the standard capacity. In fact, for a given airspace volume and hourly flow of aircraft into it, the difficulty of the monitoring, separation, and coordination tasks is highly dependant on the topology of the traffic. In particular, Gianazza and Guittet (2006) exhibit a set of metrics – including geometric measures like angles between flows of aircraft or the presence of ascending or descending aircraft – that better match the experienced workload, modelled as the decision process to merge or split sectors.

1.4.3 Flight Management System and Trajectory Prediction

One of the main challenges inherent to most (future) ATM tools is trajectory prediction, as it is the starting point of workload computation and conflict detection. However, many sources of uncertainty might lead to prediction errors, in particular flight management system (FMS) accuracy for the on-board part and the prediction model for the ground-based part. Current FMSs have a fair accuracy in the horizontal plane, but are poor at following a given vertical profile (during climb and descent) or a time constraint (they can honour only one time constraint with a ±30-second accuracy on a 30-minute horizon). Furthermore, takeoff times are highly subject to uncertainty, due to many potential constraints on the airport surface, so that slots allocated by the CFMU must be honoured within −5 to +10 minutes.

Trajectory prediction tools must take these uncertainties into account, as well as a wind estimate, in order to be as close as possible to actual trajectories. As an example, Alliot et al. (2001) prove that if one wants to detect conflicts within a 10-minute time horizon, upon accepting the detection of at most two times the number of actual conflicts, then the accuracy should be at least 1.4 % for horizontal speed and 10 % for vertical speed, which is out of reach for current FMS technology. Trajectory prediction could also benefit, at least for tactical tools, from information that is transmitted to the ground and regularly updated by aircraft systems, such as wind speed and direction (which are accurately known by aircraft), mass of the aircraft, or pilot intentions. However, airlines consider such information as strategic and are thus unwilling to share them. Many models have been proposed for trajectory prediction in order to overcome these difficulties. Musialek et al. (2010) provides a good survey on these studies.

Nevertheless, the accuracy of FMSs is constantly increasing, therefore enabling better trajectory predictions. Ambitious objectives, in the context of the SESAR programme (SESAR Consortium, 2007), are accuracy up to ±0.3 nautical miles laterally, ±150 feet vertically, and ±10 seconds on several temporal constraints by the year 2020.

1.4.4 Perspectives

Air traffic management, under its current paradigm, is reaching its structural limits, considering the continuously growing demand, as stated in Section 1.4.2. In this context, EUROCONTROL came up with new concepts for ATC, such as *sector-less* (Duong et al., 2001), where controllers are in charge of a flow of aircraft rather than a control sector, or *free flight* (Duong et al., 1997), where aircraft have more freedom to choose their trajectories and where conflict detection and resolution can be partly delegated to aircraft. However, the necessary changes for implementing these concepts are too drastic, so that it is safer and cheaper to optimise inside the current system.

As advocated by Leal de Matos and Ormerod (2000) as well as Barnhart et al. (2003) to promote the usage of techniques from operational research in ATM, the need for a decrease in traffic workload opens numerous problems for optimisation, from capacity balancing to conflict solving, using many different degrees of freedom, such as re-routing, flight level changes, or ground-holding schemes. These problems are usually of a large dimension (there are 30,000 daily flights in Europe in the year 2012) and highly combinatorial, hence challenging for current problem

solving technologies. They also induce a lot of constraints that have to be considered in order to fit real-world operations.

Among the various combinatorial problem solving technologies, constraint programming, introduced in the next section, is a good candidate to solve such large-scale problems, as it provides a quick and flexible modelling tool – which is particularly interesting for refining incrementally a problem – and allows experiments with various search strategies without changing the rest of the model.

2 Constraint Programming

First, we explain the notion of combinatorial problem (Section 2.1) and show that they appear in many real-world situations. Then, we show how to model combinatorial problems at a very high level of abstraction with the help of the declarative notion of *constraint* (Section 2.2). Next, we describe two methods for solving combinatorial problems by a mixture of inference and search, upon reusing algorithms that implement these constraints (Section 2.3). After pointing to success stories and major solvers exploiting this constraint programming (CP) paradigm (Section 2.4), we discuss its hybridisation perspectives with other combinatorial problem solving technologies (Section 2.5).

2.1 Combinatorial Problems

Solving a combinatorial problem involves finding a combination of discrete finite objects that satisfies some given constraints. Candidate solutions are combinations that may be encountered during a solution attempt, but they need not satisfy all given constraints. Solutions are candidate solutions that satisfy *all* the given constraints. Many combinatorial problems are hard to solve because there are very many candidate solutions and very few solutions: this is because a candidate solution to an individual constraint need not be a candidate solution to another individual constraint, but a solution has to be found that satisfies all given constraints. Often the only known way to solve a combinatorial problem is by intelligently searching through all the possible candidate solutions.

These problems occur in many real-world situations. Examples are scheduling, timetabling, planning, logistics, transportation, configuration, personnel allocation (rostering), sequencing, flow management, bin packing, and floor design. For solving these problems, one must find values for some given *decision variables* (which are the unknowns) within their given *domains* (which are sets of possible values), so that some given *constraints* (which are relations) on these variables are satisfied and possibly that some cost (or benefit) expression, called the *objective function*, on these variables is minimised (or maximised). One distinguishes between *satisfaction problem*s (where there is no objective function) and *optimisation problems* (where there is an objective function).

For example, consider the following scheduling problem. We have a set of flights, each of which will pass through a sequence of control sectors that is assumed to be known in advance. We assume that the airspeed is predetermined, so the flight time of each flight in each of its control sectors is known. Each control sector has a load limit that specifies the maximum number of flights that can be present in it at any time. For each flight, we use a decision variable that denotes its takeoff time. For each flight, there are constraints on when it can take off. For each control sector, there is a constraint that enforces its load limit. A satisfaction problem is to find takeoff times that respect all constraints. An objective function could include a measure of load balancing to ensure that at all times each control sector has a similar number of flights present.

In real life, combinatorial problems are often crucial, as the world is becoming increasingly complex and competitive, and as we need a more sustainable industry and society. Prime objectives are the efficient handling of scarce or expensive resources and the minimisation of environmental impact. Modern combinatorial problem solver software usually outperforms the

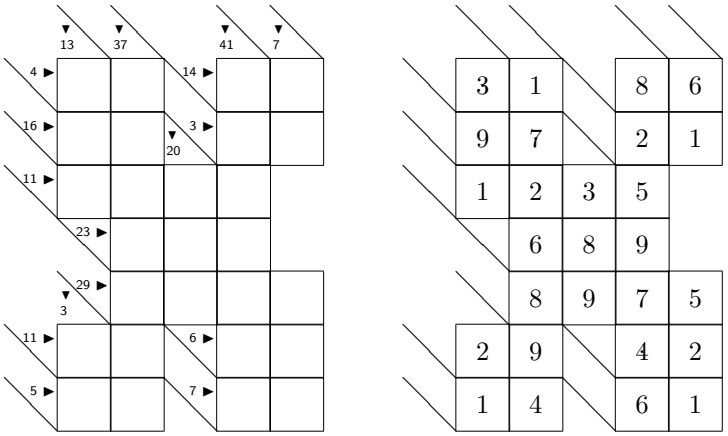


Figure 6 A Kakuro puzzle (on the left) and its solution (on the right), both taken from the crossword package for LaTeX by Gerd Neugebauer at `CTAN.org`.

best human experts by a wide margin in solving time and solution quality, hence research is very active in this area.

2.2 Problem Modelling via Constraints

Constraint programming (CP) is a successful approach to the modelling and solving of combinatorial problems. Its core idea is to capture islands of structure, also called combinatorial substructures, that are commonly occurring within such problems and to encapsulate declaratively their procedural inference algorithms by designing specialised reusable software components, called *constraints*.

As a running example, let us consider the Kakuro puzzle, with the caveat that this is *not* meant to imply that the discussed techniques are limited to (small) puzzles. See the left side of Figure 6: Each word of a Kakuro puzzle is made of non-zero digits, under the two constraints that the letters of each word are pairwise distinct and add up to the number to the left (for horizontal words) or on top (for vertical words) of the word. Hence there is one decision variable for each letter of the puzzle, with the integer set $\{1, 2, \ldots, 9\}$ as domain. This is a satisfaction problem, as there is no objective function. The solution to this puzzle is given on the right side of Figure 6; note that a Kakuro puzzle is always designed so as to have a unique solution.

Beside the usual binary comparison constraints ($<$, $\leq$, $=$, $\neq$, $\geq$, $>$), with arguments involving the usual binary arithmetic operators ($+$, $-$, $\cdot$, etc), a large number of *combinatorial constraints* (usually called *global constraints* in the literature), involving more complex structures and a non-fixed number of arguments, have been identified.

For example, the AllDifferent($\{x_1, \ldots, x_n\}$) constraint requires its n decision variables x_i to take pairwise different values. This constraint is useful in many challenging real-life problems, as well as for modelling the first constraint of the Kakuro puzzle, as there is an AllDifferent constraint on each word. Each such constraint wraps a conjunction of $\frac{n \cdot (n-1)}{2}$ binary $x_i \neq x_j$ constraints into a single n-ary constraint; this enables a more global view of the structure of the problem, which is a prerequisite for solving it efficiently (as we shall see in Section 2.3).

As another example, the Sum($\{x_1, \ldots, x_n\}, Rel, c$) constraint requires the sum $x_1 + \cdots + x_n$ of the n decision variables x_i to be in relation Rel with the constant c. This constraint can be used for modelling the second constraint of the Kakuro puzzle, as there is a Sum($\{x_1, \ldots, x_n\}, =, s$) constraint for each word $[x_1, \ldots, x_n]$ with required sum s.

Hundreds of useful constraints have been identified and are described in the *Global Constraint Catalogue* (Beldiceanu et al., 2007), the most famous ones being ELEMENT (Van Hentenryck and Carillon, 1988), CUMULATIVE (Aggoun and Beldiceanu, 1993), ALLDIFFERENT (Régin, 1994), and CARDINALITY (Régin, 1996).

Constraints can be combined to *model* a combinatorial problem in a declarative and very high-level fashion. For example, the Kakuro puzzle can essentially be modelled as follows, assuming that a hint for word w with required sum s is encoded as the pair $\langle w, s \rangle$:

$$\begin{array}{c} \textbf{for each } \textit{hint } \langle [x_1, \ldots, x_n] , s \rangle : \\ \textsc{AllDifferent}(\{x_1, \ldots, x_n\}) \wedge \textsc{Sum}(\{x_1, \ldots, x_n\} , =, s) \end{array} \quad (1)$$

For combinatorial problems, CP is *not* limited to decision variables with finite *integer* domains: arbitrary enumerations can be used as domains, as well as domains over higher-order data such as sets (of integers) and graphs.

In the following sub-section, we show that constraints are not just a convenience for high-level modelling, but can also be exploited during the solution process.

2.3 Problem Solving by Inference and Search

By *intelligent search*, we mean search where at least some form of inference takes place at every step of the search in order to reduce the cost of brute-force search (Hooker, 2007):

$$\text{combinatorial problem solving} = \text{search} + \text{inference} + \cdots$$

In the following, we discuss two CP ways of solving problems that have been modelled using constraints. The classical approach is to perform systematic search (Section 2.3.1), and a more recent approach is to trade for time the guarantees of systematic search by performing stochastic local search (Section 2.3.2): the two approaches use completely different forms of inference, which is encapsulated in reusable fashion within the constraints (Section 2.3.3).

2.3.1 Systematic Search

Classically, CP solves combinatorial problems by systematic tree search, together with backtracking, and performs at every node of the search tree a particular kind of inference called *propagation*. After explaining propagation, we will explain in more detail how such systematic search is conducted.

Propagation of One Constraint. For each individual constraint, a *propagation algorithm* (or *propagator*) prunes the domains of its decision variables by eliminating impossible values according to some desired level of *consistency*. For example, under *domain consistency* (DC) every domain *value* of every decision variable participates in some solution to the constraint that involves domain values of the other decision variables. Also, under *bound consistency* (BC) every domain *bound* of every decision variable participates in some solution to the constraint that involves domain values of the other decision variables.

For example, consider the constraint $\textsc{AllDifferent}(\{x, y, z\})$. Let x, y range over the domain $\{1, 3\}$ and z over $\{1, 2, 3, 4\}$: we denote this state by $\{x, y \mapsto \{1, 3\} , z \mapsto \{1, 2, 3, 4\}\}$. From this state, propagation to DC leads to the state $\{x, y \mapsto \{1, 3\} , z \mapsto \{2, 4\}\}$ since x and y must split the values 1 and 3 between themselves so that z cannot take any of these two values. From the same start state $\{x, y \mapsto \{1, 3\} , z \mapsto \{1, 2, 3, 4\}\}$, propagation to BC leads to the state $\{x, y \mapsto \{1, 3\} , z \mapsto \{2, 3, 4\}\}$ since there do exist solutions to the constraint where z takes its new lower bound value 2 or its old (and new) upper bound value 4, so that the unfeasibility of the intermediate value 3 is not even checked. Note that, in this case, the resulting DC state is strictly stronger than the resulting BC state: while the initial state encodes a set of $2 \cdot 2 \cdot 4 = 16$ candidate solutions, the BC state encodes a subset thereof with $2 \cdot 2 \cdot 3 = 12$ candidate solutions,

and the DC state encodes a subset of both with only $2 \cdot 2 \cdot 2 = 8$ candidate solutions, including the 4 solutions. From a second start state $\{x, y, z \mapsto \{1, 2\}\}$, propagation to DC or BC leads to the propagator signalling *failure*, because it is impossible to assign two values to three variables so that the latter are pairwise distinct. From a third start state $\{x \mapsto \{1\}, y \mapsto \{3\}, z \mapsto \{2, 4\}\}$, propagation to DC or BC leads to no propagation, but the propagator can simultaneously detect that all $1 \cdot 1 \cdot 2 = 2$ candidate solutions actually are solutions, so that the propagator can signal *subsumption* (or *entailment*).

The propagation of a constraint amounts to *reasoning with possible domain values*, but there is *no* obligation to prune *all* the impossible domain values, as just witnessed when comparing DC and BC. If a constraint has multiple propagators achieving different strengths of consistency (under different time complexities), then there is a default propagator but the modeller may also choose one of them, possibly via experiments to find out which one leads to the best trade-off in search effort; this is the first non-declarative annotation that may be added to an otherwise declarative constraint model (and we will encounter a second one below). Also, the propagator of a constraint only reasons *locally*, namely about the decision variables of that constraint, rather than globally, about all the decision variables of the entire problem.

Propagation of Multiple Constraints. Let us now consider the propagation of an entire conjunction of constraints, such as a constraint model like (1). Each constraint is propagated to its desired strength of consistency: this may take multiple invocations of some propagators and is choreographed by a fixpoint algorithm, which computes the weakest common fixpoint of all the propagators.

For example, consider the conjunction $x \neq y \wedge y \neq z \wedge z \neq x$ of binary disequality constraints, which is equivalent to the constraint AllDifferent$(\{x, y, z\})$ of the previous example. From the first start state $\{x, y \mapsto \{1, 3\}, z \mapsto \{1, 2, 3, 4\}\}$, propagation to the fixpoint of the three propagators, each propagated to DC or BC, achieves no pruning at all (unlike in the previous example), because each individual $\alpha \neq \beta$ constraint is unsatisfiable only when its two variables α and β range over the same singleton domain. From the second start state $\{x, y, z \mapsto \{1, 2\}\}$, propagation under DC and BC to the fixpoint also achieves no pruning at all (unlike in the previous example), so that search (see below) is necessary to reveal the inconsistency of the conjunction. This example actually mostly illustrates the importance of combinatorial constraints compared to their decompositions into conjunctions of more basic constraints, but a more interesting fixpoint computation will be given in the next example.

Search. We are now in position to explain how systematic tree search is conducted in order to solve a combinatorial problem modelled as a conjunction of constraints. It is an interleaving of propagation and search. Each node consists of the currently non-subsumed constraints and the current fixpoint state of all the propagators. The root of the tree has the fixpoint of the propagators for all the constraints of the model. As long as there exist non-subsumed constraints, the tree is grown at some leaf by branching into at least two sub-problems, obtained by mutually exclusive *decisions* on some decision variables. Decisions must be constraints themselves, hence they have propagators: the fixpoint algorithm is launched for computing each child of the considered leaf. Backtracking is performed when the fixpoint algorithm fails because the propagator of some constraint signals failure.

For example, at the fixpoint $\{x, y \mapsto \{1, 3\}, z \mapsto \{1, 2, 3, 4\}\}$ of the previous example for $x \neq y \wedge y \neq z \wedge z \neq x$, no propagator is subsumed. Upon exploring the sub-problems obtained for the decisions $x = 1$ and $x \neq 1$, the search tree of Figure 7 is obtained. Note that on each leaf all three propagators are subsumed. The same branching would have been obtained from the node $\langle \{\text{AllDifferent}(\{x, y, z\})\}, \{x, y \mapsto \{1, 3\}, z \mapsto \{2, 4\}\}\rangle$ that was obtained in a previous example.

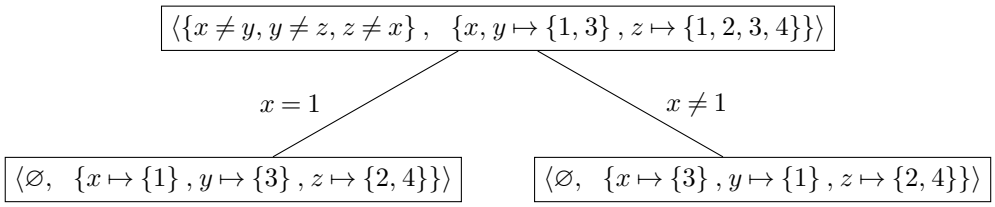


Figure 7 Search tree, with propagation at every node

Note that all communication between two constraints happens via the state of their shared decision variables: if the propagator of one constraint makes a suitable change to the state of a shared variable, then the propagator of the other constraint may be invoked.

Branching is subject to heuristics for the choice of the leaf as well as the choice of the decision variables, domain values, and predicates used in the decisions made. The leaf is usually chosen under depth-first search. Pre-defined branching heuristics can be combined for designing the decisions, or entirely new branching heuristics can be programmed. Many branching heuristics can be written in problem-independent fashion. Intuitively, such generic heuristics tackle the hardest parts of a problem first, such as the critical resources in scheduling problems.

For example, a commonly used variable selection heuristic is to follow the *fail-first principle* and choose one of the decision variables that currently has the smallest domain, as one can thus provoke failure (if any) earlier, since in the presence of propagation not all decision variables need to take a value before detecting failure. Thus, exponentially larger parts of the search tree are pruned upon earlier failure. Conversely, a commonly used value selection heuristic is to follow the *best-first principle* and choose a value that is most likely to lead to success (if possible), since one need not consider all values to find a first solution. Recently, *adaptive heuristics* have been designed, such as *weighted degree* (Boussemart et al., 2004): they make decisions based on information collected during the search.

Other than the already mentioned indication of the desired strength of consistency for the constraints of the model, the provision of these branching heuristics constitutes the second non-declarative annotation that may be added to an otherwise declarative constraint model. The resulting *search strategy* is often the part of a constraint model that requires the most expertise. Novice constraint programmers can rely on default strategies.

It is important to stress that propagation at every node of the search tree, instead of just at its leaves, usually reduces the size of the search tree in a very drastic way, so that the overhead at every node of the search tree is well amortised by overall time gains.

Return to Modelling. If experiments reveal that the current combination of model, consistency strengths, and branching heuristics is not powerful enough to solve problem instances with suitable effort, then one may have to reconsider the declarative model if no alternatives to the procedural choices turn out good enough.

For example, the Kakuro model (1) is very naïve, as there is poor communication via propagation between the AllDifferent and Sum constraints. Consider in Figure 6 the top-left horizontal word $[x, y]$: it is of length 2 and has the desired sum 4, hence the constraints AllDifferent$(\{x, y\})$ and Sum$(\{x, y\}, =, 4)$ occur in the model, with the initial state $\{x, y \mapsto \{1, 2, \ldots, 9\}\}$. However, the fixpoint even under domain consistency (DC) of the propagators of these two constraints is $\{x, y \mapsto \{1, 2, 3\}\}$, hence the unfeasibility of value 2 is *not* inferred! Fortunately, it is easy to design, with the help of the Regular meta-constraint (Beldiceanu et al., 2004; Pesant, 2004), a DC propagator for a custom constraint AllDiffSum so that AllDiffSum$(\{x, y\}, =, 4)$ infers $\{x, y \mapsto \{1, 3\}\}$. The resulting model

$$\textbf{for each } \textit{hint } \langle [x_1, \ldots, x_n], s\rangle : \text{AllDiffSum}(\{x_1, \ldots, x_n\}, =, s)$$

is orders of magnitude faster than the original one, especially upon hint-specific refinements such as rather using AllDifferent for words of length 9, because their sum can only be 45.

2.3.2 Stochastic Local Search

Systematic search (as just described) explores the *whole* search space, though *not* by explicitly trying all possible combinations of domain values for the decision variables, but *implicitly* thanks to the interleaving of search with inference. Suitable values are found *one-by-one* for the decision variables. Systematic search offers the guarantee of eventually finding a solution (or finding and proving an optimal solution, in the case of an optimisation problem), if one exists, and proving unsatisfiability otherwise. However, this may take too long and it may be more interesting in some situations to find quickly solutions that may violate some constraints (or may be sub-optimal). The idea of *stochastic local search* (SLS; see Hoos and Stützle (2004), for example) is to trade this guarantee for speed by not exploring the whole search space. Unsatisfiability of the constraints is *a priori* not detectable by SLS, and optimality of solutions is a priori not provable by SLS.

SLS starts from a possibly random assignment of domain values to *all* the decision variables, without concern for whether some constraints are violated. It then tries to find a better assignment (in the sense of violating fewer constraints, or violating some constraints less, or yielding a better value of the objective function) by changing the values of a few decision variables, upon probing the impacts of many such small changes, which are called *moves*, and then actually selecting and making one of these moves. The set of candidate moves is called the *neighbourhood*. This iterates, under suitable heuristics and meta-heuristics, until a sufficiently good assignment has been found, or until some allocated resource (such as running time or a number of iterations) has been exhausted.

SLS is an area of intensive research on its own, but the CP concept of constraint can be usefully imported into SLS, giving rise to what is known as *constraint-based local search* (CBLS; see Van Hentenryck and Michel (2005) for example). In principle, the declarative part of a constraint model is thus the *same* as when solving the problem by classical CP (by systematic tree search interleaved with propagation). The inference counterpart of the propagator of a constraint are its violation functions and its differentiation functions, discussed next.

Violation and Differentiation of One Constraint. For each individual constraint, the following functions are required in a CBLS system:

- The *constraint violation function* gives a measure of how much the constraint is violated under the current assignment. It must be zero if and only if the constraint is satisfied, and positive otherwise.
- The *variable violation function* gives a measure of how much a suitable change of a given decision variable may decrease the constraint violation.
- The *assignment delta function* gives the increase in constraint violation upon a probed $x := d$ assignment move for decision variable x and domain value d.
- The *swap delta function* gives the increase in constraint violation upon a probed $x :=: y$ swap move between two decision variables x and y.

A negative delta reflects a decrease in constraint violation, hence smaller deltas identify better moves. Differentiation functions for other kinds of moves, such as multiple assignments, can be added. Ideally, violations are updated incrementally in constant time upon the actual making of a move, but this is not always possible. Similarly, deltas are ideally computed differentially in constant time rather than by subtracting the constraint violations after and before the probed move.

For example, consider the constraint AllDifferent$(\{v, w, x, y, z\})$. Under the assignment $\{v \mapsto 4, w \mapsto 4, x \mapsto 5, y \mapsto 5, z \mapsto 5\}$, the constraint violation could be 3, because three variables need to take a suitable new value in order to satisfy the constraint, and the variable violation of y

could be 1, because the constraint violation would decrease by one if y were assigned a suitable new value, such as 6. Upon the assignment moves $y := 4$ and $y := 6$, the constraint violation increases by 0 and -1, respectively, so the latter probed move is better. Upon *any* swap move, the constraint violation increases by 0. When maintaining for every domain value the number of variables currently taking it, the violations can be updated in constant time upon an actual move, and the deltas can be computed in constant time for a probed move.

Violation and Differentiation of Multiple Constraints. The constraint violation of an entire conjunction of constraints, such as a constraint model, is obtained by summing the violations of its constraints. The same holds for the variable violations, assignment deltas, and swap deltas of a conjunction. All these sums — called *system constraint violation*, *system variable violation*, *system assignment delta*, and *system swap delta* — can be maintained in constant time upon an actual or probed move.

A neighbourhood can often be designed so that some constraints of the model remain satisfied if they are satisfied under the starting assignment. Such constraints are called *implicit* constraints, since they need not be given in the constraint model, whereas the constraints to be satisfied through search are called *explicit* constraints and must be given in the constraint model. Since the explicit constraints can be violated under the current assignment, they are often called *soft* constraints. Conversely, since the implicit constraints can never be violated, they are often called *hard* constraints.

For example, in a Sudoku puzzle, there is an AllDifferent constraint on each of the nine rows, columns, and 3×3 blocks: the row AllDifferent constraints can be made implicit upon using a neighbourhood with swap moves inside rows, since these constraints can be satisfied under the starting assignment (obtained by generating nine random permutations of the sequence $[1, 2, \ldots, 9]$) and remain satisfied upon swap moves.

Search. CBLS by itself makes *no* contributions to the design on SLS heuristics and meta-heuristics, but it facilitates their formulation. The constraint violation function helps to select a promising constraint for selecting decision variable(s) to change in a move. The variable violation function helps to select promising decision variable(s) to change in a move. The delta functions help to make a move in the good direction for a constraint or decision variable.

For example, the *min-conflict heuristic* amounts to making a best move among those that modify the value of a so-called conflicting variable: this amounts to picking a random move among those where the system assignment delta is smallest, considering only the assignment moves that alter the value of a variable whose system variable violation is positive.

Many heuristics and meta-heuristics, such as tabu search (Glover and Laguna, 1993) and simulated annealing (Kirkpatrick et al., 1983), can now be written in problem-independent fashion.

The violation functions are the counterpart of the subsumption checking of systematic search. The delta functions are the counterpart of the propagation of systematic search.

2.3.3 Conclusion about Search and the Role of Constraints in Search

Both in constraint-based systematic search and in constraint-based local search, a problem solver software (or simply: *solver*) need only provide the master search algorithm, as well as implementations of the built-in (meta-)heuristics and constraints that are used in the problem model. The modeller is free to design custom (meta-)heuristics and constraints. A constraint fully declaratively encapsulates inference algorithms (propagators or violation functions and delta functions), which have been written once and for all and are invoked by the master search algorithm and the (meta-)heuristics in order to conduct the search for solutions.

The usage of constraints achieves code *reusability*. It also entails a clean separation between the declarative and non-declarative parts of the problem model (which together form the input

to the solver), as well as a clean separation between search and inference within the solver itself. The slogan of constraint programming is:

$$\text{constraint program} = \text{model} + \text{search}$$

because we also have more code *modularity*.

2.4 Success Stories and Solvers

CP has been an area of research since the late 1980s and is now a mature technology that has been successfully used for tackling a wide range of real-world complex problems, especially for scheduling (where the foundations were laid in (Baptiste et al., 2001)), personnel rostering, and configuration problems. Practical applications are discussed, for example, in (Freuder and Wallace, 2000; Rossi et al., 2006; Wallace, 2007b).

Many CP solvers have been designed, the currently most used ones including CHIP (by `www.cosytec.com`), Choco (Choco Team (2010), at `choco.emn.fr`), Comet (Van Hentenryck and Michel (2005), at `www.dynadec.com`), ECLiPSe (Apt and Wallace (2006), at `elipseclp.org`), IBM ILOG CPLEX CP Optimizer (formerly known as ILOG Solver), JaCoP (at `www.jacop.eu`), Gecode (Gecode Team (2006), at `www.gecode.org`), and SICStus Prolog (Carlsson et al. (1997), at `www.sics.se/sicstus`). It is worth noting that the modelling languages Comet (Van Hentenryck and Michel, 2005) and (Mini)Zinc (Garcia de la Banda et al. (2006), at `www.g12.csse.unimelb.edu.au/minizinc`) are equipped with *multiple* solvers of different combinatorial problem solving technologies, including CP and SLS.

2.5 Perspectives

We have seen (in Section 2.3.2) that the core idea of CP, namely the concept of constraint, can be successfully exported to another combinatorial problem solving technology, namely stochastic local search (SLS), giving rise to constraint-based local search (CBLS). This creates opportunities for a deeper hybridisation of propagation-based solving and CBLS solving (Wallace, 2007a). But there is no reason to stop with SLS. In fact, the concept of constraint is being exported to other combinatorial problem solving technologies, such as Boolean satisfiability (SAT), giving rise to satisfiability modulo theories (SMT), and integer programming (IP) (Hooker, 2007). We argue that CP is an ideal integration technology for hybrid solving.

Note that SLS traditionally has no notion of modelling language. Also, SAT and IP traditionally rely on modelling languages that are much lower-level (but still declarative) than those of CP and CBLS, but their solvers offer a *unique* black-box search heuristic, so that the user cannot investigate (as in CP and SLS) the design of a custom search (meta-)heuristic in order to override the default one in case the latter is deemed not efficient enough. For many (non-expert) users, the inconvenience of the lower-level modelling languages of SAT and IP is well offset by the availability of highly-tuned black-box search, as search tuning is to them a daunting aspect of CP and SLS. Such a reflex recourse to SAT and IP misses opportunities, though, as no combinatorial problem solving technology can outperform all the other ones on all problems. Anecdotal evidence about when to apply which technology, or which hybridisation of technologies, is slowly emerging. For example, propagation-based CP is emerging as the technology of choice for scheduling, personnel rostering, and configuration problems.

The remainder of this paper is an annotated bibliography on the usage of CP solvers, both propagation-based ones and CBLS ones, to problems in air traffic management. We also compare CP results with those of other techniques from artificial intelligence and knowledge representation, especially non-CBLS approaches to SLS, such as genetic programming; we are not aware of any papers using SAT or SMT solvers in this area. There is a very vast literature on using IP solvers

and custom ad hoc algorithms, and we cannot do justice to it within reasonable size limitations (but see Leal de Matos and Ormerod (2000) for instance).

It is important to note that most of the discussed papers do not describe an application that could in principle be deployed. Instead, most works describe *tools for long-term research* about investigating the impact of new approaches to air traffic management, especially that current approaches may not scale to future airspace demands. We argue that CP is an ideal technology for this scenario, at it allows very fast prototyping of new or revised constraints compared to ad hoc algorithm design, without sacrificing (much) in efficiency. The stringent real-time demands of actually deployed applications are secondary at this moment since faster hardware will be available by the time of deployment and since extra effort can be put until then into increasing the performance (if not into designing from scratch a custom algorithm).

3 Annotated Bibliography on Using CP for ATM Problems

In the context of air traffic management (ATM), most combinatorial problems are large-scale and generally NP-hard, so that their exact optimal solution may be unreachable in practical time. Pre-tactical to real-time algorithms may thus have to make many approximations, or trade completeness and optimality for bounded completion times. Moreover, the many sources of uncertainty (parameters of aircraft models in trajectory prediction, weather, failure of ATC systems, cancellation of flights, etc) and the degree of safety required imply some dynamic and reactive properties that ATM optimisation algorithms should fulfil. We discuss them in the next two paragraphs, before embarking on the literature survey.

Scale of the Problem and Operational Constraints. The whole problem of generating an optimal set of conflict-free 4D trajectories at continental scale (currently there are about $30,000$ flights daily in Europe) with all degrees of freedom (3D geometry and time, therefore with possibly millions of constraints) is out of reach for current combinatorial problem solving technologies. Even if it were possible, robustness to the many uncertainties (which increase with the time horizon considered) inherent to ATM is far too costly to be taken into account at the pre-tactical level. Algorithms therefore need to be chained (according to their time horizon and granularity), dynamic at the tactical level (such as near-real-time updates with a rolling horizon, in order to handle reasonable amounts of uncertainty), as well as possibly sub-optimal (in order to meet computation time requirements in this dynamic context), and therefore they cannot yet address simultaneously all degrees of freedom. However, with increasing computing power and versatile optimisation paradigms (such as stochastic local search, large neighbourhood search, and variable neighbourhood search), attempts at solving combinations of former models (such as the optimisation of the time *and* vertical dimensions by Flener et al. (2007b) as well as Barnier and Allignol (2011)) constitute current research activities.

Uncertainties and Dynamic Properties. The duration of the considered time windows must be carefully chosen, that is both large enough to provide enough opportunity (manoeuvres, delays, etc) to solve the constraints (capacity, conflicts, etc) and small enough to consider only a limited amount of uncertainty. Similarly, the time shift of the time window should be both as small as possible in order to take updates into account as soon as possible and thus to have enough anticipation to solve the constraints, and large enough to match the computation time of the algorithm. The 'past' part of the current solution is then fixed and the concerned decision variables are considered as constants, and the 'future' part of the problem is re-solved with the new horizon. Operationally, manoeuvres or other regulation measures must be dispatched sufficiently in advance to allow pilots or flight management systems to implement them; so the first few minutes of the decision variables of the 'future' part should be fixed as well.

We have divided into nine categories the literature on using constraint programming (CP) toward the modelling and solving of ATM problems. The first eight categories are ordered from strategic to tactical to operational management, and the last one is a category apart. We start with the *strategic issues* of route network design (see Section 3.1), contingency planning following a CFMU system disruption (see Section 3.2), and airspace sectorisation (see Section 3.3). Next come the *tactical issues*, namely sector opening scheme design (see Section 3.4), air-traffic controller workload management (see Section 3.5), and slot allocation (see Section 3.6). Note that *re-routing*, which is considered a part of pre-tactical flow management (see Section 1.2.1), is not discussed in this survey as the only relevant papers stem from operational research (Barnhart et al., 2003; Bertsimas and Stock Patterson, 1998). The *operational issues* are departure management and runway allocation (see Section 3.7) and arrival management (see Section 3.8). Note that real-time *conflict resolution* with horizontal, vertical, and speed manoeuvres to automate ATC is also out of the scope of this survey, as there are no CP papers on this topic, the main related studies being based on meta-heuristics (Granger et al., 2001), operational research (Vela et al., 2009), or expert systems (Alliot and Colin de Verdière, 2003). Finally, we discuss the usage of CP outside the traditional scope of operations management, namely for the purpose of ATM *software verification* (see Section 3.9).

3.1 Route Network Design

The current route network in Europe is based on radio beacons, as described in Section 1.1.1. However, with the increasing precision of navigation systems, other types of routes might be considered in order to enhance traffic flow management (shorter routes, lower workload, etc). In this context, Barnier and Brisset (2004) try to build a network of direct routes (i.e., straight routes from origin to destination) over the French airspace. Aircraft that share the same route are aggregated into flows, and a pairwise detection of intersections is performed between these flows. A constraint programming (CP) algorithm then performs a flight-level allocation, so that two intersecting flows are vertically separated. The application of this method to several days of French domestic traffic shows that at least 20 to 24 flight levels are necessary to avoid all crossings. However, unlike (Barnier and Allignol, 2011), this study does not take into account the preferred flight levels, which reflect the performance of aircraft.

Rather than building direct routes, which might be difficult to handle by controllers at the continental scale, Brisset and Rivière (2005) propose to forge a route network over Europe that minimises the travel distances, starting from a uniform grid in which cells are 240 km side-length squares; this methodology was driven by a new ATM concept called *sector-less* (Duong et al., 2001). A simulated annealing meta-heuristic is used to move nodes in the grid, thus modifying the network. Evaluation is performed using the Floyd-Warshall shortest-path algorithm. A second approach uses *invariants*, the analogue of constraints in constraint-based local search solvers (see Section 2.3.2), to maintain shortest paths, improving the results of the meta-heuristic. The results obtained under this approach lead to travel distances comparable with those on the current route network, and showed the non-viability of the sector-less concept.

Perspectives. The design of a route network is a challenging problem: the use of direct routes makes the traffic difficult to handle by controllers, whereas creating a network from an initial grid proved not to be better than the current network. More efficient approaches could consist in defining multiple routes for each origin-destination pair, and performing an allocation of flights to such routes, as proposed by Leal de Matos and Powell (2003). This would also help the process of dynamic re-routing in case of hazardous meteorological phenomena or active military zones for example.

3.2 Contingency Planning

Upon any failure of air traffic flow and capacity management (ATFCM), such as downtime of the computer-assisted slot allocation (CASA) system, no timely updates from ATFCM are available and the air traffic controllers of each aerodrome have no idea whether it is proper to release a flight or not. During the last thirteen years, such a situation has occurred once, for a few hours. Toward this, EUROCONTROL requires the publication of a *contingency plan*, which indicates for each major aerodrome and geographical direction a series of flow directives, each consisting of a time span and a maximum number of hourly departure slots for that time span. For instance, for destinations to the west of Brussels National Airport (Belgium), from 09:00 to 12:00 a maximum of 7 flights might be allowed to take off, but only 4 from 12:00 to 14:00. These flow directives are computed in such a way that some safety, efficiency, and fairness objectives are satisfied, also taking into account the traffic from minor aerodromes.

The generation of contingency plans within the Central Flow Management Unit (CFMU) of EUROCONTROL is currently done by human experts, with inadequate tool support for constructing *globally* optimal flow directives. Biannually, namely for the winter and summer timetables, they develop a three-fold contingency plan, namely one for weekdays, one for Saturdays, and one for Sundays. The total effort takes about two person-months per year. The current contingency plan can always be downloaded from the CFMU website (`http://www.cfmu.eurocontrol.int/`).

Contingency planning that is (more) automated and (more) geared toward constructing *globally* optimal flow directives is desirable, especially that it could then be done at the *tactical* level instead of the current strategic level, so that the quality of the generated contingency plans would be increased. Toward this, a feasibility study using constraint-based local search (CBLS) was done, initially for the sub-problem where the flows and time spans have already been identified, so that only the optimal hourly rates are to be inferred: these initial results are reported from a CP perspective by Sundequist Blomdahl et al. (2010a) and from an ATM perspective in Sundequist Blomdahl et al. (2010b). Subsequently, entire flow directives (including the time spans) for given flows were inferred (Sundequist Blomdahl et al., 2012). Flows typically do not change much between contingency plans anyway, but could also be pushed into the optimisation. Tested on the *entire* European airspace (with over 30,000 daily flights in the year 2008), the CBLS model (written in Comet) outperforms the human experts in terms of flight takeoff delays and sector capacity violations of the generated contingency plans.

Perspectives. The feasibility study was deemed successful and further development of the prototype is currently being considered by the CFMU, possibly leading to actual deployment.

3.3 Airspace Sectorisation

The design of control sectors described in Section 1.1.2 is largely based on expertise and simulation tools, and is carried out locally and incrementally to alleviate recurring congestion situations. Sectors are designed so as to keep the workload of controllers manageable for the typical flow of traffic that crosses the corresponding airspace along predefined routes (see Section 1.1.1). As mentioned in Section 1.3.2, the workload induced by the flow of traffic includes the monitoring of aircraft trajectories, the detection and resolution of conflicts, and the coordination with adjacent sectors whenever flights enter or exit the sector. The design of control sectors should therefore encompass these costs, as well as structural ATC hard constraints like convexity (and connectivity) with respect to air routes, minimal distance between a border and a crossing point, etc.

The main approaches to optimise the design of sectorisation are either graph-based models with vertices representing intersections and edges the available routes, or geometry-based models that tile the airspace with basic cells. In both cases, the solution consists of a partition of the

components of the model (vertices or cells) to form optimised control sectors that minimise controller workload and balance the traffic. The graph-based model is more compact than the geometry-based model, but the former does not directly define the sector borders as does the latter, so geometrical sectors still have to be constructed from the partition of vertices.

Building on the model of Delahaye et al. (1998), which solves the graph-based version of the problem with a genetic algorithm and uses a Voronoï diagram to define precisely the borders of sectors, Tran Dac et al. (2005) present a constraint programming (CP) approach taking most of the aforementioned operational constraints into account. To be able to solve large instances, the problem is solved with a hybrid algorithm using first a good solution obtained with a heuristic and then locally improving the solution with an exact CP formulation on small subsets of the sectors. Geometrical sectors are then built with triangulation techniques similar to the ones used by Delahaye et al. (1998). Problem instances of up to 1000 vertices to partition in 80 sectors are solved.

Jägare (2011) starts from a division of the airspace into hexagonal cells, say with a diameter of five nautical miles and a height of 1000 feet, and aims at clustering these cells into a given number of sectors, so that some hard constraints are satisfied (namely balanced sector workloads, no flight re-entries into the same sector, and no flight crossings of sectors in less than a minute) and the number of sector entry points is minimised. Custom propagators had to be designed for these constraints. The data for the experiments is provided by the *Arithmetic Simulation Tool for ATFCM and Advanced Concept* (ASTAAC) tool of the EUROCONTROL Experimental Centre. This tool actually pre-clusters some cells into abstract functional blocks (AFBs) according to additional constraints (sufficient distance from potential conflicts and trajectories to sector boundaries), so that the input to sectorisation is actually a division of the airspace into cells and AFBs. Experiments were run on up to a few hundred thousand cells, to be clustered into five sectors. In comparable run-times on the same machine, the constraint program produces much better sectorisations than *NEVAC Sector Builder*, which is a greedy algorithm that comes with ASTAAC and considers the no-re-entry and no-short-crossing constraints to be soft and yet does not systematically yield fewer sector entry points. Two hard constraints, namely the physical contiguity and compactness of the resulting sectors were initially not implemented: the resulting sectors are sometimes disconnected or of highly irregular shapes, so that current air traffic controllers would be very uncomfortable in working with them. The design of new constraints for enforcing contiguity and compactness is on-going work: when ready, they can be plugged into the current constraint program without changing anything else, except possibly the heuristics of the search procedure.

On a larger scale, the sectorisation problem once more occurs while designing air-traffic control centres (there are currently 75 ATCCs in Europe), which have the responsibility for a group of sectors located in a limited zone of airspace. Traditionally, European ATCCs were essentially limited to national airspace zones, but in the context of the Single European Sky, EUROCONTROL was mandated to establish transnational functional airspace blocks (FAB, not to be mixed up with AFB) so as to enforce regional cooperation in the seamless management and control of traffic flows in the ECAC zone. To this end, Bichot and Durand (2007) propose a novel meta-heuristic called *fusion fission* (and inspired by the corresponding nuclear-physical phenomenon), which is particularly well-suited to solve the graph partitioning problem used to model this airspace design problem: vertices are sectors, weighted by the number of aircraft crossing it, and edges between two neighbouring sectors are weighted by the number of aircraft passing from a sector to another one; the algorithm then attempts to minimise the weight of edges cut by the partition, which corresponds to coordination workload between ATCCs, and maximise the weight of vertices in each part, keeping an imbalance factor of 2 between parts (as the largest ATCCs may contain twice as many sectors as the smallest ones). The fusion fission meta-heuristic seems to outperform consistently the compared partitioning libraries and outputs

FABs that much improve over the coordination workload and imbalance of the current ATCC partition.

Perspectives. The implementation of airspace optimisations that alter the shape of basic control sectors implies a heavy cost in controller training, as controllers are highly specialised in the management of their specific sectors and it takes several years to qualify them on new sectors. Moreover, the redesign of ATCCs would induce a new dispatch of radar data and probably the building of costly new infrastructures to host them. Contrary to lighter optimisations that only concern current control structures without modifying them, like the optimisation of opening schemes described in the next section, changes in airspace design have very important transition costs and must be very carefully planned.

3.4 Sector Opening Scheme

In order to adapt the capacity of air traffic control centres (ATCCs) to traffic demand, flow management experts (called flow management position: FMP) work out an open-sectors plan for each ATCC (as described in Section 1.1.2), depending on the number of available controllers. The latter may vary during the day from a single pair handling the whole centre during off-peak hours to enough people to control each elementary sector individually, according to a predetermined schedule roughly fitting the traffic forecast. Only predetermined configurations can be used[8] and the schedule of open sectors should match the forecast as closely as possible without exceeding the declared capacity (associated with each possible configuration) in order to guarantee the safety of flights while avoiding over-sized ATCC configurations and the wasting of resources. Along with flight plans, these *opening schemes* define the main input to the *slot allocation* phase of the pre-tactical regulations (see Sections 1.2 and 3.6).

Delahaye et al. (1995) propose to model the optimisation of opening schemes as a problem of partitioning a graph into connected components, similarly to the optimisation of the sectorisation presented by Delahaye et al. (1998) and mentioned in Section 3.3. For a fixed number of regroupings (merged or individual sectors), the suggested optimisation criterion combines the balancing of the control workload in each sector and the minimisation of the coordination workload between sectors. As in (Delahaye et al., 1998), the problem is solved with a genetic algorithm and the process is iterated hourly in order to match dynamically the traffic, but the variation of the number of open sectors is not taken into account.

Kerlirzin et al. (2000) present an interactive tool for FMPs to minimise the number of required ATFCM regulations. Starting with all the capacity constraints, the corresponding slot allocation problem is solved with constraint programming (CP) as described by Chemla et al. (1995) and mentioned in Section 3.6. Only the most penalising sectors are then constrained to be regulated with a capacity limit, according to some user-defined participation threshold, and the slot allocation is computed again. If overloads appear to be too much in excess (above 20% in the study), the process is iterated with a smaller participation threshold.

To automate the determination of opening schedules, both Barnier (2002), using CP, and Gianazza et al. (2002), comparing two tree search methods (A* and branch-and-bound) and a genetic algorithm, solve the problem for predefined operational groups of sectors only and compute an optimal partition with respect to the distance to the maximal capacity for each hour, penalising overloads *and* underloads to avoid wasting ATC resources. The number of available controllers during the day is taken into account as an upper bound for the number of open sectors for each hourly configuration, which contributes to the optimisation criterion as well. Both studies report optimal results and short computation times for CP and the ad hoc branch-and-bound algorithm, resulting in less costly opening schemes, both in terms of required controllers and in terms of slot

[8]Not all subsets of elementary sectors may be used as a group of sectors, for obvious reasons like connectivity, or more subtle ones like the qualification of a controller being limited to given volumes of airspace.

allocation delay. Furthermore, Barnier (2002) attempts to take the transition between successive configurations into account as an additional constraint, in order to generate operationally feasible schemes at the price of a slight increase in the cost.

However, as explained in Section 1.4.2, the hourly capacity defined as a number of flights entering a sector may not be the most appropriate criterion to model the workload of controllers. Moreover, as remarked by Barnier (2002), actual sector openings observed during operations are only remotely connected to pre-tactical opening schemes, which are much more dynamic and almost constantly exceeding declared capacities. To model the ATC workload better, Gianazza and Guittet (2006) investigate the links between a set of metrics from the literature and the grouping or ungrouping decisions taken by controllers, which should correspond to workload thresholds. A neural network is trained on actual data and the most significant factors are subsequently extracted. Simulations with a neural network, considering a reduced set of metrics only, show the relevance of the approach, as the splitting and merging decisions are accurately predicted.

Perspectives. Sector opening schemes seem to be a good candidate for optimisation technologies like constraint programming, as they replace human expertise by sound optimisation algorithms without changing operational procedures, and as they may reduce the cost of both the slot allocation phase *and* ATC by better matching the offered capacity to the traffic demand. However, as capacities only approximately represent workload and as real openings generally do not closely adhere to the opening plan, the operational impact of optimised plans should be further assessed by real-life experimentation.

3.5 Workload Management

As stated in Section 1.4.2, reducing the workload of controllers is mandatory in order to be able to cope with an increasing European traffic demand. Considering the sector-based structure of the current ATC system, it is a way of increasing airspace capacity. In this context, many metrics have been defined to model workload and act as criteria for various optimisation methods: linear programming, stochastic local search, constraint programming, etc.

Hildum and Smith (2007) propose to reduce workload by constructing a movement plan that is conflict-free. Rather than considering conflict avoidance from a local and tactical perspective, a scheduling process in space and time is applied, taking into account likely areas of congestion. The airspace is modelled using octrees, a three-dimensional extension of the well-known quadtrees, which greatly improve the computational complexity of finding such constrained areas. This scheme is applied on a set of military-like missions where aircraft must reach a target from one of the departure bases and come back; the largest solved instance involved 1000 targets and 10 bases. Building upon this preliminary approach, a technique for improving robustness is introduced in (Hildum and Smith, 2012). A coarser-grained conflict relation models an envelope within which aircraft are allowed to move in order to avoid conflicts in a reactive manner. An octree structure is again used to generate conflicting neighbourhoods in a limited computation time.

The ERASMUS project (En-Route Air traffic Soft Management Ultimate System) of Villiers (2004) aims at dynamically reducing traffic workload by means of small speed adjustments. Trajectories are predicted over a 20-minute anticipation window and checked for potential conflicting situations. When possible, conflicts are avoided by slightly changing aircraft speed (usually within a -3% to $+6\%$ margin). Otherwise, that is when speed regulation is not enough, the controller in charge solves the conflict with usual techniques. Therefore, this method can be seen, from the controller's point of view, as a filter that could be integrated in two different ways into the current system: either as a *subliminal filter* where speed modifications are small enough to fit within the uncertainty on the controller's perception of speeds, or as an *informative filter* by asking the controller for validation of the speed change. The first prototypes for this project

use an evolutionary algorithm to solve conflicts, minimising the number of manoeuvres as a first criterion and the overall delay induced by the speed changes as a second one.

Instead of performing conflict avoidance in a tactical setting with a short notification time, Barnier and Allignol (2009) propose a conflict-free planning method based on constraint programming. An *exact* 4D conflict model is introduced, where each violated constraint implies a conflict in a fast-time simulation. The only degrees of freedom during planning are takeoff times, taken for each flight in a bounded interval (usually 90 minutes for national traffic). Starting from simulated 4D trajectories, the problem of allocating a takeoff time to each flight so that no conflict occurs during the considered time period (usually 24 hours) is modelled as a constraint satisfaction problem. The optimisation phase aims at minimising the maximum allocated delay, so as to preserve equity among airlines and be able to obtain an optimal solution in reasonable computation time. The global performance of the planning, which can be measured by the sum of all delays, is maintained at a low value thanks to adapted search heuristics. Instances involving up to 9000 flights (for 24 hours of traffic in the French airspace) are solved within one minute. More details about the model and results on numerous traffic samples are found in (Barnier and Allignol, 2012). In order to enhance further the planning, a prior flight-level allocation is introduced in (Barnier and Allignol, 2011), aiming at reducing the workload of the takeoff time allocation problem. This prior phase is also solved with constraint programming, minimising the deviation between allocated flight levels and optimal flight levels.

Flener et al. (2007a,b) resolve traffic workload in a multi-sector planning context. They use a definition (QinetiQ, 2004) of the *moment workload* of a sector s at a moment m as a weighted average of the number of flights that are at m inside s, the number of flights that are at m in climbing or descending phase inside s, and the number of flights that are at m at most two minutes beyond the entry to s or before the exit from s; note that the workload does *not* depend on potentially interacting pairs of flights, because it was found that traffic volume and vertical state already capture this effect. The *interval workload* of a sector s during a time interval $\mathcal{I}$ is the average of the moment complexities of s at sampled moments within $\mathcal{I}$. On a rolling horizon basis, the objective is to balance the complexities of the sectors of the considered airspace during a short interval some 20 to 90 minutes into the future. This is achieved as a side-effect of minimising the sum of the interval complexities of the considered sectors during that interval. The allowed forms of workload resolution, and hence the constraints, are any combination of the following:

- Temporal Re-Profiling. Change the entry time of a flight into the chosen airspace: if it is not airborne yet, then change its takeoff time by an integer amount of minutes in $[-5, \ldots, +10]$, else change its remaining approach time by a speed-up rate of at most 5% or a slow-down rate of at most 10%.
- Vertical Re-Profiling. Change the altitude of passage of a flight over a way-point in the chosen airspace by an integer amount of flight levels within $[-30, \ldots, +10]$, so that the flight climbs at most 10 levels per minute, or descends at most 30 levels per minute if it is a jet, or descends at most 10 levels per minute if it is a turbo-prop.

The underlying assumptions are that sector entry times can be controlled with an accuracy of one minute (so that a flight profile is just shifted in time upon temporal re-profiling) and that flight time between two waypoints does not change if one restricts the vertical re-profiling to be "small." The resulting re-profiling can be subliminal from the air traffic controller perspective. In order to make this a non-zero-sum game, the user can set the percentage of flights that must be kept within the chosen airspace (rather than risk being delayed into the next interval); also note that vertical re-profiling can reduce the workload of a sector without increasing it in another sector. A prototype was designed by constraint programming to experiment on the upper-airspace sectors of the BeNeLux countries, which have very high traffic demands, during the busiest day of the year 2004: compared to the original CFMU flight profiles, two runtime minutes suffice to almost halve the complexities and balance them much better. Constraints (such as lower and

upper workload bounds after the re-balancing, first-scheduled first-served ground-holding, airline equity on re-profiling, etc) can be added in a modular fashion to make the scenario more realistic.

Perspectives. Handling uncertainties remains one of the main challenges for workload management, which is often implemented as a strategic filter (see Section 1). Indeed, the prediction of aircraft positions far in advance is currently subject to numerous sources of uncertainty (wind force and direction, takeoff time, vertical profile, etc) that can only be dispelled at a shorter term. Therefore, regulation measures are too costly for large uncertainties, so that a dynamic iteration process can be necessary in order to update periodically the flight information and thus enhance the prediction, as mentioned in the introduction of Section 3.

3.6 Slot Allocation

Since the early 1990s, several studies have been carried out to improve ATFCM regulations, in particular by research and development teams at the French civil aviation research centre, CENA (Centre d'Études de la Navigation Aérienne), such as the early work of Chemla et al. (1995), which laid the foundation for the modelling by constraint programming (CP) of the slot allocation problem, and at the EUROCONTROL Experimental Centre (EEC) in Brétigny, France. The algorithm that computes the slot allocation in the CASA software, described by CFMU (2011) and in Section 1.2.2, is a greedy algorithm, very robust and efficient in a dynamic operational context, but not based on a rigorous mathematical modelling and suffering from some flaws that combinatorial optimisation technologies like CP can help to prevent.

First, CASA does not try to maintain a valid solution whenever a flight is subject to multiple delays, as the two main constraints of its specification, namely sector capacity and the first-scheduled first-served principle, can be violated. However, most studies, especially the one by Dalichampt et al. (2001), question the pertinence of adherence to the latter, which has too costly an impact on the overall amount of delay to justify the notion of fairness it implements. A few other studies, such as those by Degrand and Mercier (2000) and Hassani Bijarbooneh et al. (2009), allow some capacity constraints to be relaxed so as to provide the best possible solutions for over-constrained instances in a dynamic operational context.

Second, CASA does not attempt to optimise globally the amount of delay – which is an NP-hard problem – besides locally improving later slots when flights are dynamically cancelled or postponed. All cited studies formulate a proper mathematical model and solve it with well-established combinatorial optimisation algorithms belonging to operational research, constraint programming, or stochastic local search. Depending on the qualities of the algorithm and the goal of the study, either an optimal solution or a feasibility proof is searched for with complete algorithms such as those by Degrand and Mercier (2000) and Barnier et al. (2001), or good solutions are obtained quickly with stochastic algorithms such as those by Hassani Bijarbooneh et al. (2009) and Junker (2012).

A side effect of these mathematical formulations of the ATFCM problem is to challenge the relevance of sector capacity being defined as a number of aircraft entering a given sector in one hour (see Section 1.1.2), because the time period over which the constraint should be enforced is not clearly specified. CASA implicitly considers a very tight version equivalent to imposing the constraint over any time period starting at multiples of the length of the slots (i.e., $60/\mathit{capacity}$), such that the capacity demand is evenly distributed in a regulated sector. However, as explained by Barnier et al. (2001) and Junker (2012), if ATFCM models literally adhere to the definition of sector capacity by opening schemes (see Section 1.1.2), then the capacity demand may reach twice the bound specified by the constraint for the intermediate time windows, which would jeopardise the safety of flights. Also, traffic peaks are likely to occur over short time periods, especially at the beginning of the hourly constraints, where flights exceeding the capacity of the previous time window would be put back by optimising algorithms to maintain the delay cost as low as possible. To address this issue, Barnier et al. (2001) compare overlapping hourly constraints

and a dual model using the *sort* constraint introduced by Bleuzen-Guernalec and Colmerauer (1997), equivalent to placing hourly constraint at any time step, whereas Junker (2012) studies the impact of *smoothing* constraints, which are capacity constraints over shorter time periods (down to 10 minutes), proportional to the hourly one.

Perspectives. Other studies focusing on controller behaviour, such as the one by Gianazza and Guittet (2006), have concluded that capacity alone does not accurately estimate the workload but that other factors have to be taken into account (see 1.4.2). Degrand and Mercier (2000) compare regulations constrained by classical capacity and by *load capacity constraint regulation* (LCCR), which restricts the number of flight simultaneously allowed in a given sector at each time step and is not as tight as capacity, concluding that the latter is much more efficient with respect to the amount of delay. Barnier and Allignol (2012) even propose to abandon the aggregated model with sectors and capacity, and to compute directly a conflict-free slot allocation to decrease controller workload and increase ATC efficiency.

The question also arose whether, instead of dynamically adjusting airspace demand to a static airspace capacity, one should rather dynamically adjust airspace capacity to airspace demand, by dynamically designing sectors that can handle the current demand, as discussed in Section 3.3.

3.7 Departure Management and Runway Allocation

At an airport, there are many flights that need to be managed both before and after takeoff. For takeoff, this is typically done by three controllers: the preflight controller, the taxiway controller, and finally the runway controller. Each controller tries to optimise the flow of flights for his section. The runway controller is given a sequence of flights by the preflight controller, and there are, at that point, only limited possibilities to reorder the flights. The runway controller then directs each plane to a specific exit point into the airspace of the airport. Each flight is allocated a departure slot, which is specified as a 15-minute time window. At an airport, the runways are a scarce resource. A flight has to be managed from leaving the airport stand to takeoff and finally to its exit point, while taking into account possible conflicts with other flights and the use of scarce physical resources.

In (van Leeuwen and van Hanxleden Houwert, 2003), a decision support tool is presented to aid the controllers in planning the movements of flights within an airport and its airspace. The problem of optimising flights is modelled first using constraint programming (CP). The constraints capture the topological layout of an airport, the required takeoff times of the flights, the characteristics of each flight (depending on the weight of the aircraft, only certain runways can be used), adequate separation between takeoffs on the same runway, and the eventual required exit point from the airspace of the airport.

The CP model is able to schedule medium-size airports, such as Prague (Czech Republic), but for larger airports either no solution is found quickly enough (the tool is intended to be used in real time) or no solution is found. This is because for larger airports, the given constraint problem is often over-constrained. That is, there is no solution that satisfies all the constraints simultaneously. An extension is proposed that uses constraint relaxations. When the CP model finds no solution in time, the controller is given the option to relax some of the constraints in order to find a solution. The system provides, when possible, reasons why no solution can be found, and gives the controller the option to reschedule individual flights. Priority is given not to overload the controllers with too much information, but to offer them enough information to provide a simple resolution of the scheduling conflict.

Perspectives. With the current trend of extending saturated airports with new runways and taxiways to accommodate the growing demand, airport traffic becomes more complex to manage, leaving room for optimisation technologies (such as CP) capable of taking into account the diverse operational constraints that taint the combined allocation and shortest path

problems inherent to airport management. Furthermore, depending on the airport configuration, departure management generally is closely intertwined with arrival management, such that airport management tools should simultaneously optimise both the takeoff rates and the landing rates, like the airport simulator and optimiser based on branch-and-bound and genetic algorithms proposed by Deau et al. (2009).

3.8 Arrival Management

As airport capacity is becoming a scarce resource in the ATM system, it is essential to plan the traffic carefully in approach areas in order to avoid traffic peaks. These approach areas have a particularly high traffic density, as many aircraft converge to a single point before the runway. Furthermore, successive landings must be sequenced, taking into account a minimal time interval corresponding to a safety spacing so as to avoid wake vortex.

Artiouchine et al. (2008a) present a scheduling approach to this problem. All possible trajectories are pre-computed, taking into account allowed manoeuvres: speed changes that slightly modify the landing time, and the use of *holding patterns* that generate a constant delay (usually four minutes). This pre-processing provides a set of possible landing times for each aircraft. A branch-and-cut method is proposed to solve the resulting scheduling problem, using constraint propagation to filter the search space. This approach outperforms the standard mixed integer programming (MIP) approach by Bayen et al. (2004), solving instances with up to 90 jobs (i.e., aircraft) within a few seconds.

Artiouchine et al. (2008b) propose an analogy between the arrival sequencing problem and the k-king problem. In this simplified model, time is discretised and airspace is represented as a two-dimensional chessboard, where the runway is a special square. Aircraft move as a king does on a chessboard. The objective is to get all aircraft to land (i.e., to empty the chessboard) as soon as possible. The k-king problem is modelled as a constraint satisfaction problem and several constraint propagation algorithms are tested. Solutions are obtained for small instances, ranging from 6 aircraft on a 5×5 chessboard to 14 aircraft on a 7×7 chessboard. Future work is to use techniques such as *no-good recording* (Schiex and Verfaillie, 1993) in order to speed up the search and thus be able to handle larger instances.

Perspectives. Both methods are limited to single-runway configurations. Taking multiple runways into account,[9] as proposed by Berge et al. (2006) with a branch-and-bound approach, might lead to better results, but at the cost of larger problem instances.

3.9 Verification

For a few years now, constraint programming (CP) has also been used for software verification. The aim is to produce test data for a given piece of code or to perform verification tasks on programs, such as finding non-executable parts. Gotlieb (2012) proposes a combination of bound-consistency filtering and linear programming for this task, with an application on a critical part of the Traffic Alert and Collision Avoidance System (TCAS). TCAS is an on-board conflict detection and resolution system that outputs warnings about possible short-term collisions and manoeuvres in order to avoid them. TCAS is part of the emergency filter (see Section 1). In this approach, the input source code is transformed into a constraint system P using predefined rules for each construct (function calls, conditionals, etc). If one wants to check an assertion A, then a solution search is performed on $P \wedge \neg A$, which either validates A (if no solution is found) or exhibits test input that violates it. The tested source code contained 173 lines of C code, with ten properties to prove. The task was performed within 20 seconds, with six properties proved and counter-examples found for each of the four remaining properties.

[9]Large airports can operate two or more runways at a time.

Perspectives. Further work is planned to address larger programs with thousands of lines of code. A dedicated finite-domain solver could be developed toward this.

4 Conclusion

Air traffic management (ATM) constitutes a very complex domain of hard real-world combinatorial problems whose instances tend to be very large. These problems pose important computational challenges, and any advances in solving them better or faster can entail significant improvements in the safety, efficiency, and fairness of air traffic.

After discussing the principal issues of ATM and giving a tutorial on the constraint programming (CP) paradigm for the modelling and solving of combinatorial problems, we have provided an annotated bibliography of papers tackling ATM problems using CP, comparing when relevant with other methods stemming from artificial intelligence.

Most of the discussed papers originate from *long*-term *research* projects, meaning that the developed models and methods are not (intended to be) deployed (yet), but are tools for investigating the impact of *new* approaches to ATM, especially that current approaches may not scale to future airspace demands. However, some of the results of these projects could in principle be deployed.

Overwhelmingly, the papers conclude that the usage of CP was critical to the success of the projects. Often, CP is lauded for allowing the very fast prototyping of new or revised constraints whose impact needs to be evaluated. CP excels at this scenario, without sacrificing much, if anything, in solving efficiency, compared to doing it all by ad hoc algorithm design; see (van Leeuwen and van Hanxleden Houwert, 2003) for instance. Sometimes, the usage of CP is credited for enabling the investigation of new ATM procedures, which turned out to improve significantly the quality of the results of the currently deployed ones; see (Dalichampt et al., 2001; Junker, 2012) for instance. The high level of abstraction of CP constraint modelling languages sometimes allowed ATM researchers who are not CP experts to read and validate CP models; see (Flener et al., 2007b) for instance.

Similarly, the prior existence of the CUMULATIVE constraint (Aggoun and Beldiceanu, 1993) of CP enabled Degrand and Mercier (2000) to investigate regulations based on the more realistic concept of sector *load* constraints rather than the currently used sector *capacity* constraints, and the results were very positive; with the new ATM procedures available in the meantime, this switch from capacity constraints to load constraints could actually be implemented today, especially that very advanced implementations of CUMULATIVE are available for reuse (Kameugne et al., 2011).

It is not unusual that state-of-the-art solutions address only a relaxation of the actual combinatorial problem, in the sense that some constraints are not enforced. This may be so because the deployed problem solving technology excels on the structure of the relaxed problem, but would perform poorly on a more realistic (and thus harder) version of the problem. Here again, CP shines with its versatility: it may not offer the fastest solving on some well-studied combinatorial structures for which (possibly ad hoc) solvers exist, but it excels when such combinatorial structures are rendered more complex (and probably less-studied) through the addition of *side constraints*; see (Barnier et al., 2001; Barnier and Brisset, 2004; Tran Dac et al., 2005; Artiouchine et al., 2008b) for instance.

However, not all constraint problems in ATM are combinatorial problems: many constraint problems have decision variables ranging over domains that are infinite or non-discrete, such as real-number intervals. While CP approaches to such problems exist, such as RealPaver (Granvilliers and Benhamou, 2006), as well as a preliminary study on real-time conflict resolution using mixed CP models (Feydy et al., 2005), these problems are outside the scope of this survey.

Other combinatorial problem solving technologies, such as integer programming, share with CP the rigorous mathematical modelling of problems, and they are being used to solve the same ATM problems. The projects surveyed here included CP experts who simply used the

technology they know best, and comparisons with other technologies were beyond the scope of these projects. The claim rather is that CP allowed these experts to program alternative or additional constraints faster than under other technologies, including ad hoc algorithm design. Currently, cross-fertilisation and hybridisation between technologies is being investigated; see (Hooker, 2007) for instance.

Regarding the future, it is our hope to see CP-based solutions actually deployed in a real-life ATM context. Toward that, a better handling of the uncertainty underlying the input data is required. CP techniques for on-line stochastic problem solving are emerging, see (Van Hentenryck and Bent, 2006) for instance.

Acknowledgements

Many thanks to the anonymous referees for their useful feedback. The first and second authors wish to thank Jean-Marc Alliot and Nicolas Durand, who were heads of the Global Optimisation Laboratory at the late French Civil Aviation Research Centre (Centre d'Études de la Navigation Aérienne), for having introduced us to the ATM optimisation realm, and Jean-Marc Garot, former head of the EUROCONTROL Experimental Centre in Brétigny, for helping us acquire some perspective on the subject during USA/Europe ATM Research and Developement Seminars. The third and fourth authors thank Mete Çeliktin, Søren Dissing, Carlos Garcia-Avello, Bernard Delmée, Jacques Lemaître, and Patrick Tasker of EUROCONTROL Headquarters, as well as Franck Ballerini, Marc Bisiaux, Marc Bourgois, Marc Dalichampt, Hamid Kadour, Bernard Kerstenne, Serge Manchon, Elisabeth Petit, and Leïla Zerrouki at the EUROCONTROL Experimental Centre, for many stimulating discussions and the provision of exciting ATM problems and datasets.

Glossary

AFB	Abstract Functional Blocks
ANSP	Air Navigation Service Provider
ASM	Airspace Management
ASTAAC	Arithmetic Simulation Tool for ATFCM and Advanced Concept
ATC	Air Traffic Control
ATCC	Air Traffic Control Centre
ATFCM	Air Traffic Flow and Capacity Management
ATM	Air Traffic Management
BC	Bound Consistency
CASA	Computer-Assisted Slot Allocation
CBLS	Constraint-Based Local Search
CFMU	Central Flow Management Unit
CODA	Central Office for Delay Analysis
CENA	Centre d'Études de la Navigation Aérienne
CP	Constraint Programming
DC	Domain Consistency
ECAC	European Civil Aviation Conference
EEC	EUROCONTROL Experimental Centre
EGNOS	European Geostationary Navigation Overlay Service
ERASMUS	En-Route Air Traffic Soft Management Ultimate System
ETFMS	Enhanced Tactical Flow Management System
FAA	Federal Aviation Administration
FAB	Functional Airspace Block
FL	Flight Level
FMP	Flow Management Position

FMS	Flight Management System
FRAM	Free Route Airspace Maastricht
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
ICAO	International Civil Aviation Organisation
IP	Integer Programming
IFR	Instrument Flight Rules
LCCR	Load Capacity Constraint Regulation
MIP	Mixed Integer Programming
NextGen	Next Generation Air Transportation System
NM	Nautical Mile
RNAV	Area Navigation
SAT	Satisfiability problem
SESAR	Single European Sky ATM Research
SLS	Stochastic Local Search
SMT	Satisfiability Modulo Theories
STATFOR	Statistics and Forecasts
TCAS	Traffic alert and Collision Avoidance System
VFR	Visual Flight Rules

References

A. Aggoun and N. Beldiceanu. Extending CHIP in order to solve complex scheduling and placement problems. *Mathematical and Computer Modelling*, 17(7):57–73, 1993.

J.-M. Alliot and D. Colin de Verdière. ATM: 20 ans d'effort et perspectives. In *Symposium de l'Académie Nationale de l'Air et de l'Espace: vers l'automatisation du vol et sa gestion*, 2003. Available at `http://pom.tls.cena.fr/papers/articles/anae03.pdf`.

J.-M. Alliot, N. Durand, and G. Granger. A statistical analysis of the influence of vertical and ground speed errors on conflict probe. In *Proceedings of ATM'01, the 4th USA/Europe R&D Seminar on Air Traffic Management*, 2001. Available at `http://www.atmseminar.org/seminarContent/seminar4/papers/p_118_DSTCDM.pdf`.

K. R. Apt and M. Wallace. *Constraint Logic Programming using ECLiPSe*. Cambridge University Press, 2006.

K. Artiouchine, P. Baptiste, and C. Dürr. Runway sequencing with holding patterns. *European Journal of Operational Research*, 189(3):1254–1266, September 2008a. (Early version in *Proceedings of the 2nd International Workshop on Discrete Optimization Methods in Production and Logistics*, pages 96–101, Omsk-Irkutsk, Russia, 2004).

K. Artiouchine, P. Baptiste, and J. Mattioli. The k king problem, an abstract model for computing aircraft landing trajectories: On modeling a dynamic hybrid system with constraints. *INFORMS Journal on Computing*, 20(2):222–233, Spring 2008b.

P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer Academic Publishers, 2001.

C. Barnhart, P. Belobaba, and A. R. Odoni. Applications of operations research in the air transport industry. *Transportation Science*, 37(4):368–391, November 2003. Available at `http://cassi.nuaa.edu.cn/download/paper/1148801073.pdf`.

N. Barnier. *Application de la programmation par contraintes à des problèmes de gestion du trafic aérien*. PhD thesis, Institut National Polytechnique de Toulouse, France, December 2002. Available at `http://pom.tls.cena.fr/papers/thesis/barnier.pdf`.

N. Barnier and C. Allignol. 4D-trajectory deconfliction through departure time adjustment. In S. Saunders-Hodge and V. Duong, editors, *Proceedings of ATM'09, the 8th USA/Europe R&D Seminar on Air Traffic Management*, 2009. Available at `http://www.atmseminar.org/seminarContent/seminar8/papers/p_143_NSTFO.pdf`.

N. Barnier and C. Allignol. Combining flight level allocation with ground holding to optimize 4D-deconfliction. In S. Saunders-Hodge and V. Duong, editors, *Proceedings of ATM'11, the 9th USA/Europe R&D Seminar on Air Traffic Management*, 2011. Available at `http://www.atmseminar.org/seminarContent/seminar9/papers/157-Barnier-Final-Paper-4-5-11.pdf`.

N. Barnier and C. Allignol. Trajectory deconfliction with constraint programming. *Knowledge Engineering Review*, (this issue), 2012. Special Issue on Constraint Programming for Air Traffic Management.

N. Barnier and P. Brisset. Graph coloring for air traffic flow management. *Annals of Operations Research*, 130(1–4):163–178, August 2004.

N. Barnier, P. Brisset, and T. Rivière. Slot allocation with constraint programming: Models and results. In H. McLaurin, editor, *Proceedings of ATM'01, the 4th USA/Europe R&D Seminar on Air Traffic Management*, 2001. Available at `http://www.atmseminar.org/seminarContent/seminar4/papers/p_119_ITFODM.pdf`.

A. M. Bayen, C. J. Tomlin, Y. Ye, and J. Zhang. An approximation algorithm for scheduling aircraft with holding time. In *Proceedings of CDC'04, the 43rd IEEE Conference on Decision and Control*, volume 3, pages 2760–2767. IEEE, 2004.

N. Beldiceanu, M. Carlsson, and T. Petit. Deriving filtering algorithms from constraint checkers. In M. Wallace, editor, *Proceedings of CP'04, the 10th International Conference on Principles and Practice of Constraint Programming*, volume 3258 of *LNCS*, pages 107–122. Springer-Verlag, 2004.

N. Beldiceanu, M. Carlsson, and J.-X. Rampon. Global constraint catalogue: Past, present, and future. *Constraints*, 12(1):21–62, March 2007. The catalogue is at `http://www.emn.fr/z-info/sdemasse/gccat`.

M. E. Berge, A. Haraldsdottir, and J. Scharl. The multiple runway planner (MRP): Modeling and analysis for arrival planning. In *Proceedings of DASC'06, the 25th Digital Avionics Systems Conference*, pages 1–11. IEEE, 2006.

D. Bertsimas and S. Stock Patterson. The air traffic flow management problem with enroute capacities. *Operations Research*, 46(3):406–422, May/June 1998. `http://www.mit.edu/~dbertsim/papers/AirTransportation/The%20air%20traffic%20flow%20management%20problem%20with%20enroute%20capacities.pdf`.

C.-E. Bichot and N. Durand. A tool to design functional airspace blocks. In S. Saunders-Hodge and V. Duong, editors, *Proceedings of ATM'07, the 7th USA/Europe R&D Seminar on Air Traffic Management*, 2007. Available at `http://www.atmseminar.org/seminarContent/seminar7/papers/p_169_DAM.pdf`.

N. Bleuzen-Guernalec and A. Colmerauer. Narrowing a $2n$-block of sorting in $O(n \log n)$. In G. Smolka, editor, *Proceedings of CP'97, the 3rd International Conference on Principles and Practice of Constraint Programming*, volume 1330 of *LNCS*. Springer-Verlag, 1997.

C. Bontemps and K. Guittet. Commentaires sur l'étude de l'Université de Westminster portant sur les coûts des délais ATC. Note technique, Centre d'Études de la Navigation Aérienne, Toulouse, France, July 2004.

F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais. Boosting systematic search by weighting constraints. In R. López de Mántaras and L. Saitta, editors, *Proceedings of ECAI'04, the 16th European Conference on Artificial Intelligence*, pages 146–150. IOS Press, 2004.

P. Brisset and T. Rivière. Shortest path in planar graph and air route network. In *Proceedings of INO'05, the 4th EUROCONTROL Innovative Research Workshop & Exhibition*, 2005. Available at `http://inoworkshop.eurocontrol.fr/Previous/index-53745.pdf`.

M. Carlsson, G. Ottosson, and B. Carlson. An open-ended finite domain constraint solver. In H. Glaser, P. Hartel, and H. Kuchen, editors, *Proceedings of PLILP'97*, volume 1292 of *LNCS*, pages 191–206. Springer-Verlag, 1997.

CFMU. Basic CFMU Handbook – General & CFMU Systems. Edition 15.0, EUROCONTROL / Central Flow Management Unit, Brussels, Belgium, March 2011. Available at `http://www.cfmu.eurocontrol.int/cfmu/gallery/content/public/library/handbook_supplements/basic_handbook/docu_general_systems_latest.pdf`.

D. Chemla, D. Diaz, P. Kerlirzin, and S. Manchon. Using clp(FD) to support air traffic flow management. In A. Roth and A. Marien, editors, *Proceedings of PAP'95, the 3rd international conference on Practical Applications of Prolog*, Paris, France, April 1995. Alinmead Software Ltd.

Choco Team. Choco: An open source Java constraint programming library. Research report 10-02-INFO, École des Mines de Nantes, 2010. Available at `http://choco.emn.fr/`.

CODA. CODA digest – Delays to air transport in Europe. Technical report, EUROCONTROL – Central Office for Delay Analysis, Brussels, Belgium, 2009. Available at `http://www.eurocontrol.int/sites/default/files/content/documents/official-documents/facts-and-figures/coda-reports/coda-reports-2009.zip`.

A. J. Cook and G. Tanner. The challenge of managing airline delay costs. In *Summer School / Workshop on Air Traffic Management Economics*, Belgrade, Serbia, September 2009. Available at `http://www.garsonline.de/Downloads/090910/Papers/Paper_Cook%20&%20Tanner.pdf`.

A. J. Cook, G. Tanner, and S. Anderson. Evaluating the true cost to airlines of one minute of airborne or ground delay: Final report. Technical report, EUROCONTROL and University of Westminster, May 2004. Available at `http://westminsterresearch.wmin.ac.uk/17/1/Cook%2CTanner%2CAnderson_2004_final.pdf`.

M. Dalichampt, E. Petit, U. Junker, and J. Lebreton. Innovative slot allocation: an overview. EEC Note No. 10/01, EUROCONTROL Experimental Centre, Brétigny, France, May 2001. Available at `http://www.eurocontrol.int/eec/gallery/content/public/document/eec/report/2001/016_Innovative_Slot_Allocation.pdf`.

R. Deau, J.-B. Gotteland, and N. Durand. Airport surface management and runways scheduling. In S. Saunders-Hodge and V. Duong, editors, *Proceedings of ATM'09, the 8th USA/Europe R&D Seminar on Air Traffic Management*, 2009. Available at `http://www.atmseminar.org/seminarContent/seminar8/papers/p_081_AO.pdf`.

J. Degrand and E. Mercier. Load capacity constraint regulation. EEC Note 02/2000, EUROCONTROL Experimental Centre, Brétigny, France, 2000. Available at `http://www.eurocontrol.int/eec/gallery/content/public/document/eec/report/2000/003_Load_Capacity_Constraint_Regulation.pdf`.

D. Delahaye, J.-M. Alliot, M. Schoenauer, and J.-L. Farges. Genetic algorithms for automatic regroupment of air traffic control sectors. In *Proceedings of the 4th Annual Conference on Evolutionary Programming (EP'95)*, 1995. Available at `http://pom.tls.cena.fr/papers/articles/ep95.pdf`.

D. Delahaye, M. Schoenauer, and J.-M. Alliot. Airspace sectoring by evolutionary computation. In *Proceedings of the IEEE International Congress on Evolutionary Computation*, pages 218–223. IEEE, 1998.

V. Duong, G. Gawinowski, J.-P. Nicolaon, and D. Smith. Sector-less air traffic management. In H. McLaurin, editor, *Proceedings of ATM'01, the 4th USA/Europe R&D Seminar on Air Traffic Management*, 2001. Available at `http://www.atmseminar.org/seminarContent/seminar4/papers/p_158_NC.pdf`.

V. N. Duong, E. Hoffman, and J.-P. Nicolaon. Initial results of investigation into autonomous aircraft concept (freer-1). In X. Fron and H. McLaurin, editors, *Proceedings of ATM'97, the 1st USA/Europe R&D Seminar on Air Traffic Management*, 1997. Available at `http://www.atmseminar.org/seminarContent/seminar1/papers/p_018_ASSP.pdf`.

EUROCONTROL. *As the Crow Flies – Free Route Airspace Maastricht*. Brussels, Belgium, March 2011. Available at `http://www.eurocontrol.int/system/files/sites/default/files/content/documents/2011march-free-route-airspace-maastricht.pdf`.

FAA. NextGen implementation plan. Technical Report 202-493-4939, Federal Aviation Administration, March 2011. Available at `http://www.faa.gov/nextgen/media/ng2011_implementation_plan.pdf`.

T. Feydy, N. Barnier, P. Brisset, and N. Durand. Mixed conflict model for air traffic control. In *Interval Analysis, Constraint Propagation, Applications (IntCP 2005), a workshop of CP 2005*, 2005. Available at `http://pom.tls.cena.fr/papers/articles/intcp2005.pdf`.

P. Flener, J. Pearson, M. Ågren, C. Garcia Avello, M. Çeliktin, and S. Dissing. Air-traffic complexity resolution in multi-sector planning using constraint programming. In C. Pusch and S. Saunders-Hodge, editors, *Proceedings of ATM'07, the 7th USA/Europe R&D Seminar on Air Traffic Management*, 2007a. Available at `http://www.atmseminar.org/seminarContent/seminar7/papers/p_023_IAC.pdf`.

P. Flener, J. Pearson, M. Ågren, C. Garcia Avello, M. Çeliktin, and S. Dissing. Air-traffic complexity resolution in multi-sector planning. *Journal of Air Transport Management*, 13(6): 323–328, November 2007b.

E. C. Freuder and M. Wallace. Constraint technology and the commercial world. *IEEE Intelligent Systems*, 15(1):20–23, 2000. Special Issue on Constraints.

M. Garcia de la Banda, K. Marriott, R. Rafeh, and M. Wallace. The modelling language Zinc. In F. Benhamou, editor, *Proceedings of CP'06, the 12th International Conference on Principles and Practice of Constraint Programming*, volume 4204 of *LNCS*, pages 700–705. Springer-Verlag, 2006.

Gecode Team. Gecode: A generic constraint development environment, 2006. Available at `http://www.gecode.org/`.

D. Gianazza and K. Guittet. Selection and evaluation of air traffic complexity metrics. In *Proceedings of DASC'06, the 25th Digital Avionics Systems Conference*. IEEE, 2006. Available at `http://pom.tls.cena.fr/papers/articles/dasc25_307giana.pdf`.

D. Gianazza, J.-M. Alliot, and G. Granger. Optimal combinations of air traffic control sectors using classical and stochastic methods. In *The 2002 International Conference on Artificial Intelligence IC-AI'02*, 2002. Available at `http://pom.tls.cena.fr/papers/articles/intl02_gianazza_alliot2.pdf`.

F. Glover and M. Laguna. Tabu search. In *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–150. John Wiley & Sons, 1993.

A. Gotlieb. TCAS software verification using constraint programming. *Knowledge Engineering Review*, (this issue), 2012. Special Issue on Constraint Programming for Air Traffic Management.

G. Granger, N. Durand, and J.-M. Alliot. Optimal resolution of en route conflicts. In *Proceedings of ATM'01, the 4th USA/Europe R&D Seminar on Air Traffic Management*, 2001. Available at `http://pom.tls.cena.fr/papers/articles/atm2001geraud.pdf`.

L. Granvilliers and F. Benhamou. Algorithm 852: RealPaver: An interval solver using constraint satisfaction technique. *ACM Transactions on Mathematical Software*, 32(1):138–156, March 2006.

F. Hassani Bijarbooneh, P. Flener, and J. Pearson. Dynamic demand-capacity balancing for air traffic management using constraint-based local search: First results. In Y. Deville and C. Solnon, editors, *Proceedings of LSCS'09, the 6th International Workshop on Local Search Techniques in Constraint Satisfaction*, volume 5 of *Electronic Proceedings in Theoretical Computer Science*, pages 27–40, 2009. Available at `http://dx.doi.org/10.4204/EPTCS.5.3`. (Also in: Schaefer, Dirk, editor, *Proceedings of INO'09, the 8th EUROCONTROL Innovative Research Workshop & Exhibition*, 2009).

D. W. Hildum and S. F. Smith. Constructing conflict-free schedules in space and time. In M. S. Boddy, M. Fox, and S. Thiébaux, editors, *Proceedings of ICAPS'07, the 17th International Conference on Automated Planning and Scheduling*, pages 184–191. AAAI Press, 2007. (Extended version in this volume).

D. W. Hildum and S. F. Smith. Scheduling safe movement of air traffic in crowded air spaces. *Knowledge Engineering Review*, (this issue), 2012. Special Issue on Constraint Programming for Air Traffic Management.

J. N. Hooker. *Integrated Methods for Optimization*. Springer-Verlag, 2007.

H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann, 2004.

ICAO. Procedures for air navigation services – Rules of the air and air traffic services. Document 4444, International Civil Aviation Organization, Montréal, Canada, November 1996. Current edition available at `http://store1.icao.int/documentItemView.ch2?ID=7139`.

ITA. *Cost of Air Transport Delay in Europe*. Paris, France, November 2000. Available at `http://www.eurocontrol.int/sites/default/files/content/documents/single-sky/pru/publications/other/cost-of-air-transport-delay-in-eu-ita.pdf`.

P. Jägare. Airspace sectorisation using constraint programming. Master's thesis, Uppsala University, Sweden, Report IT 11 021, Faculty of Science and Technology, 2011. Available at `http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-155783`.

U. Junker. Air traffic flow management with heuristic repair. *Knowledge Engineering Review*, (this issue), 2012. Special Issue on Constraint Programming for Air Traffic Management.

R. Kameugne, L. P. Fotso, J. Scott, and Y. Ngo-Kateu. A quadratic edge-finding filtering algorithm for cumulative resource constraints. In J. Lee, editor, *Proceedings of CP'11, the 17th International Conference on Principles and Practice of Constraint Programming*, volume 6876 of *LNCS*, pages 478–492. Springer-Verlag, 2011.

P. Kerlirzin, S. Manchon, C. Plusquellec, and J.-B. Gotteland. Building and evaluating a minimal regulation scheme. In S. Saunders-Hodge and V. Duong, editors, *Proceedings of ATM'00, the 3rd USA/Europe R&D Seminar on Air Traffic Management*, Napoli, Italy, June 2000. Available at `http://www.atmseminar.org/seminarContent/seminar3/papers/p_035_AMSMP.pdf`.

S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

P. Leal de Matos and R. Ormerod. The application of operational research to European air traffic flow management – Understanding the context. *European Journal of Operational Research*, 123 (1):125–144, 2000.

P. Leal de Matos and P. Powell. Decision support for flight re-routing in Europe. *Decision Support Systems*, 34(4):397–412, 2003.

B. Musialek, C. F. Munafo, H. Ryan, and M. Paglione. Literature survey of trajectory predictor technology. Technical Report DOT/FAA-AJP-661, Federal Aviation Administration, November 2010. Available at `http://acy.tc.faa.gov/cpat/docs/SepMgmtResearchSurveyTechNoteFinalDeliveryNov2010.pdf`.

G. Pesant. A regular language membership constraint for finite sequences of variables. In M. Wallace, editor, *Proceedings of CP'04, the 10th International Conference on Principles and Practice of Constraint Programming*, volume 3258 of *LNCS*, pages 482–495. Springer-Verlag, 2004.

QinetiQ. Complexity algorithm development: The algorithm. Technical report, EUROCONTROL EATMP Infocentre (Brussels, Belgium) and QinetiQ, April 2004.

J.-C. Régin. A filtering algorithm for constraints of difference in CSPs. In B. Hayes-Roth and R. E. Korf, editors, *Proceedings of AAAI'94, the 12th (US) National Conference on Artificial Intelligence*, pages 362–367. AAAI Press, 1994.

J.-C. Régin. Generalized arc-consistency for global cardinality constraint. In D. Weld and B. Clancey, editors, *Proceedings of AAAI'96, the 13th (US) National Conference on Artificial Intelligence*, pages 209–215. AAAI Press, 1996.

F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

T. Schiex and G. Verfaillie. Nogood recording for static and dynamic constraint satisfaction problems. *International Journal of Artificial Intelligence Tools*, 3:48–55, 1993.

SESAR Consortium. SESAR concept of operations. Technical Report DLT-0612-222-01-00, SESAR Consortium, July 2007. Available at `http://www.eurocontrol.int/sites/default/files/content/documents/sesar/20070717-sesar-conops.pdf`.

STATFOR. EUROCONTROL long-term forecast: IFR flight movements 2010–2030. Technical Report 10/11/22-134, EUROCONTROL – Air Traffic Statistics and Forecasts, Brussels, Belgium, December 2010. Available at `https://www.eurocontrol.int/sites/default/files/content/documents/official-documents/forecasts/long-term-forecast-2010-2030.pdf`.

K. Sundequist Blomdahl, P. Flener, and J. Pearson. Contingency plans for air traffic management. In D. Cohen, editor, *Proceedings of CP'10, the 16th International Conference on Principles and Practice of Constraint Programming*, volume 6308 of *LNCS*, pages 643–657. Springer-Verlag, 2010a.

K. Sundequist Blomdahl, P. Flener, and J. Pearson. Contingency plans for air traffic flow and capacity management. In D. Schaefer, editor, *Proceedings of INO'10, the 9th EUROCONTROL Innovative Research Workshop & Exhibition*, 2010b. Available at `http://inoworkshop.eurocontrol.fr/2010/download.php?pid=147_21277`.

K. Sundequist Blomdahl, P. Flener, and J. Pearson. Contingency plans for air traffic flow and capacity management using constraint programming. *Journal of Aerospace Operations*, 1, March 2012. Forthcoming.

H. Tran Dac, P. Baptiste, and V. Duong. Airspace sectorization with constraints. *RAIRO Operations Research*, 39(2):105–122, April 2005.

P. Van Hentenryck and R. Bent. *Online stochastic combinatorial optimization*. MIT Press, 2006.

P. Van Hentenryck and J.-P. Carillon. Generality versus specificity: An experience with AI and OR techniques. In T. Mitchell and R. Smith, editors, *Proceedings of AAAI'88, the 7th (US) National Conference on Artificial Intelligence*, pages 660–664. AAAI Press, 1988.

P. Van Hentenryck and L. Michel. *Constraint-Based Local Search*. The MIT Press, 2005.

P. van Leeuwen and N. van Hanxleden Houwert. Scheduling aircraft using constraint relaxation. In *Proceedings of the 22nd Workshop of the UK Planning and Scheduling Special Interest Group*, 2003. Available at `http://planning.cis.strath.ac.uk/plansig/pastsigs/glasgow-22/PimVanLeeuwen.pdf.gz`.

A. Vela, S. Solak, E. Féron, K. Feigh, W. Singhose, and J.-P. Clarke. A fuel optimal and reduced controller workload optimization model for conflict resolution. In *Proceedings of DASC'09, the 28th Digital Avionics Systems Conference*. IEEE, 2009.

J. Villiers. *Automatisation du contrôle de la circulation aérienne : ERASMUS – Une voie conviviale pour franchir le mur de la capacité*, volume 58 of *ITA études & documents*. Institut du Transport Aérien, June 2004.

M. Wallace. Hybrid algorithms in constraint programming. In F. Azevedo, P. Barahona, F. Fages, and F. Rossi, editors, *Selected and Invited Papers of CSCLP'06*, volume 4651 of *LNAI*, pages 1–32. Springer-Verlag, 2007a.

M. Wallace. Constraint programming – The paradigm to watch. *Constraint Programming Letters*, 1:7–13, 2007b. Special Issue on the Next 10 Years of Constraint Programming. Available at `http://www.constraint-programming.org/letters/Papers/v1/wallace.pdf`.