Crew Assignment by Constraint Logic Programming

George Christodoulou and Panagiotis Stamatopoulos

Department of Informatics and Telecommunications
University of Athens
Panepistimiopolis, 157 84 Athens, Greece
{gchristo,takis}@di.uoa.gr

Abstract. In this paper, we deal with the crew assignment problem, which is a subproblem of the airline crew scheduling problem. The aim of the crew assignment problem is the optimal allocation of a given set of crew pairings to crew members, in a way that a set of constraints is satisfied. The optimality criterion we employ in this work requires the flight time fair distribution among all crew members. This problem has been traditionally tackled with Operations Research techniques. In recent years, the Constraint Logic Programming paradigm has been successfully used for solving hard combinatorial optimization problems. We propose a formulation of the crew assignment problem as a constraint satisfaction problem and we use a branch-and-bound technique combined with some heuristics in order to find quickly a solution identical, or at least very close, to the optimal.

1 Introduction

There is a very broad class of problems which fall under the general areas of *planning*, *scheduling* and *resource allocation* and which are difficult to model and even more difficult to solve. The solution of such a problem consists of an appropriate assignment of values to the variables that model the problem's domain in such a way that various constraints are respected. These problems are often referred to as *combinatorial search problems*, in the sense that what we have to search for is a feasible combination of values for the incorporated variables.

In a combinatorial search problem, someone might look for one, some or all feasible solutions. Depending on the solution density of the search space, finding one or a few solutions might equally be a quite easy or an extremely difficult task. On the other hand, finding all feasible solutions might be out of the question, or even out of usefulness, in case there is a huge number of them. However, what is actually required in most cases is to find an optimal solution according to a given objective function. Then, we are talking about optimization problems, which is the kind of problems that the Operations Research (OR) community is attacking for many years now.

Combinatorial search problems have attracted the attention of Artificial Intelligence (AI) researchers as well, who have developed a variety of methods and heuristics to deal with them. However, a major contribution of the AI community to the area is the idea of an active exploitation of constraints, in the sense that they may be used to prune inconsistent values of the involved variables, before getting to the point of choosing values for these variables. The effect of this pruning may be propagated then, through another constraint, to the possible values of other variables, leading in this way to a data-driven form of ensuring consistency. The overall result may be a significant reduction of the search space, depending, of course, on the nature of the involved constraints. This method is supported by the *constraint programming* paradigm [13, 19], that has emerged and heavily been exploited during the last decade, in order to deal with real world combinatorial search problems. Initially, the constraint programming idea arose as an extension of logic programming and the Prolog language, giving birth to constraint logic programming [11, 18]. However, nowadays, the constraint programming philosophy has been transferred to other programming paradigms as well, such as object-oriented programming, etc.

Scheduling flying crews of airline companies is a combinatorial problem, which is extremely hard, given the complexity of the constraints that have to be satisfied and the huge search space that has to be explored [6]. The problem is often tackled by breaking it down into the *crew pairing* and the *crew assignment* subproblems, which are still hard problems. The crew pairing subproblem has been studied extensively and tackled with OR techniques [10, 1, 20], genetic algorithms [14], neural networks [4, 12], constraint programming [15] etc. Much work has been done also for the crew assignment problem, where pure OR methods have been applied [2, 16, 5] or hybrid methods that combine OR and constraint programming [8, 3, 17, 21, 7].

In this paper, we discuss the crew assignment problem and we propose a formulation of it as a constraint satisfaction problem that may be solved by a specific constraint logic programming system, the language Prolog IV, developed by the French company PROLOGIA.

2 The Airline Crew Assignment Problem

The crew assignment problem for airlines refers to the allocation of cockpit and cabin crew members to pairings during a predefined rostering period, usually one month. A *pairing* is a sequence of flight legs; it starts from the home base and ends at the home base and it is constructed in such a way that labour regulations are respected. A pairing may span from one to few days long. The set of flight legs in a day constitute a duty. A crew assignment system is responsible for allocating crew members to preconstructed pairings that cover all flight legs of an airline company for a given rostering period. A system of this kind has to guarantee that no regulation is violated (actually, among the ones that cannot be checked at the pairing construction phase).

The full crew assignment process is usually performed separately for the cabin and cockpit crew, since their duty is governed by different constraints and regulations. Cockpit crew assignment can sometimes be broken into smaller independent subproblems corresponding to different fleets and groups of crew members of the same rank (e.g. captains, first officers and flight engineers). However, this is not always possible, if there exist constraints (e.g. crew composition ones) that relate different ranks to each other, or even if vertical constraints are to be satisfied. A vertical constraint is one that relates roster attributes of different crew members.

Apart from ensuring the validity of all rules and regulations, a crew assignment system must follow a specific assignment methodology as well. Three main methodologies exist:

Fair Assignment: The workload is allocated to crew members in a fair way. Flight time, days off, stand-by duties, early/late flights and any other work affecting attributes are being distributed evenly.

Bid Lines: Anonymous schedules for the whole rostering period (lines of work) are constructed and published, so that the crew members bid on them and the system assigns them according to the bids (usually, respecting the seniority criterion).

Preferential Bidding: The crew members express general and specific preferences (e.g. avoiding early flights, wishing to fly the OA202 flight next Tuesday, etc.) and the system tries to award such kind of bids, either by following a direct assignment methodology keeping in mind the expressed preferences or by generating personalized lines of work and, then, attempting to find a subset of them that covers all pairings and satisfies the crews as much as possible.

In the context of this paper, we are dealing with the crew assignment problem of a specific airline, namely Olympic Airways. In the next paragraph, a very short presentation of the rules and regulations at Olympic Airways is given.

2.1 Rules and Regulations of Olympic Airways

Crew scheduling in Olympic Airways is governed by a set of rules and regulations that have to be obeyed in order for a flight schedule to be legal. A complete reference of these regulations falls outside the scope of this paper, so a subset of them has been selected, in order to demonstrate the modelling of constraints. Some required definitions are the following:

Duty time: Any continuous period during which a crew member is required to carry out daily tasks at the company's behest.

Flight time of a duty: The period of the duty time that the crew member is on air.

Days off: Periods available for leisure and relaxation, no part of which shall form part of duty time. A time interval contains N days off if it is longer than $N * 24 + 16$ hours and contains N calendar days.

A subset of the rules taken into account by Olympic Airways is the following:

1. At most one duty intersects with any calendar day.
2. In each gliding window of N consecutive days, the total duty time has to be less than H hours, in the following cases:
   - $N = 7$ and $H = 40$
   - $N = 30$ and $H = 160$
3. *(for cockpit only)* In each gliding window of N consecutive days, the total flight time has to be less than H hours, in the following cases:
   - $N = 7$ and $H = 32$
   - $N = 30$ and $H = 80$
4. In each gliding window of N consecutive days, D days off are required, in the following cases:
   - $N = 7$ and $D = 2$
   - $N = 30$ and $D = 9$

As far as the assignment methodology is concerned, Olympic Airways follows the fair assignment option, having the total flight time of a crew member as a measure of the equal workload allocation.

3 Constraint Logic Programming

Constraint Logic Programming (CLP) refers to a class of programming languages that support a hybrid scheme combining the features of traditional logic programming and the efficiency of constraint solving. CLP profits from all advantages of logic programming, such as declarativeness and non-determinism, while overcoming limitations due to the inefficiency in exploring the search space of combinatorial problems.

Constraint logic programming is based on the idea that a myriad of real-world combinatorial search problems from many different contexts can be modelled as *Constraint Satisfaction Problems* or CSP's. In a CSP, there is a set V of variables, each of which is associated with a domain, the set of values the variable can possibly assume.

A constraint c_j applies to a subset V_{c_j} of the variables in V. If the size of V_{c_j} is n and each variable has a domain of size m, then the set $S_{V_{c_j}}$ of different possible assignments of values to the variables in V_{c_j} contains m^n elements. A constraint divides this set of possible assignments into *consistent* and *inconsistent* ones. Inconsistent assignments do not respect the constraint and are not acceptable.

In a CSP, there is a set C of constraints, each of which applies to a possibly different subset of the variables in V. A solution S is every assignment of values to variables which respects all constraints. In other words, in order for an assignment S to be a solution, for every constraint c_j in C, the assignments in S to the variables in V_{c_j} should be consistent.

Given a CSP, the goal could be to find one solution, all solutions or even an optimal solution according to a given objective function. *Constraint propagation*

is the mechanism which controls the interaction of the constraints. Each constraint can deduce necessary conditions on the domains of its variables. Whenever a variable's domain is altered, the constraint propagation will trigger all relevant to this variable constraints, in order to detect further consequences.

The structure of a CLP program is the following:

```
solve(List):-
    domain_initialization(List),
    constrain(List),
    enumerating(List).
```

The argument `List` is a list of domain variables representing the problem solution. In the `domain_initialization` step, each variable of the list is restricted to an initial domain. In the `constrain` step, constraints dealing with the problem are imposed upon the list's domain variables. In the `enumerating` step, each domain variable gets a value in a random or systematic way. Each time a value is assigned to a variable, the propagation mechanism is triggered and the constraint solver prunes variable domains, in order to satisfy the set of constraints. At the end of the enumerating step, either each variable is restricted to a single value (feasible solution) or failure is returned.

Prolog IV, the successor of Prolog III, is a compiled constraint logic programming language.[1] It allows the programmer to process a wide variety of constraints, describing relations over real and rational numbers, integers, booleans and lists in a sound and unified framework. The Prolog IV constraint solving techniques are based on exact and approximation methods.

4 Modelling the Problem

In this section, we intend to present our modelling of the crew assignment problem. We follow the general idea for modelling problems using CLP. Firstly, we define variables and the corresponding domains. Then, we introduce some constraints, in order to restrict the problem's search space. Finally, we discuss optimization issues.

4.1 Declaration of Domain Variables

As we have already mentioned, the crew assignment problem takes as its input data a set of pairings. In the first step, we transform the input data into a set of Prolog IV facts, the pairing facts. Each of these facts has the form `pairing(Id, Sd, Ed, Dt, Ft, S_day, E_day)`, where

- `Id` refers to the pairing's identification
- `Sd`, `Ed` refer to the pairing's departure and arrival dates
- `Dt`, `Ft` refer to the duty time and flight time of the pairing

[1] `http://prologianet.univ-mrs.fr/Us/prolog4.html`

– `S_day`, `E_day` refer to the departure and arrival calendar days of the pairing

Let M be the number of discrete pairings and N be the number of crew members. We intend to assign each pairing to a crew member, so we create a list X_List of size M.

$$X\_List = [X_1, X_2, \ldots, X_M] \text{ with } X_i \in [1, N],\ X_i \in \mathbb{N}$$

If X_i is equal to j, then pairing i is assigned to crew member j. The enumeration of each domain variable which belongs in the X_List corresponds to a solution of the problem. Another very useful list, which interacts with the X_List, is the C_List.

$$C\_List = [C_1, C_2, \ldots, C_N],\ C_i = [C_{i1}, C_{i2}, \ldots, C_{iM}],\ C_{ij} \in \{0,1\}$$

If $C_{ij} = 1$ then pairing j is assigned to crew member i. These two lists comprise the core of the program. The existence of both may look redundant, but it contributes to the flexible handling of the constraints.

4.2 Constraints Definitions

In this section, we discuss some constraints of the problem. The constraints declaration is an extremely important point that affects both memory requirements and execution time. The nature of the problem constraints is dual. Each constraint which could be declared as pairing-oriented could equivalently be declared as crew-oriented, but with different effects on program's efficiency. So, in this way, it is possible to select either X_List or C_List for modelling a constraint, depending on the achieved efficiency from each option.

Some of the constraints that apply to the crew assignment problem we deal with follow:

1. We need to bind, in some way, the domain variables of X_List with these of C_List, so as possible reductions of the domain (probably due to propagation) of the first will affect the domain of the second and vice versa. This programming trick provides us the dual flexibility for constraints handling. The constraints which have to be stated are the following:

$$\begin{aligned} X_i = j &\Rightarrow C_{ji} = 1 \wedge C_{ki} = 0 \qquad \forall k : k \neq j \\ X_i \neq j &\Rightarrow C_{ji} = 0 \\ C_{ij} = 1 &\Rightarrow C_{ik} = 0 \wedge X_j = i \qquad \forall k : k \neq j \\ C_{ij} = 0 &\Rightarrow X_j \neq i \end{aligned}$$

   This set of constraints can be elegantly imposed using Prolog IV boolean relations.

$$C_{ij} = 1 \ \texttt{iff}\ X_j = i$$

2. Pairings which overlap in time should not be assigned to the same crew member. This constraint is set upon the variables of X_List. Firstly, we locate all pairs (P_i, P_j) which are overlapped in time. So:

$$\forall i, j \quad i \neq j \text{ and overlapped}(P_i, P_j) \Rightarrow X_i \neq X_j, \quad i, j = 1, 2, \ldots M$$

This constraint ensures that X_i and X_j will not take the same value, so P_i and P_j will not be assigned to the same crew member.

3. Another example is part of the day-off rule mentioned in a previous section. For each crew member C_i, we define a list D_i. Each element of this list D_{ij} for $j = 1, 2, \ldots, 30$ corresponds to a calendar day and its domain is {0,1}. $D_{ij} = 1$ if crew member i has a pairing assignment on day j, otherwise $D_{ij} = 0$. A constraint that has to be satisfied is:

$$\sum_{i=k}^{k+6} D_{ji} \leq 5, \qquad k = 1, \ldots, 24, \quad \forall j = 1, \ldots, N$$

4. Another set of constraints refer to rules that apply to gliding time windows over the whole rostering period. These constraints are activated each time a pairing assignment takes place. Let pairing j be assigned to crew member i. Let also S_{7+} and S_{7-} the subsets of pairings which overlap with the time intervals of 7 days before the start time and 7 days after the end time of the just assigned pairing.

$$\sum_{j:P_j \in S_{7+}} C_{ij} * Dt_j \leq 40h$$

$$\sum_{j:P_j \in S_{7-}} C_{ij} * Dt_j \leq 40h$$

An important note on this set of constraints is that a time interval of N days does not correspond to an interval of N calendar days, which affects the level of easiness of its implementation.

4.3 Optimization

As we have already mentioned, the objective of the problem is not only to find a feasible solution, but the optimal one. The optimality criterion is the flight time fairness among crew members.

Objective Function. The objective function should measure the flight time fairness criterion. A possible objective function could be the following:

$$Z = \sum_{i=1}^{N} |F_i - F_{av}|$$

F_i is the flight time of crew member i. This function does not represent sufficiently the optimality criterion, because it does not "punish" large divergences from the average flight time F_{av}. The objective function that we use for minimization is the following:

$$Z = \sum_{i=1}^{N} (F_i - F_{av})^2$$

Enumeration. A set of constraints typically reduces a variable's domain, but sometimes unifies it with a single value. Enumeration takes place in the labeling phase of a constraint logic program. A general scheme of a predicate which implements enumeration of a list `L` of finite domain variables is the following:

```
my_enum(L):-
     stop_condition(L),
     !.
my_enum(L):-
     variable_selection(L, X),
     value_selection(X, M),
     my_enum(L).
```

The predicate `stop_condition/1` succeeds if every domain variable is unified to a single value. The predicate `variable_selection(L, X)` selects the next variable `X` of `L` that is going to be assigned a value from its domain. The variable selection phase is a key point of the search. Different selection strategies affect the efficiency of the assignment and objective function's value. The variable selection strategy that we used is the following: *Select the variable that corresponds to the pairing with the largest flight time.* The predicate `value_selection(X, M)` assigns to domain variable `X` the value `M`. The selection of the value is important. The selection strategy that we used is the following: *Select the value that corresponds to the crew member with the smallest current flight time.* These greedy heuristics work good enough with this problem, as it is proved by extensive experimentation. The intuition behind them is that we want to get rid early of the large pairings that are difficult to manipulate, while small ones are more flexible. The whole enumeration procedure is packed inside an iterative branch-and-bound process, which whenever finds a solution with some cost, let C, iterates and starts searching from the beginning trying to find a solution with cost better than C.

5 Experimental Results

In this section, we present the results of our work. The implementation was based on the modelling of the previous section. The rules that were implemented are those presented in section 2.1. We ran our experiments on a dual Sun Ultra 450Mhz SPARC workstation with 2 GB main memory. The trial runs had as

input a real world dataset of 475 pairings and 33 crew members of Olympic Airways. The memory requirements of the program are 400 MB. Experiments were carried out with datasets of other sizes as well. Although the quality of solutions was not affected, it was proved that both the execution time and memory requirements had a quadratic relation to the size of the input data.

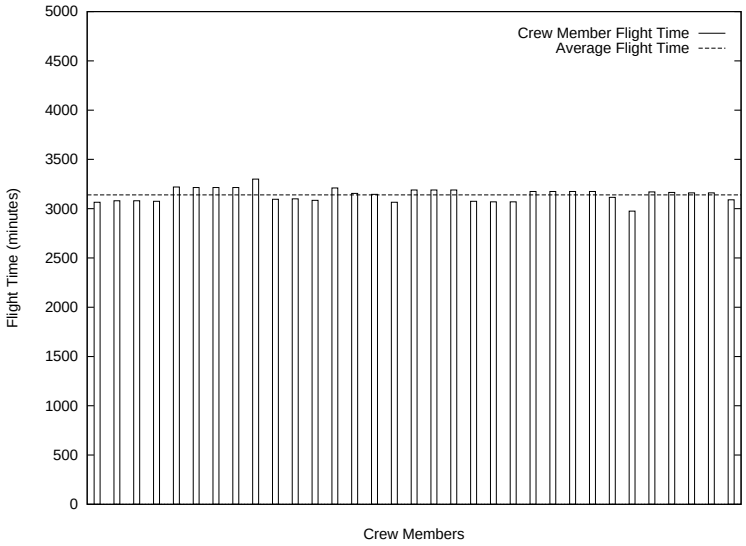

Fig. 1. A bar graph which depicts the assignment of the first solution

The bar chart in Fig. 1 depicts the first feasible solution found by branch-and-bound in 145 seconds of CPU time. Each bar depicts the flight time of a single crew member. The average flight time which would correspond to the ideal assignment, without taking into consideration the set of constraints, is 3140 minutes. As a matter of fact, an assignment like this is rather improbable to exist (the average is a rational number) even if there were no constraints at all. The average of the absolute deviations of data points from their average value that corresponds in our first solution is 54. Similar experiments for other datasets give evidence that it is preferable to accept the first solution, than a further one, with respect to the execution time. This is because the level of quality improvement of a solution decreases considerably in time. So, what such an application provides? The idea is to find quickly a slightly worse solution rather than find a better one by waiting for a large amount of time.

6 Conclusions

In this paper, we discussed the crew assignment problem, a subproblem of the crew scheduling problem faced by airline companies. We proposed a formulation

of it as a constraint satisfaction problem and we discussed the way we tackled it in a constraint logic programming environment, namely the language Prolog IV. The results we obtained on real world datasets were very satisfactory both in quality of the solution and the execution time. However, there is room for improvements, since the approach consumes big amounts of memory, leading to the requirement of very strong machines in case considerably larger datasets are given to the system.

References

1. Andersson E., Housos E., Kohl N., Wedelin D. Crew Pairing Optimization. in Yu G. (ed.) *Operations Research in the Airline Industry*. Kluwer Academic Publishing, 1997.
2. Bianco L., Bielli M., Mingozzi A., Ricciardelli S., Spadoni M. A Heuristic Procedure for the Crew Rostering Problem. *European Journal of Operational Research*, 58:272–283, 1992.
3. Caprara A., Focacci F., Lamma E., Mello P., Milano M., Toth P., Vigo D. Integrating Constraint Logic Programming and Operations Research Techniques for the Crew Rostering Problem. *Software Practice and Experience*, 28:49–76, 1998.
4. Croall I. F., Mason J. P. (editors). *Industrial Applications of Neural Networks — Project ANNIE Handbook*. Springer-Verlag, 1992.
5. Dawid H., König J., Strauss C. An Enhanced Rostering Model for Airline Crews. *Computers & Operations Research*, 28:671–688, 2001.
6. Etschmaier M. M., Mathaisel D. F. Airline Scheduling: An Overview. *Transportation Science*, 19:127–138, 1985.
7. Fahle T., Junker U., Karisch S., Kohl N., Sellmann M., Vaaben B. Constraint Programming Based Column Generation for Crew Assignment. *Journal of Heuristics*, 8:59–81, 2002.
8. Guerinik N., van Caneghem M. Solving Crew Scheduling Problems by Constraint Programming. In *Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming (CP '95)*, pages 481–498, 1995.
9. Halatsis C., Stamatopoulos P., Karali I., Bitsikas T., Fessakis G., Schizas A., Sfakianakis S., Fouskakis C., Koukoumpetsos T., Papageorgiou D. Crew Scheduling Based on Constraint Programming: The PARACHUTE Experience. In *Proceedings of the 3rd Hellenic-European Conference on Mathematics and Informatics HERMIS '96*, pages 424–431, 1996.
10. Hoffman K. L., Padberg M. Solving Airline Crew Scheduling Problems by Branch and Cut. *Management Science*, 39:657–682, 1993.
11. Jaffar J., Lassez J.-L. Constraint Logic Programming. In *Proceedings of the 14th Annual ACM Symposium on Principles of Programming Languages (POPL '87)*, pages 111–119, 1987.
12. Lagerholm M., Peterson C., Söderberg B. Airline Crew Scheduling Using Potts Mean Field Techniques. *European Journal of Operational Research*, 120:81–96, 2000.
13. Marriott K., Stuckey P. J. *Programming with Constraints: An Introduction*. The MIT Press, 1998.
14. Ozdemir H. T., Mohan C. Flight Graph Based Genetic Algorithm for Crew Scheduling in Airlines. *Information Sciences*, 133:165–173, 2001.

15. Pavlopoulou C., Gionis A., Stamatopoulos P., Halatsis C. Crew Pairing Optimization Based on CLP. In *Proceedings of the 2nd International Conference on the Practical Applications of Constraint Technology PACT '96*, pages 191–210, 1996.
16. Ryan D. M. The Solution of Massive Generalized Set Partitioning Problems in Aircrew Rostering. *Journal of the Operational Research Society*, 43:459–467, 1992.
17. Sellmann M., Zervoudakis K., Stamatopoulos P., Fahle T. Integrating Direct CP Search and CP-based Column Generation for the Airline Crew Assignment Problem. In *Proceedings of the 2nd International Workshop on Integration of AI and OR Techniques in Combinatorial Optimization Problems (CP-AI-OR '00)*, pages 163–170, 2000.
18. van Hentenryck P. *Constraint Satisfaction in Logic Programming.* The MIT Press, 1989.
19. van Hentenryck P., Saraswat V., et al. Strategic Directions in Constraint Programming. *ACM Computing Surveys*, 28:701–726, 1996.
20. Yan S., Chang J.-C. Airline Cockpit Crew Scheduling. *European Journal of Operational Research*, 136:501–511, 2002.
21. Yunes T., Moura A., de Souza C. C. Solving Very Large Crew Scheduling Problems to Optimality. In *Proceedings of the ACM Symposium on Applied Computing (SAC '00)*, pages 446–451, 2000.