A DECISION SUPPORT SYSTEM FOR CREW PLANNING IN PASSENGER TRANSPORTATION USING A FLEXIBLE BRANCH-AND-PRICE ALGORITHM

RICHARD FRELING, RAMON M. LENTINK AND

ALBERT P.M. WAGELMANS

ERIM REPORT SERIES *RESEARCH IN MANAGEMENT*	
ERIM Report Series reference number	ERS-2001-57-LIS
Publication	October 2001
Number of pages	23
Email address corresponding author	rlentink@fbk.eur.nl
Address	Erasmus Research Institute of Management (ERIM) Rotterdam School of Management / Faculteit Bedrijfskunde Erasmus Universiteit Rotterdam P.O. Box 1738 3000 DR Rotterdam, The Netherlands Phone: +31 10 408 1182 Fax: +31 10 408 9640 Email: info@erim.eur.nl Internet: www.erim.eur.nl

Bibliographic data and classifications of all the ERIM reports are also available on the ERIM website:
www.erim.eur.nl

ERASMUS **R**ESEARCH **I**NSTITUTE OF **M**ANAGEMENT

REPORT SERIES
RESEARCH IN MANAGEMENT

BIBLIOGRAPHIC DATA AND CLASSIFICATIONS		
Abstract	This paper discusses a decision support system for airline and railway crew planning. The system is a state-of-the-art branch-and-price solver that is used for crew scheduling and crew rostering. We briefly discuss the mathematical background of the solver, of which most part is covered in the Operations Research literature. Crew scheduling is crew planning for one or a few days that results in crew duties or pairings, and crew rostering is crew planning for at least one week for individual crew members. Technical issues about the system and its implementation are covered in more detail, as well as several applications. In particular, we focus on a specific aircrew rostering application. The computational results contain an interesting comparison of results obtained with, on one hand, the approach in which crew scheduling is carried out before crew rostering, and, on the other hand, an approach in which these two planning problems are solved in an integrated manner.	
Library of Congress Classification (LCC)	5001-6182	Business
	5201-5982	Business Science
	HD 30.23	Decision Support systems
Journal of Economic Literature (JEL)	M	Business Administration and Business Economics
	M 11	Production Management
	R 4	Transportation Systems
	C 44	Statistical Decision Theory, Operations Research
European Business Schools Library Group (EBSLG)	85 A	Business General
	260 K	Logistics
	240 B	Information Systems Management
	255 G	Management Science, Operations Research
Gemeenschappelijke Onderwerpsontsluiting (GOO)		
Classification GOO	85.00	Bedrijfskunde, Organisatiekunde: algemeen
	85.34	Logistiek management
	85.20	Bestuurlijke informatie, informatieverzorging
	85.03	Methoden en technieken, operations research
Keywords GOO	Bedrijfskunde / Bedrijfseconomie	
	Bedrijfsprocessen, logistiek, management informatiesystemen	
	DSS, Scheduling, Roostermodellen, Luchtvaart, Treinverkeer	
Free keywords	decision support systems, crew planning, branch and bound	

A Decision Support System for Crew Planning in Passenger Transportation using a Flexible Branch-and-Price Algorithm

RICHARD FRELING[1, 2*], RAMON M. LENTINK[1, 2] AND ALBERT P.M. WAGELMANS[1]

[1] *Erasmus Center for Optimization in Public Transport (ECOPT), Erasmus University Rotterdam (The Netherlands)*

[2] *ORTEC Consultants b.v., Gouda (The Netherlands)*

E-mail: freling@few.eur.nl

***Abstract*:** This paper discusses a decision support system for airline and railway crew planning. The system is a state-of-the-art branch-and-price solver that is used for crew scheduling and crew rostering. We briefly discuss the mathematical background of the solver, of which most part is covered in the Operations Research literature. Crew scheduling is crew planning for one or a few days that results in crew duties or pairings, and crew rostering is crew planning for at least one week for individual crew members. Technical issues about the system and its implementation are covered in more detail, as well as several applications. In particular, we focus on a specific aircrew rostering application. The computational results contain an interesting comparison of results obtained with, on one hand, the approach in which crew scheduling is carried out before crew rostering, and, on the other hand, an approach in which these two planning problems are solved in an integrated manner.

Keywords: decision support systems, crew planning, branch and bound

1. Introduction

Crew planning for passenger transportation has received a lot of attention in the Operations Research literature. Yet, only very recently, cases are reported where companies in the bus, railway and airline industry are using advanced OR techniques for solving crew planning problems (almost) optimally. With the rapidly increasing computer power in the past decade, advanced OR techniques such as column generation are gradually becoming more and more applicable to real life crew planning problems (see e.g. Day and Ryan (1997), Andersson et al. (1998), Desrosiers et al. (2000), and Ryan (2000)). Crew planning occurs on several levels, depending on the length of the planning period and whether the planning is for strategic, tactical or operational purposes. The two most widely applied crew planning problems are the crew scheduling problem for grouping tasks into duties, and the crew rostering problem for assigning duties to weekly, monthly or seasonal rosters for individual crew members.

1.1 Definition

The crew scheduling problem (CSP) is formally defined as follows: given a set of tasks with fixed starting and ending times and locations, and given a set of *rules* and *criteria*, find the *minimum cost* set of *duties* such that each task is included in a duty and all rules are satisfied.

* Corresponding author

A task is the smallest amount of work that can be assigned to one crew member. Each duty consists of a sequence of tasks that satisfy certain rules like maximum work time, minimum rest time, etc. The cost function consists of a weighted sum of several criteria, such as the number of duties or the total work time. An example of a task is Amsterdam 9:10 – London 10:50, an example of a duty is Amsterdam 8:30 – Amsterdam 17:00. A duty may also cover more than one day, which is then often called a *pairing*.

The crew rostering problem (CRP) is formally defined as follows: given the same information as for the CSP, and given a set of crew members with certain *characteristics*, find the *minimum cost* set of *rosters* such that each task is included in a roster, all rules are satisfied, and the characteristics of each crew member are taken into account. Just like a duty for the CSP a roster is a sequence of tasks that satisfy certain rules. The difference is that individual crew characteristics are taken into account for the CRP. Examples of such characteristics are qualifications, pre-assigned tasks, individual requests, and the past rosters for the crew member. The crew's past is necessary for checking laws and optimizing criteria that extend beyond the planning period. Usually, the CSP is solved first and the resulting duties serve as tasks for the CRP. We have experimented with integrating these two problems, where rosters are constructed directly from the tasks that served as input for the CSP in the definitions above (see Section 5).

Crew scheduling and rostering problems are usually modeled with set partitioning type of formulations, and solved with column generation techniques. The focus of this paper is not on these basic models and techniques for crew scheduling and rostering. Many papers have already been published on column generation techniques for crew planning, see e.g. Desaulniers et al. (1997) and Vance et al. (1997) for recent papers on crew scheduling, and Gamache et al. (1999) for a recent paper on crew rostering. In this paper, the focus is on how to implement these models and techniques successfully in a decision support system, taking all kinds of practical issues into account. Although practical applications are considered in the theoretical papers mentioned above, most practical details (often also complicating ones) are left out, or at least it is not mentioned how these are incorporated in the computer system. Anyone who has implemented a branch-and-price algorithm will confirm that it is a tedious task to do so, taking a lot of effort, and trial-and-error. It requires a combination of experience with designing algorithms and implementing decision support systems.

1.2 Historical development

Since 1995 a decision support system for crew planning called CDR (Crew Duty Rostering) has been developed at ORTEC Consultants BV, the Netherlands. This company is specialized in the development, implementation and application of intelligent planning and decision

support systems, encompassing models and methods from Operations Research and Management Science. ORTEC produces systems in various areas, including aviation, vehicle routing, human resource management, production planning, railways, and asset liability management. The CDR system was originally designed to create and maintain rosters for airline cockpit and cabin crew, and later also for crew working on trains.

In 1996, we implemented a simple column generation heuristic, which performed well for a particular application. However, for other applications this heuristic algorithm did not perform satisfactorily at all. In 1998 a new abstract implementation of the automatic planning tool in the DSS was started. The purpose was to set up a tool that can easily be used for different clients and applications, and that contains state-of-the-art mathematical techniques. Many practical issues came up during the development of the system for different applications. In Section 4, we discuss several implementation issues.

1.3 Contribution

In our view, the main contribution of our paper on a practical level is twofold:

1. We show that most of the complicating details arising in practice can be incorporated in a branch-and-price algorithm because column generation is a very powerful and flexible technique. Sometimes this introduces a heuristic feature in the algorithm, but this is generally not a problem in a practical context.
2. We provide insight into implementation issues, and therefore make it easier for novices in this field to implement branch-and-price algorithms and for the more experienced to get ideas for improvements.

Another contribution on a more theoretical level is that we perform experiments with solving rostering problems directly from tasks for an airline application, without first solving scheduling problems, thus integrating crew scheduling and crew rostering. Caprara et al. (2000) deal with a similar integration for railway applications, but they use a different algorithmic approach.

1.4 Outline

In the next section, we discuss the mathematical background of the decision support system for crew planning. Several general algorithmic aspects will not be discussed here because they are already presented well in the aforementioned papers. Therefore, we briefly describe the underlying mathematical model and a branch-and-price algorithm. In Section 3, we discuss the functionality and purpose of the system, and the classification of constraints and objectives used in the system. In Section 4, we focus on the implementation issues. In

particular, we consider abstract implementation issues, the efficient use of computer memory and the reduction of run time. Four applications are discussed in Section 5, of which one airline crew rostering problem in more detail. Finally, we summarize our work in Section 6.

2. Mathematical Background

In this section, we briefly consider the underlying mathematical formulation and the solution methodology used in the system. Throughout this paper we assume that the reader is familiar with column generation and branch-and-price. See for example Barnhart et al. (1998) for a general discussion. We would like to stress again that the mathematical model and techniques to be presented next, can be found in several aforementioned papers and that they are not the focus of this paper.

2.1 Set partitioning formulation

The generalized set partitioning model (GSP) presented below can be used to formulate most crew planning problems arising in practice. The crew members are grouped into *crew groups*, where each group consists of all crew members with identical characteristics. Let the set R denote the set of all feasible duties or rosters, let K denote the set of all crew groups, and let I denote the set of tasks which need to be covered. Furthermore, let R_k denote the set of feasible duties or rosters for group $k \in K$. GSP contains binary decision variables x_r which equal 1 if duty or roster r is selected and 0 otherwise, and continuous decision variables s_i which equal the number of uncovered tasks. The cost of duty or roster $r \in R$ is denoted by c_r and the penalty for not covering task $i \in I$ is denoted by p_i. Finally, b_i is the number of duties or rosters which need to cover task $i \in I$, and d_k denotes the maximum number of duties or rosters allowed in group $k \in K$. The generalized set partitioning problem on which the automatic planning part of the DSS is based is mathematically formulated as follows:

(GSP) Minimize $\sum_{r \in R} c_r x_r + \sum_{i \in I} p_i s_i$

subject to

$$\sum_{k \in K} \sum_{r \in R_i} x_r + s_i = b_i \qquad \forall\, i \in I \qquad \text{(i)}$$

$$\sum_{r \in R_k} x_r \leq d_k \qquad \forall\, k \in K \qquad \text{(ii)}$$

$$x_r \in \{0,1\} \qquad \forall\, r \in R$$

$$s_i \geq 0 \qquad \forall\, i \in I$$

The objective usually comes down to primarily minimizing the number of uncovered tasks, and secondarily minimizing the total cost of the duties or rosters selected in the solution. Constraints (i) are generalized set partitioning constraints, which ensure that each task is covered by at most b_i duties or rosters. In case of planning crew members as a team, one duty is assigned to the entire team for the CRP. That is, for the right-hand-side of constraint (i) b_i is equal to the number of crewmembers in the team. Variables s_i are added to allow tasks to remain uncovered. Constraints (ii) guarantee that at most d_k duties or rosters can be assigned to each group. We assume that the intersection of the sets R_k for all $k \in K$ is empty, that is, each duty or roster $r \in R$ is uniquely defined for one group. Variations for both Constraints (i) and (ii) are possible by interchanging equality signs and inequality signs. Additional constraints may be added for modeling global constraints, which deal with sets of duties or rosters.

This model is a generalization of the model proposed in Ryan (1992) for the CRP (see also Gamache and Soumis (1998) and Gamache et al. (1999) for slight variations). The novelty in this formulation is the notion of groups, which generalizes the set partitioning formulations for the CSP and CRP. In case of the CSP, there is one group and constraint (ii) only exists in case a maximum number of duties are allowed in the solution. For the CRP with unique characteristics for each crew member, every crew member corresponds to one group. In practice, several crew members may have identical characteristics and can therefore be joined in a group.

2.2 Solution approach

A disadvantage of the model above is a large number of feasible duties or rosters, which corresponds to a large number of columns. Therefore, in order to use the model in the solution approach, we need a column generation procedure. The state-of-the-art techniques in the automatic planning tool are related to the general framework in the context of time constrained routing and scheduling problems proposed by Desrosiers et al. (1995) and Desaulniers et al. (1999). In particular, our solution approach consists of a branch-and-price algorithm for the GSP. Branch-and-price is a special application of branch-and-bound, where column generation is used to solve LP relaxations with a huge number of variables. Since the late eighties several papers deal with column generation approaches for crew planning in passenger transportation (recent papers are Desaulniers et al. (1997), Vance et al. (1997), and Gamache et al. (1999)). The main steps of the general solution approach in the system are:

Step 1. Read the input and choose the initial set of columns.

Step 2. Solve the LP relaxation with the current set of columns; this is usually called the *master problem*.

Step 3. Column management: delete columns from the current LP model, and *price*: generate new columns with negative reduced costs and add them to the current LP model.
If new columns are found return to Step 2.

Step 4. If stopping criteria are satisfied write the output and stop. Otherwise perform a new branch-and-bound step, that is:

4.1 *select* a node to *branch* on,

4.2 calculate the lower bounds of the two new nodes by solving the LP relaxations with the procedures in Steps 2 and 3,

4.3 perform fathoming rules; if an integer solution has been found that is better than the best integer solution so far, update the best integer solution,

4.4 return to Step 4.

The crucial aspects of this approach are *price*, *select* and *branch*, that is, how columns are generated in Step 3, and how nodes are selected and which branching rule is used in Step 4. In our implementation, nodes can be selected by depth-first-search, best-first-search, or a combination of these two. The system contains several algorithms for generating columns based on one or more acyclic networks. For crew scheduling a single network is used, while for crew rostering a network is used for each crew member. The networks are defined such that each node corresponds to a task (or trip, flight, duty, etc.) and each arc corresponds to a feasible sequence of two tasks in one duty or roster. In addition a source and sink arc are added to each network, denoting the start and end of a duty or roster. A path in a network corresponds to a feasible duty or roster if the path constraints are satisfied. See also Desrochers et al. (1992) for an example of a similar network for crew scheduling. Paths through the networks are constructed by solving a resource constrained shortest path problem (see Desrosiers et al. (1995)). The algorithms that we have implemented are a dynamic programming algorithm, a depth-first search algorithm, and an all-pairs shortest path algorithm (see Freling (1997)). The choice of the algorithm depends on the application considered, and combinations of two algorithms may also be used.

We have also implemented several branching rules. The branching rule for the exact branch-and-price algorithm consists of branching on the arcs in the underlying networks (see e.g. Desrosiers et al. (1995)). A branch consists of either forbidding or forcing one arc to be in the solution. The consequence for the child nodes is that several variables are fixed to zero, and that the corresponding networks are adjusted by deleting arcs to either force or forbid an

arc to be in the solution. In case of forcing an arc to be in the solution, all other arcs leaving and entering the corresponding nodes are deleted.

This solution approach is very robust in the sense that both the constraints defining the feasibility of each duty or roster, and the constraints defining the feasibility of a set of columns can be easily incorporated without changing the basic structure of the algorithm (see Section 3.2). In addition, for large scale problems several exact and heuristic techniques can be used in a straightforward manner to speed-up the algorithm (see Section 4).

3. A Decision Support System for Crew Planning

In this section, we discuss the functionalities and the purpose of the DSS, the classification of the constraints and objectives in the system, and the crew classes.

3.1 Functionality and purpose of the DSS

The CDR system supports the entire planning process, which is divided in long term planning, short term planning (rostering and operations control), realisation and evaluation. This is depicted in **Error! Reference source not found.**:

[PLEASE INSERT FIGURE 1 AROUND HERE]

Starting Points are defined based on long term analyses and company rules. For example, a starting point can be the desired service level, the desired number of crew, training needs, hollidays, and other specific issues that influence the crew availability. The long term planning concerns a planning for an entire season, that is based on the defined starting point. The system helps the management to determine:

- the capacity needed to perform the tasks.
- the permanent and temporarily staffing levels needed to meet the required capacity.
- allowed vacations, standbys required etc. for the given period.

The long term planning delivers several norm figures for use in the planning department. When the norm figures are available, a planner can start with creating rosters. Of course these norm figures influence the planner's freedom of movement. An example of an application of the short term module is to determine daily duties first by solving the CSP seven times for each day of the week, and determining weekly rosters afterwards by solving the CRP. After the roster has been published, several changes can occur due to, for example, sickness of crew, schedule changes, delays etc. CDR supports the planner during the coordination stage and the operations control stage. Based on reports and a user friendly planning board, the planner will be able to handle the operational changes.

Once a specific duty has been performed, corresponding data are saved into the CDR database. For example, this concerns crew notification data such as sign-in/sign-out. To complete the planning circle, CDR supports the planner to help making an evaluation of the realization. The best rosters are those that take into account management as well as crew wishes. But these are often conflicting, so it is very difficult to create a roster manually that will keep both parties satisfied. The automatic planning tool facilitates the planner in creating a roster that satisfies both economic and social criteria (see Section 3.2). The system concept relies on the possibility of calculating several alternatives within a relatively short time (What-if analysis). What-if analysis can be used on every planning process level supported by CDR. For example, what-if analysis can be used to analyse the effects of certain policies, to support negotiations with respect to legal regulations, to make a capacity plan, or to select the best alternative out of several generated rosters.

3.2 Classification of constraint and objective types in the DSS

The duties and rosters that are generated by the automatic planning tool have to meet certain labor laws and agreements in order to be feasible. Assume that we have an underlying network representation as proposed in Section 2.2. Recall that network representation is such that each node corresponds to a task and each arc corresponds to a feasible sequence of two tasks in one duty or roster. A path in a network corresponds to a feasible duty or roster if the path constraints are satisfied. We suggest three levels of constraints:

1. *Coupling constraints*. This concerns rules that relate to the whole set of selected duties or rosters. Such a constraint could specify, for example, that at most 5% of all selected duties have durations longer than 9 hours. These constraints are added to the master problem.
2. *Path feasibility constraints*. Rules at this level determine the feasibility of a duty or roster (a path in a network representation). For example, the maximum length of a duty is 9½ hours. Usually, most constraints are at this level.
3. *Node/arc feasibility constraints*. A constraint at this level determines if a particular task (a node in a network representation) can be assigned to a certain crew member due to licenses, etc. or if two tasks (or duties) can be assigned consecutively to the same duty or roster (an arc in a network representation). For example, an arc between tasks i and j exists if task i starts after task j has finished and a short buffer time has passed.

A similar classification holds for other network representations. Sometimes the network representation needs to be modified in order to be able to check constraints efficiently. For example, several home bases may exist, and each duty or roster must start and

end at the same home base. In that case, it may be better to use one network for each home base. In case of one network for each crew member, this problem would be solved if each crew member has a unique home base. However, in one of the airline applications we worked on (see Section 5), some crew members could be assigned to two home bases because they lived somewhere in between.

For the classification of objectives we can distinguish three types of possibly conflicting objectives:

1. *Efficiency*: cost minimization with respect to the number of uncovered tasks and the number of required duties or crew; other cost factors may be the number of layovers and the total time between tasks.
2. *Welfare*: the workload of the rosters should be equally spread among the crew members, and requests should be granted equally as much as possible.
3. *Robustness*: duties or rosters must be robust with respect to, for example, delays.

A similar classification as for the constraints is also possible: some objectives have effect on a set of duties or rosters, some on paths, and some on nodes and/or arcs. Note that the constraints and objectives that need to be checked at each level depend on the application, the client, the crew type and the planning horizon. For example, for the CSP on a daily basis only efficiency and robustness are relevant. Or, for the short term CRP, crew welfare and robustness could be more important than efficiency if all crew members receive fixed salaries anyway.

3.3 Crew classes

Usually, in the airline and railway industry several crew classes like cockpit and cabin crews need to be planned. Cockpit and cabin crews usually consist of different crew functions, for instance pilots and co-pilots for a cockpit crew. Although crew can be planned for each function separately, it may be efficient to use crew teams of, for example, cockpit crew only or of both cockpit and cabin crew together. Often, different duty functions can be assigned to one crew function, for example a pilot can do both a pilot duty and a co-pilot duty. In that case, and if crews do not need to be planned as teams, a logical decomposition is to solve the co-pilot CRP first and all the duties that are uncovered are added to the pilot problem. Another example of different crew classes would be train drivers and guards, see Fischetti and Kroon (2000)). The context of crew planning for urban transit has been discussed in detail in Odoni and Rousseau (1994). Furthermore, Desaulniers et al. (1998) have provided an overview of different crew planning problems.

4. Implementation Issues

As mentioned in the introduction, implementing a branch-and-price algorithm is not an easy task, let alone implementing a flexible tool, suitable for a variety of applications. However, in the literature little or no attention has been paid to the implementation of such an algorithm. Therefore, we provide some insights into our implementation, and discuss some difficulties we ran into and the corresponding solutions we came up with.

4.1 Abstract implementation

The framework is implemented in the C++ programming language, making fully use of its object-oriented nature. Figure 2 shows an overview of the implementation.

[PLEASE INSERT FIGURE 2 AROUND HERE]

The optimization coordinator steers the various calls to the branch-and-price algorithm, like solving the CSP first and then the CRP, and solving the problem for different crew functions. There is an abstract interface with application specific information, that is, rules checking, input & output translator (via file interface) and parameter tuning. There is also an abstract interface with the linear programming solver (currently CPLEX 6.6, XPRESS 12.1, and a subgradient algorithm), and an abstract interface with the pricing algorithm.

An important part in such an abstract implementation is the rules checker, which also incorporates the functionality to calculate (elements of) the objective function. Recall that three levels of constraints and objectives can be defined. The link between the rules checker and the integer programming (IP) solver in the figure above, correspond to the checking of the coupling constraints that are added to the GSP. The path constraints are checked during the network algorithm, and the node/arc constraints are checked during the construction of the network(s). Also note that the constraints and objectives that need to be checked at each level, depend on the application and the planning horizon. For computational efficiency it is crucial that the path constraints can be checked as fast as possible. In our opinion, a column generation algorithm only works well if the path constraints are checked in sequence, that is, when building a path, each time a node is added to the path, it is not necessary to check the entire path so far. This is achieved by keeping constraint information of the path so far. An exception may be that one or two rules are checked at the end, when a complete path is found. Thus the feasibility of a path up to a certain node (no matter which algorithm is used), is checked by combining the constraint information of the path so far with the relevant information of adding the node. For this purpose, two vectors are used: a consumption vector with the *consumption* of each resource for a path up to a node, and an arc vector with the consumption of each resource for an arc (incorporating the addition of the node to which the

arc points). The network algorithm is not aware of details on these general vectors. These details are only known at the rules checker level, which keeps the implementation of the network algorithm general and abstract.

4.2 Efficient use of computer memory

The memory usage of a branch-and-price algorithm grows very fast with the size of the problem. To give an idea about the memory usage, we use a CRP example where a dynamic programming algorithm is used for the pricing problem. The larger part of memory is used by four categories:

1. Size of the network for each crew member (mainly determined by the size of the arc vector).
2. Size of the LP model.
3. Size of the branch-and-bound tree.
4. State space for the dynamic programming algorithm.

Advanced data structures are necessary to prevent the memory requirements from exploding to unmanageable size, while keeping run times as low as possible. For the first category we did not use an entire new network for each crew member. Instead, we have one network for all crew members where we avoid the duplication of node and arc information. For a certain crew member, the network differs by linking only those nodes and arcs in the network that are relevant for that particular crew member. For an application with 109 crew members, a memory reduction of a factor 6 is obtained (from 680 Mb to 112 Mb), compared to creating a complete new network for each crew member.

The size of the LP model is managed by keeping in memory only the size of the current model (if possible only in the internal data structures of the LP solver) at each iteration of the algorithm. We also have an option to delete columns from the model such that convergence of the column generation is still guaranteed (see Section 4.3). The branch-and-bound tree is implemented as a heap, where active nodes are kept. Choosing a depth-first-search strategy can also reduce memory usage, because the number of open problems remains relatively small. The state space for the dynamic programming algorithm consists of a linked list of paths at each node. The memory of these paths is never released during the algorithm to prevent wastage of computer time due to allocating and deleting memory.

4.3 Exact strategies for computer time reduction

In Freling (1997) and in Desaulniers et al. (1999) several acceleration strategies for column generation algorithms are suggested. In the latter paper, the strategies are categorized as pre-processor strategies, subproblem strategies, master problem strategies, branch-and-bound strategies, and post-optimizer strategies (i.e. reoptimization). We have not used pre-processor

or post-optimizer strategies, except for some network reduction techniques (see Freling et al. (2001)). Here, we provide a list of the strategies that we have implemented and tested. In particular, in this section we discuss exact strategies and in the subsequent section heuristic strategies. We have implemented all these strategies, except when mentioned otherwise.

Efficient column management

The more columns are added to the master problem in each iteration, the less column generation iterations are necessary but the more time it takes to solve each LP problem. Therefore, experimentation is necessary to determine the right column management strategy. A discussion about column management strategies or related topics can be found in Bixby et al. (1992), Kohl (1995) and Freling (1997). By experimenting with various applications, we found a strategy that works well in general. The idea is to generate a relatively high number of columns, add them to the LP problem, and delete a large percentage of those columns after solving the LP problem. In that way many columns are considered but the LP problem does not grow too fast. The number of columns that can be generated depends on the pricing algorithm used. For all the algorithms we used in our implementation it is possible to influence the number of columns generated. The strategy for deleting columns after solving the LP problem is as follows (see also Freling (1997)): let q be the positive reduced cost columns after solving the LP problem, that are selected from those columns that are added to the master problem just before solving the LP problem. Then, delete the $0.9q$ columns with the largest reduced cost.

Our computational testing confirmed that it is very beneficial to use partial pricing, that is, to not generate columns for each group in one iteration (see also Gamache et al. (1999)). In each iteration, we generate columns once either negative reduced columns have been found for p_1 groups, or the pricing problem has been solved for all groups. The value of p_1 is set to $\max(5,|K|/20)$ in our implementation (recall that K is the set of crew groups). If p_2 is the number of negative reduced cost columns generated, then $\min[r, p_2]$ columns are added to the master problem. The value for r may also be adjusted dynamically during the algorithm and needs to be tuned for each particular application.

Although we have not tested it, an idea to improve the partial pricing is to group crew members if their characteristic are similar. Then, columns are generated for these groups at once and then divided among the crew members.

Temporary relaxation of the pricing problem.

Recall that the purpose of the pricing problem is twofold: generating ‘good’ columns, and providing a criterion for convergence of the column generation algorithm. For the second purpose, the pricing algorithm needs to be exact, for the first purpose not. Therefore, it may

be efficient to solve the pricing problem heuristically until some point and then change to an exact algorithm (see also Sol (1994) and Gamache et al. (1999)). We have tested a strategy called *dynamic network size*. We start with a sparse subnetwork by leaving out arcs, and update the network size until the original network size is obtained. At each iteration of the column generation procedure, the arcs in the network are sorted by increasing reduced cost. Thus, when selecting the subnetwork at each iteration, we take the subset of arcs with the lowest reduced cost out of each node. The reduced cost of an arc is defined as the cost of an arc (which includes the cost of the node it points to) minus the dual value of the task corresponding to the node it points to. The maximum number of arcs to be selected in each iteration is increased during the column generation procedure.

Reduction of degeneracy.

A typical phenomena in column generation is that at the end of the procedure negative reduced cost columns are still found but the LP objective value is not or almost not improving. This is called *tailling-off*. Two ways to deal with it are lower bound approximation (see Freling (1997)) and stabilized column generation (see du Merle et al. (1999)). We have performed some preliminary tests with stabilized column generation, but did not get significant improvements. However, Desrosiers et al. (2001) mention a large reduction in computer time for a particular application.

4.4 Heuristic strategies for computing time reduction

The branch-and-price algorithm is very well suited to add heuristic features that speed up solution time considerably while the solution remains close to optimality. See for example Gamache et al. (1999). Here we discuss several of such features.

Fixed relaxation of the pricing problem

When using dynamic programming to solve the pricing problem, some constraints or criteria can cause the state space to grow very fast. Let s and t be the source and the sink of a network, respectively. The size of the state space depends on the number of paths from s to each node, that is, the number of *partial paths*. To reduce the state space, dominance criteria are used to remove partial paths that will not lead to the shortest s-t path. Let $P_i(s)$ and $Q_i(s)$ denote two partial paths from s to node i. Let $U_i(t)$ denote a partial path from node i to the sink t. Then $P_i(s)$ dominates $Q_i(s)$ if the following two conditions are satisfied:

1. The (reduced) cost of $P_i(s)$ is less or equal to the cost of $Q_i(s)$.
2. If $P_i(s) \cup U_i(t)$ is infeasible, then $Q_i(s) \cup U_i(t)$ is also infeasible for all $U_i(t)$.

The first condition is only valid when the cost function is (monotonically) increasing with the number of nodes in the path. The second condition is only valid when an ordering exists for all the constraints. For example, in case of a maximum work time, the ordering follows from the fact that less work time is always better. Thus, if maximum work time is the only constraint, $P_i(s)$ dominates $Q_i(s)$ if both the cost and the work time of $P_i(s)$ are less or equal to the cost and the work time of $Q_i(s)$. Now suppose we also have a minimum work time constraint. Then, the dominance rule above can only be applied when both partial paths satisfy this constraint, thus resulting in a much larger state space. Finally, suppose we have an objective that the work time should be as close as possible to a certain fixed value, that is, the cost function is not (monotonically) increasing. Then, dominance can only be applied once both paths considered have exceeded the fixed value.

In practice, we have encountered several of these types of constraints and objectives that cause the state space to grow very fast. But in none of those cases the user of the system had a problem when we modified the constraint or objective slightly in order to deal with them in a computationally more attractive way. For example, the minimum work time is relaxed, and only when the path is completed (that is, it reaches t) a penalty is added when the work time is below its minimum. In case of the objective that the work time should be as close as possible to a fixed value, costs are only involved once the work time exceeds the fixed value. Thus, dominance can be applied below that value as well. Experience shows that because the intention of the constraint is to balance work time among crew members, this can still be achieved approximately by setting the fixed value a little lower. Because a nonlinear penalty function is used (the higher the work time the faster the penalty increases), the work time is pushed down towards this value from above.

Another way to heuristically relax the pricing problem is to modify the exact strategy with an increasing density of the network to a constant sparse density of the network. That is, dynamically update the arcs considered using reduced costs, but never increase the size of the network.

Preliminary convergence of column generation and/or branch-and-price

Other heuristic strategies that we have implemented are:

1. Stop column generation once detecting tailling-off, that is, converge without a true lower bound.
2. Only perform a few iterations of column generation when solving the LP problem in other nodes than the root node.

3. Try to find a good feasible solution as quickly as possible using a depth-first-search strategy during branch-and-price. Stop once a feasible solution is obtained. A possible extension is to fix several variables to one at once in each node of the tree (see also Gamache et al. (1999)). In this case, we branch on individual variables instead of arcs.

5. Applications of the automatic crew planning module

The automatic planning module of the DSS has been implemented for several applications. In this section, we briefly consider a charter airline and two railway applications and we consider a case study for a European regional airline company in more detail in Section 5.2.

5.1 Example applications

A first version of the crew planning system with application to crew rostering has been in operation since 1996 at a Dutch charter airline with flights within Europe. At first a simple heuristic was implemented (see Nicoletti (1975) and Bodin et al. (1983)), but this turned out to be unsatisfactory because the planner could manually produce better solutions. The algorithm used successfully in the DSS was a simple column generation heuristic with a search procedure to generated a large number of columns in advance, and then use column generation to solve the linear programming (LP) formulation. The columns are generated by selecting them from the pre-generated set of columns, and no new columns are generated on the fly. After solving the LP relaxation, an integer solution was obtained by a simple branch-and-bound algorithm without further column generation. The branch-and-bound procedure was halted once an integer solution had been found. The input consisted of short pairings, that is a combination of a flight from the homebase to another city and back to the homebase. Rosters were built directly from these pairings, so no CSP problem was solved first to build duties. Despite the simplicity of this algorithm, computational results were satisfactory from a practical point of view. Both CPLEX and a subgradient algorithm were used to get lower bounds, but better results were obtained with CPLEX. In a computational study, two types of problems were tested, namely problems with up to 60 pairings and 30 crew members, and problems with up to 225 pairings and 60 crew members. The problems in the first category were all solved to optimality within a second of run time (on a Pentium I, 90 MHz), while the problems in the second category were solved with a gap of at most 10% within 15 seconds of run time. One rare exception that had been send to us by a user because he noted a large gap, was an instance with a gap of 27%.

In Freling et al. (2001), the authors consider a case study for the Dutch Railways in detail. We use a heuristic branch-and-price algorithm to solve CSP's with at most 1114 tasks.

This largest instance is solved in about 32 minutes (on a Pentium II/400Mhz/128Mb) with a gap of 2.32%. The pricing problem is solved exactly by dynamic programming. The heuristic feature is in the branching step by only performing a few iterations of column generation when solving the LP problem in other nodes than the root node, and to try to find a good feasible solution as quickly as possible using a depth-first-search strategy during branch-and-price. The algorithm stops once a feasible solution is found. The extension by fixing several variables to one at once in each node of the tree has also been tested, and resulted in a solution with 3.69% gap in about 11 minutes on the same instance.

One of the users of the DSS is the European company Railmasters that operates catering of the Thalys, a high speed train between major cities in west Europe. The DSS is used for planning catering personnel working on trains. Duties are constructed manually in the DSS, and the automatic planning module is used for a monthly CRP. Because we are dealing with catering personnel and run time is considered more important than quality of the solution, the branch-and-price algorithm is a bit heavy for this type of problem. Furthermore, the problem size is about 126 crew members and about 2200 duties, which is relatively large. An additional complication is the objective function. To minimize the number of uncovered duties is not anymore the primary objective. This is because balancing the workload is a company rule, and is considered more important. The uncovered tasks are assigned manually afterwards to external personnel. Even when sequentially solving the problem week by week and using several heuristic features the run time is still about 2 hours and 40 minutes (again on a Pentium II/400Mhz/128Mb), which is considered too long. Therefore, the weekly problem is split in four smaller problems by logically dividing the crews and the duties in two subproblems. The four crew classes are obtained as follows: one type for crews with a 38-hour per week contract, two types of crews with a 32-hour contract, and one type for crew with a 20 and 25-hour contract. Then, the total run time is about 80 minutes. This is acceptable, but we have also developed a simple algorithm where we sequentially solve an assignment problem for each pair of days. The quality of this solution is generally acceptable, and the run time is only about 10 to 15 minutes in total.

5.2 Application to crew rostering for a regional airline

The case study for a European regional airline company is interesting from both a practical and a computational point of view, because the laws, regulations and company rules are very complex. The aim of the study was to build automatically efficient weekly rosters for cockpit crew. Distances are relatively short since the company operates domestic flights and flights between the home country and neighbouring countries. Tasks that are input for the daily CSP consist mainly of couples of flights. After solving the daily CSP's, the input of the CRP is

about 500 duties and about 55 crew members. Because co-pilots and pilots are planned separately, this resulted in two problems of about half the size. Other characteristics are:

- Multiple aircraft types.
- A highly connected network of flights.
- Several crew specific characteristics, among which fixed tasks, crew availability, past workload, and licenses for aircraft types and for airports.
- Multiple home bases per crew member.
- A night stop (layover) off home base is allowed.
- No dead-headings on flights, only by car or bus rides between homebases.

Each crew member has a pre-defined *profile* that form the input of the CRP module. Such a profile consists mainly of fixed tasks and already granted requests. However, most requests are not granted a priori, but are assigned during the optimization process.

Constraints & Objectives

A complicating company constraint is that each crew member needs to end a duty at the same homebase where she/he started. In case of layovers, the crew member needs to end at the same homebase it started it's off homebase work period. This is incorporated in the system by keeping an extra variable in the consumption vector that denotes the start homebase.

There are too many laws and regulations to mention them all here. Some examples of complex constraints are:

- In every period of 7 days at least 36 hours rest period, or in every period of 10 days at least 48 hours rest period.
- Between each two daily duties at least 18 hours rest, and if violated, a 15-day period starts with variable minimum allowed cumulative rest. This period ends once the cumulative rest exceeds a norm value.
- The maximum allowed working time is dependent on the number of landings, on the start time of the daily duty, and, in case the minimum rest time is violated, also on the rest period and on the working time of the previous day.

The first constraint can be interpreted in several ways. Contacting the Labour Inspection Office that designed the rule did not clarify it because the person who designed the rule does not work there anymore and there is nobody else who can clarify it. So, we chose the interpretation that the company was used to, which is relatively easy to implement in a column generation context. We keep two variables (resource consumptions), namely the number of days since a 36 hour rest period and the number of days since a 48 hour rest

period. A path cannot be extended once the extension incurs that both rules will be violated, that is, in the past 7 days no 36 hour rest period occurred and in the past 10 days no 48 hour rest period occurred.

Because we are dealing with weekly rosters, minimizing costs is not a main objective. After the minimization of the number of unassigned flights, the most important objective is to balance the workload among crew members. This considers the hours working time, the number of dead-headings, layovers, ground time, heavy duty sequences, and weekends off. The balancing includes the workload of the past two weeks, that is, the workload is balanced in rolling periods of three weeks.

Algorithm

We used the same branch-and-price algorithm for the CSP and the CRP, that is, with the same parameter values. The CSP problems are relatively small and can be solved within a second. Although the size of the CRP problem is moderate (about 250 duties and 35 crew members), it is still a complex problem because of the highly complex objectives and constraints. We use dynamic programming for column generation. The only heuristic feature of the algorithm is that not all constraints are incorporated in the dominance testing, that is, optimal paths could be dominated. For the tests we performed the results are optimal or close to optimal, while the memory usage and computation time is reduced significantly. We used the set partitioning version of our algorithm, because set covering is not feasible. Overcovers cannot always be removed afterwards, because that may cause duties to start and end at different locations.

Computational results

We also implemented a version where rosters are built directly from flights, that is, without first solving the CSP. In Table 1 below we compare the results for the regular sequential approach of first scheduling then rostering with the integrated approach. For the sake of easy comparison, the objective here is to minimize the number of uncovered tasks primarily, and the number of crew members secondarily.

		First scheduling			Integrated		
Data Set	# tasks	CPU (sec.)	# crew members	# not scheduled	CPU (sec.)	# crew members	# not scheduled
A	196	37	32	8	96	32	3
B	244	359	39	0	722	39	0
C	521	41	53	17	96657	51	0

Table 1 - Integrated scheduling & rostering

The second column in the table shows the number of tasks, the third to fifth column show, respectively, the run time (again on a Pentium II/400Mhz/128Mb), the number of crew members, and the number of tasks that have no roster for the sequential approach. The sixth to the eighth column show the same numbers for the integrated approach. For data set B there is no difference in solution quality, but for data sets A and C there is a large difference. In data set A, 5 tasks that are not assigned in the sequential approach can be assigned in the integrated approach, and the run time is about 3 times higher. In data set C, all 17 unassigned tasks of the sequential method could be assigned in the integrated method while using two crews less. The run time explodes from 41 seconds to almost 27 hours. The conclusion is that the way the duties are built can have an enormous influence on the quality of the roster solution and the run time.

In Table 2 below, we show the results of a comparison of the branch-and-price algorithm (B&P) with an earlier branch-and-bound version of the system (B&B) where we used column generation only in the root node. The same objective is used as in the previous table.

Data Set	Algorithm	CPU (sec.)	Duality Gap (%)		# crew members	# not scheduled	# branches	
			co-pilot	pilot			co-pilot	pilot
A	B & B	61	11	5	35	8	200	200
A	B & P	37	0	0	32	8	17	3
B	B & B	124	15	17	43	0	200	200
B	B & P	359	0	0	39	0	28	40
C	B & B	47	7	44	54	24	200	200
C	B & P	41	0	0	53	17	12	13

Table 2 - Branch-and-bound vs. Branch-and-price

As mentioned in Section 5.1. this earlier version produced good quality solutions for an application of crew rostering for a Dutch charter airline. However, as can be seen from the table, the solutions of the B&B version can produce a gap of up to 44% which is highly unacceptable. Note the number of branches in the last two columns. The maximum number of branches for the B&B version is 200, but the solutions do not improve significantly when increasing this number.

6. Summary

This paper has discussed a decision support system for airline and railway crew planning. The focus is not on the mathematical part, since the basic ideas have been reported in other papers, but on the system, the implementation and the applications. We give insight in several implementation issues that usually come up when implementing a branch-and-price algorithm for a practical application. When looking at the development of the algorithms in the system, it is interesting to note that for the first airline application before 1996, a simple day-by-day

heuristic was not appropriate. On the contrary, in the last railway application a similar simple heuristic turned out to be more appropriate. The cause can be found in several factors, among which the type of constraints and objectives, and the definition of the duties. In case of regular duties of similar length a heuristic seems to have more success as compared to a more exact method. Thus, when working in practice for each application the best method needs to be determined. Finally, we can conclude for the comparison with the integrated scheduling and rostering, that the way duties are built can have a huge impact on the quality of the rosters at the cost of a huge increase in computing time.

Acknowledgements

The authors like to thank Daan Ament from ORTEC Consultants for his valuable comments.

References

Andersson, E., Housos, E., Kohl, N., Wedelin, D., 1998. Crew pairing optimization. In: Yu, G. (Ed.), Operations Research in the airline industry, Kluwer Acadamic Publishers, Dordrecht, pp. 228-257.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., Vance, P. H., 1998. Branch-and-Price: column generation for solving huge integer programs, Operations Research 46 316-329.

Bixby, R., Gregory, J., Lustig, I., Marsten, R., Shanno, D., 1992. Very large-scale linear programming: a case study in combining interior point and simplex methods, Operations Research 40 885-897.

Bodin, L., Golden, B., Assad, A., and Ball, M., 1983. Routing and scheduling of vehicles and crews: the state of the art, Computers and Operations Research 10 (2) 63-211.

Caprara, A., Monaci, M., Toth, P., 2001. A global method for crew planning in railway applications. In: Voß, S., Daduna, J.R. (Eds.), Computer-Aided Scheduling of Public Transport, Lecture Notes in Economics and Mathematical Systems, 505, Springer, Berlin, pp. ?????????.

Day, P.R., Ryan, D.M., 1997. Flight attendant rostering for short-haul airline operations, Operations Research 45 649-661.

Desaulniers G., Desrosiers J., Dumas Y., Marc S., Rioux B., Solomon M.M., Soumis F., 1997. Crew pairing at Air France, European Journal Operational Research (97)2 245-259.

Desaulniers, G., Desrosiers, J., Gamache, M., Soumis, F., 1998. Crew scheduling in air transportation. In: Crainic, T.G., Laporte, G. (Eds.), Fleet Management and Logistics, Kluwer Acadamic Publishers, Boston, MA, pp. 169-185.

Desaulniers, G., Desrosiers, J., Solomon, M.M., 1999. Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems, Technical report G-99-36, GÉRAD.

Desrochers, M.,Gilbert, J., Sauve, M., Soumis, F., 1992. Subproblem modeling in a column generation approach to the urban crew scheduling problem. In: Desrochers, M., Rousseau, J.M. (Eds.), Computer-Aided Transit Scheduling: Proceedings of the Fifth International Workshop, Springer Verlag, Berlin, pp. 395-405.

Desrosiers, J.. Dumas, Y., Solomon, M.M., Soumis F., 1995. Time constrained routing and scheduling. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (Eds.), Handbooks in Operations

Research and Management Science, volume 8: Network Routing, North-Holland, Amsterdam, pp. 35-139.

Desrosiers, J., A. Lasry, D. McInnis, M.M. Solomon, F. Soumis, 2000. Air transat uses ALTITUDE to manage its aircraft routing, crew pairing, and work assignment. Interfaces 30 (2) 41-53.

Desrosiers, J., Solomon, M.M., Villeneuve, D., Ben Amor, H., 2001. Stabilized column generation for crew scheduling problems, presentation at TRISTAN IV conference, Azores Islands.

Fischetti, M, Kroon, L.G., 2000. Solving large-scale crew scheduling problems at the Dutch railways. In: Voß, S., Daduna, J.R. (Eds.), Computer-Aided Scheduling of Public Transport, Lecture Notes in Economics and Mathematical Systems, 505, Springer, Berlin, pp. 181-201.

Freling, R., 1997. Models and techniques for integrating vehicle and crew scheduling, PhD Thesis, Tinbergen Institute, Erasmus University, Rotterdam.

Freling, R., Lentink R.M., Odijk, M.A., 2001. Scheduling train crews: a case study for the Dutch Railways. In: Voß, S., Daduna, J.R. (Eds.), Computer-Aided Scheduling of Public Transport, Lecture Notes in Economics and Mathematical Systems, 505, Springer, Berlin, pp. 153-165.

Gamache, M., Soumis, F., 1998. A method for optimally solving the rostering problem. In: Yu, G. (Ed.), Operations Research in the airline industry, Kluwer Acadamic Publishers, Dordrecht, pp. 124–157.

Gamache, M., Soumis, F., Marquis, G., Desrosiers, J., 1999. A column generation approach for large-scale aircrew rostering problems, Operations Research 47 247-263.

Kohl, N., 1995. Exact methods for time constrained routing and scheduling problems, PhD Thesis, Technical University of Denmark.

du Merle, O., Villeneuve, D., Desrosiers, J., Hansen, P., 1999. Stabilized column generation, Discrete Mathematics 194 229-237.

Nicoletti, B., 1975. Automatic crew rostering, Transportation Science 9 33-42.

Odoni, A.R., Rousseau, J.M., 1994. Models in urban and air transportation. In: Pollock, S.M., Rothkopf, M.H., Barnett, A. (Eds.), Handbooks in Operations Research and Management Science, volume 6: Operations Research and the Public Sector, North-Holland, Amsterdam, pp. 129-150.

Ryan, D.M., 1992. The solution of massive generalized set partitioning problems in aircrew rostering, Operations Research 43 459-467.

Ryan, D.M., 2000. Optimization earns its wings, OR/MS Today 27(2) 26-30.

Vance P.H., Atamtürk, A., Barnhart, C., Gelman, E., Johnson, E.L., Krishna, A., Mahidhara, D., Nemhauser, G.L., Rebello, R., 1997. A heuristic Branch-and-Price approach for the airline crew pairing problem, Technical report, Georgia Institute of Technology.

Sol, M., 1994. Column generation techniques for pickup and delivery problems, PhD thesis, Eindhoven University of Technology.

Warburton, A., 1987. Approximation of pareto optima in multi-objective, shortest path problems, Operations Research 35 70-79.

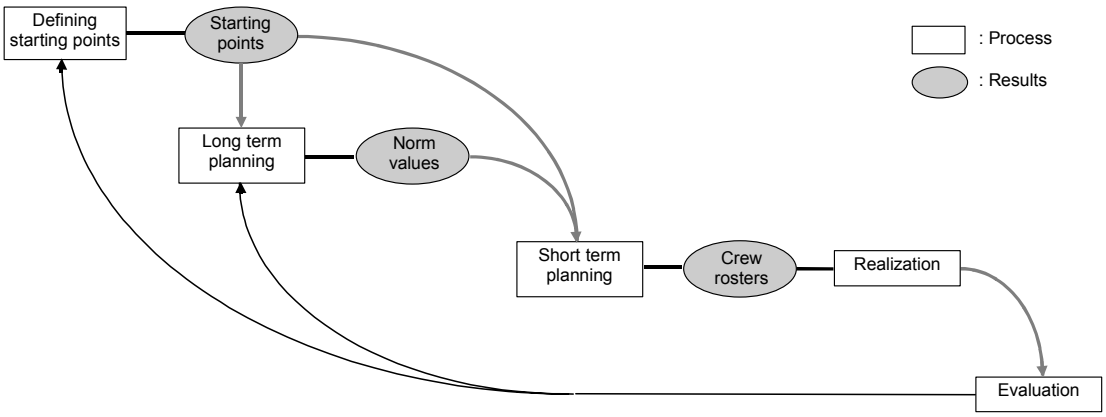

Figure 1: The planning process

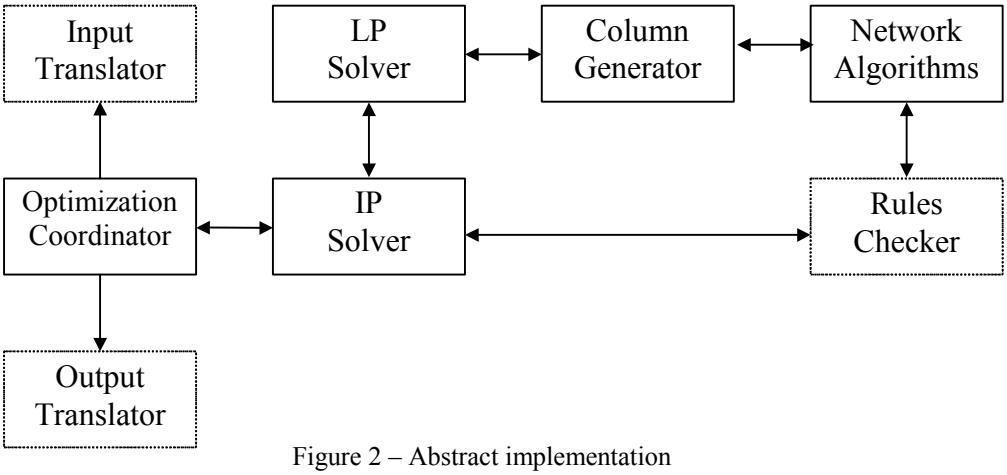

Figure 2 – Abstract implementation

Publications in the Report Series Research* in Management

ERIM Research Program: "Business Processes, Logistics and Information Systems"

2001

Bankruptcy Prediction with Rough Sets
Jan C. Bioch & Viara Popova
ERS-2001-11-LIS

Neural Networks for Target Selection in Direct Marketing
Rob Potharst, Uzay Kaymak & Wim Pijls
ERS-2001-14-LIS

An Inventory Model with Dependent Product Demands and Returns
Gudrun P. Kiesmüller & Erwin van der Laan
ERS-2001-16-LIS

Weighted Constraints in Fuzzy Optimization
U. Kaymak & J.M. Sousa
ERS-2001-19-LIS

Minimum Vehicle Fleet Size at a Container Terminal
Iris F.A. Vis, René de Koster & Martin W.P. Savelsbergh
ERS-2001-24-LIS

The algorithmic complexity of modular decompostion
Jan C. Bioch
ERS-2001-30-LIS

A Dynamic Approach to Vehicle Scheduling
Dennis Huisman, Richard Freling & Albert Wagelmans
ERS-2001- 35-LIS

Effective Algorithms for Integrated Scheduling of Handling Equipment at Automated Container Terminals
Patrick J.M. Meersmans & Albert Wagelmans
ERS-2001-36-LIS

Rostering at a Dutch Security Firm
Richard Freling, Nanda Piersma, Albert P.M. Wagelmans & Arjen van de Wetering
ERS-2001-37-LIS

Probabilistic and Statistical Fuzzy Set Foundations of Competitive Exception Learning
J. van den Berg, W.M. van den Bergh, U. Kaymak
ERS-2001-40-LIS

Design of closed loop supply chains: a production and return network for refrigerators
Harold Krikke, Jacqueline Bloemhof-Ruwaard & Luk N. Van Wassenhove
ERS-2001-45-LIS

---

* A complete overview of the ERIM Report Series Research in Management:
http://www.ers.erim.eur.nl

ERIM Research Programs:
LIS Business Processes, Logistics and Information Systems
ORG Organizing for Performance
MKT Marketing
F&A Finance and Accounting
STR Strategy and Entrepreneurship

Dataset of the refrigerator case. Design of closed loop supply chains: a production and return network for refrigerators
Harold Krikke, Jacqueline Bloemhof-Ruwaard & Luk N. Van Wassenhove
ERS-2001-46-LIS

How to organize return handling: an exploratory study with nine retailer warehouses
René de Koster, Majsa van de Vendel, Marisa P. de Brito
ERS-2001-49-LIS

Reverse Logistics Network Structures and Design
Moritz Fleischmann
ERS-2001-52-LIS

Pattern-based Target Selection applied to Fund Raising
Wim Pijls, Rob Potharst & Uzay Kaymak
ERS-2001-56-LIS

A Decision Support System for Crew Planning in Passenger Transportation using a Flexible Branch-and-Price Algorithm
ERS-2001-57-LIS
Richard Freling, Ramon M. Lentink & Albert P.M. Wagelmans

2000

A Greedy Heuristic for a Three-Level Multi-Period Single-Sourcing Problem
H. Edwin Romeijn & Dolores Romero Morales
ERS-2000-04-LIS

Integer Constraints for Train Series Connections
Rob A. Zuidwijk & Leo G. Kroon
ERS-2000-05-LIS

Competitive Exception Learning Using Fuzzy Frequency Distribution
W-M. van den Bergh & J. van den Berg
ERS-2000-06-LIS

Models and Algorithms for Integration of Vehicle and Crew Scheduling
Richard Freling, Dennis Huisman & Albert P.M. Wagelmans
ERS-2000-14-LIS

Managing Knowledge in a Distributed Decision Making Context: The Way Forward for Decision Support Systems
Sajda Qureshi & Vlatka Hlupic
ERS-2000-16-LIS

Adaptiveness in Virtual Teams: Organisational Challenges and Research Direction
Sajda Qureshi & Doug Vogel
ERS-2000-20-LIS

Assessment of Sustainable Development: a Novel Approach using Fuzzy Set Theory
A.M.G. Cornelissen, J. van den Berg, W.J. Koops, M. Grossman & H.M.J. Udo
ERS-2000-23-LIS

Applying an Integrated Approach to Vehicle and Crew Scheduling in Practice
Richard Freling, Dennis Huisman & Albert P.M. Wagelmans
ERS-2000-31-LIS

An NPV and AC analysis of a stochastic inventory system with joint manufacturing and remanufacturing
Erwin van der Laan
ERS-2000-38-LIS

Generalizing Refinement Operators to Learn Prenex Conjunctive Normal Forms
Shan-Hwei Nienhuys-Cheng, Wim Van Laer, Jan Ramon & Luc De Raedt
ERS-2000-39-LIS

Classification and Target Group Selection bases upon Frequent Patterns
Wim Pijls & Rob Potharst
ERS-2000-40-LIS

Average Costs versus Net Present Value: a Comparison for Multi-Source Inventory Models
Erwin van der Laan & Ruud Teunter
ERS-2000-47-LIS

Fuzzy Modeling of Client Preference in Data-Rich Marketing Environments
Magne Setnes & Uzay Kaymak
ERS-2000-49-LIS

Extended Fuzzy Clustering Algorithms
Uzay Kaymak & Magne Setnes
ERS-2000-51-LIS

Mining frequent itemsets in memory-resident databases
Wim Pijls & Jan C. Bioch
ERS-2000-53-LIS

Crew Scheduling for Netherlands Railways. "Destination: Curstomer"
Leo Kroon & Matteo Fischetti
ERS-2000-56-LIS