Journal of the Operational Research Society (2005) 56, 649–658

www.palgrave-journals.com/jors

Heuristical labour scheduling to optimize airport passenger flows

S Casado[1], M Laguna[2]* and J Pacheco[1]

[1]*Facultad C. Económicas y Empresariales, Pza. Infanta Elena, s/n. Burgos 09001, Spain; and* [2]*University of Colorado, Boulder, CO, USA*

We describe the development and implementation of a decision support system for the optimization of passenger flow by trading off service quality and labour costs at an airport. The system integrates a simulation module with an optimization module that requires that Dantzig's labour scheduling problem be solved in the order of thousands of times. We developed a customized scatter search to give the system the capability of finding high-quality solutions to the labour scheduling problems in short computational times. Our experiments verify that our scatter search implementation meets the needed requirements.

Journal of the Operational Research Society (2005) **56,** 649–658. doi:10.1057/palgrave.jors.2601859
Published online 17 November 2004

Keywords: labour scheduling; scatter search; GRASP; path relinking

Introduction

An operational problem in many service organizations is to determine the number of employees—and their work schedules—needed to minimize labour costs, while reaching a desired level of service quality. This problem is known in the literature as the *labour scheduling problem*. Thompson[1] states the importance of this problem as follows:

> *Labor scheduling—the process of matching, over the operating day, the number of employees working to the number of employees needed to provide the desired level of customer service—is frequently a large determinant of service organization efficiency.*

The development of efficient labour schedules has long been recognized as an important factor for improving productivity in services. Many service organizations deal with demands that vary significantly from hour-to-hour and day-to-day. If these organizations lack the capacity to satisfy demand as it occurs, they may incur lost sales or other shortage costs. Similarly, if their service capacity exceeds the current rate of demand, the excess capacity may go to waste.

Dantzig[2] was the first to model the labour scheduling problem as a mathematical programme. His classical integer programming formulation is as follows:

$$\min \sum_{j=1}^{m} c_j x_j \tag{1}$$

s.t.

$$\sum_{j=1}^{m} a_{ij} x_j \geqslant r_i, \quad i = 1, \ldots, h \tag{2}$$

$$x_j \geqslant 0; \quad x_j \in Z \tag{3}$$

where

h the number of work periods in the planning horizon (normally expressed in hours)

M the master set of allowable tours (shifts) $\{ j = 1, \ldots, m\}$

$$a_{ij} = \begin{cases} 1, & \text{if } i \text{ is a work period in tour (shift)}_j, \\ & i = 1, \ldots, h \text{ and } j = 1, \ldots, m \\ 0, & \text{otherwise} \end{cases}$$

c_j cost of having an employee working in tour (shift) j, $j = 1, \ldots, m$

r_i the number of employees required in period i, $i = 1, \ldots, h$

x_j the number of employees assigned to tour (shift) j, $j = 1, \ldots, m$

The objective of this model is to minimize the cost of the scheduled shifts in the planning horizon (eg, a day or a week), subject to the restriction that the employee requirements in every period are satisfied. Extensions to this labour scheduling model have been proposed by Betchold and Jacobs[3,4] who implicitly matched meal breaks to explicitly represent shifts. Betchold *et al*[5] rigorously evaluated a wide range of scheduling heuristics. Implicit modelling has been used by Thompson[6] for scheduling employees having limited availability and by Thompson[7] for scheduling work in services over which management has some temporal control.

*Correspondence: M Laguna, Leeds School of Business, University of Colorado, Boulder, CO 80309-0419, USA.
E-mail: laguna@colorado.edu

Since service organizations (whether public or private) commonly deal with a high level of scheduling flexibility, the number m of allowable shifts tends to be large. Bartholdi[8] showed that this family of labour scheduling problems is NP-complete, motivating the development of both conventional stand-alone heuristics and metaheuristic solution procedures. Several successful service organizations, including LL Bean,[9–11] United Airlines,[12] and the San Francisco Police Department[13] have obtained significant performance improvements after adopting optimization techniques to deal with their labour scheduling problems.

The most effective conventional heuristics for labour scheduling in the literature combine linear programming and local search.[5,14,15] The local search procedures in most conventional labour scheduling heuristics do not allow the search to cross the feasibility boundary and visit infeasible solutions. Infeasible solutions in Dantzig's model are such that for at least one period the number of scheduled employees is strictly less than the number of required employees. In other words, the solution is such that at least one of the constraints (2) is violated. When an optimization procedure is not allowed to cross the feasibility boundary, the search paths are confined to the feasible region of the search space, which may limit the procedure's ability to find improved solutions. See, for example, Glover and Kochenberger[16] and Laguna *et al.*[17]

Metaheuristic procedures have also been developed for a variety of labour scheduling problems. Specifically, Brusco and Jacobs,[18,19] Brusco *et al*,[20] and Thompson[21] developed procedures based on *simulated annealing*. Easton and Mansour[22] experimented with *genetic algorithms*. Glover and McMillan,[23] Taylor and Huxley,[13] Easton and Rossin,[24] Downsland,[25] and Alvarez-Valdes *et al*[26] proposed *tabu search* procedures.

In the present development, we report on a component of a decision support system designed to trade off labour costs and service quality (as measured by waiting time) at an airport. Specifically, the system has been developed for the Madrid–Barajas Airport, where the labour costs are associated with security control and check-in personnel. The system is such that, for each desired level of service, it carries out the following steps:

1. Optimization of a simulated model of the airport to establish the labour requirements needed at check-in and security control points to obtain the targeted service level.
2. Given the requirements obtained in the previous step, a *labour scheduling problem* is solved for each check-in and security control point accounting for the shift sets established for different types of workers.

The characteristics of the problem create two conflicting objectives: minimizing costs and maximizing quality of service. For instance, if the labour costs were reduced (lowering the number of workers), there would be an increase in passenger waiting times, and thus service quality would deteriorate. In such a situation, being able to estimate labour costs associated with different levels of service quality or *vice versa*—that is, estimating quality of service for a given labour budget—is of great interest to the decision maker. To achieve this, the two steps above would have to be executed many times as several combinations of service levels and budgetary scenarios are investigated. In addition, it is necessary to execute the entire process whenever changes take place in the normal operation of the airport, due, for instance, to flight cancellations and changes in time schedules.

Executing the two steps above within an interactive decision support system requires fast evaluation procedures. These steps have to be executed efficiently using methods that can provide approximately 100 high-quality solutions in about 3 h. Each solution represents a combination of labour cost and service level that the decision maker is considering. This means that each solution must be found in at most 108 s. The first phase (simulation-optimization) requires about 1 min of computational time, leaving approximately 48–50 s to complete the second phase. In the second phase, a labour scheduling problem must be solved for each of 50–60 stations. Hence, a high-quality solution to the labour scheduling problem must be found in less than a second. All these computational times are estimated considering a Pentium 4 machine at 2.53 GHz.

The main contribution of this paper relates to step 2 and consists of a metaheuristic method that searches for solutions to the labour scheduling problem and that can be embedded in an interactive decision support system for the optimization of passenger flow. We opted for metaheuristic strategies rather than optimal methods because of the need to find high-quality solutions fast even though optimality of the solutions may not have been confirmed. The labour scheduling problem to be solved for each station (both check-in and security) fits the Dantzig model. In this case, all the shifts have the same cost, hence, we can simply make $c_j = 1$ for all j. Our metaheuristic implementation exploits this special structure to produce high-quality solutions fast even when tackling problems of realistic sizes, which involve a large number of shifts and a planning horizon of up to 1 week.

The remainder of this paper is organized as follows. The next section describes the metaheuristic procedure based on the scatter search methodology. The subsequent section deals with determining the values for the search parameters associated with the procedure described in the previous section. The section thereafter discusses the results obtained from the computational testing, including comparative tests with other methods. The last section shows an example of the decision support system in which our metaheuristic procedure is embedded and then we draw some conclusions.

Solution approach

The solution approach we have developed is an adaptation of the scatter search (SS) methodology. SS is an instance of the so-called evolutionary methods, with the main distinction that its mechanisms for searching are not based solely on randomization. More about the origins and multiple applications of SS can be found in Glover,[27] Laguna[28] and Laguna and Martí.[29]

SS is characterized by the use of a *Reference Set* (*RefSet*) of solutions. At each step of the solution procedure, reference solutions are combined to generate new solutions, which in turn are used to update the current *RefSet* according to some systematic rules. For our labour scheduling problem, we have developed a version of SS that uses a static update of the *RefSet*. For more details on alternative SS designs, the reader is referred to the book by Laguna and Martí.[29] Figure 1 shows a general structure of our SS implementation in the form of a pseudocode.

The size of P is denoted by *PSize* (step 1). In addition, $b = b_1 + b_2$ denotes the size of the *RefSet*. In order to build the initial *RefSet* (step 3), we start by selecting the best, according to the objective function value, b_1 solutions in P. Then, the remaining b_2 elements are added to increase the diversity of the initial *RefSet*. We measure the 'diversity' of a candidate solution x in relation to those elements already in *RefSet*. In particular, we calculate the minimum distance (*MinDist*) between a candidate solution x and all the solutions x' in the reference set:

$$\text{MinDist}(x,\ RefSet) = \min\{d(x, x') : x' \in RefSet\}$$

where $d(x,x') = \sum_j |x_j - x_j'|$. The solution $x \in P$ that maximizes *MinDist*(x, *RefSet*) is added to *RefSet* and deleted from P. The process is repeated until b_2 solutions are added to *RefSet*.

The updating of *RefSet* (step 4.4) is carried out by taking into account only the quality of the solutions. In other words, a new trial solution x that improves the objective function of the worst reference solution x^b is included in *RefSet*. Since the *RefSet* is ordered according to solution quality, the worst reference solution is always the last one in the set and is denoted by x^b.

Diversification method

Our diversification method is based on GRASP constructions. GRASP (greedy randomized adaptive search procedure) is a heuristic that constructs solutions with controlled randomization and a greedy function. Most GRASP implementations also include a local search that is used to improve upon the solutions generated with the randomized greedy function. GRASP was originally proposed in the context of a set covering problem.[30] Details of the methodology and a survey of applications can be found in Feo and Resende[31] and Pitsoulis and Resende.[32]

The greedy function that we have selected in our current setting is denoted by Δ_j and represents the number of infeasible periods included in shift j. The procedure starts from a solution x obtained by rounding to the closest integer the solution to the continuous relaxation of Dantzig's model. The rounded solution x^r is integer but may be infeasible with respect to meeting the labour requirements in every period of the planning horizon. Figure 2 shows a pseudocode of the Diversification method.

The α parameter ($0 \leqslant \alpha \leqslant 1$) controls the level of randomization for the greedy selections. Randomization decreases as the value of α increases. When $\alpha = 1$ the candidate list CL contains only one element, the shift with the maximum number of infeasible periods. When $\alpha = 0$ all shifts with at least one infeasible period are members of the candidate list CL. The controlled randomization from choosing an α-value that is strictly between 0 and 1 results in a sampling procedure where the best solution found is typically better than the one found by setting α to either 1 or 0. A judicious selection of the value of α provides a balance between diversification and solution quality.

Improvement method

Our improvement method, used in steps 2 and 4.3 of Figure 1, is a local search procedure with a neighbourhood structure defined by the compound moves described in this section. We start by noticing that the objective function of our labour scheduling problem is such that any improving move involves eliminating workers from one or more shifts.

1. Generate an initial set P of solutions with a Diversification Generation Method
2. Improve these solutions with an Improvement Method
3. Build an initial *RefSet* from the improved P set
4. Do until *RefSet* converges (i.e., no new trial solutions are admitted to the set)
    - 4.1. Obtain all pairs of reference solutions for which at least one of the two solutions is new in the pair
    - 4.2. Apply a Combination Method to these subsets and obtain new trial solutions
    - 4.3. Improve the new trial solutions with an Improvement Method
    - 4.4. Update *RefSet* considering both the current reference solutions and the new trial solutions

Figure 1 High-level view of the SS implementation.

1. x = rounded LP solution and $w_i = \sum_{j=1}^{m} a_{ij} x_j$ for $i = 1, \ldots, h$
2. Do until $w_i \geq r_i \; \forall \; i = 1, \ldots, h$
   - 2.1. Calculate $\Delta_j = \sum_{i / w_i < r_i} a_{ij}$, $j = 1..m$
   - 2.2. Calculate $\Delta_{max} = \max \{\Delta_j : j = 1, \ldots, m\}$ and $\Delta_{min} = \min \{\Delta_j : j = 1, \ldots, m\}$
   - 2.3. Build $CL = \{ j : \Delta_j \geq \alpha \Delta_{max} + (1\text{-}\alpha)\Delta_{min}, j = 1, \ldots, m\}$
   - 2.4. Choose j^* randomly from CL and make $x_{j^*} = x_{j^*} + 1$
   - 2.5. Update $w_i = w_i + a_{ij^*}$, $i = 1, \ldots, h$

Figure 2 Diversification generation method based on GRASP constructions.

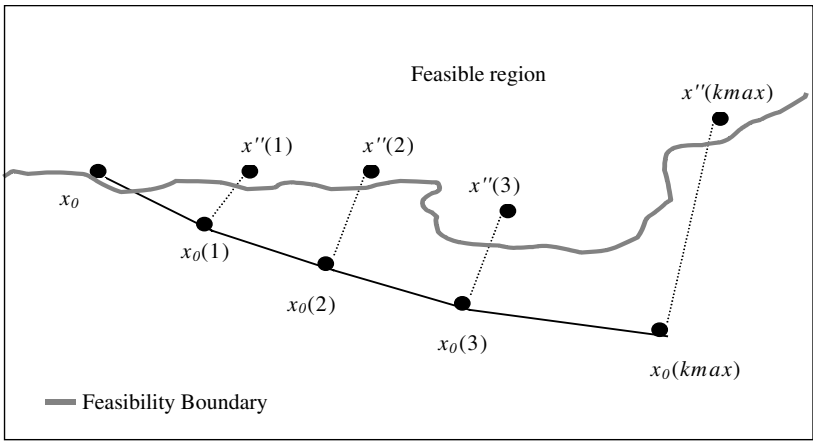

Figure 3 Compound move with projections.

This means that feasible solutions of high quality tend to be near or at the feasibility boundary. Infeasible solutions are reached when the elimination of workers results in a labour shortage in any period during the planning horizon.

Preliminary experimentation showed that a higher level of performance is achieved when the improvement method is allowed to follow a search trajectory that includes both feasible and infeasible solutions. Therefore, the procedure that we have developed crosses the feasibility boundary in both directions. When the search crosses the boundary toward the infeasible region, a projection mechanism is used to map infeasible solutions onto feasible ones. The projection procedure, based on the ideas of Brusco and Jacobs,[18] is quite simple. It consists of executing the steps in Figure 2 starting from the infeasible solution x and setting α to 1.

The procedure to recover feasibility is employed within a move mechanism that consists of a sequence of changes involving the elimination of a worker from a shift. Hence, a single compound move is defined by a collection of simple moves. The mechanism, which is also inspired in some ideas by Brusco and Jacobs,[18] is illustrated in Figure 3. This shows the sequence of simple moves taken from a seed solution x_0. The k_{max} parameter is used to control the number of simple moves that are performed within the compound move. Each simple move produces a new solution $x_0(k)$, for $k = 1, \ldots, k_{max}$. Each infeasible $x_0(k)$ solution is projected to the feasibility region using the projection mechanism. After k_{max} simple moves, the best feasible solution $x''(k)$ is chosen. The procedure is reinitiated by making $x_0 = x''(k)$.

Combination method

New solutions are generated from combining pairs of reference solutions (step 4.2 in Figure 1). The number of solutions generated from each combination depends on the relative quality of the solutions being combined. Let x^p and x^q be two reference solutions being combined, where $p < q$. We work with a reference set that is ordered in a way that x^1 is the best solution and x^b is the worst. Then, the number of solutions generated from each combination is:

3 if $p \leqslant b_1$ and $q \leqslant b_1$
2 if $p \leqslant b_1$ and $q > b_1$
1 if $p > b_1$ and $q > b_1$

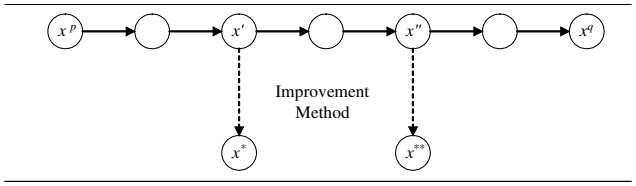

Figure 4 Path relinking to generate new trial solutions.

Hence, every pair of reference solutions is combined to generate new solutions but not all pairs generate the same number of trial solutions. The combination method is based on a strategy known as *path relinking*, which was originally proposed in the context of *tabu search*[33] and has also been used in scatter search[29] as well as GRASP.[34,35]

The underlying premise of path relinking is that in the path between two good solutions, other solutions of similar quality (and perhaps better) may be found. The basic idea is to build a path to join two solutions: an initiating solution and a guiding solution. This path generates a number of intermediate solutions that are projected and improved using the improved method described previously. In our implementation, the intermediate solutions are generated as equidistant as possible from each other.

Figure 4 shows a schematic representation of the path relinking process. The initiating solution is x^p and the guiding solution is x^q. During the relinking process, solutions x' and x'' are chosen for the application of the improvement method, which includes the projection mechanism. The application of the improvement method results in the improved solutions denoted as x^* and x^{**} in Figure 4.

The path between x^p and x^q is built by concentrating on the shifts that have different values (number of employees) in both solutions. Let an intermediate solution in the path between x^p and x^q be x, then initially $x = x^p$. In the path relinking process, we first consider the set of shifts $E = \{j : x_j > x_j^q\}$ and find j^* such that $\beta(j^*) = \max_{j \in E}\{\beta(j)\}$, where

$$\beta(j) = |\{i : a_{ij} = 1,\ w_i \geqslant r_i + 1\}|$$

As long as $E \neq \varnothing$, one worker is removed from shift j^*, that is, $x_{j^*} = x_{j^*} - 1$. When $E = \varnothing$, we consider the set of shifts $F = \{j : x_j < x_j^q\}$ and find j^* such that $\Delta_{j^*} = \max_{j \in F}\{\Delta_j\}$, where Δ_j is the function defined in Figure 2. As long as $F \neq \varnothing$, one worker is added to shift j^*, that is, $x_{j^*} = x_{j^*} + 1$. In this way, the intermediate solution x moves closer to x^q at each step of the relinking process.

Parameter fine tuning

One of the most time-consuming tasks in the development of metaheuristic procedures for optimization is the tuning of search parameters. Díaz and Laguna[36] propose a semi-automated parameter tuning system called CALIBRA that employs statistical analysis techniques and a local search procedure to create a systematic way of fine-tuning algorithms. The goal of CALIBRA is to provide a system to fine-tune algorithms, where a user needs only to specify a range for each parameter to be tuned, a training set of instances and a measure of performance. The quality of a set of parameter values is tested on the specified set of problem instances. CALIBRA is available at http://opalo.etsiig.uniovi.es/~adenso/file_d.html, where a user manual can also be found.

Our complete set of test problems consists of 800 instances. We randomly selected a relatively small training set of eight problems, because we considered that this sample would be representative and that the parameter values found with CALIBRA would also perform well when applied to the entire test set. The parameters to be adjusted were α in the (0.1, 0.9) range, $k_{\max}$ in the (3, 7) range, *PSize* in the (12, 20) range, b_1 in the $(1, b-1)$ range where $b = \min(10, \lfloor PSize/2 \rfloor)$ and $b_2 = b - b_1$. After 100 CPU minutes on a Pentium III machine at 600 MHz, CALIBRA obtained the following parameter values: $\alpha = 0.7$, $k_{\max} = 6$, $PSize = 14$ and $b_1 = 3$. We used these parameter values for all of the experiments reported in the next section.

Computational experiments

In order to assess the efficiency of our SS procedure, a series of tests were carried out to compare its performance with the Simulated Annealing procedure (SA) described in the paper of Brusco and Jacobs,[19] the Tabu search algorithm by Alvarez-Valdés *et al*,[26] and the results obtained from solving Dantzig's model with CPLEX 8.0.

Four sets consisting of 200 artificial problems each were generated. The problem generator is described in the Appendix. In sets I and II, the number of shifts was set to $m = 221$, while in sets III and IV the number of shifts was set to $m = 320$. In all cases, the number of periods was $h = 168$. Since we seek a fast procedure to solve the labour scheduling problem repeatedly, the total solution times under consideration were 0.4 and 1 s. The tests were carried out on a Pentium 4 at 2.53 GHz and 512 MB of RAM. In addition, in order to measure the quality of the different solutions, a lower bound for each problem instance was obtained with a 2-min CPLEX run.

The proposed method described in this paper, as well as of the procedures by Brusco and Jacobs,[19] and Alvarez-Valdés *et al*,[26] were programmed in PASCAL, using the compilers BORLAND PASCAL 7.0 and BORLAND DELPHI 5.0. Tables 1–3 summarize the experimental results. Table 1 contains the following information:

- For each set of problems, the number of instances that have been solved optimally executing CPLEX during 2 min (Known Opt).

Table 1 Number of optima and percent deviation

			0.4 s				*1.0 s*			
Type	*Known opt*		*Cplex*	*SS*	*TS*	*SA*	*Cplex*	*SS*	*TS*	*SA*
I	65	Opt	7	25	6	0	17	31	6	0
		Dev	1.4	0.91	1.45	3.74	1.09	0.76	1.43	3.73
		LBDev	3.33	2.58	3.28	5.2	3.05	2.43	3.24	5.18
II	81	Opt	14	28	8	0	25	33	9	0
		Dev	1.34	0.93	1.5	3.73	0.99	0.85	1.48	3.72
		LBDev	3.29	2.51	3.37	5.35	3.02	2.39	3.33	5.32
III	41	Opt	0	11	2	0	5	13	2	0
		Dev	6.79	1.11	1.64	4.14	1.34	1.04	1.64	4.12
		LBDev	7.92	2.9	3.62	5.76	3.66	2.79	3.62	5.71
IV	46	Opt	0	8	4	0	10	10	4	0
		Dev	6.81	1.19	1.48	4.1	1.14	1.11	1.47	4.1
		LBDev	7.46	2.83	3.63	5.49	3.46	2.7	3.61	5.49

Table 2 Number of best solutions found and average objective function values

		0.4 s				*1.0 s*			
Type		*Cplex*	*SS*	*TS*	*SA*	*Cplex*	*SS*	*TS*	*SA*
I	ObjVal	68.73	68.325	68.745	70.15	68.555	68.235	68.735	70.145
	Best	118	196	112	10	132	195	101	9
II	ObjVal	68.285	67.91	68.36	69.815	68.095	67.85	68.35	69.81
	Best	124	195	111	8	144	193	100	8
III	ObjVal	68.73	65.33	65.74	67.19	65.71	65.275	65.74	67.18
	Best	0	198	121	10	112	193	104	8
IV	ObjVal	68.285	65.08	65.48	66.78	65.35	65.02	65.475	66.78
	Best	3	197	113	12	126	191	96	9

- Among the instances that CPLEX solved optimally in 2 min, the number of instances (Opt) that were solved optimally by each alternative and for each computational time considered (0.4 and 1 s). The table also shows the average deviation from optimality (Dev) for the problems for which optimal solutions are known.
- Among the instances that CPLEX did not solve optimally in 2 min, the mean percentage deviation from the best lower bound (LBDev) is shown.

Table 2 shows, for each set of problems and for each computational time under consideration, the number of times that each procedure obtained the best solution among all alternatives (Best), and the average objective function values (ObjVal).

Finally, Table 3 shows, for each set of problems and for each computational time under consideration, the number of times that Cplex obtained better solutions than scatter search (column labelled Cplex), the number of times that SS obtained better solutions that CPLEX (column labelled SS) and the number of times that both procedures obtained the same objective function value (Draws).

Table 3 Performance of SS compared to Cplex

	0.4 s			*1.0 s*		
Type	*Cplex*	*SS*	*Draws*	*Cplex*	*SS*	*Draws*
I	3	81	116	5	68	127
II	5	76	119	7	56	137
III	0	200	0	6	87	107
IV	0	197	3	7	72	121

The following observations can be made regarding the results reported in Tables 1–3:

- Our proposed SS procedure outperforms the alternative metaheuristics that we used for comparison. The performance of SS is superior regardless of the problem type

and the computational time. All statistics (number of optima, number of best, percent deviation from optima and from lower bounds) reported in Tables 2 and 3 favour SS over TS and SA.

- Solving the MIP formulation with Cplex is the second best alternative, except for type III and IV problems when the computational time is 0.4.
- SS is capable of finding solutions of high quality, as indicated by the average deviations from optima, which range between 0.76 and 1.19%.

To conclude this section, we present the results of two statistical tests that compare the performance of the proposed SS procedure and Cplex. The first is the well-known *t*-Student's test for differences of means and the second is a paired test that ignores instances in which both methods obtain the same results (reported as Draws in Table 3). Specifically, we define

A_i the objective function value obtained by Cplex in the ith problem instance

B_i the objective function value obtained by SS in the ith problem instance

and consider the following sets of hypotheses for the *t*-test and the paired test, respectively:

$$H_0 : \bar{A} - \bar{B} \leqslant 0$$
$$H_1 : \bar{A} - \bar{B} > 0$$

$$H_0 : \Pr(A_i > B_i / A_i \neq B_i) \leqslant 0.5$$
$$H_1 : \Pr(A_i > B_i / A_i \neq B_i) > 0.5$$

Table 4 shows the t and Z statistics obtained for each type of problem instances and length of run. The table also shows the tail probability p.

The results in Table 4 indicate that the *t*-test detects significant difference between the performance of SS and Cplex for type II and IV problems when the executing time is limited to 0.4 s. The test does not detect significant differences in performance in any of the other cases. The results of the paired test are such that, when draws are ignored, the probability that SS obtains a better solution than Cplex is significantly larger than 0.5.

In sum, although SS always obtains an average objective function value that is better than the one obtained by running Cplex, these average values are in general not significantly better (as indicated in Table 4). This is due to all the cases where Cplex and SS end up finding the same solution. However, if these cases are ignored, then the probability that SS gives better results than Cplex significantly exceeds 0.5.

The DSS

In this section, we provide a high-level view of how the decision support system for the optimization of passenger flow works. Figure 5 provides a schematic representation of the steps that are performed to determine the number of employees that will be needed in each station in the airport in order to meet a desired level of service. The steps in Figure 5 are performed for each station and for each service

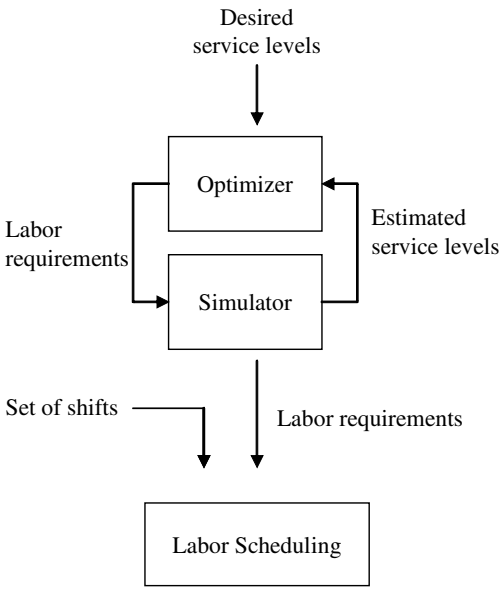

Figure 5 How the sytems works.

Table 4 Results of the Student's *t*-test and paired test

	Type I		*Type II*		*Type III*		*Type IV*	
	t	p	t	p	t	p	t	p
t-test								
0.4 s	0.2383	0.4058	0.2204	0.4128	2.0788	0.0188	1.9562	0.0252
1.0 s	0.188	0.4254	0.1439	0.4428	0.2768	0.391	0.2095	0.417
Paired test	Z	p	Z	p	Z	p	Z	p
0.4 s	8.51	<0.0001	7.89	<0.0001	14.1421	<0.0001	13.72	<0.0001
1.0 s	7.37	<0.0001	6.17	<0.0001	8.4	<0.0001	7.31	<0.0001

level that the decision maker would like to explore. Service level in this context typically is defined in terms of the percentage of passengers that go through a particular station within a predetermined time. Service level may be specified also as a mean waiting time.

Figure 5 shows that the process at each station involves first the optimization of a simulation model. The model mimics the behaviour of the station under consideration and the optimizer is such that it attempts to find the labour requirements that would meet the desired level of service. Once the labour requirements have been established, then the labour scheduling problem is solved to determine the actual labour costs given a set of allowed shifts for each station. The process is performed for every station in the airport and then a final simulation is executed using the number of employees and shifts determined by the solution of the labour scheduling problem. The final simulation gives an estimate of total labour costs and service levels achieved by the entire operation. This final level of service will be at least the same or slightly better than the desired level specified as an input.

The system facilitates decision-making at an airport because the final output provides a set of possible alternatives regarding labour costs and service levels. To illustrate this, consider a problem based on the typical traffic at the Barajas Airport in Madrid:

- 75 000 daily passengers
- 770 daily flights (arrivals and departures)
- 20 000 connections
- 36 arrival gates
- 18 luggage pick-up points
- 90 departure gates
- 50 airlines (and therefore 50 check-in points) with an average of 12 check-in desks per airline
- 9 security control points at arrivals, with 20 gates available for each one
- 9 security control points at departures, with 16 gates available for each one

For this illustration, we consider that the duration of the service follows an exponential distribution with the following mean values:

- 60 s at check-in counters
- 6 s at arrival security controls
- 30 s at departure security controls

Although we have used these mean values for all passengers, they in fact depend on the type of passenger (eg, national vs international, Madrid–Barcelona link, etc.). Finally, 150 weekly shifts have been identified for the check-in personnel and 180 for the security personnel.

Figure 6 shows one of the outputs obtained from using the proposed DSS on the data summarized above. This figure depicts the trade-off between labour costs (given, in this case, by number of employees) and service quality, as measured by the mean waiting time. The trade-off is such that, for instance, to reduce the average waiting time from 40 to 9 min, the number of employees must be increased from 243 to 443. A reduction of the average waiting time from 3 to 2 min requires going from 675 employees to 1235. Hence, the plot in Figure 6 can be used either to predict a service level for a given labour cost or to estimate the necessary budget given a desired level of service.

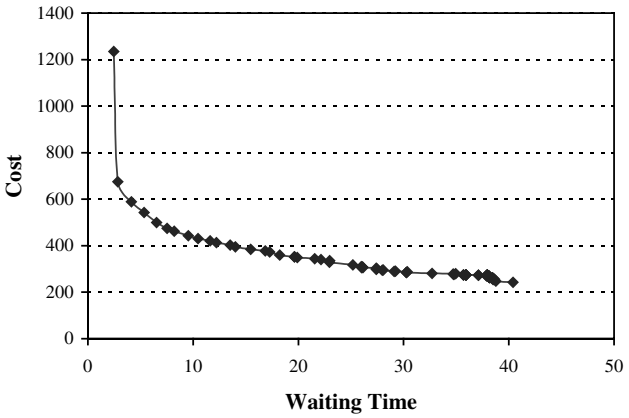

Figure 6 Trade-off between labour cost and average waiting time.

When tackling this problem, the DSS makes 68 calls to the SS procedure that finds solutions to the labour scheduling problem (1 call per station). If the system is used to construct a plot such as the one shown in Figure 5 with 100 solutions, then the labour scheduling problem needs to be solved 6800 times. This justifies the need for a procedure, such as the one based on SS that we have described in this paper, capable of quickly finding high-quality solutions to the labour scheduling problem.

Conclusions

The labour scheduling problem is highly relevant in the literature because of the vast amount of applications it has in the real world, both for public institutions and private companies. Many models and versions have been analysed and a large variety of solutions methods have been proposed.

However, we have not found procedures in the literature capable of finding high-quality solutions within a very limited computational time. Some practical applications, such as the one described in this paper, require solving a large number of labour scheduling problems as part of an overall optimization system. The contribution of this paper is the development of a method based on the scatter search metaheuristic that is capable of providing high-quality solutions in very short computational times for moderate- to large-size problems.

References

1 Thompson GM (1995). Improved implicit optimal modeling of the labor shift scheduling problem. *Mngt Sci* **41**: 595–607.
2 Dantzig GB (1954). A Comment on Edie's 'Traffic Delays at Toll Booths'. *Opns Res* **2**: 339–341.
3 Bechtold SE and Jacobs LW (1990). Implicit optimal modeling of flexible break assignments in labour staffing decisions for service operations. *Mngt Sci* **36**: 1339–1351.
4 Bechtold SE and Jacobs LW (1991). Improvement of labour utilization in shift scheduling for services with implicit optimal modeling. *Int J Opns Ann Prod Mngt* **11**: 54–69.
5 Betchold SEM, Brusco M and Showalter M (1991). A comparative evaluation of labour tour scheduling methods. *Decis Sci* **22**: 683–699.
6 Thompson GM (1990). Shift scheduling when employees have limited availability: an L.P. approach. *J Opns Mngt* **9**: 352–370.
7 Thompson GM (1992). Improving the utilization of front-line service delivery system personnel. *Decis Sci* **23**: 1072–1098.
8 Bartholdi JJ (1981). A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering. *Opns Res* **29**: 501–510.
9 Andrews B and Parsons H (1989). L.L. Bean chooses a telephone agent scheduling system. *Interfaces* **19**(6): 1–9.
10 Andrews B and Parsons H (1993). Establishing telephone-agent staffing levels through economic optimization. *Interfaces* **23**(2): 14–20.
11 Quinn P, Andrews B and Parsons H (1991). Allocating telecommunications resources at L.L. Bean. *Interfaces* **21**(1): 75–91.
12 Holloran T and Byrn J (1986). United Airlines station manpower planning system. *Interfaces* **16**(1): 39–50.
13 Taylor P and Huxley S (1989). A break from tradition for the San Francisco police: patrol officer scheduling using an optimization-based decision support system. *Interfaces* **19**(1): 4–24.
14 Easton EF and Rossin DF (1991). Equivalent alternate solutions for the tour scheduling problem. *Decision Science* **22**: 985–1007.
15 Aykin T (1996). Optimal shift scheduling with multiple break window. *Mngt Sci* **42**: 591–602.
16 Glover F and Kochenberger G (1996). Critical event tabu search for multidimensional knapsack problems. In: Osman IH and Kelly JP (eds). *Meta-Heuristics: Theory & Applications*. Kluwer Academic Publishers, Boston, MA, pp 407–427.
17 Laguna M, Kelly JP, González Velarde JL and Glover F (1995). Tabu search for the multilevel generalized assignment problem. *Eur J Opl Res* **82**: 176–189.
18 Brusco M and Jacobs L (1993). A simulated annealing approach to the cyclic staff-scheduling problem. *Nav Res Logist* **40**: 69–84.
19 Brusco M and Jacobs L (1993). A simulated annealing approach to the solution of flexible labour scheduling problems. *J Opl Res Soc* **44**: 1191–1200.
20 Brusco M, Jacobs L, Bongiorno R, Lyons D and Tang B (1995). Improving personnel scheduling at airline stations. *Opns Res* **43**: 741–751.
21 Thompson GM (1996). A simulated annealing heuristic for shift scheduling using non-continuously available employees. *Comput Opl Res* **23**: 275–278.
22 Easton F and Mansour N (1999). A distributed genetic algorithm for employee staffing and scheduling problems. In: Forrest S (ed). *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan-Kaufmann, San Mateo, CA, pp 360–367.
23 Glover F and Mcmillan C (1986). The general employee scheduling problem: an integration of management science and artificial intelligence. *Comput Opl Res* **13**: 563–593.
24 Easton F and Rossin D (1996). A stochastic goal program for employee scheduling. *Decis Sci* **27**: 541–568.
25 Downsland KA (1998). Nurse scheduling with tabu search and strategic oscillation. *Eur J Opl Res* **106**: 393–407.
26 Alvarez-Valdes R, Crespo E and Tamarit JM (1999). Labour scheduling at an airport refuelling installation. *J Opl Res Soc* **50**: 211–218.
27 Glover F (1998). A template for scatter search and path relinking. In: Hao J-K, Lutton E, Ronald E, Schoenauer M and Snyers D (eds). *Artificial Evolution, Lecture Notes in Computer Science, 1363*. Springer, Berlin, pp 3–51.
28 Laguna M (2002). Scatter search. In: Pardalos PM and Resende MGC (eds). *Handbook of Applied Optimization*. Oxford University Press, New York, pp 183–193.
29 Laguna M and Martí R (2003). *Scatter search. Methodology and Implementations in C*. Kluwer Academic Publishers: Boston, MA.
30 Feo TA and Resende MGC (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Opns Res Lett* **8**: 67–71.
31 Feo TA and Resende MGC (1995). Greedy randomized adaptive search procedures. *J Global Optim* **2**: 1–27.
32 Pitsoulis LS and Resende MGC (2002). Greedy randomized adaptive search procedures. In: Pardalos PM and Resende MGC (eds). *Handbook of Applied Optimization*. Oxford University Press, New York, pp 168–182.
33 Glover F and Laguna M (1997). *Tabu search*. Kluwer Academic Publishers: Boston.
34 Laguna M and Martí R (1999). GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization. *INFORMS J Comput* **11**: 44–52.
35 Resende MGC and Ribeiro CC (2003). A GRASP with path-relinking for private virtual circuit routing. *Networks* **41**: 104–114.
36 Díaz A and Laguna M (2001). *Fine-tuning of Algorithms using Fractional Experimental Designs and Local Search*. University of Colorado at Boulder.

Appendix

Problem instance generator

The problem instance generator used to create data for our computational experiments works as follows:

The generator considers a 1-week planning horizon, that is, $h = 168$. For sets types I and II, $m = 221$ and for sets type III and IV $m = 320$. The a coefficients are the same in all types I and II instances and are randomly generated considering 8-h shifts. Therefore, the only difference between types I and II instances is in the set of requirement values (ie, they use different r-values but the same set of shifts). Sets of types III and IV are also identical except for the requirement values.

To generate the r-vectors, seven daily demand patterns have been considered. Three patterns are selected to create a periodic weekly pattern for problem types I and II. Let P1, P2 and P3 be the chosen patterns (out of the seven available ones). Then, types I and II problems would have P1 assigned

to Monday, P2 to Tuesday, Wednesday and Thursday and P3 to Friday, Saturday and Sunday. The result week would be represented as (P1, P2, P2, P2, P3, P3, P3). This scheme results in $7^3 = 343$ different weeks, from which 200 are randomly chosen for the set of type I and II problems.

Types II and IV problems are generated by perturbing the patterns corresponding to Wednesday, Thursday, Saturday and Sunday. In particular, a random value between -3 and 3 is added to the requirements corresponding to periods 49–96 and 121–168. Hence, the requirements for problems types II and IV are not periodical.

Received September 2003;
accepted June 2004 after two revisions