Finding Near-Optimal Rosters Using Column Generation

Han Hoogeveen

Eelko Penninkx

Department of Information and Computing Sciences,
Utrecht University

Technical Report UU-CS-2007-002

www.cs.uu.nl
ISSN: 0924-3275

Finding Near-Optimal Rosters Using Column Generation

Han Hoogeveen* Eelko Penninkx
Department of Information and Computing Sciences
Utrecht University
P.O. Box 80089, 3508 TB Utrecht, The Netherlands.
slam@cs.uu.nl, penninkx@cs.uu.nl

December 22, 2006

Abstract

We consider the following basic rostering problem. The days are divided in three fixed shifts (Early, Late, Night) and for each day (seven days a week) a number of personnel required is given. To man these shifts, a given number of employees are available, who all have their individual preferences. We have to find a year-roster for each person with minimum total cost. This cost consists of two components: penalties due to deviations from the desired number of personnel and 'dissatisfaction cost' due to a deviation from the preferences. Each roster must obey the legal regulations and some other specific constraints. Furthermore, we require a certain minimum satisfaction level for each employee.

We have applied column generation to solve the above problem. Here a column corresponds to a year-roster for each employee. To reduce the computational burden, we solve the problem by a step-by-step procedure, in which we plan per four-week period, where we optimize the link between the two consecutive four-week periods. When applied in practice, we find near-optimal solutions.

1 Introduction

We look at the problem of finding good rosters for the security personnel of some hospital in the vicinity of Utrecht (The Netherlands). The security personnel have to man several posts in the hospital, seven days a week, 24 hours a day. Therefore, the people work in three shifts: Early, Late, and Night; the required number of persons varies per shift.

*Supported by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society)

Except for a number of part-timers, each worker has a contract for 36 hours per week, but they are assigned approximately 34 hours per week on average to compensate beforehand for extra work that has to be carried out to replace people that are ill. Next to the ordinary working shifts, there are the so-called stand-by shifts, during which a worker can be called in as a replacement. Finally, it is allowed to deviate from the desired number of workers per shift. Here we have that Night shifts and shifts occurring in the weekend should not be over-booked; if there are too few workers in such a shift, then we can hire some outsiders. For the early and late shifts during week-days, it is allowed to have more workers than necessary. This is even encouraged during the Wednesday morning shift, since the superfluous workers can then follow a training session.

Our task is to produce a set of good rosters for a one-year period. These rosters have to satisfy several regulations. The rosters have to be personalized, such that they reflect the personal preferences as much as possible; we wish to maximize the total satisfaction, with the side-constraint that the unluckiest person is not extremely unlucky. Note that in the current situation there are standard cyclic rosters: in week 1, person i gets roster i, and when the week is over, he moves up to the next roster in the list, until the whole cycle of weeks has been run. We further want to find a roster in which the total weighted deviation from the desired number of personnel is minimum; thereto, we minimize the weighted sum of both components.

The above problem can be seen as a mixture between nurse rostering and physician scheduling. It differs from the standard nurse rostering problem, since all people have equal skills and minimizing the salary costs is not an issue; the difference with the standard physician scheduling problem is that more than one employee is needed per shift. We refer to Burke et al. (2004) and to Gendreau et al. (2006) for an overview of the respective fields. We attack the problem by the method of column generation. Although this has become a traditional approach in many fields by now, it has been hardly used to solve these kinds of rostering problems. The only references we could find in the literature are the two rather inaccessible references Labit (2000) and Vovor (1997) mentioned by Gendreau et al.

The remainder of this paper is organized as follows. In Section 2 we describe the problem. In Section 3 we present our solution approach, which is based on solving an integer linear programming problem for a restricted set of variables. In Section 4 we show how to solve the corresponding LP-relaxation by applying column generation. Since solving the ILP for the restricted set of variables does not work out when we work with a whole year, we show how this can be circumvented in Section 5 by applying a rolling horizon approach. Finally, we discuss our results and draw some conclusions in the Sections 6 and 7.

Our contribution. In this paper, we present the first solution of such a basic rostering problem based on column generation. We present and apply several tricks to reduce the computational burden it takes to solve the pricing problem and the resulting ILP.

2 Problem description

We have to find a set of year-rosters for n employees, where in our application we have $n = 35$. The day is divided in three shifts: Early (7.00-15.00), Late (15.00-23.00), and Night (23.00-7.00). For each day and shift, the number of required employees is given. Weekend shifts and day shifts are not allowed to be over-booked. For each employee, we must find a roster for one year. Each roster must satisfy the following basic sequence. The employee must

1. start with working 3 to 5 early shifts;
2. have one or more days off;
3. continue with working 3 to 5 late shifts;
4. have one or more days off;
5. continue with working 2 to 3 night shifts;
6. have three or more days off.

In practice, working two early shifts, for example, is allowed but non-preferred. Since we can find good solutions with these tighter constraints, we simply do not consider any other, nonpreferred options. Except for some part-timers, each employee has a contract for 36 hours per week, but there is a tendency to roster only 34 hours. This implies 17 shifts per four-week period, which should be spread 'regularly' over the weeks. The constraints mentioned so far are the hard constraints. We have one soft constraint, which states that over-booking is preferred on the Wednesday morning to allow employees to follow a training session regularly. Concerning the quality of a roster, each employee has specified some preferences. These preferences concern:

- The parameters of the basic rotation, that is, the number of consecutive early-late-night shifts and the number of days off in between;
- Having a fixed day off each week.

There are *stand-by* shift, which will be converted in working shifts in case of illness. We ignore these when constructing the rosters, but we add them afterwards. Furthermore, it is possible to hire some outsiders. This is a non-preferred option.

3 Solution approach

We model the problem as an integer linear programming problem. Here we cannot use a model that is based on variables that correspond to decisions of the kind 'does employee j work shift i?', since the quality of a solution depends on the sequence of shifts that is worked by an employee. To cover these dependencies between the shifts worked by an

employee, we use a model that is based on selecting a year-roster for each employee. We denote the set of feasible year-rosters for employee j ($j = 1, \ldots, n$) by S_j, and we assume for the time being that we know each roster (j, s) in this set and its cost level c_{js}, which measures the dissatisfaction of the employee. A roster (j, s) is encoded through a set of indicator values a_{ijs}, where $a_{ijs} = 1$ if shift i is covered by the year-roster and $a_{ijs} = 0$, otherwise. We introduce the additional notation of using m to denote the number of shifts, and we use b_i ($i = 1, \ldots, m$) to denote the number of employees required in shift i. For each roster (j, s) we introduce a binary variable x_{js} indicating whether it gets selected for employee ($j = 1, \ldots, n$). We use the variables y_i and z_i to measure the deviation of the number of workers from the desired number of workers in shift i ($i = 1, \ldots, m$). The cost of over-booking and under-booking are measured by f_i and g_i respectively; hard constraints regarding over-booking of shift i are implemented by setting f_i to a high value.

We have to find the x_{js}, y_i, and z_i values that solve the ILP

$$\min \quad \sum_{j=1}^{n} \sum_{s \in S_j} c_{js} x_{js} + \sum_{i=1}^{m} (f_i y_i + g_i z_i)$$

subject to

$$\sum_{j=1}^{n} \sum_{s \in S_j} a_{ijs} x_{js} = b_i + y_i - z_i, \text{ for each } i = 1, \ldots, m, \tag{1}$$

$$\sum_{s \in S_j} x_{js} = 1, \text{ for each } j = 1, \ldots, n, \tag{2}$$

$$x_{js} \in \{0, 1\}, \text{ for each } s \in S_j, j = 1, \ldots, n. \tag{3}$$

$$y_i, z_i \geq 0, \text{ for each } i = 1, \ldots, m. \tag{4}$$

Note that in case of two or more identical employees, that is, with equal preferences, we can aggregate equations 2 for these employees. We will solve the ILP formulated above for a restricted set $Q_j \subset S_j$, for each $j = 1, \ldots, n$. The set Q_j will contain the best year-rosters with respect to the optimal solution of the LP-relaxation.

We obtain the linear programming relaxation by replacing conditions (3) by the conditions $x_{js} \geq 0$ for all $s \in S_j; j = 1, \ldots, n$; we do not need to enforce the upper bound of 1 for x_{js}, since this follows immediately from the conditions (2). We solve the LP-relaxation using column generation.

4 Column generation

We first solve the linear programming relaxation for a small initial subset of the columns. Given the solution to the linear programming problem with the current set of variables, it is well-known from the theory of linear programming (see for instance Bazaraa et al. (1990)) that the reduced cost of a variable x_{js} is given by

$$c'_{js} = c_{js} - \lambda_j - \sum_{i=1}^{m} \pi_i a_{ijs}$$

where $\lambda_1, \ldots, \lambda_n$ are the dual multipliers corresponding to the constraints (2) and where $\pi_1, \ldots, \pi_m$ are the dual multipliers corresponding to the constraints (1) of the solution of the current LP. If for each variable x_{js} we have that $c'_{js} \geq 0$, then the solution with the current set of variables solves the linear programming problem with the complete set of variables as well. To check whether this condition is fulfilled, we maximize the reduced cost over all feasible year-rosters. This yields the so-called pricing problem. We solve it for each employee independently, which implies that we can 'ignore' the term λ_j.

We solve the pricing problem by solving the problem of minimizing

$$\sum_{i=1}^{m} \pi_i a_{ijs} - c_{js}$$

over all feasible year-rosters (j, s) in S_j for $j = 1, \ldots, n$.

At first sight, this seems doable by solving a shortest path problem in a layered graph, where we have a layer for each shift. As the quality of a solution depends on the sequence of shifts that is worked by an employee, it becomes almost impossible to set the length of each arc or vertex such that the length of a path through this graph contains the term corresponding to the satisfaction of a year-roster. Moreover, the constraint that an employee should work approximately 34 hours per week can only be maintained at the expense of a lot of overhead, which makes the usage of such a layered network unsuitable. We have therefore decided to solve the pricing problem as a shortest path in a layered network, but each layer now corresponds to a feasible roster for a four-week period. Hence, we enumerate all feasible four-week rosters for employee j $(j = 1, \ldots, n)$; as an additional bonus, this approach enables us to fulfil the constraint that we only select year-rosters that are not 'too bad' by simply removing the four-week rosters that are above a certain maximum cost level.

We construct our layered graph by putting in each layer a vertex for each of the remaining feasible four-week rosters plus added to it the first shift succeeding this four-week roster. Hence, a feasible four-week roster can lead to several vertices. Note that we assign a cost value to this vertex that is equal to the cost of the corresponding four-week roster. Next, we connect each pair of vertices in two consecutive layers for which the combination of these two four-week rosters is feasible. Because we fix the first shift directly succeeding a four-week roster, the cost of the transition between two consecutive rosters can be included in the first four-week roster. The cost of the connecting arc in the graph corresponds to the contribution of the total value of the dual multipliers of the shifts that are worked in the four-week roster the arc originates from. Finally, we add a dummy vertex after layer 13 and we connect each vertex in layer 13 to it by an arc. Note that we do not have to adjust the graph in the next iteration; we only have to update the cost of the arcs. Since the vertices in two non-consecutive layers do not influence each other, the shortest path in this graph solves the pricing problem.

We have computed the optimum of the LP-relaxation in this way. But when we tried to solve the reduced ILP, in which we included the feasible year-rosters that we encountered when solving the LP-relaxation together with the top 2500 year-rosters per employee, our

ILP-solver CPLEX had major problems. To counter these, we apply a *rolling horizon* approach.

5 Rolling horizon approach

We solve the ILP step by step. Schematically, this works as follows.

1. Compute an optimal solution for the LP-relaxation for the first 2 periods (8 weeks);
2. Use the generated columns plus the 2500 most-preferred columns based on the dual multipliers of the optimal solution of the LP-relaxation to solve the ILP;
3. Fix the first period (four weeks) accordingly;
4. Repeat the procedure for the next two periods, given the current partial solution.

The 2500 columns added in step 2 are quite similar to those used in the optimal solution to the LP-relaxation. This is due to the fact that we used high cost terms for hard constraints with respect to over-booking: only columns resembling columns used in this optimal solution will be added. We use only two periods, so obtaining the best columns can be easily implemented by finding all shortest paths in the two-layer graph.

6 Computational results

We have first applied our approach to the real-world problem with $n = 35$. For every employee we generate all feasible four-week rosters having a preferred day off, consisting of exactly 17 shifts and having at most 3 shifts difference between the first and the last two weeks. Next, we assign to every roster a cost between 0 and 1 based on the preferences of the employee and delete every four-week roster with a cost more than 0.5. Finally we remove all four-week rosters not having enough successors or predecessors.

One layer of the graph that we use to solve the pricing problem contains approximately 1500 vertices, and between two consecutive layers there are approximately 70000 arcs. Solving the LP-relaxation for the whole year takes 100 minutes. We use CPLEX 9.1.2 to solve the LPs and ILPs; the program is coded in JAVA and runs on a machine with two dual core Intel 3.2GHz processors and 2GB of RAM. We use the outcome value as a lower bound for the quality of the solution we find through the rolling plan approach for the ILP. Solving the problem by the rolling plan approach takes 260 minutes. We allow the ILP solver to use 60 minutes for the first step of 2 periods, and 15 for the other steps where we already have the first shift fixed. The total outcome value of the rolling plan approach is 8 percent higher than the complete year LP-relaxation. The average gap between the LP-relaxation for 2 periods and the corresponding ILP is 3 percent, so only 5 percent is lost due to the rolling plan approach. The running time is dominated by solving the ILPs: we spend 240 out of 260 minutes here. In all observed cases we need less than a minute

per step before the integrality gap is less than 10 percent. This implies that we can also solve the problem in roughly 30 minutes when we allow slightly worse solutions.

The average cost of the four-week schedules used in the final solution is only 0.25, and never exceeds 0.5 because of preprocessing. As costs are normalized between 0 and 1 we conclude that the problem is solved with respect to the initial demands of the hospital: the average employee is highly satisfied, and the most unlucky employee is not too unlucky.

To test the general applicability of our approach, we have applied it to some other instances. Thereto, we have generated two other instances: one with $n = 14$ and $n = 70$, where we have changed the b_i values correspondingly. The instance with $n = 14$ suffers from feasibility problems in the ILP phase and hence is not solved for a complete year. The instance with $n = 70$ is solved in the same running time as the original problem, but with even better results: only a 1.4 percent gap remains between the complete year LP-relaxation and the rolling plan approach, where only 0.1 percent is due to the rolling plan approach.

These differences in behavior can be explained by looking at the number of non-zero values of x_{js} in the solution of the optimal LP-relaxation. This gives us the total number of columns used in this solution, and indicates how 'fractional' the optimal solution for the LP is. For the case where $n = 14$ we have 114 columns in the optimal solution of the LP-relaxation; for $n = 35$ we have 127; for $n = 70$ only 158. We observe that the optimal solution of the LP-relaxation is less fractional for higher values of n, resulting in better results for the ILP. For small values of n it is even hard to find feasible solutions sometimes; probably a constraint-satisfaction approach would be more successful for these instances.

7 Conclusions

We conclude that column generation works very well. Moreover, since we decompose the year-roster in 13 four-week rosters, which we enumerate when solving the pricing problem, we can include each additional constraint that affects no more than two consecutive four-week periods. Examples of these are

- Specific days per year off
- Vacations

If we want to include constraints that spread over a longer period, then either we need to work with additional state-variables in the computation of the shortest path, or we need to decompose the problem in sub-periods that are longer than four weeks. In both cases, we expect that this will slow down the evaluation of the pricing problem tremendously. If there are only a few of such constraints, like 'no worker has to work on both Christmas and Easter', then this will not pose any real problem.

When computing our solution, we ran into a different, non-mathematical, but extremely difficult problem: the management of the hospital decided to regroup the department, which disqualified our solutions.

Acknowledgements

The authors want to thank Guido Diepen for his support with the CPLEX package.

References

[1] M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali (1990). *Linear Programming and Network Flows*, Wiley, New York.

[2] E.K. Burke, P. de Causmaecker, G. vanden Berghe, and H. van Landeghem (2004). The state of the art of nurse rostering. *Journal of Scheduling 7,* 441–499.

[3] M. Gendreau, J. Ferland, B. Gendron, N. Hail, B. Jaumard, S. Lapierre, G. Pesant, and P. Soriano (2006). Physician scheduling in emergency rooms. E.K. Burke and H. Rudová (Eds.) *PATAT 2006*, 2–14.

[4] P. Labit (2000). *Amélioration d'une methode de génération de colonnes pour la confection d'horaire d'infirmières.* Master's thesis, École Polytechnique, Montréal, Canada.

[5] T. Vovor (1997). *Problème de chemins bicritère ou avec constraintes de ressources: algorithmes et applications* Ph.D.-thesis, École Polytechnique, Montréal, Canada.