Column Generation Method for an Agent Scheduling Problem

Balázs Dezső Alpár Jüttner Péter Kovács

Dept. of Algorithms and Their Applications, and Dept. of Operations Research
Eötvös Loránd University, Budapest, Hungary

Abstract

This paper discusses a real life problem of daily schedule planning for customer visiting agents. An optimization scheme using a combination of column generation and rounding techniques is proposed for solving this problem. In order to realize an efficient implementation, a polynomial time algorithm is presented for the column generation subproblem. Some technical implementation issues are also discussed and finally, experimental results are shown on real-life problem instances. An implementation of the presented solution is currently in production use by one of the leading contact center service providers of Hungary.

Keywords: resource planning, scheduling, column generation

1 Introduction

The problem discussed in this paper is the planning of the daily program of customer visiting agents. Consider an insurance company employing hundreds of agents in order to personally persuade potential customers to take out an insurance. To do that, they first call the potential customers by phone to make an appointment, then one of the agents will go to the place of the meeting at the right time. For the sake of flexibility, every agent makes phone calls and visits customers, but an appointment will not necessarily be assigned to the same agent who contacted the customer.

[1] Email: deba@inf.elte.hu, alpar@cs.elte.hu, kpeter@inf.elte.hu
[2] This work was supported by OTKA grant K060802

Organizing a marketing campaign of this scale is far from being trivial. The traditional way of doing it manually is typically suboptimal and very inflexible. Instead, this paper proposes an optimization method for designing the schedule for all agents at once taking the various constraints and objectives into accounts. The presented solution is currently in production use by one of the leading contact center service provider of Hungary.

A more formal definition of the optimization problem is the following. We are given a set A of business agents who have to meet customers. Let M denote the set of meeting requests. Each request has a given time, duration and location. In addition, when the agents are not visiting customers, they have to do other jobs in their office (e.g. they contact potential customers by phone). Our task is to plan the work schedule of the agents for a single day.

The workday starts when the agent arrives at the office or at place of the first meeting. Between the meetings or office shifts, they have to travel to the next location (or to the office) either by car or by public transport depending on the agent. Furthermore, the schedules must satisfy various *constraints* in order to meet the working regulations and ensure the acceptable working conditions of the agents.

- The agents must be provided with a 30-minute free lunch sometime between 12:00 and 14:00. (These parameters are in fact configurable.)
- The working time of the agents is limited.
- The idle time of the agents and the traveling time between any two locations are also limited.
- There is a given minimum allowed duration of the office work.

The *objective function* of the optimization task comprises various quality criteria of the proposed schedule. Namely,

- The productive activities of the agents should be maximized, therefore negative cost is assigned to the meetings and the office work. Meetings have a priority over office work, hence the cost of them is less.
- The traveling time should also be minimized, therefore an extra penalty is assigned to traveling times.
- We prefer bigger blocks of the same working type, thus a penalty is applied when an agent switches between office work and meeting participation.

The rest of the paper is organized as follows. Section 2 presents an ILP formulation of the problem (of exponential size) and also defines an LP relaxation of it. Then Section 3 discusses how to solve this enormous ILP problem

efficiently. Finally, Section 4 summarizes some practical experiences about the performance of the solution.

2 ILP Formulation

This section presents a column generation based heuristic algorithm for the optimization problem. The column generation approach is extensively used in resource planning problems [1].

The master problem (MP) is formalized as an integer program of exponential size. Let A be the set of the agents and V be the set of visiting requests. Let S_a be the set of the feasible schedules of the agent a, and c_{s_a} is the cost of a schedule $s_a \in S_a$. Then the integer programming formulation is the following.

$$\min \quad \sum_{a \in A} \sum_{s_a \in S_a} c_{s_a} x_{s_a} \tag{1a}$$

$$\text{s. t.} \quad \sum_{s_a \in S_a} x_{s_a} = 1 \qquad \forall a \in A \tag{1b}$$

$$\sum_{m \in s_a} x_{s_a} = 1 \qquad \forall m \in M \tag{1c}$$

$$x_{s_a} \in \{0, 1\} \qquad \forall a \in A, s_a \in S_a \tag{1d}$$

The cost c_{s_a} can be calculated from the properties of the schedule. If the agent travels in t_t minutes, waits for meetings in t_r minutes and works in t_w minutes in the office, moreover there are n_c changes between meetings and office shifts, then we can compute the overall cost with the following expression:

$$c_{s_a} = C_t t_t + C_r t_r - B_w t_w + C_c n_c \tag{2}$$

We assign a binary variable x_{s_a} to each possible schedule s_a of each agent a. The constraints enforce that each meeting is fulfilled by exactly one agent and each agent has exactly one work schedule.

In addition, let RMP denote the following linear programming relaxation of the above formulation with the introduction of slack variables for constraints (1c) in order to ensure that the problem is always feasible.

$$\min \quad \sum_{a \in A} \sum_{s_a \in S_a} c_{s_a} x_{s_a} + \sum_{m \in M} H y_m \tag{3a}$$

$$\text{s. t.} \quad \sum_{s_a \in S_a} x_{s_a} = 1 \qquad \forall a \in A \tag{3b}$$

$$\sum_{v \in s_a} x_{s_a} + y_m = 1 \qquad \forall m \in M \tag{3c}$$

$$x_{s_a} \geq 0 \qquad \forall a \in A, s_a \in S_a \tag{3d}$$

$$y_m \geq 0 \qquad \forall m \in M \tag{3e}$$

where H is a sufficiently large number. The dual linear program of RMP is the following.

$$\max \quad \sum_{a \in A} w_a + \sum_{m \in M} z_m \tag{4a}$$

$$\text{s. t.} \quad w_a + \sum_{m \in s_a} z_m \leq c_{s_a} \qquad \forall a \in A, \forall s_a \in S_a \tag{4b}$$

$$y_m \leq H \qquad \forall m \in M \tag{4c}$$

$$w_a \in \mathbb{R} \qquad \forall a \in A \tag{4d}$$

$$z_m \in \mathbb{R} \qquad \forall m \in M \tag{4e}$$

3 Solution Method

The proposed solution is based on an iterative rounding of the LP relaxation of the above formulation. We fix the schedule of one agent in each iteration. For this, (a) we find an (approximate) solution to RMP, then based on this solution, we choose an agent and (b) fix a schedule for her by rounding the corresponding fractional variables x_{s_a}, $s_a \in S_a$. Then we delete this agent and the requests covered by her schedule from the problem and repeat steps (a) and (b) until we find a schedule for each agent.

3.1 Solving RMP

To deal with the fact that the size of RMP can be enormous (exponential in the number of the meeting requests), we use the usual column generation approach [6], as briefly described below.

Instead of keeping the whole problem in the memory, we maintain a reasonably sized subset of the columns, i.e. a limited number of possible schedules $S'_a \subset S_a$ for each agent $a \in A$. Then we find a solution x_{s_a} to this subset of RMP together with the corresponding dual solution w_a, y_m. Now, we check whether (4b) holds for all agents $a \in A$ and for all schedules $s_a \in S_a$. If yes, then x_{s_a} is in fact the optimal solution. If not, then we add the column corresponding the failed dual constraint to the problem and iterate the above steps until the optimum is found or a certain time limit is reached.

Therefore the core of this scheme is the subroutine that checks the feasibility of a dual solution and finds a failed constraint if it is not feasible.

3.1.1 Column Generation with Dynamic Programming Method

During the algorithm, we have to find feasible schedules which violate the inequality (4b). It can be done by searching the schedule with the smallest reduced cost for agent a, i.e.

$$\min c_{s_a} - w_a + \sum_{m \in s_a} z_m \tag{5}$$

The problem can be formulated as finding a resource constrained shortest path. Let us define a directed graph $G = (V, E)$ as follows. We assign a node in the graph to each meeting request. Between two meetings m_1 and m_2, the agents can do only a couple of different things. We add a directed edge $m_1 \longrightarrow m_2$ to each of these actions whenever it is feasible (parallel edges are allowed), and assign the cost corresponding to the action. Namely, the choices are the following.

- The agent can go directly to the place of m_2.
- If there is enough time between the meetings, she can go to the office for internal work.
- The agent can also have a lunch time between the meetings.

So, the traveling time is calculated between every pair of meetings m_1 and m_2, both directly and through the office location. The travel time has to fit in to the time gap, and the travel lengths must be shorter than a threshold value. Moreover, if the agent goes to the office, she has to stay there for at least a certain time period, which is necessary to do effective work.

Each path in this graph almost determines a schedule, but the periods before the first and after the last requests are not fixed. Therefore additional edges are assigned to the graph which belong to the exterior schedule. The schedules must consist of an internal path and from an external edge between the same starting and finishing nodes.

A cost function $c : E \to \mathbb{R}$ is defined on the edges, which can be calculated based on equation (2). The cost of a schedule can be determined as the total cost on the corresponding path and external edge. To get the reduced cost of a schedule, we have to subtract the dual values z_m from the path cost.

The agents' working time must include the lunch time. Thus we assign another function $r : E \to \{0, 1\}$ to the edges depending on whether it includes a lunch period (the external edge may include a lunch period, as well).

Claim 3.1 *The optimal schedule is the minimum cost feasible combination of the internal paths and external edges for all pairs of nodes.*

We use label setting resource constrained shortest path algorithm [4] for finding a schedule with minimum cost and exactly one lunch period. Since both an internal path and an external edge have to be find, we calculate the shortest paths on the internal edges between each pair of nodes in the graph. Note that the edges are directed from an earlier event to a later one, thus the walks obtained by the label setting algorithm are simple paths.

The label setting method is not polynomial in general, but in our case, the resource function has only two values, which ensures the polynomiality.

Claim 3.2 *The dynamic programming algorithm for column generation runs in polynomial time.*

If n is the number of the meeting requests, then the graph can have $O(n^2)$ edges. Therefore the shortest paths from an arbitrary starting node can be calculated in $O(n^2)$ time. This yields $O(n^3)$ total running time for finding the optimal schedule.

3.2 Rounding Phase

Once we have a solution for RMP, the rounding phase is easy. We simply evaluate each of the K largest primal variables, and compute the increase in the cost function when this variable is set to 1 and all others for the corresponding agent are set to 0. We choose the one resulting in the least increment and fix the corresponding schedule s_a for the agent a. Then we remove agent a and all covered requests from the problem and repeat the column generation and rounding phases until the schedule of each agent is fixed. Similar greedy rounding technique is used in airline crew scheduling, as well [2].

3.3 Implementation Details

This optimization problem came from the real life and the proposed solution is currently in production use. To develop a reliable and efficient solution, a couple of technical issues must be dealt with.

Because of running time efficiency, the algorithm was implemented in C++ language, and several third party libraries were used. For example, various minor tools of LEMON [5] turned out to be very useful, as well as its high-level and solver independent LP and MIP interface. The GLPK LP solver library [3] was used as the back-end.

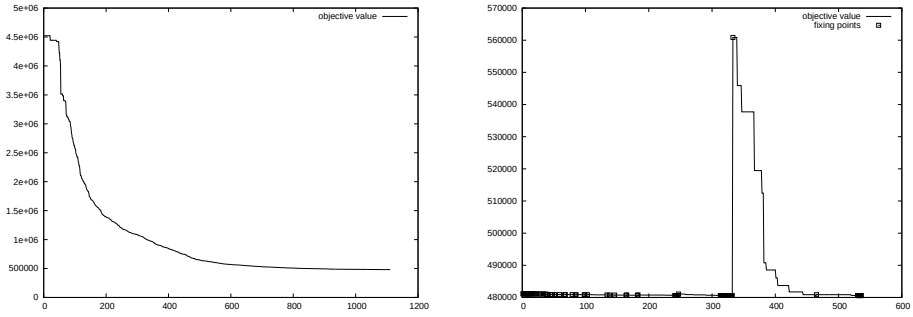
Estimating the travel times between the geographical locations is a major

difficulty in practice. Accurate values can be obtained using on-line route planner services, but it is a costly operation, both in terms of money and in terms of time. Therefore a mixed strategy was developed for this purpose. First, a simple straight line distance based estimation is used in the dynamic programming algorithm (using a pessimistic calibration), but the accurate travel times are queried when a column is added to the problem. Moreover, once a travel time between two locations has been queried, this value is stored and it will be used even in the dynamic programming method later on.

4 Experimental Results

The algorithm has been evaluated with respect to the running time and objective value aspects using 10 real-life input instances. In these test cases, the number of the agents is between 75 and and 95, and the number of meeting requests is between 100 and 185.

Fig. 1(a) shows the change of the objective value in the improvement phase. For numerical comparison, the optimal solution of the relaxed problem was also calculated. Our experiments show that the gap between this value and the obtained integer solution is quite small. It also means that the optimal fractional and integer solutions are usually close to each other for this problem.



(a) Improvement phase (b) Rounding phase

Fig. 1. Change in the objective value

Fig. 1(b) shows a typical result of the rounding phase. Most of the fixations do not raise the objective value, therefore they follow each other directly. On the other hand, some fixations increase the objective value significantly, but it can be decreased back close to its original value by some improvement steps.

The algorithm was tuned by setting the maximum number of generated new columns per rounding phase in order to meet the customer's request. For usual input sizes, the results are obtained within a couple of minutes. This running time is dominated by the web service access.

5 Concluding Remarks

During the real life application of the presented method, various new requirements was requested by the customer. We conclude the paper by shortly sketching how these extensions could be integrated into the original framework.

A general goal is that a balanced schedule has to be achieved, i.e. both office and visit shifts should be assigned to each agent. Additionally, the experienced agents should get more meetings in a day, while at most one visit should be assigned to the beginners. These requirements are assumed to be soft constraints, thus they are encoded into the objective function.

Such constraints can be handled easily with introducing new resources in the dynamic programming algorithm. The number of meetings and the existence of an office shift are registered in the labels. If the meeting number exceeds or does not reach the limit, then the cost of the schedule is increased proportionally to the deficit or the surplus. Moreover, if the agent does not have office or meeting shift, an additional constant is added to the schedule cost value. Since the number of meetings is limited by a small constant, which is independent of the number of agents and requests, the modified algorithm is also polynomial.

References

[1] Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh and P. H. Vance, *Branch-and-price: Column generation for solving huge integer programs*, Oper. Res. **46** (1998), pp. 316–329.

[2] Borndörfer, R., U. Schelten, T. Schlechte and S. Weider, *A column generation approach to airline crew scheduling*, Technical report, Konrad Zuse Institute, Berlin (2005).

[3] *GLPK – GNU Linear Programming Kit*, http://www.gnu.org/software/glpk/ (2009).

[4] Irnich, S. and G. Desaulniers, *Shortest path problems with resource constraints*, Column Generation (2005), pp. 33–65.

[5] *LEMON – Library for Efficient Modeling and Optimization in Networks*, http://lemon.cs.elte.hu/ (2009).

[6] Lübbecke, M. E. and J. Desrosiers, *Selected topics in column generation*, Oper. Res. **53** (2005), pp. 1007–1023.