The Manpower Allocation Problem with Time Windows and Job-Teaming Constraints

Anders Dohn and **Esben Kolind** and **Jens Clausen**
Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads, 2800 Lyngby, Denmark
Corresponding author (Anders Dohn): adh@imm.dtu.dk

Abstract

The Manpower Allocation Problem with Time Windows, Job-Teaming Constraints and a limited number of teams (m-MAPTWTC) is a crew scheduling problem faced in several different contexts in the industry. The number of teams is predetermined and hence the objective is to create a schedule that will maximize utilization by leaving as few tasks uncompleted as possible. The schedule must respect working hours of the teams, transportation time between locations, and skill requirements and time windows of the tasks. Furthermore, some tasks are completed by multiple cooperating teams. Cooperating teams must initiate work simultaneously and hence this must be maintained in the schedule. The problem is solved using column generation and Branch-and-Bound. Optimal solutions are found in 11 of 12 test instances originating from real-life problems. The paper illustrates a way to exploit the close relations between scheduling and vehicle routing problems. The formulation as a routing problem gives a methodological benefit, leading to optimal solutions. A constraint on synchronization is imposed and successfully dealt with in the branching scheme.

Introduction and Problem Description

The Manpower Allocation Problem with Time Windows, Job-Teaming Constraints and a limited number of teams (m-MAPTWTC) is the problem of assigning m teams to a number of tasks, where both teams and tasks may be restricted by time windows outside which operation is not possible. Tasks may require several individual teams to cooperate. Due to the limited number of teams, some tasks may have to be left unassigned. The objective is to maximize the number of assigned tasks.

The problem arises in various crew scheduling contexts where cooperation between teams/workers, possibly with different skills, is required to solve tasks. An example is the home care sector, where the personnel travel between the homes of the patients who may demand collaborative work (e.g. for lifting). The problem also occurs in hospitals where a number of doctors and nurses are needed for surgery and the composition of staff may vary for different tasks. Another example is in the allocation of technicians to service jobs, where a combination of technicians with individual skills is needed to solve each task.

This study focuses on the scheduling of ground handling tasks in some of Europe's major airports. Between arrival and the subsequent departure of an aircraft, numerous jobs including baggage handling and cleaning must be performed. Typically, specialized handling companies take on the jobs and assign crews of workers with different skills. Any daily work plan must comply with the time windows of tasks, the working hours of the staff, the skill requirements of tasks, and union regulations. It may be necessary to have several teams cooperating on one task in order to complete it within the time window. The workload has to be divided equally among the cooperating teams. Furthermore, all teams involved must initiate work on the task simultaneously (synchronized cooperation), as only one of the team leaders is appointed as responsible supervisor. In the remainder of this paper, a team is a fixed group of workers, whereas when referring to job-teaming or cooperation, we refer to a temporary constellation of teams joined together for a specific task. In the airport setting, all tasks require exactly one skill each.

The problem instances of this paper are currently solved by a software tool built on a Simulated Annealing heuristic. An efficiency increase of up to 20% compared to a manual planning approach has been reported. Unfortunately, a comparison to our approach is not possible, as the details of the heuristic are confidential.

MAPTWTC has previously been treated by Lim, Rodrigues, & Song (2004) and Li, Lim, & Rodrigues (2005) in a metaheuristic approach. They study an example originating from the Port of Singapore, where the main objective is to minimize the number of workers required to carry out all tasks, rather than carrying out the maximum number of tasks with a given workforce. Both papers describe secondary objectives as well.

Our problem is closely related to the Vehicle Routing Problem with Time Windows (VRPTW) which has been studied extensively in the literature. Look to (Selensky, Prosser, & Beck 2006) for a generic comparison of vehicle routing and job shop scheduling formulations.

The most promising recent results for exact solution of VRPTW problems use column generation. Column generation for VRPTW was initiated by Desrochers, Desrosiers,

& Solomon (1992). They solve the pricing problem as a *Shortest Path Problem with Time Windows (SPPTW)*. Their approach proved to be very successful and has recently been applied with success by e.g. Righini & Salani (2006), and Irnich & Villeneuve (2006).

Recently, Feillet *et al.* (2004) suggested solving the pricing problem as an *Elementary Shortest Path Problem with Time Windows (ESPPTW)* building on the ideas of Beasley & Christofides (1989). Chabrier (2006), Danna & Pape (2005), and Jepsen *et al.* (2006) have extended the ideas and achieved very promising results.

The remainder of this paper is structured as follows. First, we present the problem definitions of m-MAPTWTC. Next, a Dantzig-Wolfe decomposition is presented, where the problem is decomposed into a master problem and a pricing problem. This decomposition allows us to solve the problem using column generation in a Branch-and-Price framework. In the following section, the necessary branching rules are described. This includes branching to enforce integrality as well as synchronized cooperation on tasks. The computational results on a number of real-life problems are presented next.

This conference paper is a compact version of the technical report (Dohn, Kolind, & Clausen 2007).

Problem Definitions

Definition of m-MAPTWTC

Consider a set $C = \{1, \ldots, n\}$ of n tasks and a workforce of inhomogeneous teams V. Each task has a number of attributes including a duration, a time window, a set of required skills, and a location. We model the attributes in the following way. For each task $i \in C$ a time window is defined as $[a_i, b_i]$ where a_i and b_i are the earliest and the latest starting times for task i, respectively. r_i is the number of teams required to fully complete task i (Task i is divided into r_i *split tasks*). Between each pair of tasks (i, j), we associate a time t_{ij} which contains the transportation time from i to j and the service time at task i. Further, g_{ik} is a binary parameter defining whether team k has the required qualifications for task i ($g_{ik} = 1$) or not ($g_{ik} = 0$).

Each team $k \in V$ also has a time window $[e_k, f_k]$, where the team starts at the service center at time e_k and must return no later than f_k. There exists only one service center, and all teams begin their shift at this location. We refer to the service center as location 0. We define the set $N = C \cup \{0\} = \{0, \ldots, n\}$. The transportation time from the service center to each task i is denoted t_{0i}. The service time of task i plus transportation time from task i to the service center is t_{i0}.

We assume that a_i, b_i, e_k, and f_k are non-negative integers and that each t_{ij} is a positive integer. We also assume that the triangular inequality is satisfied for t_{ij}. The assumptions on t_{ij} are naturally fulfilled in all problem instances as t_{ij} includes service time at task i.

Relations to Vehicle Routing

As mentioned earlier, m-MAPTWTC is closely related to VRPTW. Consider the teams as vehicles driving from one customer to another as they in m-MAPTWTC move from one task to another. The service that the teams deliver is an amount of their time, unlike the vehicles that deliver goods which have taken up a part of the total volume. Hence, in that sense m-MAPTWTC is uncapacitated. Except for the binding between teams inflicted by the possibility of cooperation on tasks, the problem is similar to the Uncapacitated Vehicle Routing Problem with Time Windows and a limited number of vehicles (m-VRPTW).

Column generation has proven a successful technique for exact solution of VRPTW and hence the solution procedure in this paper is built on the principles of column generation in a Branch-and-Bound framework (a so called Branch-and-Price algorithm).

Decomposition

We present the Dantzig-Wolfe decomposition (Dantzig & Wolfe 1960) of m-MAPTWTC. First, we introduce the notion of a *path*. A feasible path is defined as a shift starting and ending at the service center, obeying time windows and skill requirements, but disregarding the constraints dealing with interaction between shifts. By this definition the feasibility of a path can be determined without further knowledge about other paths. We define $\mathcal{P}_k$ as the set of all feasible paths for team $k \in V$. Each path is defined by the tasks it visits. Let $a_{ik}^p = 1$ if task i is on path p for team k and $a_{ik}^p = 0$ otherwise.

Master Problem

In the *Integer Master Problem* we solve the problem of optimally choosing one feasible path for each team, maximizing the total number of assigned tasks. In the master problem, the set $\mathcal{P}_k$ is used to guarantee this feasibility. We are in this model not able to enforce synchronized cooperation and this hence has to be enforced by the branching scheme. We choose to consider the problem as a minimization problem by introducing δ_i as the number of unassigned split tasks of task i. Finally, to decrease the size of the problem, a set of promising paths $\mathcal{P}'_k$ ($\subseteq \mathcal{P}_k$) is used instead of $\mathcal{P}_k$. In a column generation context $\mathcal{P}'_k$ contains all paths generated for team k in the pricing problem so far. We introduce the binary decision variable λ_k^p, which for each team k is used to select a path p from $\mathcal{P}'_k$. To be able to solve the model efficiently λ_k^p and δ_i are LP-relaxed. We arrive at the *Restricted Master Problem (RMP)*:

$$\min \sum_{i \in C} \delta_i \tag{1}$$

$$\delta_i + \sum_{k \in V} \sum_{p \in \mathcal{P}'_k} a_{ik}^p \lambda_k^p \geq r_i \qquad \forall i \in C \tag{2}$$

$$\sum_{p \in \mathcal{P}'_k} \lambda_k^p = 1 \qquad \forall k \in V \tag{3}$$

$$\lambda_k^p \geq 0 \qquad \forall k \in V, \forall p \in \mathcal{P}'_k \tag{4}$$

$$\delta_i \geq 0 \qquad \forall i \in C \tag{5}$$

The sum of δ_i over all tasks is minimized (1). (2) penalizes inadequate assignment to a task by incrementing δ_i

sufficiently. $a_{ik}^p \lambda_k^p$ is larger than 0 only when paths containing task i are chosen. E.g. if only one path containing task i is selected, $\sum_{k \in V} \sum_{p \in \mathcal{P}'_k} a_{ik}^p \lambda_k^p = 1$ and if the requsted number was 2 ($r_i = 2$), we set $\delta_i = 1$, to satisfy (2), and hence a contribution to the objective function of 1 is induced. $\sum_{k \in V} \sum_{p \in \mathcal{P}'_k} a_{ik}^p \lambda_k^p$ and δ_i may also be fractional as we have relaxed the integrality constraints.

(3) ensures that exactly one path is selected for each team. (4) and (5) are non-negativity constraints on our decision variables. This formulation allows tasks to be done more times than required, which is useful in a column generation setting, where an existing column may enter the solution basis, and we do not have to generate a new, almost identical column containing a subset of the tasks. As a consequence, the estimates of the final dual variables improve (see Kallehauge *et al.*, 2005). The master problem has the form of a generalized set-covering problem.

On the downside, any solution may contain *overcovering*, i.e. we may have tasks which are assigned to more teams than requested. However, in this formulation, overcovering can be removed without altering the objective value by unassigning the superfluous number of teams for each task. In the case of overcovering of task i we avoid the unassignment-penalty (i.e. $\delta_i = 0$), but the additional team assignments to the task do not improve the objective value further as δ_i is a non-negative variable. Hence, by removing all but r_i of the assignments, the objective value remains unchanged (we still have: $\delta_i = 0$). The modified solution is still feasible and the overcovering can hence easily be removed from an optimal solution.

If the master problem contains no columns representing paths from the outset of the column generation procedure, the problem will be infeasible due to the team constraints (3). Therefore, we add an *empty path* λ_k^0 ($a_{ik}^0 = 0, \forall i \in C$) for each team to ensure feasibility whether regular paths are present or not.

The solution to the restricted master problem may not be integer. In addition, we have relaxed the constraint on synchronization of tasks. Both of these properties must be enforced by a branching scheme.

The solution to the restricted master problem is not guaranteed to be optimal either, since only a small subset of feasible paths is considered. For each primal solution λ to the restricted master problem we obtain a dual solution $[\pi, \tau]$, where π and τ are the dual variables of constraints (2) and (3), respectively.

The dual variables express, for each constraint, the marginal value of decreasing the right hand side of the constraint, e.g. π_i gives the marginal value of decreasing the right hand side of the corresponding constraint of type (2). The value of decreasing the right hand side is always the same as the value of increasing the left hand side. Marginally, we can hence gain π_i by adding a path containing task i. At the same time, we would have to devote one of the teams k to this task, at the cost of τ_k.

In column generation, the dual solution is used in the pricing problem to ensure the generation of columns leading to an improvement of the solution to the master problem. In our case, the gain of including the tasks in the path (the sum over π_i for all tasks i in the path) must be larger than the cost of moving the team from the route they would otherwise be assigned to (τ_k).

Pricing Problem

The *pricing problem* specifies all the requirements of a feasible path. The objective is to find the path with the lowest possible reduced cost. In m-MAPTWTC with inhomogeneous teams as described above, we obtain $m = |V|$ separate pricing problems. Each pricing problem is an *Elementary Shortest Path Problem with Time Windows (ESPPTW)*. The binary variable x_{ij} is defined as $x_{ij} = 1$ if the team goes directly from task i to task j and $x_{ij} = 0$ otherwise. s_i is an integer variable and defines the start time of task i. We only include tasks where $\pi_i > 0$ as other tasks are certain not to be in an optimal path. Further, we only consider tasks where the team has the required skill and where team time windows and respective task time window have an overlap. If this is not the case, such a task is not in any feasible path. With the unwanted tasks removed, we have the new sets: $C_{k'}$ and $N_{k'}$. For a team $k' \in V$ the pricing problem is formulated as:

$$\min \sum_{i \in C} \sum_{j \in C} -\pi_i x_{ij} - \tau_{k'} \tag{6}$$

$$\sum_{j \in N} x_{0j} = 1 \tag{7}$$

$$\sum_{i \in N} x_{ih} - \sum_{j \in N} x_{hj} = 0 \qquad \forall h \in N_{k'} \tag{8}$$

$$e_{k'} + t_{0j} - M(1 - x_{0j}) \leq s_j \qquad \forall j \in C_{k'} \tag{9}$$

$$s_i + t_{i0} - M(1 - x_{i0}) \leq f_{k'} \qquad \forall i \in C_{k'} \tag{10}$$

$$s_i + t_{ij} - M(1 - x_{ij}) \leq s_j \qquad \forall i \in C_{k'}, \forall j \in C_{k'} \tag{11}$$

$$a_i \leq s_i \leq b_i \qquad \forall i \in C_{k'} \tag{12}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in N_{k'}, \forall j \in N_{k'} \tag{13}$$

$$s_i \in \mathbb{Z}^+ \cup \{0\} \qquad \forall i \in C_{k'} \tag{14}$$

The objective (6) is to minimize the reduced cost of the path as dictated by column generation theory. We have to ensure that all shifts start at the service center (7). Constraints (8) ensure that no shifts are segmented. Any task visited by a team must be left again. The next four constraints deal with the time windows. First, we ensure that a team can only be assigned to a task during their working hours (9)-(10). Next, we check if the time needed for traveling between tasks is available (11). If a customer i is not visited, the large scalar M makes the corresponding constraints non-binding. Constraints (12) enforce the task time windows. Finally, constraints (13)-(14) are the integrality constraints. The introduction of a service start time removes the need for sub-tour elimination constraints, since each customer can only be serviced once during the scheduling horizon because t_{ij} is positive.

The pricing problem can be perceived as a graph problem. Consider a graph $G(N_G, E_G, c, t)$, where the nodes N_G are all tasks plus the service center and E_G is the set of edges connecting all nodes. With each edge $e \in E_G$ is associated a travel time $t_e = t_{ij}$ and a cost $c_e = c_{ij} = -\pi_i$, where i and j are the two nodes connected by e. To simplify, the service center is usually split into two vertices: a start vertex 0 and an end vertex $n+1$. The objective is to find a path in G from 0 to $n+1$ with a minimum sum of edge costs that does not violate any time windows.

Solution methods to the Shortest Path Problem with Time Windows have been studied extensively in the literature and successful algorithms for solving SPPTW have been built on the concept of dynamic algorithms. We solve the elementary version of the problem (ESPPTW), where no cycles are allowed. Dror (1994) proves that the problem is NP-hard in the strong sense and thus no pseudo-polynomial algorithms are likely to exist. We use a label setting algorithm built on the ideas of Chabrier (2006) and Jepsen *et al.* (2006). The authors of both papers have recently succeeded in solving previously unsolved VRPTW benchmarking instances (from the Solomon Test-sets, Solomon, 1987) by ESPPTW-based column generation. Furthermore, Feillet, Dejax, & Gendreau (2005) and Feillet *et al.* (2004) address the Vehicle Routing Problem with Profits (similar to the Vehicle Routing Problem with a limited number of vehicles) and state that solving the elementary shortest path problem as opposed to the relaxed version is essential to obtain good bounds.

We will not go into the details of the label setting algorithm, since the problem is almost identical to the pricing problem of VRPTW. We have a shortest path problem where all arc costs out of a node are identical and hence can be moved to the node. The pricing problems are first solved in a heuristic label setting approach and if no columns can be added, we switch to the exact label setting algorithm.

Branching

When an optimal solution to the relaxed master problem is reached, and when the solution is not feasible in the original problem, branching is applied. When branching, the relaxed solution space is split into two disjunctive subspaces. The current infeasible solution is excluded from the two subspaces, possibly along with other infeasible solutions. All feasible solutions remain in one subspace or the other. The branching is carried out in the master problem. Branching decisions are transmitted to the pricing problem when they have an impact.

Branching to Get Integral Solutions

Various branching strategies for VRPTW have been proposed. See (Kallehauge *et al.* 2005) for a more thorough review of branching strategies for VRPTW.

We focus on a 0-1 branching on $\sum_j x_{ijk}$. For a chosen (i,k), each branching decision fixes $\sum_j x_{ijk} = 0$ and $\sum_j x_{ijk} = 1$, respectively. This implies that team k is either forced to or banned from completing task i. In the pricing problem, the node corresponding to task i is either removed from the network (along with *all* arcs incident to it) or given a very low (negative) cost to ensure its inclusion in any optimal solution.

Synchronized Cooperation Using Branching

Consider an optimal solution to the relaxed master problem, fractional or integral, and let s_i^p be the point in time where execution of task i begins on path p (if i is not a part of p, s_i^p is irrelevant). The solution violates the synchronized cooperation constraint for some task i if there exist positive variables $\lambda_{k_1}^{p_1}$ and $\lambda_{k_2}^{p_2}$ associated with the two paths p_1 and p_2 ($p_1 \neq p_2$), both containing i where

$$s_i^{p_1} \neq s_i^{p_2}$$

If the solution is fractional, the teams k_1 and k_2 may be identical.

Define $s_i^* = \lceil (s_i^{p_1} + s_i^{p_2})/2 \rceil$ as the *split time*. Now, split the problem into two branches and define new time windows for task i as

$$[a_i; s_i^* - 1] \qquad \text{and} \qquad [s_i^*; b_i]$$

respectively. Existing columns not satisfying the new time windows are removed from the corresponding child nodes and new columns generated must also respect the updated time window. In this way, the current solution is cut off in both branches and the new subspaces are disjoint. Since time has been discretized the branching strategy is guaranteed to be complete.

The idea behind this branching scheme is to restrict the number of points in time, where the execution of task i can begin. If the limited time window makes it inconvenient for the teams to complete task i, the lower bound will increase and the branch is likely to be pruned at an early stage. On the other hand, if the limited time window contains an optimal point in time for the execution of task i, it may be necessary to continue the time window branching until a singleton interval is reached. However, since the label setting algorithm for the pricing problem aims at placing tasks as early as possible (see Desrochers, Desrosiers, & Solomon, 1992), the actual number of different positions in time for any task is rather small. In fact, as the time windows are reduced, the tasks are more and more likely to be placed at the very beginning of their time window. This property greatly reduces the number of branching steps needed.

Using time window branching, the solution will eventually become feasible with respect to the synchronized cooperation constraint. It is not guaranteed to be integral, though, and it may therefore be necessary to apply the regular $\sum_j x_{ijk}$ branching scheme, branching on a combination of a task and a team. As both schemes have a finite number of branching candidates, the solution algorithm will terminate when they are used in combination. In general, when none of the feasibility criteria (integrality and synchronized cooperation) are fulfilled, we have a choice of branching scheme.

Our algorithm has been set to use time window branching whenever applicable. The restricted time windows reduce flexibility in the column generation which, in turn, limits the possibilities of combining fractional columns when solving the master problem. Thus, time window branching is also expected to have a positive influence on the integrality of the solution as observed by Gélinas *et al.* (1995) for VRPTW. This property has also been observed in practice when testing the algorithm, hence the choice of prioritizing time window branching.

We now focus on how good branching candidates are selected for branching. Let P_i be the set of all paths p including task i with $\lambda_k^p > 0$ in the current solution of the restricted master problem. If

$$s_i^{p_1} \neq s_i^{p_2}$$

for any two paths $p_1, p_2 \in P_i$, task i is stored in the set C' of possible candidates. We determine the split time as

$$s_i^* = \left\lceil \frac{\min_{p \in P_i} (s_i^p) + \max_{p \in P_i} (s_i^p)}{2} \right\rceil, \forall i \in C'$$

When ranking the branching candidates, we prefer candidates that provide a balanced search tree. That is, the paths in P_i should be divided equally into the two child nodes when weighted according to the variable values λ_k^p. Define

$$S_i = \sum_{k \in V, p \in P_i} \lambda_k^p, \forall i \in C'$$

as the sum of all positive variables containing i and let

$$S_i^< = \sum_{p \in P_i | s_i^p < s_i^*, k \in V} \lambda_k^p, \forall i \in C'$$

be the same sum restricted to the variables where task i is executed before the split time. The branching candidate i^* is now determined by

$$i^* = \arg\min_{i \in C'} \left| \frac{S_i^<}{S_i} - 0.5 \right|$$

Computational Results

The Branch-and-Price algorithm has been implemented in the Branch-and-Cut-and-Price framework of COIN-OR (Coin 2006) and tests have been run on 2.7 GHz AMD processors with 2 GB RAM. The implementation has been tuned to the problems at hand and parameter settings have been made on the basis of these problems. The algorithm is set to do strong branching (Achterberg, Koch, & Martin 2005) with 25 branching candidates. Strong branching is a technique that facilitates the deployment of beneficial branching decisions. In this case, 25 possible branching decisions are investigated. By intermediate calculations each branching decision is evaluated and the most promising one is used. Such evaluations are time consuming, but they tend to decrease the size of the branch-and-bound tree enough to make it worth while. Up to 10 columns with negative reduced cost are added per pricing problem.

The test data sets originate from real-life situations faced by ground handling companies in two of Europe's major airports. This gives rise to four different problem types, since the two airports each produce problems of two distinctive types. Each type is represented by three problem instances, each spanning approximately one 24-hour day, thus, a total of 12 test instances are available.

Generally, the four problem types can be summarized as (In brackets: The total number of tasks after splitting into requested split tasks):

Type A Small instances, Airport 1.
12-13 teams and 80 (120) tasks

Type B Medium instances, Airport 2.
27 teams and 90 (150) tasks.

Type C Small instances, Airport 2.
15 teams and 90 (110) tasks.

Type D Large instances, Airport 1.
19-20 teams and 270 (300) tasks.

The problem instance **A.1** and its optimal solution is illustrated in Figure 1. The figure depicts the distribution of tasks over the day and the skill requirements for these. The execution time of tasks and the length of their time windows are similar in the other problem types. In our problem instances, each team must be given a predefined number of breaks during their day and within certain time windows. Breaks are treated as regular tasks, with the exceptions that they can only be assigned to the related team, and they cannot be left unassigned in a feasible solution.

The individual schedules of the teams are captured in the 13 boxes, which clearly show the start and end time of each shift. Each task is represented by one or more small boxes labeled with the task ID (Breaks have ID: "BR"). The superscript denotes the number of teams that the task must be split between. This number therefore corresponds to the total number of boxes labeled with the task ID of this task. Above each task is a thin box depicting the time window of the task. Furthermore, each task has a color pattern revealing its skill requirement. Each team has between one and three skills, identified by the small squares to the left of the team ID. To assign a task to a team, the color pattern of the task must match that of one of these small squares.

To illustrate how to read the figure, we go through the work plan of team 9. The first task carried out is task 6 which requires skill C. The task is scheduled from 6:10 to 7:10 and hence the time window of the task is respected, since execution cannot start before 6 o'clock and must be finished by 7:30. The task is completed in collaboration with team 6. The light gray box in front of the task gives the required travel time. Next, the team takes care of task 52 (requires skill A), this time cooperating with team 7. After this, team 9 is given their daily break. Subsequently, they will carry out 71, 49, and 22, where task 49 and task 22 are dealt with by team 9 alone.

In Table 1 the results from the 12 datasets are given. From the table we conclude the following. 6 of the 12 datasets were solved to optimality within one hour. The remaining 6 instances are split in two cases: one case for the small and medium-sized problems (**Type A-C**) and one case for the large instances (**Type D**). For the unsolved problems of **Type A-C** we see an explosion in the size of the branching tree. In these cases the time-out limit is never reached, since

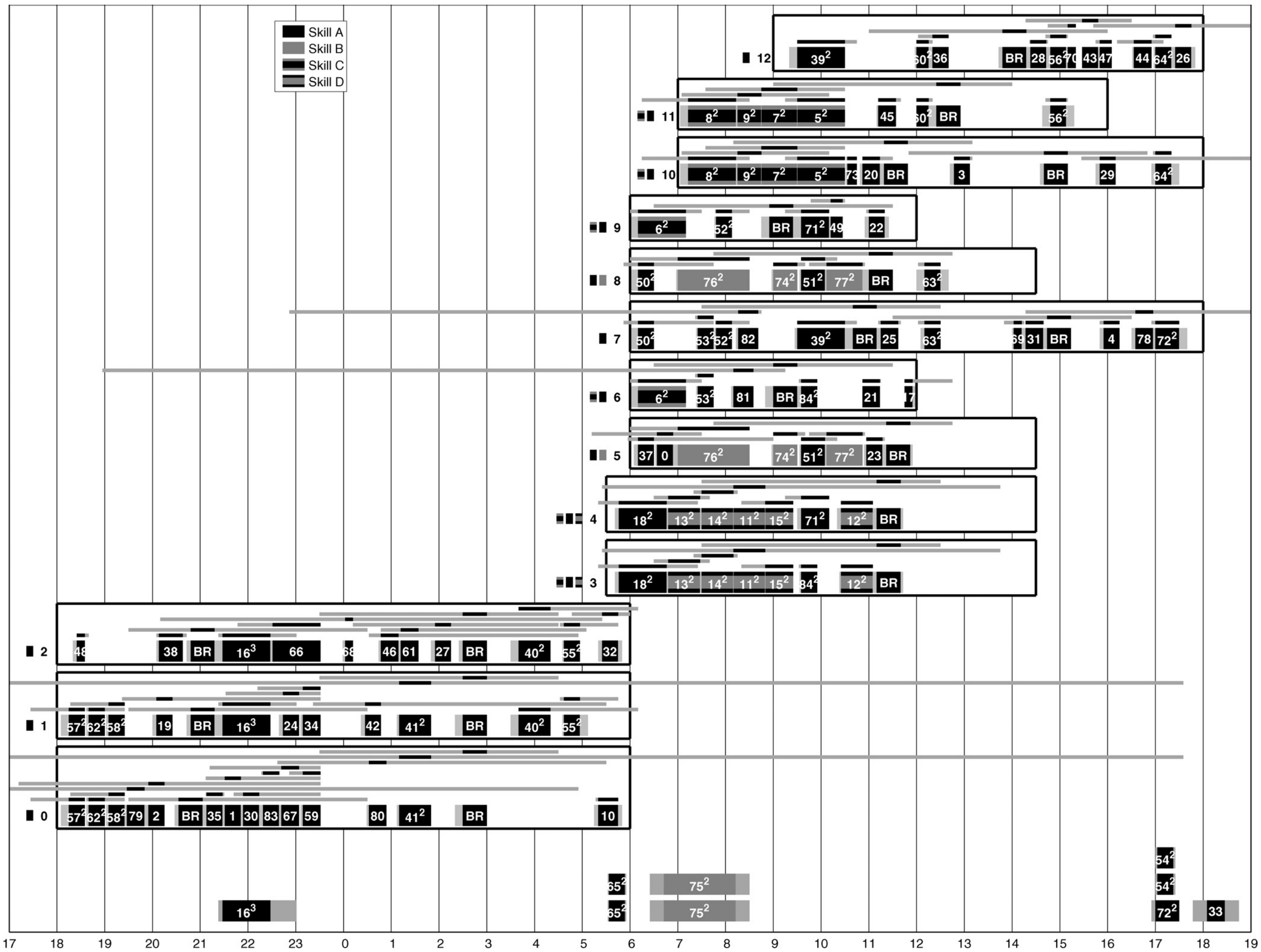

Figure 1: Problem instance **A.1** and its optimal solution.

we run out of memory before time out. The reported results for these instances have been recorded after 2 hours, which in these cases is just before the memory limit is reached. For **Type D** the results indicate that the generation of columns is now in itself a time consuming task and time-out is encountered with a relatively small tree size.

The branching trees from the above test have been built without a good initial solution. For each of the unfinished problems, we restart the algorithm with an initial solution, namely the best feasible solution of Table 1. The results of the new test are displayed in Table 2.

It is interesting that most of these instances are now solved to optimality within seconds. It clearly indicates that inexpedient branching decisions were made in the first run and more reliable branching is possible when promising columns exist initially. Another observation is that solving **C.1** under default settings leads to another out-of-memory failure, whereas changing the settings slightly gives an optimal solution within one second. This is another indication of the importance of making the right branching decisions and the consequence of not doing so. It has been tested that the settings giving a fast solution in this case are not superior in general.

To reveal the complexity added by the synchronized cooperation requirement, we also show results for a version of the problem where no branching on time windows is done (Table 3). This means that cooperation is no longer synchronized, but we are able to reach optimal solutions faster. Since the latter is a relaxation of the original problem, we are able to use the solution values as lower bounds on our problem.

Solution times of Table 3 should be compared to the times of Table 1 and reveal that solving the relaxed problem evidently is much faster and optimal solutions are found in all cases. The running times for the small and medium problems are up to 2 seconds, where one of the large problem instances uses around 37 minutes.

It is conspicuous that all the optimal solutions found in Table 1 are equal to the lower bound found in Table 3. The lower bound found by the unsynchronized model is naturally

	A.1	A.2	A.3	B.1	B.2	B.3	C.1	C.2	C.3	D.1	D.2	D.3
Unassigned split tasks	9	* 7	1	0	3	5	* 3	* 6	* 10	* 29	24	* 31
Lower Bound$^{\otimes}$	9	6	1	0	3	5	2	4	9	27	24	30
Time (s)	133	OM	2663	120	172	97	OM	OM	OM	TO	2719	TO
- LP (%)	15	46	20	10	10	11	29	9	34	2	5	3
- Branching (%)	68	7	70	82	82	78	34	81	32	5	10	4
- Pricing Problem (%)	4	8	2	1	2	2	4	4	9	93	83	91
- Overhead (%)	13	39	8	7	6	9	33	6	25	0	2	2
Tree size	605	42435	3207	537	597	507	188623	87843	69637	4961	487	2741
Max. depth	160	162	168	264	291	253	122	166	204	219	235	228
# Pricing Problems	13292	$3 \cdot 10^6$	107320	15554	17240	14813	$3 \cdot 10^6$	$2 \cdot 10^6$	$2 \cdot 10^6$	379799	20728	247634
# Vars added	12268	$2 \cdot 10^6$	109810	4074	5223	4321	$2 \cdot 10^6$	$1 \cdot 10^6$	$1 \cdot 10^6$	231209	16659	204614

Table 1: Results of the Branch-and-Price algorithm with no initial solution.
OM = Out-of-Memory was encountered. TO = The Time-Out limit of 10 hours was reached.
* The solution given is the best feasible solution found.
$^{\otimes}$ Lower Bound (more details in Table 3).

	A.1	A.2	A.3	B.1	B.2	B.3	C.1	C.2	C.3	D.1	D.2	D.3
Unassigned split tasks	9	7	1	0	3	5	$^{\times}$ 3	4	9	* 29	24	31
Lower Bound$^{\otimes}$	9	6	1	0	3	5	2	4	9	27	24	30
Time (s)		0.84					0.80	36	0.97	TO		235
- LP (%)		33					25	21	17	0		5
- Branching (%)		5					8	25	8	0		0
- Pricing Problem (%)		18					6	14	8	100		95
- Overhead (%)		44					61	40	67	0		0
Tree size		11					19	981	59	447		9
Max. depth		3					5	46	28	40		4
# Pricing Problems		530					561	32921	1358	42284		6415
# Vars added		785					758	16406	475	37212		6104

Table 2: Results of the Branch-and-Price algorithm with initial solution from the test of Table 1.
TO = The Time-Out limit of 10 hours was reached.
* The solution given is the best feasible solution found.
$^{\times}$ After OM on the first run, the pricing problem solver was in this case changed to not create heuristic columns.
$^{\otimes}$ Lower Bound (more details in Table 3).

closely related to the lower bound found in the root node of the branching tree of the problems in Table 1 and these results stress how important a good lower bound is.

Conclusion and Future Work

The Manpower Allocation Problem with Time Windows, Job-Teaming Constraints and a limited number of teams is successfully solved to optimality using a Branch-and-Price approach. By relaxing the synchronization constraint and using Dantzig-Wolfe decomposition, the problem is divided into a generalized set covering master problem and an elementary shortest path pricing problem. Applying branching rules to enforce integrality as well as synchronized execution of divided tasks enables us to arrive at optimal solutions in half of the test instances. Running a second round of the optimization, initiated from the best solution found in round one, uncovers the optimal solution to all but one of the 12 test instances. The test instances are all full-size realistic problems originating from scheduling problems of ground handling tasks in major airports. Synchronization between teams in an exact optimization context has not previously been treated in the literature. We have successfully integrated the extra requirements into the solution procedure and the results are promising.

Future work could aim at creating a structured approach to utilize the effect of restarting the branching mechanism. By simply restarting the algorithm once, we see a remarkable increase in the number of solvable problems, and an extended strategy may shorten solution time significantly and it may further increase the chance of finding optimal solutions. Beck (2006) describes a more sophisticated approach, where a number of promising solutions are saved and the tree search is restarted from one of these solutions, when the search seems to be stuck. A similar methodology may prove to be very efficient in our case.

Extending the model to include more properties of realistic problem instances is another desirable enhancement. Implementing such extensions, and at the same time preserv-

	A.1	A.2	A.3	B.1	B.2	B.3	C.1	C.2	C.3	D.1	D.2	D.3
Unassigned split tasks	9	6	1	0	3	5	2	4	9	27	24	30
Time (s)	0.96	1.10	1.37	0.64	0.77	0.80	1.18	1.86	1.65	75.18	413.26	2195.59
- LP (%)	16	8	15	6	4	3	19	25	10	17	10	2
- Branching (%)	7	0	0	0	0	1	39	24	49	17	8	1
- Pricing Problem (%)	45	74	69	19	22	30	9	11	17	62	81	97
- Overhead (%)	32	18	16	75	74	67	33	40	24	4	1	0
Tree size	3	3	1	1	3	3	11	21	19	35	83	97
Max. depth	1	1	0	0	1	1	5	8	9	17	41	22
# Pricing Problems	163	93	291	103	80	81	288	481	367	4450	8783	9811
# Vars added	407	350	663	309	222	212	586	683	435	4489	7773	14111

Table 3: Results of the Branch-and-Price algorithm with no constraint on synchronized coordination.
All solution values can be used as lower bounds on the original formulation.

ing the Branch-and-Price structure for effectiveness, may be a difficult task. Work with a dynamic choice on the number of split tasks for the individual tasks has shown that such an extension is feasible in a Branch-and-Price context.

References

Achterberg, T.; Koch, T.; and Martin, A. 2005. Branching rules revisited. *Operations Research Letters* 33(1):42–54.

Beasley, J. E., and Christofides, N. 1989. An algorithm for the resource constrained shortest path problem. *Networks* 19:379–394.

Beck, J. C. 2006. An empirical study of multi-point constructive search for constraint-based scheduling. *ICAPS 2006 - Proceedings, Sixteenth International Conference on Automated Planning and Scheduling* 2006:274–283.

Chabrier, A. 2006. Vehicle routing problem with elementary shortest path based column generation. *Computers and Operations Research* 33(10):2972–2990.

Coin. 2006. COmputational INfrastructure for Operations Research (COIN-OR). http://www.coin-or.org/.

Danna, E., and Pape, C. L. 2005. *Branch-and-Price Heuristics: A Case Study on the Vehicle Routing Problem with Time Windows*. Desaulniers G., Desrosiers J., Solomon M.M.: Column Generation, Springer, New York. chapter 4, 99–129.

Dantzig, G. B., and Wolfe, P. 1960. Decomposition principle for linear programs. *Operations Research* 8(1):101–111.

Desrochers, M.; Desrosiers, J.; and Solomon, M. M. 1992. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* 40:342–354.

Dohn, A.; Kolind, E.; and Clausen, J. 2007. The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby.

Dror, M. 1994. Note on the complexity of the shortest path models for column generation in VRPTW. *Operation Research* 42(5):977–978.

Feillet, D.; Dejax, P.; Gendreau, M.; and Gueguen, C. 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks* 44(3):216–229.

Feillet, D.; Dejax, P.; and Gendreau, M. 2005. Traveling salesman problems with profits. *Transportation Science* 39(2):188–205.

Gélinas, S.; Desrochers, M.; Desrosiers, J.; and Solomon, M. M. 1995. A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research* 61:91–109.

Irnich, S., and Villeneuve, D. 2006. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing* 18(3).

Jepsen, M.; Petersen, B.; Spoorendonk, S.; and Pisinger, D. 2006. A non-robust branch-and-cut-and-price algorithm for the vehicle routing problem with time windows. Technical report, Department of Computer Science, University of Copenhagen, Denmark.

Kallehauge, B.; Larsen, J.; Madsen, O. B. G.; and Solomon, M. M. 2005. *Vehicle Routing Problem with Time Windows*. Desaulniers G., Desrosiers J., Solomon M.M.: Column Generation, Springer, NY. chapter 3, 67–98.

Li, Y.; Lim, A.; and Rodrigues, B. 2005. Manpower allocation with time windows and job-teaming constraints. *Naval Research Logistics* 52:302–311.

Lim, A.; Rodrigues, B.; and Song, L. 2004. Manpower allocation with time windows. *Journal of the Operational Research Society* 55:1178–1186.

Righini, G., and Salani, M. 2006. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization* 3(3):255–273.

Selensky, E.; Prosser, P.; and Beck, J. C. 2006. A case study of mutual routing-scheduling reformulation. *Journal of Scheduling* 9(5):469–491.

Solomon, M. M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* 35(2):254–265.