A Branch-and-Price Algorithm for a Hierarchical Crew Scheduling Problem*

Diego B.C. Faneyte,[1] Frits C.R. Spieksma,[2,†] Gerhard J. Woeginger[3]

[1] *Epitec, P.O. Box 240, NL-6200 AE, Maastricht, The Netherlands*

[2] *Katholieke Universiteit Leuven, Department of Applied Economics, Naamsestraat 69, B-3000 Leuven, The Netherlands*

[3] *University of Twente, Faculty of Mathematical Sciences, P.O. Box 217, NL-7500 AE Enschede, The Netherlands*

Received 15 February 2000; revised 23 August 2001; accepted 6 May 2002

Abstract: We describe a real-life problem arising at a crane rental company. This problem is a generalization of the basic crew scheduling problem given in Mingozzi et al. [18] and Beasley and Cao [6]. We formulate the problem as an integer programming problem and establish ties with the integer multicommodity flow problem and the hierarchical interval scheduling problem. After establishing the complexity of the problem we propose a branch-and-price algorithm to solve it. We test this algorithm on a limited number of real-life instances. © 2002 Wiley Periodicals, Inc. Naval Research Logistics 49: 743–759, 2002; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/nav.10044

Keywords: branch-and-price; column generation; crew scheduling; integer multicommodity flow; interval scheduling

1. INTRODUCTION

A branch of a crane rental company has at its disposal a number of cranes of different types. These cranes are used to perform lifting tasks mostly at construction sites. The company operates in the following way: Each crane leaves the central depot in the morning to the location of its first job, performs this job, travels to the next one and so on, and returns to the depot when all its jobs are finished. An important characteristic of a type of cranes is its *capacity* or the maximum amount of weight it can lift. Also, for each type of cranes a *cost rate* is known.

The planning process for assigning jobs to cranes on some day d is handled in the following way: Starting 2 weeks before day d up to day $d - 1$, orders come in by telephone for the company's planners. A customer then specifies the time at which the job must start, the length

* Part of this research was performed when the second author visited the Technical University of Graz.
† The research of this author was partially supported by EU Grant 14084 (APPOL).
Correspondence to: F.C.R. Spieksma

of the period needed to perform the job (usually in hours), the capacity required for the job, and the location. Based upon a tentative schedule for day d, the planner decides to accept or reject the order. In case the order is accepted, we call it a job. Notice that a job requiring capacity c can only be performed by a crane of a type with capacity c or more. Some other constraints are as follows: A crane can only perform one job at a time, and preemption is not allowed. It follows from this description that, at the end of day $d - 1$, an assignment of jobs to cranes for day d has been constructed by the planners.

What determines the quality of such an assignment? The amount of money payed by the customer depends on two things: the length of the period that the crane is required (this is specified by the customer herself) and the cost rate of a crane type with minimum capacity to perform the job. The product of these two factors equals the price that the customer pays. Notice that the formulation above implies that, although a customer pays for a c-ton crane (since she has a c-ton job), she may receive a crane from the rental company from a type with larger capacity. It follows that total revenue in a day is independent of the actual assignment (as long as all jobs are performed). Therefore, we deal in this paper with minimizing costs.

The following cost structure was judged to be appropriate after discussions with the management of the company. Let us call the *working period* of a crane on a day as the amount of time worked on the jobs (as specified by the customers) plus total time spent traveling between the various locations. Notice that this does not include potential waiting times at locations. The cost of a crane is computed by multiplying its working period by the cost rate of the associated type of cranes. Total costs are then found by summing over all cranes. For a more elaborate description of the planning process and the problem, see Faneyte [11].

There are, in our view, at least two interesting issues related to this problem. First of all, if all cranes are identical and each job can be performed by each crane, the problem described above boils down to the so-called "basic" crew scheduling problem described in Mingozzi et al. [18] and Beasley and Cao [6] (with cranes corresponding to crews). These crew scheduling problems combine aspects from interval scheduling with routing decisions. Indeed, if there had been freedom in determining the starting times of the jobs, the problem would become similar to an m-traveling salesman problem with the cranes in the role of the salesmen. If, on the other hand, one would be able to choose the location of each job, one would presumably choose the depot as the location for each job, and the problem would become a type of interval scheduling problem with the cranes in the role of machines.

Second, from a scheduling point of view, the cost-structure of this problem is unorthodox in the following sense: Most scheduling environments have a cost for using a crane or not, independent of the amount of time the crane is used. In our case, time matters. For instance, the cost structure implies that, given a set of jobs all to be performed at the depot, it doesn't matter how many cranes are used to perform all jobs. In fact, we will see in Section 3 that this cost structure is more general than an orthodox one.

The goal of this paper is threefold:

(i) to present a formulation of our problem (which we will refer to as the hierarchical crew scheduling problem),
(ii) to establish ties between this problem, on the one hand, and the integer multicommodity flow problem and an interval scheduling problem, on the other hand, thereby (partly) establishing its complexity, and
(iii) to present a branch-and-price approach that is used for solving real-life instances.

1.1. Related Literature and Applications

It is well-known that (basic) crew scheduling problems and variants thereof have many applications in airline and railway industries (see, e.g., Mingozzi et al. [18] and Beasley and Cao [6] and references contained therein). More generally, routing problems with time windows are obviously related to our hierarchical crew scheduling problem. These type of problems often occur in manufacturing industries where one needs (service) vehicles to perform tasks at given moments in time (see, e.g., Fisher, Jörnsten, and Madsen [12] and Ge and Yih [14]).

As described above, our problem can be described as an interval scheduling problem with routing aspects. In fact, an early reference to such a problem (using the assumption that each job can be handled by each machine, using the triangle inequality and minimizing the number of machines used) is Dantzig and Fulkerson [7] (see also Ford and Fulkerson [13], pp. 64–67). Work concerning interval scheduling problems with machines of different types that does not involve routing aspects is Arkin and Silverberg [3], Dondeti and Emmons [9], and Kolen, Lenstra, and Papadimitriou [16]. In Kroon et al. [17] an interval scheduling problem is discussed where a fixed cost depending upon the machine type is incurred each time a machine performs a job.

Finally, it turns out (see Section 3) that our problem can be modeled as a special case of the integer multicommodity flow problem (see Ahuja, Magnanti, and Orlin [2], Aggerwal, Oblak, and Vegumanti [1], and Pióro and Gajowniczek [19]). A branch-and-price algorithm for the general integer multicommodity flow problem is described in Barnhart, Hane, and Vance [4]. For a general introduction to column generation techniques and branch-and-price algorithms we refer to Barnhart et al. [5], Desrosiers et al. [8], Sol [20], and Vanderbeck [22].

Section 2 of this paper gives a formal description of the hierarchical crew scheduling problem and discusses some of the assumptions we used. In Section 3 we establish ties between this crew scheduling problem and the integer multicommodity flow problem and the so-called hierarchical interval scheduling problem, and we settle the complexity of our problem. Section 4 proposes a branch-and-price algorithm for the problem, and in Section 5 we present limited computational results. Section 6 contains the conclusions.

2. PROBLEM FORMULATION

We give a formal description of the hierarchical crew scheduling problem introduced in the previous section.

2.1. Assumptions

Here we list explicitly some of the assumptions we use when providing a formulation of the hierarchical crew scheduling problem as described in the previous section (see also Faneyte [11]):

1. Planning horizon is a single day: This is reasonable, because a job rarely takes more than 1 day.
2. Constant and identical speed of each crane: In reality different types of cranes may have different speeds; moreover, this speed is not constant over time. For reasons of simplicity we assume that each crane has the same speed. However, our approach remains valid when different speeds are allowed (see a remark made in

Subsection 2.3). The fact that we assume constant speed is a close-enough approximation of reality.

3. Fixed starting times: In reality, it seems possible in some cases that the starting time is negotiable. This, however, is not recommended by the crane rental company, and it seems difficult to formalize this aspect.
4. Capacities: In reality, the capacity of a crane is not always observed, that is, sometimes a crane is used for a job that is slightly too heavy for that crane. However, management regards this as something to be avoided.
5. Single depot: The crane rental company consists of 4 branches, each with its own depot where cranes can be hired. At present, these locations work independently (although the possibility of coordinating the different branches is recognized). We obey this independence assumption; however, in Section 5 we discuss the implications for solving a multi-depot model.
6. No breakdowns: We assume that no breakdowns, traffic congestion, or other catastrophes occur.

Another issue concerns the cost rates of crane types. These cost rates reflect the variable costs of operating a crane of a specific type (including costs for petrol, maintenance, etc.). Although we do not explicitly assume that the cost rate of a crane is a non-decreasing function of its capacity, all instances that we consider (see Table 1 in Section 5) satisfy this property.

2.2. The Input

For each type of cranes $t = 1, \ldots, m$, let

- m_t = number of cranes of type t available.
- cap_t = capacity of a crane of type t.
- r_t = cost rate of a crane of type t.

For each job $i = 1, \ldots, n$, let

- s_i = starting time of job i.
- p_i = length of period needed to perform job i (processing time).
- w_i = capacity required by job i.
- $d_{0i}(d_{i0})$ = time needed to travel from the depot to the location of job i (and vice versa).

Finally, for each pair of jobs i, j, where $i, j = 1, \ldots, n$, let

- d_{ij} = time needed to travel from the location of job i to the location of job j.

We assume that all input parameters are nonnegative integers. We seek an assignment of jobs to cranes such that

1. For each job i assigned to a crane of type t we have $w_i \leq cap_t$.
2. For each pair of jobs i, j carried out consecutively by a single crane, we have $s_i + p_i + d_{ij} \leq s_j$.
3. No more than m_t cranes of type t are used.

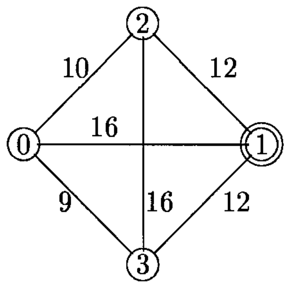

Figure 1. The distances.

4. Total costs are minimized.

To compute total costs, suppose that some crane of type t performs jobs $i_1, i_2, \ldots, i_l$ in that order. Then the costs associated to this crane are: $r_t(p_{i_1} + \cdots + p_{i_l} + d_{0,i_1} + \cdots + d_{i_{l-1},i_l} + d_{i_l,0})$. Summing this expression over all cranes gives total costs.

We will refer to the problem described above as the Hierarchical Crew Scheduling Problem with m crane types or HCSPm for short.

Consider, as an illustration, the following example.

EXAMPLE: Here is an instance with 2 types of cranes and 3 jobs: $m = 2$, $m_1 = m_2 = 1$, $cap_1 = 0$, $cap_2 = 1$, $r_1 = 1$, $r_2 = 3$, $n = 3$, $s_1 = 20$, $s_2 = 40$, $s_3 = 60$, $p_1 = p_2 = p_3 = 0$, $w_1 = 1$, $w_2 = w_3 = 0$, and with distances d_{ij} as in Figure 1. Notice that the distances d_{ij} satisfy the triangle inequality and are symmetric.

One (of the 2) optimal solution(s) for this instance can be described as follows. Starting at time 0, a crane of type 2 travels to job 1, waits 4 time units, performs the job, travels to job 3, waits 28 time units, performs it, and returns to the depot. Distance traveled is 37, so its costs are 111. The crane of type 1 simply travels to job 2, performs it, and returns for a total cost of 20. Thus, the value of an optimal solution to this instance is 131.

2.3. A Mathematical Model

We model HCSPm as a problem on a weighted, directed graph $G = (V, A)$ as follows. Construct a vertex for each job $i = 1, \ldots, n$ and let $V = \{1, \ldots, n\} \cup \{s, f\}$. The vertices s and f each represent the depot and can be regarded as the source and sink of the graph G. There is an arc from vertex i to j in A if $s_i + p_i + d_{ij} \le s_j$ for all $i, j \in V \backslash \{s, f\}$. Further, there is an arc $\{s, i\}$ and $\{i, f\}$ in A for all $i \in V \backslash \{s, f\}$. Finally, there is a weight vector c^t_{ij} associated with each arc $\{i, j\} \in A$. We compute this vector as follows:

$$c^t_{ij} = \begin{cases} r_t(d_{ij} + p_j) & \text{if } w_j \le cap_t, \\ M & \text{if } w_j > cap_t, \end{cases} \quad \text{for all } t, \quad \text{for all } \{i, j\} \in A, \tag{1}$$

where M is a large number and $p_f = w_f = 0$.

Notice that by using (1), costs associated with jobs are transferred to costs on arcs. Also, notice that for a fixed t this operation preserves the triangle inequality, or, in other words, if the d_{ij} satisfy the triangle inequality, so do the c^t_{ij}. Finally, observe that in case the d_{ij}'s would depend on the type of crane t (the case of different speeds for different types of cranes), the above computations would go through.

Graphs corresponding to instances of HCSPm have a special structure. For instance, such a graph is acyclic. Also, if the triangle inequality holds, it is transitive, that is, if arcs $\{i, j\}$ and $\{j, k\}$ are in A, then arc $\{i, k\}$ is in A. Using this graph, we devise the following integer linear programming model. Let

$$x_{ij}^t = \begin{cases} 1 & \text{if a crane of type } t \text{ travels directly from the location of job } i \text{ to the} \\ & \text{location of job } j, \\ 0 & \text{otherwise.} \end{cases}$$

The model is

$$(\text{HCSP}m) \qquad \text{minimize} \quad \sum_{t=1}^{m} \sum_{\{i,j\}\in A} c_{ij}^t x_{ij}^t$$

$$\text{subject to} \quad \sum_{t=1}^{m} \sum_{j:\{j,i\}\in A} x_{ji}^t = 1 \qquad \text{for all } i \in V\backslash\{s,f\}, \tag{2}$$

$$\sum_{j:\{j,i\}\in A} x_{ji}^t - \sum_{j:\{i,j\}\in A} x_{ij}^t = 0 \qquad \text{for all } t, \quad \text{for all } i \in V\backslash\{s,f\}, \tag{3}$$

$$\sum_{j:\{s,j\}\in A} x_{sj}^t \le m_t \qquad \text{for all } t, \tag{4}$$

$$x_{ij}^t \in \{0,\ 1\} \qquad \text{for all } t, \quad \text{for all } i,j \in V\backslash\{s,f\}. \tag{5}$$

Constraints (2) imply that each vertex is visited by selecting one incoming arc for each vertex, equalities (3) represent the flow conservation constraints, inequalities (4) state that not more than m_t cranes of type t can leave the source, and constraints (5) are the integrality constraints.

There is a close relation between HCSPm and integer multicommodity flow. In fact, HCSPm is a special case of an integer m-commodity circulation problem as shown in the next section.

3. COMPLEXITY ISSUES

In this section we show that HCSPm is a special case of the integer m-commodity flow problem, and that m-hierarchical interval scheduling is a special case of HCSPm. We refer to the Appendix for the proofs of the theorems in this section.

THEOREM 1: HCSPm is a special case of the integer multicommodity flow problem.

PROOF: See the Appendix. □

In fact, the reduction shows that HCSPm can be seen as an integer multicommodity circulation problem. The connection between HCSPm and multicommodity flow goes even further: When one would write down the integer programming formulation of a multicommodity flow problem corresponding to the instances described here, and one would make straightforward substitutions, formulation (HCSPm) would arise.

COROLLARY 2: CSP1 can be solved in polynomial time.

Indeed, Theorem 3.1 implies that HCSP1 is a min cost flow problem and hence can be solved in polynomial time (see Ahuja, Magnanti, and Orlin [2] for achievable time-bounds). More generally, any positive result for the integer multicommodity flow problem implies a positive result for HCSPm (see Srivastav and Stangier [21]).

In our next result we use the hierarchical interval scheduling problem (see Kolen, Lenstra, and Papadimitriou [16] or the Appendix for a definition of this problem).

THEOREM 3: The hierarchical interval scheduling problem is a special case of HCSPm.

PROOF: See the Appendix. □

COROLLARY 4: HCSPm is $\mathcal{NP}$-hard for all $m \geq 3$.

Indeed, this follows from Theorem 3.3 and the fact that Kolen, Lenstra, and Papadimitriou [16] proved that the hierarchical interval scheduling problem with K processor types is $\mathcal{NP}$-hard for any fixed $K \geq 3$. More generally, any negative result for the hierarchical interval scheduling problem with $K \geq 3$ implies a negative result for HCSPm, $m \geq 3$. Since the hierarchical interval scheduling problem for $K = 2$ can be solved efficiently (see Dondeti and Emmons [9]), whereas the integral 2-commodity flow problem is $\mathcal{NP}$-hard (see Even, Itai, and Shamir [10]), we need a separate theorem to establish the complexity of HCSP2.

THEOREM 5: HCSP2 is $\mathcal{NP}$-hard.

PROOF: See the Appendix. □

4. BRANCH-AND-PRICE

This section describes a branch-and-price algorithm for HCSPm. First we propose a formulation for HCSPm using variables associated to paths instead of variables associated with arcs. Then we show how one can solve this formulation using a column generation procedure. We show that the pricing problem in this procedure can be efficiently solved. Finally, we describe a branching scheme.

4.1. Formulation

Given an instance of HCSPm and its associated graph $G = (V, A)$ (see Section 2), consider the following formulation of HCSPm using the following parameters:

- R = the set of paths in G from s to f,

for $i = 1, \ldots, n$, $r \in R$:

- $\delta_{ir} = \begin{cases} 1 & \text{if vertex } i \text{ is in path } r \text{, and} \\ 0 & \text{otherwise} \end{cases}$

for $t = 1, \dots, m$, $r \in R$:

- c_{tr} = cost incurred when a crane of type t takes path r [observe that, for a given r, t, we can easily compute this quantity using (1); notice that if a path r contains a vertex that cannot be served by a crane of type t, we set the corresponding c_{tr} to a large number], and using, for $t = 1, \dots, m$, $r \in R$, the decision variables

$$y_{tr} = \begin{cases} 1 & \text{if a crane of type } t \text{ takes path } r, \\ 0 & \text{otherwise,} \end{cases}$$

$$(\text{CGHCSP}m) \qquad \text{minimize} \quad \sum_{t=1}^{m} \sum_{r \in R} c_{tr} y_{tr}$$

$$\text{such that} \quad \sum_{t=1}^{m} \sum_{r \in R} \delta_{ir} y_{tr} = 1 \qquad \text{for } i = 1, \dots, n, \tag{6}$$

$$\sum_{r \in R} y_{tr} \le m_t \qquad \text{for } t = 1, \dots, m, \tag{7}$$

$$y_{tr} \in \{0, 1\} \qquad \text{for } t = 1, \dots, m, r \in R. \tag{8}$$

Constraints (6) state that each vertex must occur once in a selected path, inequalities (7) express that no more than m_t cranes of type t can be used, and constraints (8) are the integrality constraints. The LP-relaxation of this model is found by replacing constraints (8) by $y_{tr} \ge 0$ for all t, r.

Notice that by setting for all t, r and for all $\{i, j\}$ in path r, $x_{ij}^t = y_{tr}$, one concludes that the value of the LP-relaxation of both formulations (HCSPm and CGHCSPm) is equal (see also Ahuja, Magnanti, and Orlin [2]).

4.2. Column Generation

Column generation is a way to solve the LP-relaxation of the model above without having to enumerate all variables (columns) y_{tr} (see, for instance, Barnhart et al. [5] for a description). Given a feasible basis for some LP the question determining whether this basis is an optimal one is: Do there exist variables with negative reduced costs? Using dual variables u_i attached to constraints (6) and dual variables e_t to constraints (7), we can deduce the following expression for the reduced costs of variable y_{tr}:

$$c_{tr} - \sum_{i=1}^{n} \delta_{ir} u_i - e_t.$$

Thus, given some LP-solution and its associated dual variables, the *pricing problem* boils down to the following question:

$$\text{Price:} \qquad \exists\, t, r \in R \quad \text{such that} \quad c_{tr} - \sum_{i=1}^{n} \delta_{ir} u_i - e_t < 0?$$

We claim that this question can be answered in polynomial time:

LEMMA 6: Problem Price can be solved by solving m shortest path problems on a directed acyclic graph.

PROOF: We claim that, for a fixed t, Price boils down to a shortest path problem which implies the lemma. This can be seen as follows: Consider the graph corresponding to the instance and consider only those nodes that can be visited by a crane of type t. Observe that no cycles occur in this network. Modify the existing arc costs c^t_{ij} by setting $h_{ij} := c^t_{ij} - u_j$ for all $\{i, j\} \in A$. Observe that the cost of a path P in this network with respect to costs h_{ij} from s to f equals: $\sum_{\{i,j\}\in P} h_{ij} = \sum_{\{i,j\}\in P} (c^t_{ij} - u_j) = c_{tr} - \sum_i \delta_{ir} u_i$. If this expression is smaller than e_t, there is a profitable column for this type t; otherwise not. □

Let us now describe the generic column generation procedure that we employ:

Step 1: Find an initial set of paths from s to f called R' that contain a feasible solution.

Step 2: Solve the model (CGHCSPm) with R' instead of R (this is the *restricted master* problem).

Step 3: Solve Price (see Lemma 4.1). If the answer is yes, update R' by adding the found path and go to Step 2; otherwise STOP, the current LP-solution is optimal.

We refer to Section 5 for more details concerning each of these steps.

4.3. A Branching Rule

Obviously, the column generation procedure only gives us the LP-relaxation of (CGHCSPm). In order to obtain integral solutions, we need a branching rule that partitions the solution space. Ideally, this rule should not destroy the solvability of the pricing problem. This is not a trivial matter (see, e.g., the discussion in [5]): Indeed, a branching rule that consists of setting a variable to 1 versus setting that variable to 0 does not fall under this category. Here we propose a branching rule that leaves the structure of the problem intact, allowing for the efficient solvability of the pricing problem throughout the search tree.

Suppose that y is a fractional feasible LP-solution, and let us call a path r from s to f a *fractional* path (with respect to y) if there exists a t with $0 < y_{tr} < 1$. We claim that y has the following property: There exist two vertices i and j (that differ from source and sink) that lie consecutively on a fractional path such that the sum of all y_{tr} such that r contains arc $\{i, j\}$ is greater than 0 and smaller than 1. Let us formally phrase this claim in the following lemma:

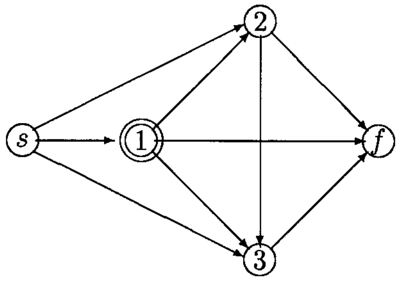

Figure 2. The graph G.

LEMMA 7: If y is fractional, there exist two nodes $i, j \in V\backslash\{s, f\}$, $\{i, j\} \in A$ with $0 < \sum_t \sum_{r:\{i,j\} \text{ is contained in } r} y_{tr} < 1$.

PROOF: Observe that if y is fractional there are at least two different fractional paths. Now, consider the first node in each fractional path. If this set of nodes has cardinality more than 1, the claim is easily seen to be true. So assume that each fractional path has the same first node. However, then we can repeat this argument replacing the sink s by this first node. Since there must be at least two fractional paths, a pair i, j as described in the lemma must exist. □

Thus, by this lemma, when y is fractional there exist nodes i and j (that differ from source and sink) that are connected by an arc whose sum of fractional values lies between 0 and 1. Now, in an optimal solution either these nodes are visited consecutively, or they are not. More specifically, given a fractional solution we identify two nodes i and j having the property described above. Then we branch as follows. In one branch we modify G into G_1 by deleting all arcs $\{i, p\}$ for $p \neq j$ and all arcs $\{p, j\}$ for $p \neq i$. Thus, in any feasible solution there is a path that contains arc $\{i, j\}$. In the other branch we modify G into G_2 by simply deleting arc $\{i, j\}$. In this case it is obvious that no feasible solution has a path with arc $\{i, j\}$ in it. Notice that the current solution is excluded by this rule. Let us illustrate this branching rule on the Example from Section 2.

EXAMPLE (continued): Consider the instance described in the example of Section 2. The graph corresponding to this instance is depicted in Figure 2.

An optimal LP-solution is described by $y_{1,s-2-3-f} = y_{2,s-1-2-f} = y_{2,s-1-3-f} = \frac{1}{2}$, and all other variables 0. Now, let 1, 2 be a pair of nodes that we branch on. The resulting graphs G_1 and G_2 are depicted in Figure 3.

Notice that in G_1 the arcs $\{s, 2\}$, $\{1, 3\}$, and $\{1, f\}$ have been deleted. When solving the LP-relaxation corresponding to G_1 we find an integral solution with value 132. G_2 is constructed by deleting arc $\{1, 2\}$. In this branch we also find an integral solution with value 131, which is therefore optimal.

5. COMPUTATIONAL RESULTS

In this section we describe limited computational experience of a branch-and-price algorithm implemented along the lines proposed in Section 4. These experiments are based on real-life

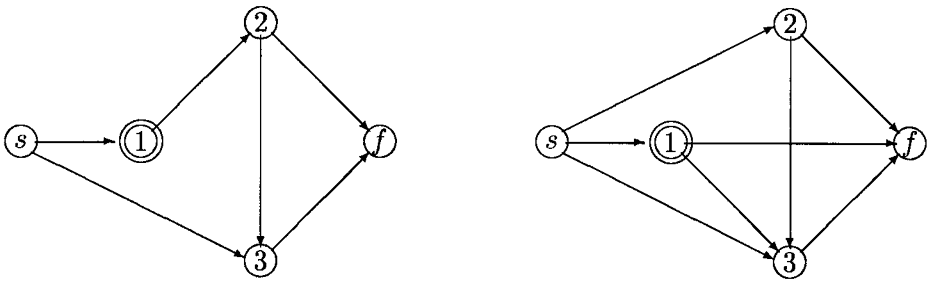

Figure 3. The graphs G_1 (left) and G_2 (right).

instances made available to us by the crane rental company. They are carried out on a Silicon Graphics Indigo workstation with 32 MB of intern memory. We use Cplex 2.1 to solve the restricted master problem, and the algorithms are coded using Visual Basic.

Subsection 5.1 describes the characteristics of 11 real-life instances. Based on these 11 instances, we create additional instances involving random distances. In Subsection 5.2 we consider the consequences of a possible extension of our work to a multi-depot environment.

Let us now specify some issues concerning the implementation of the branch-and-price algorithm. Consider the three steps in the column generation procedure described in Section 4.

Ad Step 1: An initial set of paths was found as follows. As sketched in Section 1, the crane rental company employs planners that manually construct a solution. We used the set of paths in their solution as our initial set in Step 1 of the column generation procedure. In general one could employ a heuristic that—starting with the smallest-capacity crane—greedily selects a feasible path until all jobs are covered.

Ad Step 2: We used Cplex 2.1 as our LP-solver to solve the restricted master problem.

Ad Step 3: Starting with $t = 1$, we solved a shortest path problem (since the graph is acyclic we can use a simple dynamic program (see [2], pp. 107–108 for details) until a path was found with negative reduced costs. At that point we simply added this path to the current set of paths and a new iteration started.

Given a fractional solution y, we identified a pair of nodes i, j as described above by simply checking whether in each of the fractional paths in y, two consecutive nodes in such a path had this property. Finally, concerning our search strategy, we used a simple depth first strategy, where we always followed the branch that ensured that nodes i and j would be consecutive on some path in the solution.

5.1. Eleven Instances

As mentioned above, the crane rental company made available to us the crane characteristics of a single branch of the company (see Table 1) and 11 instances corresponding to 11 days.

We solved these 11 instances using the branch-and-price algorithm as described above. The results can be found in Table 2. Each row corresponds to an instance, the explanation of columns

Table 1. The cranes.

Crane Type t	1	2	3	4	5	6	7	8	9	10	11	12	13
m_t	1	2	3	11	1	4	3	2	2	2	1	1	1
cap_t	7	20	25	30	35	40	45	60	70	90	120	150	300
r_t	585	675	680	715	715	735	735	855	980	990	1555	1610	1670

1 and 2 is straightforward, v_{PLAN} denotes the value of the solution found by the planners of the crane rental company, v_{IP} denotes the value of an optimal integral solution found by the branch-and-price algorithm, the fifth column gives the improvement as a percentage $\left(\frac{v_{PLAN} - v_{IP}}{v_{PLAN}}\right)$, the sixth column gives the depth of the search tree (where a '0' denotes solved in the root node), the seventh column gives the total number of iterations of the column generation procedure, and the final column gives the time needed to solve the instance in seconds. These computation times do not include times for reading an instance or outputting a solution.

Observe that the solutions that we computed are significantly better than the ones used in practice. From these experiments one can conclude that an improvement of 10% in solution value is possible [and recall that some of the solutions carried out in practice are not feasible in our models due to violation of the capacity constraint (see Subsection 2.1)]. The solutions themselves usually differed completely from the manually constructed solutions. Also notice that in 10 out of the 11 real-life instances the LP-relaxation has an integral solution! This seems to indicate that at least for these instances the LP-relaxation is quite strong. One possible explanation for this phenomenon could be the length of the jobs involved in our instances. The average length of a job is about 2 h, which implies that a crane rarely performs more than 5 jobs. Finally, computation times are satisfactory, at least for practical purposes.

To further experiment with our algorithm and to test the possible explanation of the strength of the LP-relaxation for our instances, we constructed artificial instances based on the characteristics of the crane rental company as described in Table 1. Specifically, we created three additional sets of each 11 instances. For the first additional set of instances we simply copied the input of each original instance, but we set the processing time p_i of each job i ($i = 1, \ldots, n$) to 25 min; see Table 3 for the results. For the second additional set of instances we also copied the input of each original instance, but we replaced the distances by random numbers. More precisely, for each pair of locations in the instance, we drew a random number from a uniform

Table 2. The instances.

Instance	Number of Jobs	v_{PLAN}	v_{IP}	Improvement	Depth of Search Tree	Iterations	Time (s)
1	48	136023.60	117997.95	13.3%	0	24	12.08
2	40	134263.73	119525.85	11.0%	0	13	3.11
3	42	118415.68	104531.70	11.7%	0	16	3.86
4	34	121382.38	107170.43	11.7%	0	10	1.19
5	41	126666.68	106570.55	15.9%	1	29	7.57
6	40	95649.53	76140.73	20.4%	0	39	13.28
7	38	99653.35	88930.43	10.8%	0	23	6.57
8	34	120173.15	105663.23	12.1%	0	5	0.58
9	44	130927.73	117512.9	10.2%	0	21	6.53
10	40	142107.45	126963.95	10.7%	0	23	7.36
11	47	138518.40	124665.85	10.0%	0	28	11.98

Table 3. The instances with processing time equal to 25 min.

Instance	Number of Jobs	v_{IP}	Depth of Search Tree	Iterations	Time (s)
1	48	15310.23	2	73	49.22
2	40	16737.47	3	38	19.11
3	42	15831.18	0	33	11.35
4	34	16953.02	0	23	4.51
5	41	13412.58	0	43	16.16
6	40	14517.88	0	34	10.18
7	38	16449.05	1	47	16.22
8	34	20845.98	0	16	2.54
9	44	17325.38	9	176	99.14
10	40	17403.31	1	35	9.92
11	47	23367.50	1	54	27.93

distribution between 5 and 45; see Table 4 for the results. Finally, for the third additional set of instances we imposed the characteristics of the two previous sets; that is, we set the processing time of each job to 25 min and each distance is drawn from $U[5, 45]$. The results can be found in Table 5.

The outcome of these experiments does not refute the possible explanation for the apparent strength of the LP-relaxation: Now, in the case with short processing times (see Table 3), about half of the instances has a nonintegral LP-relaxation. For the case with random distances all instances turn out to have an integral LP-relaxation (Table 4). When, in addition, processing times are short, one instance has a nonintegral LP-relaxation (Table 5). Further, computation times stay very reasonable.

5.2. A Possible Extension: The Multi-Depot Case

As discussed in Section 2, the crane rental company has different branches located on different sites. To establish cooperation between the various branches, it would be desirable to have the ability to solve a multi-depot version of CSPm with a large number of jobs. It is not very difficult to extend the formulations given here to deal with different depots: Add a source and a terminal for each depot, and both formulations go through. (Notice, however, that the number of cranes of some type at a single depot may then vary over the days.)

Table 4. The instances with random distances.

Instance	Number of Jobs	v_{IP}	Depth of Search Tree	Iterations	Time (s)
1	48	22409.13	0	50	34.54
2	40	20902.44	0	26	8.22
3	42	23215.84	0	33	12.07
4	34	20458.33	0	21	3.52
5	41	18202.92	0	55	23.28
6	40	14260.30	0	38	12.50
7	38	15041.12	0	41	13.66
8	34	24879.31	0	29	6.20
9	44	21965.63	0	69	36.48
10	40	18955.08	0	36	11.55
11	47	24715.14	0	48	27.04

Table 5. The instances with processing time equal to 25 min and random distances.

Instance	Number of Jobs	v_{IP}	Depth of Search Tree	Iterations	Time (s)
1	48	20499.21	1	135	125.17
2	40	20472.78	0	20	5.07
3	42	21631.30	0	28	9.45
4	34	19647.01	0	31	5.70
5	41	19050.51	0	60	29.63
6	40	16247.01	0	43	17.00
7	38	15937.15	0	44	15.18
8	34	25381.88	0	14	2.28
9	44	17787.30	0	74	45.84
10	40	26319.67	0	50	18.40
11	47	32838.91	0	56	30.68

Since no data were available for all branches, we generated—as an experiment—16 instances having a single depot with a large number of jobs. In fact, we created each of the 16 instances by adding three of the original instances, while multiplying the number of available cranes of each type (see Table 1) by 3. Our motivation for this experiment is to see whether this approach could be used for solving multi-depot instances. The results of the branch-and-price algorithm can be seen in Tables 6 and 7. The first column in each of these tables indicates which 3 original instances are used to construct a large instance. In the first table we used original distances, and in the second table we used distances drawn from a uniform distribution between 5 and 45. It turns out that, although computation times increase, they stay within reasonable bounds. Notice also that apart from two instances in Table 6, all instances have an integral valued LP-relaxation. This seems to suggest that the approach proposed here could be used to facilitate cooperation between the different branches of the crane rental company.

6. CONCLUSIONS

In this paper we formulated a hierarchical crew scheduling problem (called HCSP*m*) and connected it with the multi-commodity flow problem and an interval scheduling problem. We established the complexity of HCSP*m*, and we proposed a branch-and-price algorithm for it. This algorithm was shown to perform well on real-life instances; it turned out most instances were solved in the root node. Finally, we give some computational evidence that our approach has potential in the sense that it allows the possibility of multiple depots working together.

Table 6. Large instances.

Instance	Number of Jobs	v_{IP}	Depth of Search Tree	Iterations	Time (s)
6, 7, 8	112	253145.05	0	66	77.02
9, 10, 11	131	357279.20	0	136	300.88
1, 4, 5	123	317328.83	3	203	343.38
2, 3, 6	122	297975.65	0	80	115.44
5, 7, 10	119	295342.08	0	126	237.26
3, 5, 10	123	324541.55	4	129	221.50
2, 6, 11	127	292954.98	0	117	216.44
5, 8, 9	119	316951.30	0	50	48.14

Table 7. Large instances with random distances.

Instance	Number of Jobs	v_{IP}	Depth of Search Tree	Iterations	Time (s)
6, 7, 8	112	256347.18	0	73	84.31
9, 10, 11	131	333535.38	0	187	442.56
1, 4, 5	123	321506.70	0	341	620.65
2, 3, 6	122	299807.15	0	119	187.91
5, 7, 10	119	364389.63	0	142	264.33
3, 5, 10	123	332586.23	0	154	270.34
2, 6, 11	127	290752.58	0	110	212.90
5, 8, 9	119	317144.50	0	100	118.42

APPENDIX

Here we prove Theorems 1, 3, and 5. To do so, let us first describe the Integer Multicommodity Flow Problem (IMFP) (see, for instance, Ahuja, Magnanti, and Orlin [2]). An instance of IMFP consists of specifying the number of commodities m', a directed graph $G' = (V', A')$, a demand $b^t(i)$ for each $i \in V'$, $t = 1, \dots, m'$, lower (upper) bounds l_{ij} (u_{ij}) for total flow through arc $\{i, j\} \in A'$, lower (upper) bounds l^t_{ij} (u^t_{ij}) for flow of commodity $t = 1, \dots, m'$ through arc $\{i, j\} \in A'$, and costs $\tilde{c}^t_{ij}$, which represent the cost of sending one unit of commodity t through arc $\{i, j\}$, $t = 1, \dots, m'$, $\{i, j\} \in A'$.

PROOF OF THEOREM 1: Given is an instance of HCSPm with its associated graph. Let $m' = m$. To construct V', introduce for each vertex $i \in V \backslash \{s, f\}$ two nodes in V', say i' and i'', and add vertices s and f to V'. For each arc $\{i, j\}(\{s, i\}, \{i, f\}) \in A$, let there be an arc in A' of the form $\{i'', j'\}(\{s, i'\}, \{i'', f\})$ with costs $\tilde{c}^t_{i'',j'} = c^t_{ij}$ $(\tilde{c}^t_{s,i'} = c^t_{si}$, $\tilde{c}^t_{i'',f} = c^t_{if})$ for $t = 1, \dots, m'$. Further, all arcs of the form $\{i', i''\}$ are in A' with costs 0 for all t and $l_{i',i''} = 1$. All demands $b^t(i) = 0$ for all $i \in V'$, for all t, and there is an arc $\{f, s\}$ in A' with $\tilde{c}^t_{fs} = 0$ for all t and $u^t_{fs} = m_t$ for all t. All arcs $\{i, j\}$ in A' except arc $\{f, s\}$ have capacity $u_{ij} = 1$, and all other remaining parameters are 0.

It is now straightforward to check that a feasible solution to IMFP corresponds to a feasible solution to HCSPm with the same cost and vice versa. $\square$

Let us now consider the Hierarchical Interval Scheduling Problem (HISP). An instance of HISP is described as follows. Given is some integer K and m'_k processors of type k, each with costs c_k, $k = 1, \dots, K$. Also given are n' jobs which are to be processed without preemption during the interval $[s'_i, s'_i + p'_i)$, $i = 1, \dots, n'$. Each job belongs to some class k, $k = 1, \dots, K$ which implies that it can only be processed by processors of type $k, k + 1, \dots, K$. When a processor of type k is used to process 1 or more jobs a cost of c_k is incurred. Given some integer T the decision problem associated with HISP is the following: Does there exist a feasible solution to HISP with cost no more than T?

PROOF OF THEOREM 3: Define $c_{\max} = \max_k c_k$, and let us construct the following instance of HCSPm. First of all, all job characteristics go through in an obvious manner (more precisely: $n = n'$, $p_i = p'_i$, $s_i = s'_i$ for $i = 1, \dots, n$), $m = K$, $r_t = c_t$ for $t = 1, \dots, m$, $m_k = m'_k$ for all k, $d_{ij} = 0$ for all jobs $i, j = 1, \dots, n$, and $d_{0,i} = d_{i,0} = \frac{1}{2}(c_{\max} + 1) \sum_j p_j$ for all i. Further, let $cap_t = t$, and set $w_i = t$ if job i is of class t and set $Q = (c_{\max} + 1)T \sum_i p_i + c_{\max} \sum_i p_i$. Intuitively speaking, all jobs have to be performed at the same location, and the depot is placed quite far away from this location.

If the answer to the decision question is yes, it is easy to see that there is a solution in HCSPm with a cost bounded by $\sum_t \alpha_t c_t(d_{0i} + d_{i0}) + c_{\max} \sum_i p_i$ (where α_t is the number of machines of class t, $t = 1, \dots, m$ that are used in the solution to HISP). This can be seen by simply copying the solution for HISP in HCSPm. The first term is then an upper bound for the costs incurred for all cranes that leave and return to the depot, and the second term is an upper bound for the costs incurred when serving all the jobs. Since $\sum_t \alpha_t c_t \leq T$, it follows that the cost of this solution in HCSPm does not exceed Q.

Now suppose there is a solution in HCSPm with cost bounded by Q. Since using a crane of type t in this solution costs at least $c_t(c_{\max} + 1) \sum_i p_i$ (the cost of leaving and entering the depot), it follows that there are numbers β_t $(\leq m_t)$ such that $\sum_t \beta_t c_t(c_{\max} + 1) \sum_i p_i \leq Q$. This simplifies to $(\sum_t \beta_t c_t - T)(c_{\max} + 1) \leq c_{\max}$ and the result follows. $\square$

Notice that the distances d_{ij} in this reduction are symmetric and satisfy the triangle inequality.

Finally, let us now establish Theorem 3.5. We give a reduction from the THREE DIMENSIONAL MATCHING Problem (3DM) which is known to be NP-complete in the strong sense and which is defined as follows (see Garey and Johnson [15]).

INSTANCE: Sets X, Y and Z, $|X| = |Y| = |Z| = q$ and a set $T \subseteq X \times Y \times Z$, with $|T| = 2q$.
QUESTION: Does T contain a matching; i.e., do there exist q triples in T such that each element in $X \cup Y \cup Z$ occurs precisely once?

PROOF OF THEOREM 5: Construct an instance of HCSP2 as follows. We will specify in total $n = 9q$ jobs, $3q$ of which will have weight 1 (referred to as the *heavy* jobs), and the remaining $6q$ jobs will have weight 0 (referred to as the *light* jobs).

Consider a triple $t = (x_i, y_j, z_k)$ in T. For each element of a triple we have a light job; so there is a light job corresponding to x_i, one corresponding to y_j, and one corresponding to z_k. Since we do this for every triple, we now have $6q$ light jobs. Further, for each element in $X \cup Y \cup Z$, we construct a heavy job. This yields $3q$ heavy jobs. Let us refer to a pair of jobs one of which is light and one of which is heavy and both of which correspond to the same element in $X \cup Y \cup Z$ as a *corresponding pair.* The processing times of all jobs are 0, that is, $p_i = 0$, $i = 1, \dots, n$, the starting times of the light jobs corresponding to elements in X (Y, Z) equal 0 (1, 2), the starting times of the heavy jobs all equal 5. The distances are specified as follows. For all $i = 1, \dots, n$, $d_{0i} = d_{i0} = 0$. The distance

- between a corresponding pair of jobs equals 0,
- between a light job and a heavy job that do not form a corresponding pair equals 1,
- between a pair of light jobs of the same triple equals 1,
- between a pair of light jobs that do not correspond to the same triple equals $2n + 1$, and
- between a pair of heavy jobs equals 1.

There are 2 types of cranes: type 1 has $m_1 = q$, $cap_1 = 0$, $r_1 = 1$ (referred to as *light* cranes) and type 2 has $m_2 = 3q$, $cap_2 = 1$, $r_2 = 2q + 1$ (referred to as *heavy* cranes). This specifies an instance of HCSP2.

We claim that this instance of HCSP2 admits a feasible solution with cost $2q$ if and only if 3DM has a feasible solution.

[If:] Suppose that T contains a matching. Consider the light jobs corresponding to elements that belong to triples in the partition. Each of these light jobs forms, together with a unique heavy job a corresponding pair. Let us use $3q$ heavy cranes to perform these $3q$ corresponding pairs of jobs. Then all heavy jobs are serviced, and one easily sees that, by using a light crane for the jobs in a triple not chosen in the matching, one incurs a cost of 2 per light crane, leading to a total cost of $2q$.

[Only If:] Assume that we have a solution of the HCSP2 instance with cost no more than $2q$. Observe that since all heavy jobs happen simultaneously, we need at least $3q$ heavy cranes to do these jobs. Is it possible for a heavy crane to do 2 or more light jobs? No, since traveling between a pair of light jobs takes at least distance 1, this would lead to a cost of at least $2q + 1$. Thus a heavy crane does at most 1 light job. Also, since each light crane can do at most 3 light jobs (otherwise total travel cost would exceed $2q$), it follows that each light crane must do 3 light jobs, and thus that each heavy crane must do exactly 1 light job. Notice now that the cost incurred by the light cranes to do their triples equals $2q$. So for the heavy cranes to serve the heavy jobs, they must use 0 distances. This is only possible by using the corresponding pairs and a solution to 3DM is found. □

ACKNOWLEDGMENTS

We thank Mr. Driessen of Marcon BV for his support and willingness to provide us with data and Stan van Hoesel for discussions. Further, we thank the referees for their comments.

REFERENCES

[1] A.K. Aggerwal, M. Oblak, and R.R. Vegumanti, A heuristic solution procedure for multicommodity integer flows, Comput Oper Res 22 (1995), 1075–1087.

[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network flows, Prentice Hall, Englewood Cliffs, NJ, 1993.

[3] E.M. Arkin and E.B. Silverberg, Scheduling jobs with fixed start and end times, Discrete Appl Math 18 (1987), 1–8.

[4] C. Barnhart, C. Hane, and P.H. Vance, Integer multicommodity flow problems, Lecture Notes Comput Sci 1084 (1996), 58–71.

[5] C. Barnhart, 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), 316–329.

[6] J.E. Beasley and B. Cao, A dynamic programming based algorithm for the crew scheduling problem, Comput Oper Res 25 (1998), 567–582.

[7] G.B. Dantzig and D.R. Fulkerson, Minimizing the number of tankers to meet a fixed schedule, Nav Res Logistics Quart 1 (1954), 217–222.

[8] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis, "Time constrained routing and scheduling," Handbooks in operations research and management science, Volume 8, Network Routing, M. Ball, T. Magnanti, C. Monma, and G. Nemhauser (Editors), Elsevier, Amsterdam, 1994.

[9] V.R. Dondeti and H. Emmons, Fixed job scheduling with two types of processors, Oper Res 40 (1992), S76–S85.

[10] S. Even, A. Itai, and A. Shamir, On the complexity of timetable and multicommodity flow problems, SIAM J Comput 5 (1976), 691–703.

[11] D.B.C. Faneyte, Kranen op (de kortste) weg, Master's thesis, University of Maastricht, 1997 (in Dutch).

[12] M.L. Fisher, K.O. Jörnsten, and O.B.G. Madsen, Vehicle routing with time windows: Two optimization algorithms, Oper Res 45 (1997), 488–492.

[13] L.R. Ford and D.R. Fulkerson, Flows in networks, Princeton University Press, Princeton, NJ, 1962.

[14] Y. Ge and Y. Yih, Crane scheduling with time windows in circuit board production lines, Int J Prod Res 33 (1995), 1187–1199.

[15] M.R. Garey and D.S. Johnson, Computers and Intractability: A guide to the theory of NP-completeness, Freeman, San Francisco, 1979.

[16] A.W.J. Kolen, J.K. Lenstra, and C.H. Papadimitriou, Interval scheduling problems, unpublished manuscript, 1986.

[17] L.G. Kroon, A. Sen, H. Deng, and A. Roy, The optimal cost chromatic partition problem for trees and interval graphs, Report 221, Erasmus University, Rotterdam, 1995.

[18] A. Mingozzi, M.A. Boschetti, S. Ricciardelli, and L. Bianco, A set partitioning approach to the crew scheduling problem, Oper Res 47 (1999), 873–888.

[19] M. Pióro and P. Gajowniczek, Solving multicommodity integral flow problems by simulated allocation, Telecommun Sys 7 (1997), 17–28.

[20] M. Sol, Column generation techniques for pickup and delivery problems, Ph.D. Thesis, Technical University of Eindhoven, Eindhoven, The Netherlands, 1994.

[21] A. Srivastav and P. Stangier, "Integer multicommodity flows with reduced demands," Proceedings of the first European Symposium on Algorithms, T. Lengauer, (Editor), Lecture Notes in Computer Science 726, Springer-Verlag, New York, 1993, pp. 360–371.

[22] F. Vanderbeck, Decomposition and column generation for integer programs, Ph.D. Thesis, Catholic University of Louvain, Louvain, Belgium, 1994.