Computers & Operations Research 39 (2012) 3305–3315

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A branch-price-and-cut algorithm for the workover rig routing problem

Glaydston Mattos Ribeiro [a], Guy Desaulniers [b,*], Jacques Desrosiers [c]

[a] Department of Engineering, Federal University of Espírito Santo, Brazil
[b] Department of Mathematics and Industrial Engineering and GERAD, École Polytechnique de Montréal, Canada
[c] Department of Management Sciences and GERAD, HEC Montréal, Canada

ARTICLE INFO

Available online 19 April 2012

Keywords:
Workover rig routing
Vehicle routing
Branch-price-and-cut
Column generation
Tabu search column generator
ng-Paths
Subset-row inequalities

ABSTRACT

In an onshore oil field, the productivity of oil wells decreases when they require maintenance. To restore full productivity at a well, it must be visited by a specially equipped vehicle, called a workover rig. Given a set of wells needing maintenance and a heterogeneous fleet of workover rigs, the workover rig routing problem (WRRP) consists of finding rig routes that minimize the total production loss of the wells over a finite horizon. The wells have different loss rates, require various services, and may not be serviced within the horizon due to rig availability. The rigs have initial positions and do not have the same equipment. This paper presents the first exact algorithm for the WRRP, namely, a branch-price-and-cut algorithm that relies on some of the most recent techniques introduced for the vehicle routing problem with time windows. Our computational experiments show that this exact algorithm can solve practical-sized instances in reasonable computational times.

1. Introduction

Onshore oil fields can contain a large number of wells spread over a wide region (more than 1000 in certain cases). When the productivity of a well decreases due to a malfunction, a request for maintenance is issued. This request specifies a production loss rate and the maintenance services (such as cleaning, reinstatement, or stimulation) required to restore full productivity. To maintain a well, a specially equipped vehicle, called a workover rig (or rig, for short), must be sent to the well. Because the available rigs do not have the same equipment, a request is serviceable only by a subset of the rigs. The maintenance requests are issued dynamically, and the rig routes (sequences of wells to service) are planned periodically at every P time periods in a rolling horizon fashion. When a planning session is performed, a finite horizon of H time periods with $H \geq P$ is considered for servicing the requests that are known at this time. In particular, these requests include the portions of the routes previously planned that will not be completed before the beginning of the horizon. Given the limited number of rigs available and the finiteness of the horizon, it might, however, be impossible to plan the service of all the maintenance requests within the horizon. The unserved ones are postponed to the next planning session together with the requests to be issued in the next P periods, that is, until the next planning session.

In this paper, we address the static problem, called the workover rig routing problem (WRRP), that must be solved in a given planning session of the rolling horizon. Given a set of wells requesting maintenance and a heterogeneous fleet of rigs, each with a specific equipment and an initial position, the WRRP consists of determining feasible routes for the rigs such that the total production loss of the wells over the next H time periods is minimized. A route is feasible for a rig if it starts at the rig's initial position, its duration (including service times) does not exceed H periods, and it services only wells for which the rig is equipped. A route ends at the location of its last visited well.

Fig. 1 illustrates a 100-well, 10-rig instance of the WRPP and its computed optimal solution. Part (a) shows the locations of the wells and the initial positions of the rigs while part (b) depicts the computed optimal set of routes, where the number of wells serviced by each route appears in parentheses. Notice the open aspect of the rig routes and that 10 wells are not planned to be serviced within the horizon.

The WRRP arises on onshore oil fields and involves enormous monetary sums. For example, Aloise et al. [1] report that the rental of 10 rigs costs yearly 10 millions USD to Petrobras, a Brazilian company. In the literature, various heuristics have been developed for the WRPP: a variable neighborhood search [1]; a tabu search and an iterated local search [20]; a greedy randomized adaptive search procedure with path-relinking [21]; and recently, a clustering search and an adaptive large neighborhood search [22]. To the best of our knowledge, no exact solution methods have been proposed.

* Corresponding author.
E-mail addresses: glaydstonribeiro@ceunes.ufes.br (G. Mattos Ribeiro), guy.desaulniers@gerad.ca (G. Desaulniers), jacques.desrosiers@hec.ca (J. Desrosiers).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.04.010

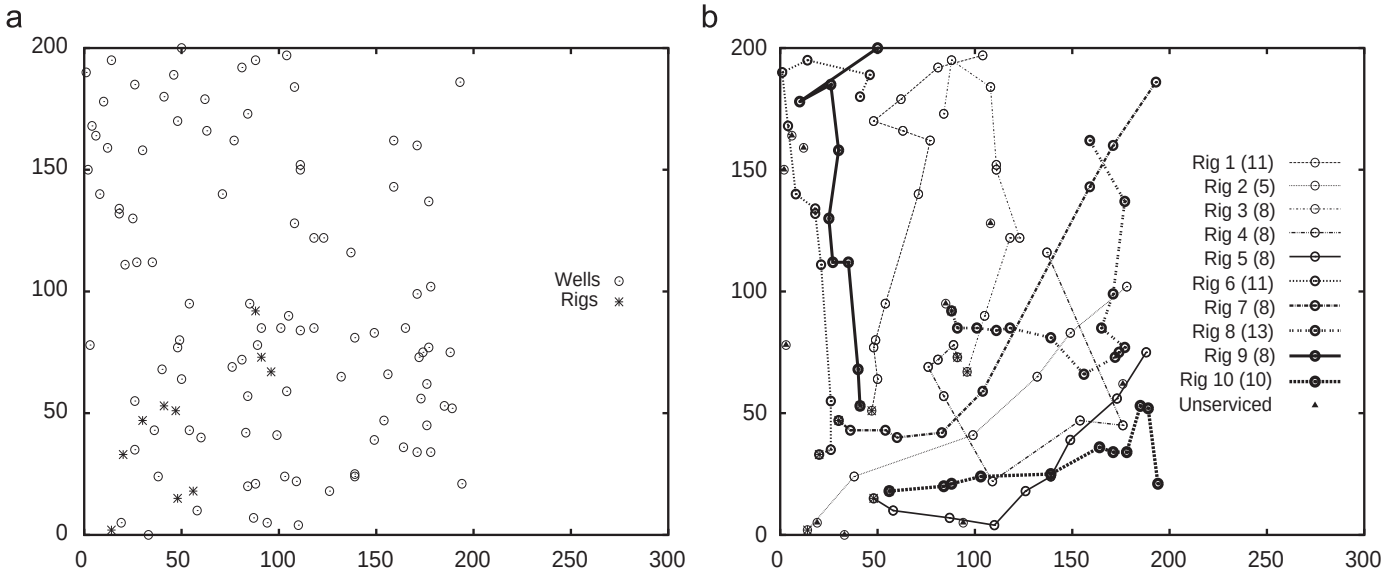

Fig. 1. A 100-well, 10-rig instance and its computed optimal solution. (a) Well and rig locations. (b) Optimal open routes.

The WRRP has similarities with various vehicle routing problems (VRPs). In particular, the vehicles do not have to return to a depot at the end of their routes as in the open VRP [18]. Also, certain wells may remain unserviced due to the limited horizon length. The serviced wells are selected on the basis of the travel time required to reach them and the production losses that can be saved. This aspect appears in VRPs with profits (equivalent to the losses saved) such as the team orienteering and the profitable tour problems [2]. In those VRPs, the profits gained by visiting a customer is known a priori whereas, in the WRRP, the production loss saved at a well depends on the time at which it is serviced. The objective function is, thus, time dependent as in the multi-depot vehicle scheduling problem with time windows and linear waiting costs [8]. In this problem, a cost per minute (representing the driver salary) is charged when a vehicle waits before starting a task due to a time window restricting its start of service time. In this case, late arrival times reduce waiting costs. At the opposite, early arrival times must be favored in the WRRP to minimize the total production loss. Finally, time plays an important role in the WRRP, both in the objective function and in the deadline constraint yielded by the end of the horizon, as in many VRPs with time windows such as the VRP with time windows (VRPTW) (see [10] for a recent survey).

The main contribution of this paper is to present the first exact algorithm for the WRPP, namely, a branch-price-and-cut (BPC) algorithm. We chose this type of algorithm because it is known to be a state-of-the-art methodology for solving VRPs involving time-dependent constraints or objective function. However, when route feasibility is weakly constrained as in the WRRP, BPC can struggle to reach and prove optimality because the shortest path labeling algorithm generating the routes as needed must get rid of numerous negative (reduced) cost cycles. In our case, the cycles should be easier to handle because the production loss at a well increases with time, yielding higher reduced cost cycles as time progresses.

The proposed BPC algorithm relies on some of the most recent techniques introduced for the VRPTW, including a tabu search column generator [9], an *ng*-path relaxation [3], and the subset-row inequalities [17]. Through a series of computational experiments, we assess the effectiveness of these techniques and show that the overall BPC algorithm can solve practical-sized instances (up to 200 wells and 10 rigs) in reasonable computational times (less than 1 h).

The rest of this paper is structured as follows. In the next section, we provide a time-constrained arc-flow model for the WRRP. Section 3 gives an equivalent set packing path-flow formulation that is suitable for BPC. We also describe the subproblems, the algorithms used to generate columns, the implemented cutting planes, and the branching strategies. Computational results are reported in Section 4, while conclusions are drawn in Section 5.

2. A time-constrained arc-flow model

In this section, we provide a time-constrained arc-flow model for the WRRP, a specialization of the unified model proposed by Desaulniers et al. [7] for deterministic time-constrained VRPs. Besides precising the WRRP definition, this model is *helpful* to define the column generation subproblems to be used in the BPC algorithm.

2.1. Networks

Let K be the set of available rigs. A network $G^k := (N^k, A^k)$ is associated with each rig $k \in K$, where N^k and A^k denote its sets of nodes and arcs, respectively. Set N^k contains three types of nodes: *source*, *sink*, and *task*. The source node $o(k)$ represents the initial position of rig k, while the sink node $d(k)$ represents the end of the horizon. A task corresponds to a maintenance service to be performed on a well and it is represented by a task node. The set of task nodes is denoted by W while the subset of task nodes that can be accomplished by rig k is denoted $W^k \subseteq W$. Therefore, $N^k := \{o(k), d(k)\} \cup W^k$.

Set A^k contains four types of arcs: *empty*, *start*, *end*, and *inter-task*. There is an empty arc linking $o(k)$ to $d(k)$ that represents an empty route when rig k is not required in the solution. There is a start arc $(o(k), j)$ between $o(k)$ and each task node $j \in W^k$. Such an arc represents the travel of rig k from its initial position to well j followed by the service at j. There is also an end arc $(i, d(k))$ between each task node $i \in W^k$ and $d(k)$ indicating that the route of rig k ends at well i. Finally, there is an inter-task arc (i,j) between each pair of task nodes $i,j \in W^k$ that represents the movement of rig k from well i to well j followed by the service

at j. Therefore, $A^k := \{(o(k),d(k))\} \cup (o(k) \times W^k) \cup (W^k \times d(k)) \cup (W^k \times W^k)$.

The planning horizon is divided into H disjoint time periods, numbered from 1 to H. With each arc $(i,j) \in A^k$, we associate a nonnegative parameter t_{ij} that specifies the elapsed time (travel time plus service time, if any) on arc (i,j). We assume that t_{ij} is a nonnegative integer number of time periods and that the parameters t_{ij}, $(i,j) \in A^k$, satisfy the triangle inequality. Note that any arc (i,j) such that $t_{ij} > H$ can be deleted from its arc set. Finally, with each node $i \in W^k$, we associate a positive parameter ℓ_i that indicates the production loss rate per time period at node i.

2.2. Mathematical formulation

The proposed time-constrained arc-flow model relies on two main types of variables: for each arc $(i,j) \in A^k$, $k \in K$, we define a binary arc-flow variable X_{ij}^k that takes value 1 if arc (i,j) is used by rig k and zero otherwise; for each node $i \in W^k \cup \{o(k)\}$, $k \in K$, we define a time variable $T_i^k \in [0,H]$ that indicates the time period at which rig k completes service at node i if it services it. We impose $T_i^k = 0$ if $i = o(k)$ or if $i \in W^k$ and well i is not serviced by rig k. Note that no time variables $T_{d(k)}^k$, $k \in K$, need to be defined at the sink nodes because $t_{i,d(k)} = 0$ for all $k \in K$.

The time variables allow to determine the production loss at each well. Indeed, if rig k services well i, that is, if $T_i^k > 0$, then well i returns to full productivity in period $T_i^k + 1$ and the production loss for this well is given by $\ell_i T_i^k$. On the other hand, if well i is not serviced by any rig during the horizon, that is, if $T_i^k = 0$ for all $k \in K$, then its production loss is equal to $\ell_i H$.

To facilitate the presentation of the model, we introduce two types of auxiliary variables: for each node $i \in W^k$, $k \in K$, we define a binary variable $X_i^k := \sum_{j:(j,i) \in A^k} X_{ji}^k$ that takes value 1 if well i is serviced by rig k and 0 otherwise; we also define a nonnegative saving variable S_i^k that is equal to the production loss saved at well i if serviced by rig k. Imposing the following constraints on the time variables:

$$0 \le T_i^k \le H X_i^k, \quad \forall k \in K,\ i \in W^k,$$

one can express the saving variables as follows:

$$S_i^k := \ell_i (H X_i^k - T_i^k), \quad \forall k \in K,\ i \in W^k.$$

Notice that $S_i^k = 0$ if rig k does not service well i, that is, if $X_i^k = 0$. With the saving variables, the total production loss is given by

$$H \sum_{i \in W} \ell_i - \sum_{k \in K} \sum_{i \in W^k} S_i^k,$$

where the first term corresponds to the maximal production loss and the second to the total losses saved by the rigs.

Using this notation, the WRRP can be formulated as follows:

$$\text{Minimize} \quad H \sum_{i \in W} \ell_i - \sum_{k \in K} \sum_{i \in W^k} S_i^k = H \sum_{i \in W} \ell_i - \sum_{k \in K} \sum_{i \in W^k} \ell_i (H X_i^k - T_i^k) \quad (1)$$

$$\text{subject to:} \quad \sum_{k \in K} \sum_{j:(i,j) \in A^k} X_{ij}^k \le 1, \quad \forall i \in W, \quad (2)$$

$$\sum_{j:(o(k),j) \in A^k} X_{o(k),j}^k = \sum_{i:(i,d(k)) \in A^k} X_{i,d(k)}^k = 1, \quad \forall k \in K, \quad (3)$$

$$\sum_{j:(i,j) \in A^k} X_{ij}^k - \sum_{j:(j,i) \in A^k} X_{ji}^k = 0, \quad \forall k \in K,\ i \in W^k, \quad (4)$$

$$X_{ij}^k (T_i^k + t_{ij} - T_j^k) \le 0, \quad \forall k \in K,\ (i,j) \in A^k,\ j \ne d(k), \quad (5)$$

$$0 \le T_i^k \le H X_i^k, \quad \forall k \in K,\ i \in W^k, \quad (6)$$

$$T_{o(k)}^k = 0, \quad \forall k \in K, \quad (7)$$

$$X_{ij}^k \in \{0,1\}, \quad \forall k \in K,\ (i,j) \in A^k, \quad (8)$$

$$X_i^k = \sum_{j:(j,i) \in A^k} X_{ji}^k, \quad \forall k \in K,\ i \in W^k, \quad (9)$$

$$S_i^k = \ell_i (H X_i^k - T_i^k), \quad \forall k \in K,\ i \in W^k. \quad (10)$$

Objective function (1) aims at minimizing the total production loss over the horizon. Constraints (2) ensure that each well (*task* node) is serviced at most once by a rig. For each rig k, constraints (3) and (4) correspond to the classical network flow conservation constraints of a path originating at source node $o(k)$ and ending at sink node $d(k)$. The nonlinear constraints (5) express the compatibility requirements between arc-flow and time variables and ensure subtour elimination. Constraints (6) impose a maximal duration of H periods to the routes. They also ensure that the time variables take value 0 when the corresponding well is not serviced by the corresponding rig. Constraints (7) set the starting time of all routes to 0. Constraints (8) impose binary values to the arc-flow variables. Finally, the auxiliary variables are defined in (9) and (10). They can easily be removed from the model by replacing them in (1) and (6) with their expressions in terms of the arc-flow and time variables.

The objective function (1) and constraints (3)–(10) are separable by rig $k \in K$. Therefore, model (1)–(10) is well suited for a solution method based on a decomposition process, as the one presented in the following section.

3. Branch-price-and-cut

Model (1)–(8) contains nonlinear constraints (5) that can be linearized. However, such a linearization typically yields a very weak linear relaxation. To obtain a stronger relaxation, we propose to apply the Dantzig–Wolfe decomposition process [6] on model (1)–(8). This process yields a set packing model that can be solved by a BPC method. This method (see [19,12]) consists of a column generation method embedded into a branch-and-cut method. Column generation is used to compute lower bounds in the search tree, cuts are added to tighten these bounds, and branching decisions are imposed to derive integer solutions. All these components are described below.

3.1. A set packing model

Following the Dantzig–Wolfe decomposition process, the service request constraints (2) of model (1)–(10) are kept in the master problem while the constraints (3)–(10), separable by rig $k \in K$, define the domain of the $|K|$ pricing subproblems. This decomposition gives rise to an alternative path-flow formulation for the WRRP that corresponds to a set packing model.

Let R^k be the set of feasible routes for rig $k \in K$. For each route $r \in R^k$, denote by s_r^k the total production loss saved by this route and by a_{ir}^k, $i \in W^k$, a binary parameter equal to 1 if well i is visited in route r, and 0 otherwise. Parameter s_r^k is computed as the sum of the loss saved at each serviced well

$$s_r^k := \sum_{i \in W^k} \ell_i (H a_{ir}^k - t_{ir}^k),$$

where parameter t_{ir}^k is equal to the time period at which rig k completes service at well i in route r if it services it and to zero otherwise. Finally, for each rig $k \in K$ and each route $r \in R^k$, define a binary path variable Y_r^k indicating whether or not route r for rig k is selected in the solution.

With this notation, the WRRP can be formulated as follows:

$$\text{Minimize} \quad H\sum_{i\in W}\ell_i - \sum_{k\in K}\sum_{r\in R^k} s_r^k Y_r^k \tag{11}$$

$$\text{subject to}: \quad \sum_{k\in K}\sum_{r\in R^k} a_{ir}^k Y_r^k \le 1, \quad \forall i\in W, \tag{12}$$

$$\sum_{r\in R^k} Y_r^k = 1, \quad \forall k\in K, \tag{13}$$

$$Y_r^k \in \{0,1\}, \quad \forall k\in K,\ r\in R^k. \tag{14}$$

Objective function (11) aims at minimizing the total production loss over the horizon, which is again computed as the maximal production loss minus the sum of the losses saved by the rigs. Set packing constraints (12) ensure that each well is serviced at most once by a rig. Convexity constraints (13) indicate that one route must be determined for each available rig. Finally, binary requirements (14) are imposed on the path variables. They are derived from the binary requirements (8) on the arc-flow variables (see [16]).

3.2. Column generation

In practice, model (11)–(14) contains a huge number of path variables. To solve its linear relaxation, called the *master problem*, we apply column generation. Column generation is an iterative method that solves, at each iteration, a master problem restricted to a subset of its variables, called the *restricted master problem* (RMP), and one or several *pricing subproblems*. Solving the RMP (using the simplex algorithm in our case) provides a pair of optimal primal and dual solutions. The primal solution is also deemed optimal for the master problem if there exist no more negative reduced cost variables among those not considered in the RMP. The role of the subproblems is to check this condition by searching for negative reduced cost variables (columns). If no such columns exist, the solution process stops. Otherwise, negative reduced cost columns identified by the subproblems are added to the RMP before starting a new iteration.

For the WRRP, there is one pricing subproblem per workover rig $k\in K$. It corresponds to an elementary shortest path problem, defined on network G^k, with an additional constraint that limits the duration of a route to a maximum of H periods and a time-dependent objective function that is defined as follows.

Let $\alpha_i \le 0, i\in W$, and $\sigma^k, k\in K$, be the dual variables corresponding to constraints (12) and (13) of the master problem, respectively. The reduced cost $\bar{s}_r^k$ of variable Y_r^k, $k\in K$, $r\in R^k$, is computed as

$$\bar{s}_r^k := -s_r^k - \sum_{i\in W^k}\alpha_i a_{ir}^k - \sigma^k = -\sum_{i\in W^k}\ell_i(Ha_{ir}^k - t_{ir}^k) - \sum_{i\in W^k}\alpha_i a_{ir}^k - \sigma^k.$$

The goal of the pricing subproblem for rig k is to find a route $r\in R^k$ with the least reduced cost $\bar{s}_r^k$. Using the notation of the time-constrained arc-flow model (1)–(10), its objective function can thus be written as

$$\text{Minimize} \quad -\sum_{i\in W^k}\ell_i(HX_i^k - T_i^k) - \sum_{i\in W^k}\alpha_i X_i^k - \sigma^k, \tag{15}$$

or, equivalently

$$\text{Minimize} \quad \sum_{(i,j)\in A^k}(\ell_j(T_j^k - H) - \alpha_j)X_{ij}^k, \tag{16}$$

by defining $\ell_{d(k)} := 0$ and $\alpha_{d(k)} := \sigma^k$, and exploiting constraints (6). Its domain is given by the constraints (3)–(10) associated with rig k. The values of the variables X_i^k, T_i^k, and S_i^k for the routes $r\in R^k$ found by the subproblem provide, respectively, the parameters a_{ir}^k, t_{ir}^k, and s_{ir}^k, associated with the variables Y_r^k added to the RMP.

Forward labeling algorithm: The elementary constrained shortest path problem associated with rig $k\in K$ can be solved using a forward labeling algorithm (see [15]). In this algorithm, labels represent partial paths that are extended, using so-called *extension functions*, in network G^k from the source node $o(k)$ toward the sink node $d(k)$. Each label E (a vector with $2+|W^k|$ components) stores the reduced cost of the partial path $Z(E)$, the elapsed time $T(E)$ along this path and, for each well $i\in W^k$, a binary component $V_i(E)$ that takes value 1 if well i has already been visited along the path associated with E or if it is unreachable from E. A well i is said to be *unreachable* from E if time does not allow to visit it within the planning horizon in any extension of E [14]. The extension along an arc $(i,j)\in A^k$ of a label E representing a partial path ending at node $i\in N^k$ proceeds as follows to create a new label E' representing a partial path ending by the arc (i,j)

$$T(E') := T(E) + t_{ij}, \tag{17}$$

$$Z(E') := Z(E) + \ell_j(T(E') - H) - \alpha_j, \tag{18}$$

$$V_w(E') := \begin{cases} V_w(E)+1 & \text{if } w = j \ne d(k), \\ \max\{V_w(E), U_w(E')\} & \text{otherwise,} \end{cases} \quad \forall w\in W^k, \tag{19}$$

where $U_w(E')$ indicates whether or not w is unreachable from E' (that is, $U_w(E')$ is equal to 1 if $T(E')+t_{jw} > H$ and 0 otherwise). Label E' corresponds to a feasible path if $T(E')\le H$ and $V_w(E')\le 1$ for all $w\in W^k$. It is discarded if this is not the case.

To avoid enumerating all feasible paths in G^k, the algorithm keeps only the Pareto-optimal labels (i.e., the labels that are not proven to be dominated by other labels associated with paths ending at the same node). Because all extension functions are non-decreasing, the following label dominance criterion can be used.

Proposition 1 (*Desaulniers et al. [7]*). *Let E and E' be two labels representing partial paths ending at the same node. Label E dominates label E' if*

$$Z(E) \le Z(E'), \tag{20}$$

$$T(E) \le T(E'), \tag{21}$$

$$V_w(E) \le V_w(E'), \quad \forall w\in W^k, \tag{22}$$

and at least one of these inequalities is strictly satisfied.

Dominated labels are discarded. Furthermore, when equality holds for all label components in the above criterion, one of the two labels is discarded.

To speed up the labeling algorithm, two strategies were developed recently. The first [23] consists of using a *bounded bidirectional search* that extends labels *both forwardly* (*from the source node*) *and backwardly* (*from the sink node*) *before joining them together to yield* feasible source-to-sink paths. In our case, this strategy cannot be applied because the backward extension functions are not non-decreasing and a straightforward adaptation of the dominance rule stated in Proposition 1 is not valid.

The second acceleration strategy [5,24], called *decremental search space*, is iterative. It starts by solving the pricing subproblem considering labels without the components $V_w(\cdot)$, $w\in W^k$, and a dominance criterion involving only (20) and (21). If no negative reduced cost paths are generated, then the algorithm stops without finding any variables to add to the RMP. Otherwise, it checks whether there exist elementary paths among the negative reduced cost paths found. If so, the algorithm stops and the variables associated with these paths are added to the RMP. Otherwise (all negative reduced cost paths contain cycles), we select a node $i\in W^k$ that is visited the most often in a path with the least reduced cost and add the corresponding

component $V_i(\cdot)$ in the labels and in the dominance rule. A new iteration then starts by solving the pricing subproblem again. For our tests, we use decremental search space. However, a component $V_w(\cdot)$ that is added for a subproblem in a column generation iteration is never omitted afterwards for the other subproblems or the subsequent column generation iterations [9].

Path relaxations: Due to the elementary constraints (see, for example, [13]), the time-constrained pricing subproblems are NP-hard. They can, therefore, be very difficult to solve in practice. Relaxations allowing cycles in the paths have been proposed to ease their solution at the expense of yielding a weaker lower bound at each node of the search tree. Indeed, in this case, the sets of routes R_k, $k \in K$, enlarge. Furthermore, if a route r for rig k visits well i several times, the coefficient a_{ir}^k associated with the variable Y_r^k in model (11)–(14) is set equal to the number of times well i is visited and the coefficient s_r^k cumulates losses saved at each of these visits.

In our computational experiments, we consider two path relaxations. In the first relaxation, all cycles are allowed except the 2-cycles, that is, a path cannot contain a subsequence i–j–i where i and j are task nodes. In this case, no components $V_w(\cdot)$ are considered in the labels and the dominance rule, increasing the number of labels dominated. The dominance rule must, however, be modified to take into account the 2-cycle elimination (see [15] for details).

Proposed by Baldacci et al. [4], the second path relaxation, called the *ng*-path relaxation, associates a neighborhood $W_i^k \subset W^k$ with each task node $i \in W^k$. This neighborhood contains node i and its $\mathcal{N}^{ng}$ closest nodes, where $\mathcal{N}^{ng}$ is a predefined value. A feasible *ng*-path can contain a cycle starting and ending with task node i if and only if this cycle includes another task node j such that $i \notin W_j^k$ (that is, i is not in the neighborhood of j, thus returning to i from j is a long detour).

The *ng*-paths can be handled in the forward labeling algorithm described above. To do so, we replace the extension function (19) by:

$$V_w(E') := \begin{cases} V_w(E)+1 & \text{if } w = j \neq d(k), \\ \max\{V_w(E), U_w(E')\} & \text{if } w \in W_j^k \setminus \{j\}, \\ 0 & \text{otherwise} \end{cases} \qquad \forall w \in W^k, \qquad (23)$$

where $W_{d(k)}^k := \emptyset$. Thus, in each label E associated with a path ending at a task node i, all the components $V_w(E)$, $w \in W^k \setminus W_i^k$, are equal to 0. At this node, the dominance rule (22) only needs to compare $\mathcal{N}^{ng}$ components $V_w(\cdot)$, increasing the chances of identifying dominated labels if $\mathcal{N}^{ng}$ is relatively small. On the other hand, if $\mathcal{N}^{ng}$ is too small, the generated paths might contain many cycles, yielding very weak lower bounds. When generating *ng*-paths, we also apply a 2-cycle elimination procedure. For our main computational experiments, we set $\mathcal{N}^{ng} = 10$, but we also tested $\mathcal{N}^{ng} = 5$ and $\mathcal{N}^{ng} = 15$ (see Section 4).

When $\mathcal{N}^{ng}$ is not too small, the pricing subproblems remain somewhat difficult to solve. However, in a column generation algorithm, there is no need to solve the pricing subproblems exactly at each column generation iteration. They can be solved heuristically as long as negative reduced cost columns are found. When this is not the case, an exact algorithm must be invoked to prove optimality. We propose two heuristics (described below) for generating columns rapidly: a heuristic labeling algorithm and a multi-start tabu search algorithm. At each column generation iteration, the multi-start tabu search heuristic tries to generate negative reduced cost columns. If it fails, then the heuristic labeling algorithm is invoked. If this algorithm also fails for all pricing subproblems, then the exact labeling algorithm is executed.

Heuristic labeling algorithm: Consider the pricing subproblem associated with rig $k \in K$. The heuristic labeling algorithm consists of applying the forward labeling algorithm on a reduced network obtained by removing inter-task arcs from network $G^k := (N^k, A^k)$ that does not seem promising. The elimination procedure depends on the current dual values α_i, $i \in W^k$, and is applied at every column generation iteration. Let $\mathcal{A}^{min}$ be a predefined positive parameter value. For each task node $i \in W^k$, a minimum of $\mathcal{A}^{min}$ outgoing arcs and $\mathcal{A}^{min}$ incoming arcs are kept (unless there are fewer in A^k) besides the start and end arcs incident to i. To select the inter-task arcs to remove, we first associate with each inter-task arc (i,j) a value

$$v_{ij} := \ell_j(0.5(t_{o(k),i}+t_{ij}+H)-H)-\alpha_j = 0.5\ell_j(t_{o(k),i}+t_{ij}-H)-\alpha_j$$

that simulates the reduced cost of an arc as given in (16). This estimation is based on the average loss saved at well j when preceded by the service at well i, where $t_{o(k),i}+t_{ij}$ *(resp. H) is the earliest (resp. latest) completion time at* j, *and* $0.5(t_{o(k),i}+t_{ij}+H)$ *is, thus, the average completion time (assuming a uniform distribution on the interval* $[t_{o(k),i}+t_{ij},H]$). Then, for every task node, its incoming arcs and its outgoing arcs are ranked in increasing order of their v_{ij} value. Denote by ρ_{ij}^{out} (resp. ρ_{ij}^{in}) the rank of an inter-task arc $(i,j) \in A^k$ as an outgoing (resp. incoming) arc of node i (resp. node j). Arc (i,j) is removed from G^k if $\min\{\rho_{ij}^{out}, \rho_{ij}^{in}\} > \mathcal{A}^{min}$.

Based on the results of preliminary tests, we chose to set $\mathcal{A}^{min} = 10$ for our main computational experiments.

Multi-start tabu search algorithm: As empirically shown by Desaulniers et al. [9], tabu search can be an efficient heuristic to generate negative reduced cost columns when there exist many such columns (that is, in all column generation iterations except the last few ones). Tabu search, a well-known metaheuristic for solving combinatorial optimization problems, considers an initial solution that is gradually modified according to a sequence of moves. For solving a pricing subproblem, the proposed tabu search algorithm relies on two types of moves: insertion of a well into the current solution path (according to a best insertion policy) and removal of a well from it. At each iteration of the algorithm, all non-tabu feasible moves are evaluated and the one yielding the least reduced cost path is retained. This best move is then stored in a tabu list of length $\mathcal{L}^{max}$: its reverse move is forbidden (tabu) for the next $\mathcal{L}^{max}$ iterations. However, we apply an aspiration criterion that consists of also evaluating all tabu moves. If such a move yields a new overall best solution, it is accepted regardless of the tabu consideration. To simplify the computations (especially to *avoid handling* the *ng*-path cycle restrictions), the solution space *of the tabu search column generator is restricted to the set of feasible elementary paths (that is, only the labeling algorithms can generate paths with cycles when they are allowed*). For every insertion evaluated, path feasibility with respect to its length is checked. No check is required for removals.

As suggested by Desaulniers et al. [9], we combine the tabu search algorithm with a multiple start procedure. The tabu search algorithm is executed for a fixed number of iterations $\mathcal{I}^{max}$ starting from different initial paths. The set of initial paths is given by the paths associated with the basic variables in the current RMP solution. All these path variables are good initial candidates because they have a zero reduced cost. Note that each initial path is associated with a pricing subproblem and determines the network G^k on which to apply the algorithm.

For our tests, the parameters $\mathcal{L}^{max}$ and $\mathcal{I}^{max}$ were set to 8 and 30, respectively.

3.3. Cutting planes: subset-row inequalities

The subset row inequalities introduced by Jepsen et al. [17] for solving the VRPTW are a set of valid inequalities directly defined on the master problem variables, whose dual values cannot be directly transferred to the arc costs of the pricing subproblems. These inequalities are special cases of the Chvátal–Gomory rank-1 valid inequalities for the set packing polytope which is defined by

(12) and the continuous relaxation of (14). They are given by

$$\sum_{k \in K} \sum_{r \in R^k} \left\lfloor \frac{1}{q} \sum_{i \in Q} a_{ir}^k \right\rfloor Y_r^k \leq \left\lfloor \frac{|Q|}{q} \right\rfloor, \quad \forall Q \subseteq W,\ 2 \leq q \leq |Q|, \tag{24}$$

where Q is a subset of the wells (task nodes). As in Jepsen et al. [17], we focus on the cuts defined for subsets Q of three task nodes and $q=2$. For elementary paths, this results in a right-hand side member of 1 and binary coefficients equal to 1 if and only if a route r for rig k visits at least two of the three wells in the subset Q defining the cut. The cuts can then be rewritten as follows:

$$\sum_{k \in K} \sum_{r \in R_Q^k} Y_r^k \leq 1, \quad \forall Q \subseteq W \text{ such that } |Q| = 3, \tag{25}$$

where $R_Q^k \subseteq R^k$ is the subset of routes servicing at least two wells in Q. For paths with cycles, the coefficients $\lfloor (1/q) \sum_{i \in Q} a_{ir}^k \rfloor$ can be larger than 1.

As shown by the computational results reported in Jepsen et al. [17], the use of the subset row inequalities can significantly improve the lower bound computed at the root node of the search tree. However, adding these cuts increases the computational complexity of the pricing subproblems. Indeed, in a label of the labeling algorithm, each cut requires a specific new component for computing the number of task nodes in Q visited along the path. This component is necessary to subtract adequately the dual variable of the associated cut in the computation of the reduced cost of a route. For more details about this procedure, see Desaulniers et al. [11].

When a large number of subset row cuts are added to the RMP, the labeling algorithm can become quite slow. To avoid this negative impact, we use the following restrictions that were proposed by Desaulniers et al. [9]. First, a maximum of $\mathcal{C}^{max}$ cuts are added simultaneously, the most violated ones that respect the next condition. Second, a well can be included in at most $\mathcal{W}^{max}$ of the subsets defining these cuts. Third, cuts are added only if the most violated cut is violated by at least $\mathcal{V}^{min}$. Finally, cuts are added only at the root node of the search tree.

For our tests, the parameters $\mathcal{C}^{max}$, $\mathcal{W}^{max}$, and $\mathcal{V}^{min}$ were set to 20, 5, and 0.1, respectively.

3.4. Branching strategies

Branching decisions are taken on binary aggregated arc-flow variables X_{ij} defined as follows:

$$X_{ij} := \sum_{k \in K:(i,j) \in A^k} X_{ij}^k = \sum_{k \in K} \sum_{r \in R^k} b_{ijr} Y_r^k, \quad \forall (i,j) \in \bigcup_{k \in K} A^k \text{ such that } i,j \in W, \tag{26}$$

where b_{ijr} is equal to the number of times that arc (i,j) is used in route r.

Given a fractional solution, the aggregated arc-flow variable X_{ij} to branch on is selected as the 1 with the value closest to 0.5. The branching decisions ($X_{ij}=0$ and $X_{ij}=1$) are treated by modifying the networks of the subproblems. To impose $X_{ij}=0$, we remove the corresponding arc (i,j) from every network G^k, $k \in K$, containing it. To impose $X_{ij}=1$, we remove all arcs leaving node i or entering node j, except arc (i,j), from every network G^k, $k \in K$, containing at least one of these two nodes. In this case, because i and j are both task nodes, one of the two set packing constraints (12) associated with them can be removed from the master problem. This type of branching decisions reduces the size of the model and does not change its structure.

The branch-and-bound search tree is explored using a best-first procedure.

4. Computational experiments

To the best of our knowledge, there are no publicly available instances of the WRRP with a heterogeneous fleet of rigs and a finite horizon. However, the instances of Neves [20] for a homogeneous fleet are available. These instances provide well locations, well production loss rates, well service times, travel times between all pairs of locations, and rig initial positions. From these instances and for the case with a heterogeneous fleet, we created 80 instances of practical sizes involving 100 or 200 wells, five or 10 rigs, and a horizon length of $H=200$ or 300 time periods (10 instances for each possible parameter combination). The missing data were randomly generated as follows. For each well $w \in W$, a uniform random integer ζ_w is drawn in [1,5] to assign a service level required at well w. For each rig $k \in K$, a level of equipment γ_k is randomly chosen following a uniform discrete distribution in [3,5]. Given these service and equipment levels, we consider that a rig k can service a well w if and only if $\zeta_w \leq \gamma_k$. The horizon lengths $H=200$ and $H=300$ correspond to approximately 14 and 21 days, respectively. The instances with $H=300$ are identical to the instances with $H=200$ except for the horizon length. All tested instances can be found on the web page www.gerad.ca/~guyd/wrrp.html.

All computational tests were run on a Linux PC with an Intel Quad Core processor of 2.66 GHz, using a customized version of the Gencol software (version 4.5) which is an implementation of a branch-and-price algorithm commercialized by Kronos Inc. Gencol uses IBM Cplex Solver version 12.2 to solve the RMPs. A 1-h time limit is imposed for solving the instances. Note that all reported lower bounds and optimal values do not include the constant term in objective function (11). Thus, they correspond to the negative of the total loss saved by the rigs.

4.1. Evaluating algorithm components

In this section, we present computational results that permit to evaluate the benefits of using certain strategies proposed for the BPC algorithm. Three series of experiments were conducted to test: different column generator configurations, different path relaxations, and the proposed cutting planes. For all these experiments, we use a subset of 10 instances that were chosen because they are of large size (200 wells and five rigs) and their optimal values are known. The horizon length is $H=200$ for five of these instances and $H=300$ for the others. Each instance is associated with an identifier: for example, 200w_5r_300h_4 indicates the fourth instance with 200 wells, five rigs, and $H=300$.

Column generators: In the first series of tests, we compare four configurations of column generators, all ensuring optimality: exact labeling (ExL), heuristic labeling (HeuL) followed by ExL, tabu search (Tabu) followed by ExL, and Tabu followed by HeuL and by ExL. For this comparison, we solved only the linear relaxation of the 10 test instances using the ng-path relaxation with $\mathcal{N}^{ng}=10$. The results obtained are reported in Table 1. For each instance and each configuration, this table indicates, for each column generator used, the number of column generation iterations that called it (including those where it did not succeed to generate negative reduced cost columns) and the computational time in seconds. Two rows provide averages computed over the instances with $H=200$ (top part) and over those with $H=300$ (bottom part).

These results show that Tabu/HeuL/ExL is clearly the best configuration. With it, the exact labeling algorithm is almost never invoked except for proving optimality. Using only HeuL/ExL or Tabu/ExL yields somewhat similar computational times when $H=300$ but faster ones for HeuL/ExL when $H=200$. Indeed, Tabu is less efficient than HeuL for finding negative reduced cost

Table 1
Linear relaxation results for different column generator configurations.

Instance	ExL		HeuL/ExL		Tabu/ExL		Tabu/HeuL/ExL	
	No. it	Time (s)	No. it	Time (s)	No. it	Time (s)	No. it	Time (s)
200w_5r_200h_5	25	160.4	26/2	48.5	30/8	72.2	30/4/1	23.5
200w_5r_200h_6	31	284.7	41/4	113.9	51/11	117.0	43/9/1	47.3
200w_5r_200h_7	26	115.5	27/5	32.3	31/14	95.7	23/6/1	16.9
200w_5r_200h_8	33	175.5	37/5	51.9	37/6	47.0	35/6/1	20.7
200w_5r_200h_9	38	278.8	30/1	37.6	46/17	181.0	29/6/1	28.4
Average	30.6	203.0	32.2/3.4	56.8	39.0/11.2	102.6	32.0/6.2/1.0	27.4
200w_5r_300h_1	>61	>3600	71/14	707.2	64/13	834.5	58/15/5	268.1
200w_5r_300h_2	40	1300.8	48/5	271.0	74/15	720.3	58/11/1	117.3
200w_5r_300h_3	>53	>3600	91/16	1696.5	65/16	1422.9	49/9/1	261.1
200w_5r_300h_4	50	2130.2	61/9	638.1	70/15	804.7	64/12/2	243.1
200w_5r_300h_10	33	1098.5	54/9	410.8	64/13	569.3	78/16/2	131.2
Average	>47.4	>2345.9	65.0/10.6	744.7	67.4/14.4	870.3	61.4/12.6/2.2	204.2

Table 2
Results for the 2-cycle-free path relaxation and with elementary paths.

Instance	2-cycle-free paths		Elementary paths	
	Gap (%)	Time (s)	Gap (%)	Time (s)
200w_5r_200h_5	3.23	1941.3	0.24	154.3
200w_5r_200h_6	2.92	>3600	0.08	>3600
200w_5r_200h_7	2.75	>3600	0.33	>3600
200w_5r_200h_8	0.50	40.9	0.00	48.2
200w_5r_200h_9	3.22	1279.0	0.00	294.8
Average	2.52	>2092.2	0.13	>1539.5
200w_5r_300h_1	3.77	>3600	0.00	>3600
200w_5r_300h_2	1.94	1265.3	0.07	>3600
200w_5r_300h_3	4.04	>3600	<0.01	>3600
200w_5r_300h_4	1.79	>3600	0.00	>3600
200w_5r_300h_10	2.61	>3600	0.67	>3600
Average	2.83	>3133.1	0.15	>3600.0

Table 3
Results for three *ng*-path relaxations.

Instance	$\mathcal{N}^{ng}=5$		$\mathcal{N}^{ng}=10$		$\mathcal{N}^{ng}=15$	
	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
200w_5r_200h_5	0.24	51.5	0.24	55.0	0.24	74.3
200w_5r_200h_6	0.51	287.4	0.18	137.7	0.09	206.7
200w_5r_200h_7	0.08	73.3	0.08	96.8	0.08	168.3
200w_5r_200h_8	0.00	27.1	0.00	20.7	0.00	24.7
200w_5r_200h_9	0.24	67.0	0.00	28.4	0.00	75.1
Average	0.21	101.3	0.10	67.7	0.08	109.8
200w_5r_300h_1	0.00	391.4	0.00	268.1	0.00	638.2
200w_5r_300h_2	0.07	304.1	0.07	262.1	0.07	336.4
200w_5r_300h_3	0.33	994.5	<0.01	686.7	<0.01	788.8
200w_5r_300h_4	0.00	164.6	0.00	243.4	0.00	199.3
200w_5r_300h_10	0.68	1689.4	0.67	1803.6	0.67	3444.8
Average	0.22	708.8	0.15	652.8	0.15	1081.5

columns when there are not many, yielding more expensive calls to ExL. On the other hand, the computational effort of HeuL and ExL increases rapidly with the length of the paths (that is, with H), while Tabu is less impacted by the value of this parameter. Finally, we observe that ExL alone takes much more computational time than the other configurations. In particular, the linear relaxation of two instances could not be solved within the 1-h time limit. All subsequent tests in this paper were realized using the Tabu/HeuL/ExL configuration.

Path relaxations: The second series of experiments concern the different path relaxations that can be used to ease the solution process of the pricing subproblems at the expense of weaker lower bounds. We compare the use of elementary paths with that of 2-cycle-free paths and *ng*-paths for three values of the neighborhood sizes, namely, $\mathcal{N}^{ng}=5,10,15$. The results are presented in Tables 2 and 3. In these tables, we report for each type of paths and each instance the *integrality gap (optimal value minus the lower bound obtained at the root node before adding any subset-row*

inequalities) in percentage of the optimal value in absolute value, and the total computational time (in seconds) to solve the instance with the BPC algorithm. Average results are reported for the instances with $H=200$ and those with $H=300$. Notice that, for the case with elementary paths, we could compute the lower bound within the 1-h time limit for only five of the 10 instances. Nevertheless, we report the lower bounds for all instances for the sake of completeness.

Table 2 allows to compare the use of the traditional 2-cycle-free paths with that of the elementary paths. We observe *from the Gap column* that the 2-cycle-free paths yield a (much) weaker lower bound than the elementary paths for each instance. For four of the 10 instances, there is no integrality gap when using the elementary paths. For the instances with $H=200$, the stronger lower bounds yield a faster computational time on average. However, when the paths are longer ($H=300$), the BPC algorithm struggles at generating elementary paths and does not succeed to solve any instance within the 1-h time limit. With 2-cycle-free paths, the algorithm performs slightly better by solving one instance.

The results obtained with the three *ng*-path relaxations are given in Table 3. All instances can be solved within 1 h of computational time with each of these path relaxations. Therefore, they yield much better results than the cases with elementary paths or 2-cycle-free paths. Among these three relaxations, the one with $\mathcal{N}^{ng}=10$ produces the best average computational time for both groups of instances (with $H=200$ and with $H=300$). We observe that the *ng*-paths with $\mathcal{N}^{ng}=5,10,15$ yield the same lower bound than with elementary paths for six, eight, and nine instances, respectively, showing that the *ng*-paths are, in practice, elementary when $\mathcal{N}^{ng}$ is large enough. On the other hand, setting $\mathcal{N}^{ng}$ to a too large value increases the average computational time because the (small) gain in the lower bound is not sufficient to compensate the increased complexity of solving the pricing subproblems. For all subsequent tests, the *ng*-path relaxation with $\mathcal{N}^{ng}=10$ is used.

Cutting planes: In the third series of experiments, we evaluate the effectiveness of the subset-row inequalities. All 10 test instances were solved using the proposed BPC algorithm with the subset-row inequalities and without them. Table 4 reports the results obtained. For each instance and each algorithm, it provides *the integrality gap in percentage (using the lower bound obtained after adding cuts, if any)*, the total number of subset-row cuts added (only for the case with cuts), the total number of nodes in the search tree, and the total computational time (in seconds). Averages are given for the instances with $H=200$ and with $H=300$.

From these results, we observe first that four instances were solved at the root node without adding any cuts. For the remaining six instances, the subset-row inequalities completely close the integrality gap and no branching is needed. Overall using these cuts yields much smaller average computational times. Consequently, the subset-row cuts were used to produce the main results presented in the next section.

4.2. Main results

With the proposed BPC algorithm, we tried to solve the 80 benchmark instances. The detailed results of these experiments can be found in Appendix A. Here, we report in Table 5 a summary of these results. In fact, average results over the instances with the same number of wells, same number of rigs, and same horizon length are provided. The groups of 10 instances are identified similarly as the instances: for example, the group 200w_5r_300h includes the instances with 200 wells, five rigs, and $H=300$. For each group, Table 5 provides the number of instances solved to optimality within the 1-h time limit, the average number of wells serviced in the computed solution, the average integrality gap (%), the average number of cuts generated, the average number of branch-and-bound nodes explored, and the average computational time (in seconds). All averages are computed over the instances solved to optimality.

The BPC algorithm is highly successful on the 100-well instances (top part of Table 5), solving to optimality 38 of the 40 instances. For the larger 200-well instances (bottom part), 12 of the 40 instances could not be solved. In particular, all instances with 200 wells, 10 rigs, and $H=300$ were unsolvable. As expected, increasing the number of wells, or the number of rigs, or the horizon length yields harder-to-solve instances. For the instances solved, we observe relatively small average integrality gaps that can be almost completely closed by the subset-row inequalities, the number of branch-and-bound nodes being very low. Finally,

Table 5
Summary of the computational results.

Instance group	No. solved	No. wells serviced	Gap (%)	No. cuts	No. nodes	Time (s)
100w_5r_200h	10	41.7	0.10	11.0	1.0	4.2
100w_10r_200h	10	71.6	0.14	25.8	2.2	16.7
100w_5r_300h	9	60.7	0.33	39.2	2.3	140.5
100w_10r_300h	9	92.4	0.22	82.8	1.4	409.8
200w_5r_200h	10	56.5	0.20	19.0	1.0	69.2
200w_10r_200h	9	99.0	0.27	60.0	1.4	534.9
200w_5r_300h	9	83.7	0.14	26.3	1.0	977.2
200w_10r_300h	0	–	–	–	–	–

Table 4
Results with and without subset-row cuts.

Instance	Without cuts			With cuts			
	Gap (%)	No. BB	Time (s)	Gap (%)	No. cuts	No. BB	Time (s)
200w_5r_200h_5	0.24	11	297.0	0.00	20	1	55.0
200w_5r_200h_6	0.18	7	387.4	0.00	30	1	137.7
200w_5r_200h_7	0.08	7	139.1	0.00	35	1	96.8
200w_5r_200h_8	0.00	1	20.7	0.00	0	1	20.7
200w_5r_200h_9	0.00	1	28.4	0.00	0	1	28.4
Average	0.10	5.4	174.5	0.00	17.0	1.0	67.7
200w_5r_300h_1	0.00	1	268.1	0.00	0	1	268.1
200w_5r_300h_2	0.07	11	2039.3	0.00	20	1	262.1
200w_5r_300h_3	<0.01	5	2842.9	0.00	20	1	686.7
200w_5r_300h_4	0.00	1	243.4	0.00	0	1	243.4
200w_5r_300h_10	0.67	>16	>3600	0.00	100	1	1803.6
Average	0.15	>6.8	>1798.7	0.00	28.0	1.0	652.8

Table A1
Results for instances with 100 wells, five rigs and $H=200$.

Instance	Opt. value	No. wells served	LB	Gap (%)	No. cuts	No. nodes	Time (s)
1	−32,275	41	−32,284.5	0.03	5	1	3.0
2	−29,068	41	−29,139.8	0.25	15	1	4.8
3	−28,466	45	−28,486.8	0.07	40	1	13.4
4	−27,929	41	−27,929.0	0.00	0	1	1.2
5	−26,398	38	−26,505.8	0.41	30	1	4.1
6	−26,661	42	−26,661.0	0.00	0	1	2.3
7	−26,128	38	−26,128.0	0.00	0	1	0.5
8	−32,912	46	−32,999.4	0.27	20	1	8.6
9	−26,,704	42	−26,704.0	0.00	0	1	1.0
10	−33,521	43	−33,521.0	0.00	0	1	2.6
Average		41.7		0.10	11.0	1.0	4.2

Table A2
Results for instances with 100 wells, 10 rigs and $H=200$.

Instance	Opt. value	No. wells served	LB	Gap (%)	No. cuts	No. nodes	Time (s)
1	−52,034	74	−52,128.0	0.18	42	1	14.9
2	−46,189	69	−46,216.4	0.06	20	1	6.5
3	−47,003	74	−47,205.3	0.43	65	11	85.9
4	−46,513	72	−46,528.0	0.03	14	1	6.6
5	−47,734	71	−47,734.0	0.00	0	1	2.7
6	−41,834	69	−41,999.6	0.39	40	1	10.2
7	−44,703	66	−44,703.0	0.00	0	1	3.5
8	−51,208	74	−51,296.0	0.17	20	1	5.8
9	−45,796	72	−45,876.3	0.18	57	3	26.2
10	−51,279	75	−51,279.0	0.00	0	1	4.8
Average		71.6		0.14	25.8	2.2	16.7

Table A3
Results for instances with 100 wells, five rigs and $H=300$.

Instance	Opt. value	No. wells served	LB	Gap (%)	No. cuts	No. nodes	Time (s)
1	−71,487	60	−71,609.0	0.17	20	1	38.1
2	−63,737	57	−63,779.1	0.07	20	1	32.9
3	–	–	−64,624.7	–	≥165	≥1	>3600
4	−60,144	59	−60,633.1	0.81	98	3	372.7
5	−60,598	59	−61,059.3	0.76	65	7	258.0
6	−60,081	64	−60,194.9	0.19	25	5	269.5
7	−59,372	54	−59,554.2	0.31	40	1	28.1
8	−74,611	68	−74,688.0	0.10	30	1	140.0
9	−61,860	63	−61,874.0	0.02	15	1	21.4
10	−72,805	62	−73,214.1	0.56	40	1	103.7
Average		60.7		0.33	39.2	2.3	140.5

Table A4
Results for instances with 100 wells, 10 rigs and $H=300$.

Instance	Opt. value	No. wells served	LB	Gap (%)	No. cuts	No. nodes	Time (s)
1	−111,649	92	−111,672.1	0.02	20	1	56.9
2	−97,802	90	−98,051.9	0.26	120	3	363.8
3	−100,913	98	−101,390.3	0.47	240	3	2356.3
4	−96,886	92	−97,047.8	0.17	65	1	205.0
5	−102,631	92	−102,668.2	0.04	20	1	48.4
6	–	–	−90,623.8	–	≥160	≥3	>3600
7	−96,842	82	−97,244.5	0.41	60	1	121.4
8	−108,731	96	−109,078.7	0.32	80	1	134.3
9	−98,843	94	−99,128.8	0.29	120	1	359.5
10	−109,304	96	−109,309.3	0.01	20	1	42.6
Average		92.4		0.22	82.8	1.4	409.8

we remark that the number of wells serviced increases with the number of rigs and the horizon length, but also with the number of wells. Indeed, more wells to service in the same region reduces the average travel time between the wells serviced consecutively on the same rig route, increasing productivity.

5. Conclusions

In this paper, we propose the first exact algorithm for the WRPP, namely a BPC algorithm that exploits some of the most recent techniques introduced for solving the VRPTW. In particular, the column generation process relies on a tabu search heuristic [9] that is able to provide negative reduced cost columns efficiently in most iterations, on *ng*-paths [3], and on exact and heuristic labeling algorithms. We also use the subset-row

Table A5
Results for instances with 200 wells, five rigs and $H=200$.

Instance	Opt. value	No. wells served	LB	Gap (%)	No. cuts	No. nodes	Time (s)
1	−40,257	54	−40,447.3	0.47	60	1	194.8
2	−35,084	55	−35,084.0	0.00	0	1	12.5
3	−40,195	59	−40,272.5	0.19	25	1	85.8
4	−40,523	54	−40,757.0	0.57	20	1	45.4
5	−40,194	57	−40,291.3	0.24	20	1	55.0
6	−42,335	62	−42,411.0	0.18	30	1	137.7
7	−33,070	52	−33,180.3	0.33	35	1	96.8
8	−39,517	60	−39,517.0	0.00	0	1	20.7
9	−45,683	58	−45,683.0	0.00	0	1	28.4
10	−38,036	54	−38,036.0	0.00	0	1	15.2
Average		56.5		0.20	19.0	1.0	69.2

Table A6
Results for instances with 200 wells, 10 rigs and $H=200$.

Instance	Opt. value	No. wells served	LB	Gap (%)	No. cuts	No. nodes	Time (s)
1	−67,670	98	−67,670.0	0.00	0	1	76.4
2	−61,377	98	−61,591.5	0.35	45	3	290.9
3	−66,032	101	−66,353.0	0.35	80	1	380.7
4	−67,445	97	−67,813.2	0.54	150	3	2734.5
5	−72,658	103	−72,773.3	0.16	40	1	288.8
6	–	–	−67,601.6	–	255	≥6	>3600
7	−60,036	93	−60,087.0	0.09	5	1	51.0
8	−60,350	99	−60,512.2	0.27	120	1	401.8
9	−79,301	107	−79,669.5	0.46	40	1	412.4
10	−61,968	95	−62,089.8	0.20	60	1	177.5
Average		99.0		0.27	60.0	1.4	534.9

Table A7
Results for instances with 200 wells, five rigs and $H=300$.

Instance	Opt. value	No. wells served	LB	Gap (%)	No. cuts	No. nodes	Time (s)
1	−90,317	86	−90,317.0	0.00	0	1	268.1
2	−83,684	82	−83,739.5	0.07	20	1	262.1
3	−91,981	87	−91,995.0	0.02	20	1	686.7
4	−91,917	84	−91,917.0	0.00	0	1	243.4
5	−91,957	82	−92,051.8	0.32	60	1	2456.1
6	–	–	−96,756.7	–	≥60	≥1	>3600
7	−76,049	78	−76,182.5	0.18	37	1	1897.0
8	−89,421	86	−89,421.0	0.00	0	1	632.7
9	−99,962	87	−99,962.0	0.00	0	1	545.3
10	−86,564	81	−87,145.2	0.67	100	1	1803.6
Average		83.7		0.14	26.3	1.0	977.2

Table A8
Results for instances with 200 wells, 10 rigs and $H=300$.

Instance	Opt. value	No. wells served	LB	Gap (%)	No. cuts	No. nodes	Time (s)
1	–	–	−149,075.6	–	≥80	≥1	>3600
2	–	–	−139,449.6	–	≥80	≥1	>3600
3	–	–	−151,256.4	–	≥40	≥1	>3600
4	–	–	−152,691.7	–	≥40	≥1	>3600
5	–	–	−159,987.7	–	≥40	≥1	>3600
6	–	–	−154,020.5	–	≥60	≥1	>3600
7	–	–	−134,694.5	–	≥100	≥1	>3600
8	–	–	−141,087.1	–	≥60	≥1	>3600
9	–	–	−170,977.2	–	≥40	≥1	>3600
10	–	–	−144,,644.5	–	≥80	≥1	>3600
Average		–		–	–	–	–

inequalities [17] at the root node to greatly improve the linear relaxation lower bound. Our computational experiments showed that the proposed BPC algorithm can solve practical-sized instances of the WRRP in reasonable computational times.

Acknowledgments

Glaydston Mattos Ribeiro acknowledges Espírito Santo Research Foundation (Process 53630742/11) and National Council for Scientific and Technological Development (Process 307002/2011-0) for their financial support. Guy Desaulniers and Jacques Desrosiers acknowledges the National Science and Engineering Research Council of Canada for its financial support.

Appendix A. Detailed results

Tables A1–A8 report the detailed results obtained for each individual instance that were used to compute the average results of Table 5. In these tables, the columns provide in order: the instance number, the optimal value (that is, the negative of the total loss saved), the number of wells served in the computed optimal solution, the lower bound computed at the root node, the integrality gap in percentage, the total number of subset-row cuts generated, the number of nodes explored in the search tree, and the total computational time in seconds. A dash (–) indicates that the instance is not solved to proven optimality within the 1-h time limit. In each table, the last row provides averages over the instances solved to optimality.

References

[1] Aloise DJ, Aloise D, Rocha CTM, Ribeiro CC, Ribeiro Filho JC, Moura LSS. Scheduling workover rigs for onshore oil production. Discrete Applied Mathematics 2006;154(5):695–702, http://dx.doi.org/10.1016/j.dam.2004.09.021.

[2] Archetti C, Feillet D, Hertz A, Speranza MG. The capacitated team orienteering and profitable tour problems. Journal of the Operational Research Society 2009;60(6):831–842, http://dx.doi.org/10.1057/palgrave.jors.2602603.

[3] Baldacci R, Bartolini E, Mingozzi A, Roberti R. An exact solution framework for a broad class of vehicle routing problems. Computational Management Science 2010;7(3):229–268, http://dx.doi.org/10.1007/s10287-009-0118-3.
[4] Baldacci R, Mingozzi A, Roberti R. New route relaxation and pricing strategies for the vehicle routing problem. Operations Research 2011;59(5):1263–1283, http://dx.doi.org/10.1287/opre.1110.0975.
[5] Boland N, Dethridge J, Dumitrescu I. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. Operations Research Letters 2006;34:58–68, http://dx.doi.org/10.1016/j.orl.2004.11.011.
[6] Dantzig GB, Wolfe P. Decomposition principle for linear programs. Operations Research 1960;8(1):101–111, http://dx.doi.org/10.1287/opre.8.1.101.
[7] Desaulniers G, Desrosiers J, Ioachim I, Solomon MM, Soumis F, Villeneuve D. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In: Crainic TG, Laporte G, editors. Fleet management and logistics. Kluwer; 1998. p. 57–93.
[8] Desaulniers G, Lavigne J, Soumis F. Multi-depot vehicle scheduling problems with time windows and waiting costs. European Journal of Operational Research 1998;111:479–494, http://dx.doi.org/10.1016/S0377-2217(97)00363-9.
[9] Desaulniers G, Lessard F, Hadjar A. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transportation Science 2008;42(3):387–404, http://dx.doi.org/10.1287/trsc.1070.0223.
[10] Desaulniers G, Desrosiers J, Spoorendonk S. The vehicle routing problem with time windows: state-of-the-art exact solution methods. In: Cochran JJ, Cox Jr. LA, Keskinocak P, Kharoufeh JP, Smith JC, editors. Wiley encyclopedia of operations research and management science, vol. 8. New York, NY: Wiley; 2010. p. 5742–5749, http://dx.doi.org/10.1002/9780470400531.eorms1034.
[11] Desaulniers G, Desrosiers J, Spoorendonk S. Cutting planes for branch-and-price algorithms. Networks 2011;58(4):301–310, http://dx.doi.org/10.1002/net.20471.
[12] Desrosiers J, Lübbecke ME. Branch-price-and-cut algorithms. In: Cochran JJ, Cox Jr. LA, Keskinocak P, Kharoufeh JP, Smith JC, editors. Wiley encyclopedia of operations research and management science, vol. 8. New York, NY: Wiley; 2010http://dx.doi.org/10.1002/9780470400531.eorms0118.
[13] Dror M. Note on the complexity of the shortest path models for column generation in VRPTW. Operations Research 1994;42:977–979, http://dx.doi.org/10.1287/opre.42.5.977.
[14] Feillet D, Dejax P, Gendreau M, Gueguen C. An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. Networks 2004;44(3):216–229, http://dx.doi.org/10.1002/net.v44:3.
[15] Irnich S, Desaulniers G. Shortest path problems with resource constraints. In: Desaulniers G, Desrosiers J, Solomon MM, editors. Column generation. Springer; 2005. p. 33–66, http://dx.doi.org/10.1007/0-387-25486-2_2.
[16] Jans R. Classification of Dantzig–Wolfe reformulations for binary mixed integer programming problems. European Journal of Operational Research 2010;204(2):251–254, http://dx.doi.org/10.1016/j.ejor.2009.11.014.
[17] Jepsen M, Petersen B, Spoorendonk S, Pisinger D. Subset-row inequalities applied to the vehicle-routing problem with time windows. Operations Research 2008;56(2):497–511, http://dx.doi.org/10.1287/opre.1070.0449.
[18] Letchford AN, Lysgaard J, Eglese RW. A branch-and-cut algorithm for the capacitated open vehicle routing problem. Journal of the Operational Research Society 2007;58(12):1642–1651, http://dx.doi.org/10.1057/palgrave.jors.2602345.
[19] Lübbecke ME, Desrosiers J. Selected topics in column generation. Operations Research 2005;53(6):1007–1023, http://dx.doi.org/10.1287/opre.1050.0234.
[20] Neves TA. Heuristics with adaptive memory applied to workover rig routing and scheduling problem. Master's thesis. Fluminense Federal University, Niterói, Brazil; 2007.
[21] Pacheco AVF, Ribeiro GM, Mauri GR. A GRASP with path-relinking for the workover rig scheduling problem. International Journal of Natural Computing Research 2010;1(2):1–14, http://dx.doi.org/10.4018/jncr.2010040101.
[22] Ribeiro GM, Laporte G, Mauri GR. A comparison of three metaheuristics for the workover rig routing problem. European Journal of Operational Research 2011 doi:10.1016/j.ejor.2012.01.031.
[23] Righini G, Salani M. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Discrete Optimization 2006;3(3):255–273, http://dx.doi.org/10.1016/j.disopt.2006.05.007.
[24] Righini G, Salani M. New dynamic programming algorithms for the resource constrained elementary shortest path problem. Networks 2008;51(3):155–209, http://dx.doi.org/10.1002/net.20212.