European Journal of Operational Research 220 (2012) 28–36

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

A comparison of three metaheuristics for the workover rig routing problem

Glaydston Mattos Ribeiro [a,b], Gilbert Laporte [a,*], Geraldo Regis Mauri [c]

[a] CIRRELT and HEC Montréal, Canada H3T 2A7
[b] Federal University of Espírito Santo, São Mateus, ES 29932-540, Brazil
[c] Federal University of Espírito Santo, Alegre, ES 29500-000, Brazil

ARTICLE INFO

Article history:
Received 27 August 2010
Accepted 17 January 2012
Available online 28 January 2012

Keywords:
Metaheuristics
Iterated local search
Clustering search
Adaptive large neighborhood search
Workover rig routing problem

ABSTRACT

The workover rig routing problem (WRRP) is a variant of the Vehicle Routing Problem with Time Windows (VRPTW) and arises in the operations of onshore oil fields. In this problem, a set of workover rigs located at different positions must service oil wells requesting maintenance as soon as possible. When a well requires maintenance, its production is reduced or stopped for safety reasons and some workover rig must service it within a given deadline. It is therefore important to service the wells in a timely fashion in order to minimize the production loss. Whereas for classical VRPTWs the objective is to minimize route length, in the WRRP the objective is to minimize the total lost production, equal to the sum of arrival times at the wells, multiplied by production loss rates. The WRRP generalizes the Delivery Man Problem with Time Windows by considering multiple open vehicle routes and multiple depots. This paper compares three metaheuristics for the WRRP: an iterated local search, a clustering search, and an Adaptive Large Neighborhood Search (ALNS). All approaches, in particular ALNS, have yielded good solutions for instances derived from a real-life setting.

1. Introduction

Onshore oil wells make use of specialized and expensive equipments in their operations. When these equipments fail, special maintenance services such as cleaning, reinstatement and stimulation, are required to restore production. These services are performed by a limited number of expensive workover rigs (or rigs, for short) which are transported by trucks. The monetary sums involved in these operations are enormous. For example, Aloise et al. (2006) state that the yearly rental cost of 10 rigs is US $10,000,000 for Petrobras. Their heuristic was capable of reducing production losses by an estimated US $2,568,000 per year. The reference by Aloise et al. (2006) contains pictures of workover rigs and of the trucks used for their transportation.

When a well requires maintenance, its production is reduced or stopped for safety reasons. Thus, a rig must be sent to perform the required service in order to reestablish production. Each well has a deadline for its maintenance service, and a rig must start performing the service before this deadline so that the well does not become inactive. The rigs are located at different positions and may take considerable time to reach the wells.

The Workover Rig Routing Problem (WRRP) consists in finding the best routing for the rigs in such a way that all requests are attended, deadlines are respected and the oil production loss is minimized. The production loss of a well is obtained as its regular average daily flow rate, multiplied by the number of the days during which it does not operate (Aloise et al., 2006). The rigs are rented and they do not need to return to a depot after servicing the wells, which means that the routes are open. Fig. 1 shows our best-known solution for an instance with 50 wells and five rigs. Note the open routes.

The WRRP is a variant of the Vehicle Routing Problem with Time Windows (VRPTW). However, whereas the objective of the classical VRPTW is to minimize route length, in the WRRP one must minimize the arrival times at the wells, weighted by production losses. The WRRP generalizes the Delivery Man Problem with Time Windows (DMPTW) known to be NP-hard, by considering multiple open vehicle routes and multiple depots (initially the rigs are located in different positions). The DMPTW consists in determining a Hamiltonian path for a delivery man, starting at the depot, so as to minimize the sum of travel durations between the depot and all customers, while respecting time windows. The literature concerning the DMPTW is very limited. To the best of our knowledge, only two publications exist on this problem: Heilporn et al. (2010) present two mixed integer linear programming formulations, polyhedral properties, as well as exact and heuristic algorithms for the problem, and Tsitsiklis (1992) provides NP-completeness results for several special cases of the DMPTW and of the *Traveling Salesman Problem with Time Windows* (TSPTW). In contrast, the Delivery Man Problem (DMP), introduced by Lucena

* Corresponding author. Tel.: +1 514 343 6143; fax: +1 514 343 7121.
E-mail addresses: glaydstonribeiro@ceunes.ufes.br (G.M. Ribeiro), gilbert.laporte@cirrelt.ca (G. Laporte), mauri@cca.ufes.br (G.R. Mauri).

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2012.01.031

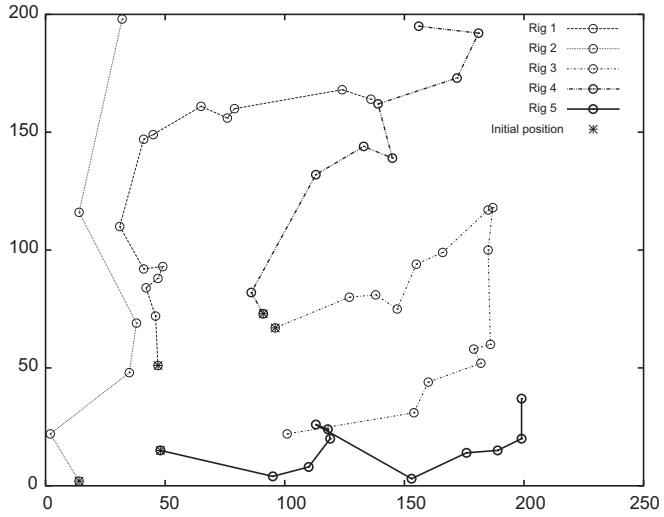

Fig. 1. Best-known solution for instance 50/5–1 (total production loss: 42682.12). This picture shows the location of the wells, of the rigs and our best-known open routes for each rig.

(1990) as the *Minimum Latency Problem*, has been the object of more publications. Lucena (1990) has proposed an integer nonlinear model, while Fischetti et al. (1993) and Méndez-Días et al. (2008) have presented several mixed integer linear programming formulations and valid inequalities for this problem.

Recently, Ngueveu et al. (2010) have studied the *Cumulative Capacitated Vehicle Routing Problem* (CCVRP) which minimizes the sum of arrival times at customers, instead of the classical route length, subject to vehicle capacity constraints. They have considered an homogeneous fleet and a single depot. The WRRP is related to the CCVRP but it has different initial positions for the rigs and deadlines, although there are no capacity constraints. It is also related to the *Open Vehicle Routing Problem* (OVRP) (see, for instance, Letchford et al. (2007) and Li et al. (2007) for more details) where all vehicles are initially based at the same depot, the objective is to minimize the total routing cost, but vehicles do not return to the depot after servicing the last customer on their route. As shown by Ngueveu et al. (2010) and Letchford et al. (2007), a solution for the open version of a VRP can be quite different from that of the closed version.

There are also a limited number of publications on the WRRP. Aloise et al. (2006) have considered a version without deadlines and proposed a variable neighborhood search heuristic for the problem. The authors have used nine neighborhoods and presented computational experiments for instances derived from a Brazilian oil field. Pacheco et al. (2010) have recently proposed a greedy randomized adaptive search procedure (GRASP) with path-relinking (PR) (Glover, 1996) to solve the problem without travel times. The authors have used instances obtained from another Brazilian oil field. Neves (2007) has solved the WRRP using GRASP, GRASP + PR, tabu search and iterated local search (ILS) algorithms. This author has proposed instances with 50, 100 and 500 wells, and with five and 10 rigs. The ILS heuristic provided better results than the other heuristics used in the comparison. It will be summarized in Section 3.

This paper solves the WRRP by means of a clustering search algorithm (CS) and of an adaptive large neighborhood search heuristic (ALNS). Our results outperform those obtained with all previous algorithms developed by Neves (2007). For example, on instances of size 500, we reduce the average gap from the best known solution values by more than 2%, which is significant given the large sums of money at stake in the petroleum industry. The remainder of this paper is organized as follows. Section 2 presents the mathematical formulation. Section 3 describes the ILS heuristic of Neves (2007). Our CS and ALNS heuristics are described in Sections 4 and 5, respectively. Computational results are presented in Section 6, followed by conclusions in Section 7.

2. Mathematical model

In order to state the WRRP in a precise fashion, we provide a mathematical formulation for the problem. Let K be the set of rigs, and associate with each of them a directed graph $G^k = (V^k, A^k)$ where V^k is the node set and A^k is the arc set for rig $k \in K$. Let $V^k = W^k \cup \{o(k), d(k)\}$, where $o(k)$ and $d(k)$ denote the origin and depot nodes corresponding to the initial and final locations of rig k, and W^k its restricted set of wells. Let $W = \bigcup_{k\in K} W^k$ which represents all wells that must be visited once for any rig k.

A deadline l_i is associated with each node $i \in W^k$. Each arc $(i,j) \in A^k, i \in W^k \cup \{o(k)\}, j \in W^k$, is characterized by a positive duration c_{ij}. We assume that the service time s_j at node j is included in the time c_{ij}. A parameter e_i defines the earliest time to reach node i by some rig k, i.e, $e_i = \min_{k\in K}\{c_{o(k)i}\}$. We assume dummy arcs $(o(k), d(k))$, $k \in K$ and arcs $(i, d(k))$, $i \in W^k$, $k \in K$, with zero duration included in A^k to account for unused rigs.

To properly define the objective function, let $p_i > 0$, $i \in W$ be a production loss per time unit for node i. Let x_{ij}^k be a binary variable equal to 1 if and only if rig k traverses edge (i,j) from i to j in graph G^k, and let t_i^k be a time variable equal to the time necessary to reach node i by rig k. Thus, adapting well-known formulations for the VRPTW with one or more depots (see for instance Kallehauge et al. (2006)), the WRRP can be formulated as follows:

$$(WRRP)\ \text{minimize} \sum_{k\in K}\sum_{i\in W^k} p_i t_i^k \tag{1}$$

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

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

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

$$x_{ij}^k \left(t_i^k + c_{ij} - t_j^k\right) \leqslant 0 \quad k \in K, (i,j) \in A^k \tag{5}$$

$$e_i \sum_{j:(i,j)\in A^k} x_{ij}^k \leqslant t_i^k \leqslant l_i \sum_{j:(i,j)\in A^k} x_{ij}^k \quad k \in K, i \in W^k \tag{6}$$

$$t_{o(k)}^k = 0 \quad k \in K \tag{7}$$

$$x_{ij}^k \in \{0,1\} \quad k \in K, (i,j) \in A^k. \tag{8}$$

The objective function (1) represents the total production loss to be minimized. Linking constraints (2) ensure that each well is visited exactly once, and hence assigned to a single rig. Constraints (3) and (4) define the classical network flow constraints for a path originating at origin node $o(k)$ and ending at a virtual depot node $d(k)$. The non-linear constraints (5) express the compatibility requirements between arc flow and node time variables. Deadline constraints are given by time window constraints (6) where if a node i is visited by rig k, then $e_i \leqslant t_i^k \leqslant l_i$, otherwise $t_i^k = 0$ and there is no impact on the objective function. Constraints (7) indicate that each rig k starts its service at time zero. Finally, constraints (8) impose binary values for the flow variables. The mathematical formulation (1)–(8) is non-linear because of Constraints (5), but can easily be linearized by using standard techniques.

3. Iterated local search

This section presents a brief description of the ILS of Neves (2007). We have used the ILS computed code provided by Neves (2007) in our computational experiments.

The ILS heuristic is a stochastic local search heuristic made up of four main steps (Lourenço et al., 2002): a procedure to generate an initial solution, a local search procedure, a perturbation scheme and an acceptance criterion to decide from which solution the search is continued. First, an initial solution s is constructed and the local search is applied to it. Second, while the stopping criterion is not met (for example, the number of iterations or the maximum computing time), a perturbation procedure based on the search history is applied on s to generate a new solution s'. Local search is then applied on s' and the algorithm determines whether to continue the search from s or from s'. The ILS heuristic has produced good solutions on different combinatorial problems such as the Traveling Salesman Problem (Johnson and McGeoch, 2002) and the Quadratic Assignment Problem (Stützle, 2006).

We now review the main elements implemented by Neves (2007) for the WRRP.

1. *Initial solution*: Neves (2007) applies a nearest neighbor heuristic which considers the travel times between wells and the associated production loss.
2. *Local search*: For the local search, given a solution s, the algorithm removes each well from s and inserts it in another position in the same route or in another route. The best insertion is performed, and the process is repeated until no improvement is possible.
3. *Perturbation*: For the perturbation procedure, 12 moves are performed, nine of which are chosen at random and three are based on the search history. A move removes a well from a route of the current solution and inserts it into another route. The frequency of pairs of wells appearing together in a solution is used as memory. Given a solution s, the algorithm selects the pairs present in s with the three highest frequencies. For each pair, the first well is assigned to another rig and inserted at some position chosen at random in its route.
4. *Acceptance criterion*: If a new solution s' improves upon the current one, it is accepted and the search continues from s'.

4. Clustering search

In this section, we present our CS algorithm for the WRRP. This heuristic contains some user-controlled parameters denoted by lower case greek letters, which will be calibrated in Section 6.

The CS heuristic was proposed by Chaves and Lorena (2010) and employs clustering for detecting promising areas of the search space. It generalizes the evolutionary clustering search proposed by Oliveira and Lorena (2004, 2007), but substitutes the evolutionary algorithm by metaheuristics such as SA, GRASP or tabu search.

In CS, solutions are put into clusters, and each new solution is assigned to its best cluster according to a distance metric. Each cluster i is defined as a triple $(\varsigma_i, \tau_i, \beta_i)$, where ς_i, τ_i and β_i are the center, the volume and the indicator of inefficiency, respectively. The center ς_i is a solution representing the cluster, the volume τ_i defines the number of solutions assigned to it, and the indicator of inefficiency β_i controls the search strategy.

When a new solution is generated by the metaheuristic, it is assigned to its closest cluster. This process activates the chosen cluster i, causing a *disturbance* to it, and an *assimilation* process is started. At regular intervals, each cluster is also evaluated searching a promising one. If some cluster i becomes promising, a search strategy is used over its center. When the search strategy is used β_{max} times without success (it does not improve the cluster center), a disturbance is applied to the center. This prevents the search intensification to be applied to poor areas or to areas that have already been sufficiently explored.

We now describe the main elements of our CS implementation for the WRRP.

1. *Solution generator*: SA is used to generate solutions for clustering. It starts from an initial solution and performs local search according to the usual SA rules. Three neighborhoods N^1, N^2 and N^3 are defined: N^1 is obtained by inserting a well i chosen at random from some route into another route in a position chosen at random, N^2 is obtained by swapping two wells from two different routes, all chosen at random, and N^3 is obtained by randomly choosing two wells of the same route, and swapping them. The initial temperature $T_{start} = 30000$ is decreased by the cooling rate $c = 0.975$, and at each fixed temperature, the moves are repeated during $SA_{max} = 4000$ iterations. The algorithm ends when the temperature reaches $T_{frozen} = 0.01$.
2. *Iterative clustering*: Given a user-defined maximum number of clusters γ, a new solution s provided by SA is assigned to the closest cluster i according to the Hamming distance H_i (Hamming, 1950), i.e., the number of different edges between ς_i and s. The algorithm assigns s to cluster i that minimizes $\min_{i\in\{1,2,\ldots,\gamma\}}\{H_i\}$.
3. *Penalized objective function*: Solution violating deadlines are considered during the search. This is done through the use of the penalized objective function

$$v(s) = \sum_{k\in K}\sum_{i\in W^k}\left(p_i t_i^k + \lambda \max\left\{0, t_i^k - l_i\right\}\right), \tag{9}$$

where λ is a user-defined penalty and s is a solution to be evaluated.

4. *Assimilation process*: When a new solution s must be assigned to cluster i, its center ς_i and its volume τ_i are updated. This process, called assimilation process by Chaves and Lorena (2010), is performed by a path-relinking method (Glover, 1996) applied on s and ς_i. By so doing, several solutions along the path between s and ς_i are evaluated. The assimilation process is useful for applying intensification and diversification strategies within the clusters. It is used in a reverse way, from ς_i ($s_{initial}$) to s (s_{target}). The process starts computing the symmetric difference $\Delta(s_{initial}, s_{target})$, i.e., the number of moves needed to reach s_{target} from $s_{initial}$. To construct the path, at each step all moves are evaluated from the current solution s_{step}. The purpose of a move is to change a well i from its position in s_{step} to the best position indicated by s_{target}. That move m^* resulting in the best solution is chosen to be performed on s. The set of available moves is then updated. The process ends when there are not more moves to be explored, and the best solution found in the path is returned by the algorithm as a new center. See Algorithm 1 for more details.
5. *Analysis process*: Each cluster i has a volume τ_i indicating its activity level. Whenever τ_i reaches a certain threshold τ_{max}, the cluster i must be better investigated to accelerate the search convergence. Chaves and Lorena (2010) indicate that the search strategy must be a problem-specific and should be applied to the cluster center.
6. *Search strategy*: Local search is used to intensify the search on the most promising cluster. It removes each well from a solution and inserts it into the best position. This procedure is repeated as long as improvements are obtained. Variable β_i controls the number of times that the local search is not effective on cluster i, and if β_i is equal to β_{max}, which represents a maximum number of attempts without success, a disturbance is applied to ς_i by neighborhood N^2.

Algorithm 1. Assimilation process: path-relinking algorithm

Input: $s_{initial}$ and s_{target}
$s_{step} \leftarrow s_{initial}$
Compute the symmetric difference $\Delta(s_{step}, s_{target})$
$v_{best} \leftarrow +\infty$
While $\Delta(s_{step}, s_{target}) \neq 0$ **do**
  Analyze all moves m
  Choose the best move m^* and apply it to s_{step} to get s'
  $s_{step} \leftarrow s'$
  Update the symmetric difference $\Delta(s_{step}, s_{target})$
  If $v(s_{step}) < v_{best}$ **then**
    $v_{best} \leftarrow v(s_{step})$ $s_{best} \leftarrow s_{step}$
  end
end
Return s_{best}

Algorithm 2. CS

Create γ new solutions (clusters) at random
Set $\tau_i = 0$ and $\beta_i = 0$ for $i = 1, \ldots, \gamma$
$s \leftarrow$ Initial Solution() $s_{best} \leftarrow s$
$c \leftarrow 0.975$ $T \leftarrow T_{start}$
While $T > T_{frozen}$ **do**
  $IterT \leftarrow 0$
  While $IterT < SA_{max}$ **do**
    $IterT \leftarrow IterT + 1$
    Choose at random a neighborhood $N^k (k = 1,2,3)$
    $s' \leftarrow N^k(s)$
    if $v(s') - v(s) < 0$ **then**
      $s \leftarrow s'$
    else
      With probability given by $e^{-(v(s')-v(s))}/T$ set $s \leftarrow s'$
    end
  end
  $T \leftarrow cT$ $i \leftarrow \arg\min_{i \in \gamma} H_i$ $\tau_i \leftarrow \tau_i + 1$ $\varsigma_i \leftarrow$ Path-reliking (s, ς_i)
  if $\tau_i = \tau_{max}$ **then**
    $\tau_i \leftarrow 0$
    $s \leftarrow$ Local Search (ς_i)
    if $v(s) = v(\varsigma_i)$ **then**
      $\beta_i \leftarrow \beta_i + 1$
      if $\beta_i = \beta_{max}$ **then**
        $\beta_i \leftarrow 0$ $\varsigma_i \leftarrow N^2(\varsigma_i)$
      end
    else
      $\beta_i \leftarrow 0$
    end
  end
  $s_{best} \leftarrow \min(s_{best}, \varsigma_i)$
end
if s_{best} *is feasible* **then**
  Return s_{best}
else
  Return "No feasible solution was found"
end

4.1. Initial solution

The initial solution is balanced: the numbers of wells assigned to any two rigs differ by at most one. First, the wells are sorted in non-decreasing order of the ratios p_i/s_i, and are then distributed for the rigs. This procedure does not respect the deadline constraints, but our computational results have shown that CS is able to regain feasibility.

4.2. Pseudo-code of the CS

The general structure of our CS algorithm implementation for the WRRP is summarized in Algorithm 2.

4.3. Illustration

Fig. 2 depicts a typical clustering process provided by CS. We again use instance 50/5–1 as an example. This figure shows the partitioned search space into 14 clusters (numbered rectangles), the solutions coming from SA (dots), the centers (stars), and the clustering process generated by CS after 10 iterations (a), 100 iterations (b), 300 iterations (c), and 604 iterations (d) (the final distribution).

This figure also illustrates some important features present in CS, such as active clusters, poor clusters, and promising clusters. Active clusters are those whose center changes during the search, such as clusters 3 and 4, providing new local-best solutions. Poor clusters are those that are not activated. For instance, the centers of the clusters 13 and 14 do not change during the search. This happens because their search subspaces are not interesting, and consequently the metaheuristic used as a solution generator is unable to find solutions to assign to these clusters. Promising clusters are the ones to be explored, such as clusters 1, 4, 6, 8 and 9. In particular, cluster 4, which contains the best-known solution, is activated several times through an intensification of the search around its center.

5. Adaptive large neighborhood search

In this section, we present our ALNS heuristic for the WRRP. This heuristic also contains some user-controlled parameters denoted by lower case greek letters, which will be also calibrated in Section 6.

The ALNS heuristic proposed by Ropke and Pisinger (2006a) extends the large neighborhood search (LNS) heuristic introduced by Shaw (1997) based on the ruin and recreate principle. At each iteration, the algorithm *destroys* part of the current solution s and *repairs* it in a different way, generating a new solution s'. This new solution is accepted according to a criterion defined by a search paradigm applied at the master level, such as an acceptance criterion from simulated annealing (SA) where if s' is better than s, the search continues from s', and otherwise the search continues from s' with a given probability.

In ALNS, the destroy and repair procedures are selected according to an adaptive probabilistic mechanism. At each iteration, the probability of selecting a given procedure depends on how well it has performed in the past. ALNS has provided good solutions for a wide variety of vehicle routing problems, see for instance Ropke and Pisinger (2006a,b), Pisinger and Ropke (2007) and Azi et al. (2010).

We now describe the main elements of our ALNS implementation for the WRRP.

1. *Large neighborhood*: Given a solution s, at each iteration, q wells are removed from the solution and are reinserted. This is accomplished by using one of several removal and insertion heuristics. At each iteration a random integer number q satisfying $2 \leqslant q \leqslant \min\{150, \xi|W|\}$ is selected. A parameter ξ controls how many wells can be removed.
2. *Adaptive search engine*: The choice of the removal and insertion heuristics is governed by a roulette-wheel mechanism in which

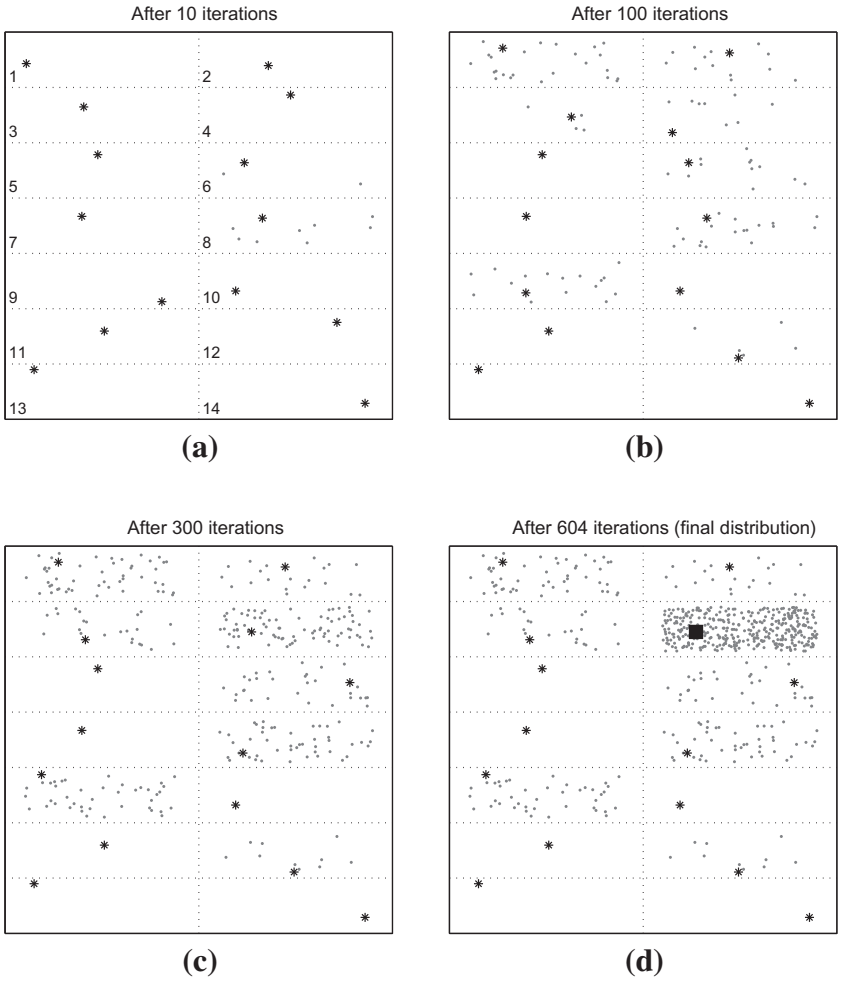

Fig. 2. CS distribution of solutions for the clusters on instance 50/5–1. Dots are the solutions provided by SA and distributed at random into the respective cluster, stars are the centers, and the square is the best-known solution found.

each heuristic is assigned a weight that depends on its past behavior. More precisely, let w_i be a measure of how well heuristic i has performed in past iterations. Then, given h heuristics with weights w_i, heuristic j is selected with probability $w_j/\sum_{i=1}^{h} w_i$. The insertion heuristic is chosen independently of the removal heuristic.

3. *Adaptive weight adjustment*: As in Ropke and Pisinger (2006a), the search is divided into a number of *segments* of φ consecutive iterations. In the first segment, all heuristics have the same weights i.e., $w_i = 1$ for $i = 1,\ldots,h$. After φ iterations, the weights used to select removal and insertion heuristics are updated considering the *score* obtained during the segment. The scores show how well the removal and insertion heuristics have performed in the last segment.
   The score of a heuristic is increased by a parameter equal to σ_1 ,σ_2 or σ_3 when it identifies new solutions. For the WRRP, if a pair of removal-insertion heuristic finds a new best solution, their scores are increased by $\sigma_1 = 50$, if it finds a solution better than the current one, their scores are increased by $\sigma_2 = 10$, and if it finds a non-improving solution which is accepted, their scores are increased by $\sigma_3 = 6$.
   When a segment ends, new weights are calculated using the scores obtained, and all scores are reset to zero for the next segment. Let π_i and o_{ij} be, respectively, the score of the heuristic i, and the number of times that heuristic i has been chosen in the last segment j. Then,

$$w_{ij+1} = \begin{cases} w_{ij} & \text{if} \quad o_{ij} = 0 \\ (1-\eta)w_{ij} + \eta\pi_i/o_{ij} & \text{if} \quad o_{ij} \neq 0, \end{cases} \tag{10}$$

   where $\eta \in [0,1]$ is parameter called the *reaction factor*. This parameter controls how quickly the weight adjustment algorithm reacts to changes in the effectiveness of the heuristics. We have used $\eta = 0.01$ in our implementation.
4. *Penalized objective function*: We also consider solution violating deadlines during the search which is evaluated by (9).
5. *Acceptance and stopping criteria*: The acceptance criterion from SA is used, i.e., given a current solution s, a neighbor solution s' is accepted if $v(s') < v(s)$, and with probability $e^{-(v(s')-v(s))/T}$ otherwise, where $T > 0$ is the current *temperature* and $v(s)$ is the penalized solution cost defined by (9). The temperature starts at T_{start} and is decreased by $T = cT$ every iteration, where $0 < c < 1$ is the *cooling rate*. We have set T_{start} to 25000 and c to 0.99975 in our computational experiments.
6. *Noise to the objective function*: Ropke and Pisinger (2006a,b) indicate that to avoid the myopic behavior of some insertion heuristics, it is necessary to add a noise term to the objective function. However, for the WRRP this strategy did not improve the results, so it was not used.

5.1. Initial solution

We starts from the same initial solution used by CS, and our ALNS is also able to regain feasibility. Different strategies were

tested, like the nearest neighbor and the regret insertion heuristics, but ALNS provided the best solutions with this balanced initial solution.

5.2. Removal (H^-) and insertion (H^+) heuristics

This section describes five removal and three insertion heuristics. Four removal heuristics remove q wells from a solution s and return a solution s'. One removal heuristic removes a route at random and the number q of removed wells depends on the corresponding rig. Where applicable, let D be a set of removed wells.

- *Shaw removal heuristic 1 – SRH1*: Following Ropke and Pisinger (2006a) and Shaw (1997), the general idea of the *Shaw removal heuristic* is to remove wells that are somewhat similar, as it is expected to be reasonably easy to shuffle similar requests around and thereby create new, perhaps better solutions. The degree of similarity between two wells i and j, serviced by rigs k_1 and k_2 respectively, is computed through a *relatedness measure* $R(i,j)$, where a lower value corresponds to more similar wells. The first Shaw Removal Heuristic (SRH1) is based on the absolute difference between the contribution of wells i and j to the objective function. More precisely, the first relatedness measure is given by

$$R_{SHR1}(i,j) = |p_i t_i^{k_1} - p_j t_j^{k_2}|. \quad (11)$$

The relatedness is used according to Shaw (1997) to remove q wells. Given a solution s and a set of removed wells D, the algorithm randomly selects a well w from D, calculates the relatedness measure between it and all wells not yet removed, and then chooses a new well to be inserted in D. The process is repeated while $|D| < q$. According to Ropke and Pisinger (2006a), the parameter $\upsilon \geqslant 1$ is used to avoid determinism when selecting a well.

- *Shaw removal heuristic 2 – SRH2*: This shaw removal heuristic (SRH2) is based on distances. The similarity between two wells i and j is measured by relatedness measure $R_{SHR2}(i,j) = d_{ij}$, where d_{ij} is the distance between i and j. Apart from the relatedness measure, this procedure is identical to the previous one.
- *Random removal heuristic – RRH1*: This simple removal heuristic removes q wells at random for the current solution s. This heuristic tends to generate a poor set of removed wells, but it helps diversifying the search.
- *Route removal heuristic – RRH2*: This is another simple removal heuristic where some route k is chosen at random and its wells are removed. However, as shown in Algorithm 3, a parameter $\phi \geqslant 1$ introduces some randomness in the order of the removed wells. A simple greedy heuristic inserts the wells according to the removal order. Randomness helps diversify the search.

Algorithm 3. RRH2 – Route removal heuristic

```
1  Input: Solution s and φ ⩾ 1
2  Let D be a set of removed wells
3  Let L be an ordered set of wells and L_i the ith well of L
4  L← wells from a non-empty route selected at random from s
5  D ← {}
6  While L ≠ {} do
7  |  Choose a random number y from [0,1]
8  |  i ← ⌊y^φ |L|⌋      D ← D ∪ {L_i}
9  |  Remove L_i from s
10 |  L ← L\{L_i}
11 end
12 Return s and D
```

- *Worst removal heuristic – WRH*: This heuristic removes wells with high cost in the current solution s and attempts to insert them in better positions. Let $ProdLoss^-(i,s) = v(s) - v_{-i}(s)$ be the production loss caused by well i in current solution s, where $v_{-i}(s)$ represents the solution cost without well i into s. WRH first sorts the wells according to $ProdLoss^-(i,s)$, chooses one to be removed, recalculates $ProdLoss^-(i,s)$ for the remaining wells, and the process is repeated. See Algorithm 4 for details and note that the removal is also randomized, but controlled by parameter $\rho \geqslant 1$.

Algorithm 4. WRH – Worst removal heuristic

```
1  Input: Solution s, q and ρ ⩾ 1
2  Let D be a set of removed wells
3  Let L be an ordered set of wells and L_i the ith well of L
4  D ← {}
5  While |D| ≠ q do
6  |  L← wells of s
7  |  Sort L such that i < j ⇒ ProdLoss^-(L_i,s) < ProdLoss^-(L_j,s)
8  |  Choose a random number y from [0,1]
9  |  i ← ⌊y^ρ |L|⌋      D ← D ∪ {L_i}
10 |  Remove L_i from s
11 end
12 Return s and D
```

- *Basic greedy insertion heuristic – BGIH*: Let Δv_{ik} be the change in solution cost incurred by inserting well i into route k in the position that increases the production loss the least in the current solution s. Then, let $ProdLoss^+(i,s) = \min_{k \in K}\{\Delta v_{ik}\}$ be the minimum production loss found representing the *minimum cost position*. If D is a set of removed wells, we calculate $ProdLoss^+(D_1,s)$ and the first well (D_1) is removed from D and inserted in the corresponding route in the best position. Note

Table 1
Values for the CS and ALNS parameters after the tuning phase.

Heuristic	Parameter	Meaning	Best value
CS	γ	Determines the maximum number of clusters.	14
	τ_{max}	Defines the maximum volume for the local search	2
	β_{max}	Defines when a disturbance must be applied	3
	λ	Penalizes a solution violating deadlines in (9)	15
ALNS	ξ	Controls the number of wells to be removed	0.40
	φ	Defines the number of consecutive iterations in a segment	50
	λ	Penalizes a solution violating deadlines in (9)	15
	υ	Avoids determinism in the Shaw removal heuristics	2
	ϕ	Avoids determinism in the route removal heuristic	3
	ρ	Avoids determinism in the worst removal heuristic	2
	κ	Indicates which regret heuristic must be used	3

that this greedy insertion heuristic respects the order of the wells in D. After the first well has been inserted, $ProdLoss^{+}(D_1,s)$ is calculated again and the process is repeated.

- *Deep greedy insertion heuristic – DGIH*: This heuristic works differently from the previous one. Instead of inserting the first well i from D into the current solution s, this heuristic inserts well i with the minimum global cost position. Formally, it inserts well i that minimizes $\min_{i\in D}\{ProdLoss^{+}(i,s)\}$ at its minimum cost position. This process is repeated until all wells have been inserted. Consequently, this heuristic is slower than BGIH.
- *Regret-κ insertion heuristic – RIH*: According to Azi et al. (2010), Potvin and Rousseau (1993) and Ropke and Pisinger (2006a), the *regret* heuristic tries to improve the myopic behavior of greedy heuristics. Given a set D of removed wells, for each well $i \in D$, this heuristic calculates a *regret value* equal to the difference in cost between two solutions in which it is inserted in its best or second best routes. The well i with the maximum regret value is chosen to be inserted in the current solution s. Formally, let $\omega_{ik} \in K$ be a variable that indicates the route for which well i has the k th-lowest insertion cost, i.e., $\Delta v_{i\omega_{ik}} \leqslant \Delta v_{i\omega_{ik'}}$ for all $k \leqslant k'$. Thus, the regret value is $RegretValue_i = \Delta v_{i\omega_{i2}} - \Delta v_{i\omega_{i1}}$ and at each iteration, the heuristic chooses well i according to $\max_{i\in D}\{RegretValue_i\}$. Ties are broken by selecting the lowest cost insertion.

This concept can be extended by considering not only the difference in cost defined above, but also considering the difference in cost of inserting a well $i \in D$ in its best, or 2nd-best, or 3rd-best,..., or its κth-best routes, where κ is a user-defined parameter. With this idea, the *regret-κ* heuristic chooses a well i according to $\max_{i\in D}\left\{\sum_{j=2}^{\kappa}(\Delta v_{i\omega_{ij}} - \Delta v_{i\omega_{i1}})\right\}$ and it is inserted in the minimum cost position. Ties are broken by selecting the lowest cost insertion.

5.3. Pseudo-code of the ALNS

This section presents the general parameters and the pseudo-code of our ALNS. The master framework is based on the SA mechanism, and the removal and insertion heuristics are repeated during 50000 iterations. See Algorithm 5 for more details.

Algorithm 5. ALNS

```
1  Initialize: set λ, set weights of all removal and insertion
   heuristics equal to 1 and all scores equal to 0
2  s ← initial solution()      s_best ← s      T ← T_start
3  While the stop criteria are not met do
4  |  s' ← s
5  |  q ← a random number in [2,min{150, ξ |W|}]
6  |  Choose a removal (H⁻) and insertion (H⁺) heuristic
   |  using the roulette-wheel selection principle based on
   |  weights of the current segment
7  |  Apply H⁻(s',D) and H⁺(s',D)
8  |  Update the number of times using H⁻ and H⁺
9  |  According to Section 5, if necessary, update the weights
   |  of H⁻ and H⁺, verify if the search continues from s or s',
   |  and update s_best
10 |  if the end of the segment is reached then
11 |  |  Update the weights of all removal and insertion
   |  |  heuristics and reset their scores
12 |  end
13 |  T ← cT
14 end
15 if s_best is feasible then
16 |  Return s_best
17 else
18 |  Return "No feasible solution was found"
19 end
```

6. Computational results

The CS and ALNS heuristics were coded in C++and run on a laptop with Pentium Core 2 Duo processor of 2.0 GHz with 3 GB of RAM Memory under Windows operating system.

Our algorithms were tested on the WRRP instances proposed by Neves (2007). These instances are based on problems arising in Brazilian oil fields. They contain 50, 100 and 500 wells, and five and 10 rigs. There are 10 instances for each pair of values, for a total of 60 instances. These instances are labeled $|W||K| - id$, where $|W|$ is the number of wells, $|K|$ is the number of rigs and id is an instance identifier for these two values. The reader can find these instances together with our best solutions at http://www.grmauri.pro.br/problems.html.

6.1. Tuning instances and parameters

To tune the parameters of CS and ALNS, we have chosen eight instances at random: 50/5–5, 50/5–7, 100/5–3, 100/5–4, 50/10–5, 50/10–7, 100/10–3 and 100/10–4. To set the main ALNS parameters, we have followed the methodology presented by Ropke and Pisinger (2006b). The parameters not yet fixed are: ξ, φ, λ, κ, υ, ϕ and ρ. A special attention was given to weight λ. Large values for this parameter often trap the search in a locally suboptimal solution. The parameter ξ also has a significant impact on the results. It was tested on the range [0.05,0.5] with a step size of 0.05. To set these parameters, each parameter was in turn allowed to take several values, while the others were kept fixed. We ran ALNS five times for each parameter setting, and the setting yielding the best average results was chosen. We used the same methodology for the CS parameters. The parameters not yet fixed are: γ, τ_{max}, β_{max} and λ. In this case, γ and λ present the greatest impact on the results. The parameter γ was tested on the range [1,50] with a step size of 1. Table 1 shows the best values found for all parameters.

6.2. Results

This section presents our comparative evaluation of ILS, CS and ALNS. All metaheuristics were applied 10 times to each instance with up to 100 wells, and five times to each instance with 500 wells on the computer used for the tests described in Section 6.

To be fair, we imposed a time limit as unique stopping criterion for all metaheuristics: 30 seconds for each instance with 50 wells, 180 seconds for each instance with 100 wells, and 7200 seconds for each instance with 500 wells. The CS and ALNS heuristics have stopping conditions based on the temperature or on number of iterations. If they reach their stopping conditions before the time limit, we restart the heuristic from the best solution identified during the search. This process is repeated until the heuristic reaches the time limit.

The computational experiments are summarized in Tables 2–4. Five columns are used for all results: the instance name, the best solution found (column Best), the average solution found (column Avge), the deviation (column Dev %) and the standard deviation (column StdDev). The deviation is calculated as Dev (%) = $100 \times$ (Avge − Best*)/Best*, where Best* is the best-known solution value obtained by any of the three heuristics for a given instance. Best values are shown in boldface.

On the 50-well instances, all three heuristics are more or less equivalent, but ILS is slightly better on instances with five rigs. However, as the instance size grows, it becomes clear that ILS has the worst performance of the three heuristics and ALNS has

Table 2
Results for instances with 50 wells (Time limit: 30 seconds).

Instance	ILS				CS				ALNS			
	Best	Avge	Dev (%)	StdDev	Best	Avge	Dev (%)	StdDev	Best	Avge	Dev (%)	StdDev
50/5–1	**42682.12**	42701.30	0.04	24.76	**42682.12**	42805.70	0.29	204.07	**42682.12**	43157.81	1.11	973.30
50/5–2	**45516.36**	45956.64	0.97	428.07	**45516.36**	45884.60	0.81	250.37	**45516.36**	45550.78	0.08	72.57
50/5–3	**41622.72**	41625.63	0.01	9.20	**41622.72**	41785.28	0.39	221.97	**41622.72**	41650.73	0.07	88.56
50/5–4	**40524.17**	40691.36	0.41	270.67	**40524.17**	40967.73	1.09	355.17	**40524.17**	41102.25	1.43	628.87
50/5–5	**42785.74**	42938.67	0.36	181.48	**42785.74**	43047.65	0.61	256.61	**42785.74**	42821.71	0.08	78.90
50/5–6	**44682.51**	44771.24	0.20	145.61	**44682.51**	45140.33	1.02	305.90	**44682.51**	44778.37	0.21	112.09
50/5–7	**51554.44**	51764.56	0.41	664.44	**51554.44**	52172.93	1.20	932.07	**51554.44**	51555.61	0.00	3.71
50/5–8	**37665.35**	37665.35	0.00	0.00	**37665.35**	37887.89	0.59	220.70	**37665.35**	37957.86	0.78	308.33
50/5–9	**43498.26**	43617.40	0.27	376.74	**43498.26**	43692.70	0.45	363.68	**43498.26**	43498.26	0.00	0.00
50/5–10	**42970.28**	43235.19	0.62	428.43	**42970.28**	43072.74	0.24	132.28	**42970.28**	43053.67	0.19	263.71
Avge	**43350.20**	**43496.73**	**0.33**	**252.94**	**43350.20**	**43645.76**	**0.67**	**324.28**	**43350.20**	**43512.71**	**0.40**	**253.00**
50/10–1	**30486.13**	30492.27	0.02	19.43	**30486.13**	30643.26	0.52	183.43	**30486.13**	30486.13	0.00	0.00
50/10–2	**30933.87**	30947.15	0.04	42.02	30952.65	31087.64	0.50	91.60	**30933.87**	30946.77	0.04	19.43
50/10–3	**29352.94**	29397.67	0.15	57.74	**29352.94**	29366.86	0.05	44.02	**29352.94**	29352.94	0.00	0.00
50/10–4	**28826.23**	28850.62	0.08	51.42	**28826.23**	29138.27	1.08	252.62	**28826.23**	28832.55	0.02	14.36
50/10–5	**29348.40**	29391.52	0.15	37.16	**29348.40**	29412.19	0.22	61.81	**29348.40**	29353.32	0.02	4.03
50/10–6	**30462.58**	30489.55	0.09	34.81	**30462.58**	30550.51	0.29	75.71	**30462.58**	30471.71	0.03	13.11
50/10–7	**37502.23**	37502.69	0.00	1.48	**37502.23**	37626.89	0.33	185.11	**37502.23**	37502.23	0.00	0.00
50/10–8	**26451.65**	26466.27	0.06	30.82	**26451.65**	26545.04	0.35	88.66	**26451.65**	26452.07	0.00	1.32
50/10–9	**31350.39**	31625.44	0.88	348.63	**31350.39**	31573.67	0.71	291.54	**31350.39**	31357.61	0.02	15.44
50/10–10	**29825.70**	29863.20	0.13	118.56	**29825.70**	29978.15	0.51	211.31	**29825.70**	29825.70	0.00	0.00
Avge	**30454.01**	**30502.64**	**0.16**	**74.21**	**30455.89**	**30592.25**	**0.46**	**148.58**	**30454.01**	**30458.10**	**0.01**	**6.77**

Table 3
Results for instances with 100 wells (Time limit: 180 seconds).

Instance	ILS				CS				ALNS			
	Best	Avge	Dev (%)	StdDev	Best	Avge	Dev (%)	StdDev	Best	Avge	Dev (%)	StdDev
100/5–1	**134885.07**	136242.70	1.01	1677.65	**134885.07**	136528.76	1.22	725.94	135207.57	138313.56	2.54	2897.98
100/5–2	**111949.03**	114351.14	2.15	2938.16	**111949.03**	112936.70	0.88	1311.38	**111949.03**	113466.74	1.36	1792.95
100/5–3	**124160.04**	125963.10	1.45	1486.90	125019.58	125863.40	1.37	752.70	**124160.04**	125579.39	1.14	972.10
100/5–4	**122246.77**	124633.75	1.95	1590.99	122649.89	123336.68	0.89	739.40	**122246.77**	124362.85	1.73	1871.52
100/5–5	**128874.17**	132071.05	2.48	1693.76	129898.97	132429.34	2.76	1365.56	**128874.17**	130381.58	1.17	1516.60
100/5–6	**114685.46**	116731.17	1.78	1927.48	**114685.46**	115334.01	0.57	661.18	**114685.46**	116038.32	1.18	1459.10
100/5–7	134487.48	136728.23	2.84	2104.01	**132946.09**	134600.91	1.24	1322.71	133027.78	136340.56	2.55	1930.44
100/5–8	128938.09	131050.13	2.37	1069.42	128752.08	130539.15	1.97	1015.04	**128012.29**	128796.90	0.61	618.58
100/5–9	**117343.92**	119279.38	1.65	1423.70	**117343.92**	118445.47	0.94	772.20	117696.52	118593.59	1.06	744.91
100 5–10	123680.15	124711.70	1.27	869.52	123364.34	124213.49	0.87	928.61	**123144.66**	124140.76	0.81	695.85
Avge	**124125.02**	**126176.24**	**1.90**	**1678.16**	**124149.44**	**125422.79**	**1.27**	**959.47**	**123900.43**	**125601.43**	**1.42**	**1450.00**
100/10–1	**82542.56**	83275.84	0.89	590.87	**82542.56**	83175.11	0.77	402.72	**82542.56**	82562.67	0.02	44.46
100/10–2	71627.93	72344.86	1.80	501.27	71129.90	71741.74	0.96	551.70	**71063.02**	71343.49	0.39	243.37
100/10–3	76160.31	76865.16	2.18	485.00	75538.50	75730.28	0.67	178.56	**75227.52**	75484.83	0.34	268.39
100/10–4	70925.80	71512.50	0.95	463.86	71024.66	71457.54	0.87	345.74	**70839.08**	71144.69	0.43	191.09
100/10–5	77635.44	78430.26	1.14	368.38	**77547.66**	78470.64	1.19	388.58	**77547.66**	78200.59	0.84	301.55
100/10–6	70647.29	70914.72	0.83	274.10	**70330.98**	70768.57	0.62	408.08	**70330.98**	70433.05	0.15	167.57
100/10–7	82487.40	83233.91	1.18	729.89	**82261.81**	82618.31	0.43	285.61	**82261.81**	82284.38	0.03	23.43
100/10–8	76728.96	77354.84	0.93	820.80	76927.81	77468.50	1.08	417.78	**76642.32**	76819.63	0.23	128.78
100/10–9	72515.97	73263.71	1.15	521.04	72488.83	72877.33	0.61	335.88	**72433.40**	72788.33	0.49	385.23
100/10–10	75105.01	75864.19	1.33	583.14	74891.56	75252.30	0.51	324.28	**74868.80**	74910.36	0.06	45.83
Avge	**75637.67**	**76306.00**	**1.24**	**533.84**	**75468.43**	**75956.03**	**0.77**	**363.89**	**75375.72**	**75597.20**	**0.30**	**179.97**

the best. When the number of wells reaches 500, ALNS identifies all best solutions, whereas ILS and CS exhibit respective deviations of 4.54% and 2.31% when the number of rigs is five, and of 3.47% and 2.65% when 10 rigs are used. The superior performance of ALNS on larger instances is consistent with the findings of Ropke and Pisinger (2006b) who observed a similar phenomenon on the pickup and delivery vehicle routing problem with time windows. Finally the standard deviation values are relatively low when compared to the mean values, and seem to be unrelated with the quality of the heuristics.

7. Conclusions

In this paper we have developed a clustering search algorithm (CS) and an adaptive large neighborhood search heuristic (ALNS) for the workover rig routing problem (WRRP). On test instances, our two heuristics yield good results and outperform the rather good metaheuristic of Neves (2007). Our results clearly show the superiority of the ALNS. Although CS is a good heuristic for large instances, it is dependent of the metaheuristic used as a solution generator for clustering. On the other hand, the behavior of ALNS

Table 4
Results for instances with 500 wells (Time limit: 7200 seconds).

Instance	ILS				CS				ALNS			
	Best	Avge	Dev (%)	StdDev	Best	Avge	Dev (%)	StdDev	Best	Avge	Dev (%)	StdDev
500/5–1	1788318.75	1807569.37	5.23	14104.61	1746744.06	1771806.34	3.15	17491.72	**1717773.25**	1738287.16	1.19	16223.25
500/5–2	1736967.30	1771071.92	5.42	20219.68	1692270.48	1709409.86	1.75	14722.60	**1679938.72**	1698992.17	1.13	17294.95
500/5–3	1774514.14	1785148.47	3.93	8167.72	1739351.73	1757175.22	2.30	12577.66	**1717592.06**	1737060.87	1.13	15869.17
500/5–4	1827894.08	1847832.75	4.81	18574.39	1772737.75	1793148.99	1.71	19644.91	**1762963.29**	1781777.39	1.07	13454.84
500/5–5	1803351.12	1823863.35	4.89	17380.35	1769305.26	1791690.15	3.04	20110.21	**1738829.49**	1757013.68	1.05	17866.48
500/5–6	1708505.61	1723329.31	4.13	14001.77	1685117.91	1696332.21	2.50	7339.03	**1655005.50**	1674112.35	1.15	15043.92
500/5–7	1777556.59	1803114.91	4.48	26718.14	1759921.91	1768316.12	2.47	5400.45	**1725762.73**	1740087.02	0.83	11169.08
500/5–8	1756629.76	1792085.55	4.20	31479.15	1722318.81	1736616.89	0.98	9376.54	**1719784.75**	1730489.35	0.62	23943.09
500/5–9	1788994.97	1812745.86	4.61	24946.02	1775926.03	1787367.08	3.14	14885.21	**1732897.88**	1776473.37	2.51	35170.77
500/5–10	1795375.84	1826407.18	3.68	17920.57	1775089.29	1797188.28	2.02	21246.43	**1761643.26**	1785499.36	1.35	17041.00
Avge	**1775810.82**	**1799316.87**	**4.54**	**19351.24**	**1743878.32**	**1760905.11**	**2.31**	**14279.48**	**1721219.09**	**1741979.27**	**1.21**	**18307.66**
500/10–1	922408.83	933890.85	3.39	6490.90	926963.87	931956.86	3.17	3961.99	**903293.37**	913586.75	1.14	11007.78
500/10–2	909522.44	922052.04	3.67	7678.67	897149.86	909834.01	2.29	8668.48	**889432.38**	895175.82	0.65	4651.35
500/10–3	910636.72	926394.04	2.51	9224.27	913846.94	917410.79	1.52	4032.93	**903713.77**	916647.23	1.43	11061.61
500/10–4	948053.29	961203.31	4.38	14517.50	934619.14	943525.87	2.46	6845.97	**920881.61**	945185.83	2.64	13796.74
500/10–5	939733.89	953965.48	3.54	10327.20	933400.38	937338.92	1.74	4144.03	**921325.42**	928800.59	0.81	7416.66
500/10–6	904980.72	911738.88	4.17	7400.32	895408.82	902480.11	3.12	4213.01	**875207.53**	889892.63	1.68	11159.17
500/10–7	923907.42	929228.41	3.73	4573.53	918879.11	924944.85	3.25	7191.96	**895856.03**	908202.17	1.38	6963.82
500/10–8	905134.93	920646.78	2.16	10130.61	920384.37	928185.15	2.99	7194.65	**901208.41**	918745.82	1.95	10966.76
500/10–9	933735.17	941773.84	4.35	8037.37	929631.90	937514.66	3.88	6609.11	**902533.41**	911338.19	0.98	6963.04
500/10–10	950785.92	961521.34	2.82	8597.87	949599.27	955040.10	2.12	4316.81	**935174.94**	953597.53	1.97	12112.11
Avge	**924889.93**	**936241.50**	**3.47**	**8697.82**	**921988.37**	**928823.13**	**2.65**	**5717.89**	**904862.69**	**918117.26**	**1.46**	**9609.90**

is dependent of several removal and insertion heuristics, but its adaptive layer makes this heuristic more robust, which explains its superior performance on the WRRP.

Acknowledgements

Glaydston Mattos Ribeiro and Geraldo Regis Mauri acknowledge Espírito Santo Research Foundation (Process 45391998/09) and National Council for Scientific and Technological Development (Processes 201509/2009-1, 300747/2010-1 and 477148/2011-5) for their financial support. Gilbert Laporte acknowledges the Canadian Natural Sciences and Engineering Research Council (Grant 39682-05). The authors thank Tiago Araújo Neves from Fluminense Federal University for kindly providing the data sets and ILS code used in this paper and for answering questions regarding the data sets and the problem. Thanks are due to the referees for their valuable comments.

References

Aloise, D.J., Aloise, D., Rocha, C.T.M., Ribeiro, C.C., Filho, J.C.R., Moura, L.S.S., 2006. Scheduling workover rigs for onshore oil production. Discrete Applied Mathematics 154, 695–702.

Azi, N., Gendreau, M., Potvin, J.-Y., 2010. An adaptive large neighborhood search for a vehicle routing problem with multiple trips. Technical Report 2010-08, CIRRELT, Montréal. https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2010-08.pdf.

Chaves, A.A., Lorena, L.A.N., 2010. Clustering search algorithm for the capacitated centered clustering problem. Computers and Operations Research 37, 552–558.

Fischetti, M., Laporte, G., Martello, S., 1993. The delivery man problem and cumulative matroids. Operations Research 41, 1055–1064.

Glover, F., 1996. Tabu search and adaptive memory programming: Advances, applications and challenges. In: Barr, R., Helgason, R., Kennington, J. (Eds.), Interfaces in Computer Science and Operations Research. Kluwer, Boston, pp. 1–75.

Hamming, R.W., 1950. Error detecting and error correcting codes. Bell System Technical Journal 26, 147–160.

Heilporn, G., Cordeau, J.-F., Laporte, G., 2010. The delivery man problem with time windows. Discrete Optimization 7, 269–282.

Johnson, D.S., McGeoch, L.A., 2002. Experimental analysis of heuristics for the STSP. In: Gutin, G., Punnen, A. (Eds.), The Traveling Salesman Problem and its Variations. Kluwer, Boston, pp. 369–443.

Kallehauge, B., Larsen, J., Madsen, O.B.G., 2006. Lagrangian duality applied to the vehicle routing problem with time windows. Computers and Operations Research 33, 1464–1487.

Letchford, A.N., Lysgaard, J., Eglese, R.W., 2007. A branch-and-cut algorithm for the capacitated open vehicle routing problem. Journal of the Operational Research Society 58, 1642–1651.

Li, F., Golden, B.L., Wasil, E.A., 2007. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. Computers and Operations Research 34, 2918–2930.

Lourenço, H.R., Martin, O., Stützle, T., 2002. Iterated local search. In: Glover, F., Kochenberger, G. (Eds.), Handbook of metaheuristics, International Series in Operations Research and Management Science. Kluwer, Boston, pp. 321–353.

Lucena, A., 1990. Time-dependent traveling salesman problem: The deliveryman case. Networks 20, 753–763.

Méndez-Días, I., Zabala, P., Lucena, A., 2008. A new formulation for the traveling deliveryman problem. Discrete Applied Mathematics 156, 3223–3237.

Neves, T.A., 2007. Heurísticas com memória adaptativa aplicadas ao problema de roteamento e scheduling de sondas de manutenç ao (Heuristics with adaptive memory applied to workover rig routing and scheduling problem). Master's thesis, Fluminense Federal University (UFF), Niterói. http://www.ic.uff.br/satoru/index.php?id=3.

Ngueveu, S.U., Prins, C., Wolfler-Calvo, R., 2010. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. Computers and Operations Research 37, 1877–1885.

Oliveira, A.C.M., Lorena, L.A.N., 2004. Detecting promising areas by evolutionary clustering search. In: Advances in artificial intelligence SBIA 2004. Lecture Notes in Artificial Intelligence, Vol. 3171. Springer, Berlin, pp. 385–394.

Oliveira, A.C.M., Lorena, L.A.N., 2007. Hybrid evolutionary algorithms and clustering search. Hybrid Evolutionary Algorithms, Studies in Computational Intelligence 75, 77–99.

Pacheco, A.V.F., Ribeiro, G.M., Mauri, G.R., 2010. A GRASP with path-relinking for the workover rig scheduling problem. International Journal of Natural Computing Research 1, 1–14.

Pisinger, D., Ropke, S., 2007. A general heuristic for the vehicle routing problem. Computers and Operations Research 34, 2403–2435.

Potvin, J.-Y., Rousseau, J.-M., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research 66, 331–340.

Ropke, S., Pisinger, D., 2006a. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40, 455–472.

Ropke, S., Pisinger, D., 2006b. A unified heuristic for a large class of vehicle routing problems with backhauls. European Journal of Operational Research 171, 750–775.

Shaw, P., 1997. A new local search algorithm providing high quality solutions to vehicle routing problems. Technical Report, University of Strathclyde, Glasgow.

Stützle, T., 2006. Iterated local search for the quadratic assignment problem. European Journal of Operational Research 174, 1519–1539.

Tsitsiklis, J.N., 1992. Special cases of traveling salesman and repairman problems with time windows. Networks 22, 263–283.