A GRASP with Adaptive Memory for a Period Vehicle Routing Problem

Luciana B. Gonçalves, Luiz S. Ochi and Simone L. Martins
Departamento de Ciência da Computação
Universidade Federal Fluminense
Niterói, Rio de Janeiro, Brazil
lgoncalves,satoru,simone@ic.uff.br

Abstract

We present some proposals to approximately solve a period vehicle routing problem used to model the extraction of oil from a set of onshore oil wells in Brazil. This problem differs from the well-known period vehicle routing problem in several aspects. One major difference between them, responsible for increasing the complexity of the problem, is that, in the proposed problem, the number of visits required by a customer during the period is not previously determined. We developed some pure GRASP heuristics and some other heuristics that include the use of memory in GRASP. Experimental results illustrate the effectiveness of GRASP with adaptive memory over pure GRASP heuristics.

1 Introduction

Vehicle routing problems (VRP) [1] consist of minimizing the cost of supplying a set of customers by a fleet of vehicles operating from a central facility. Several versions of the VRP have been studied in the literature modeling practical applications that present specific objectives and constraints. One of these versions does not oblige that all customers are visited, i.e, only a subset of them may be supplied. Another class of VRP, called the Period Vehicle Routing Problem (PVRP) [2], deals with the problem of designing the visits to the customers for each day of a given period. In this paper, we develop a model for a real application found in the Northeastern part of Brazil concerning the exploitation of oil in onshore oil wells by joining the constraints found in these two problems.

In the state of Rio Grande do Norte, the onshore oil field has approximately 5.000 wells and produces about 10% of the Brazilian oil production. Most of these wells rely on artificial lift methods to bring the oil to the surface [3]. The service of oil lifting is performed by mobile units that carry a bump used to lift the oil from the wells. Due to their high operation costs, there are relatively few bump mobile units (BMUs) compared to the number of wells. Every day of a predetermined period, the BMUs should leave a Treatment Oil Station (TOS) and visit some of the wells to extract oil, returning to the TOS. After bumping the oil from a well, a BMU may revisit the well only after a day interval to allow the oil well to recover and to make the next bumping profitable. The number of days for recovering is different for each well. As there are fewer BMUs than wells, sometimes a well may not have its oil lifted for a period longer that its recovery time, introducing a loss of oil production. The aim of this problem is to generate daily tours for each BMU, starting and finishing in TOS, so that the amount of oil extracted in a predetermined period is maximized.

We can find in the literature some works to solve this problem [3], but all of them try to find good feasible routes for each day, not considering all days of the period. We think that it is possible to find better solutions by planning the routing for the predetermined period, treating it as a Period BMU Routing Problem (PBMURP). We developed a mathematical formulation for a predetermined planning period based on a mixed integer linear programming model. Due to its high complexity, constructive and local search heuristics were proposed in order to develop Greedy Randomized Search Procedure (GRASP) [4] heuristics able to find optimal or near optimal solutions in feasible computational time. Experimental analyses were performed to show the influence of constructive and local search heuristics over GRASP heuristics performance. GRASP is a method that executes several independent iterations. We use well-known methods to introduce some memory in this method [5]. Computational results obtained for randomly generated instances are reported, showing that the proposed algorithms outperform pure GRASP in terms of quality solution.

In Section 2, we present a mathematical formulation developed for this problem. In Section 3, the proposed heuristic algorithms are described and in Section 4, computational results for these methods are shown. Some conclusions are discussed in Section 5.

2 The Period Bump Mobile Units Routing Problem(PBMURP)

In this section we present a mathematical formulation for PBMURP. Consider the undirected complete graph $G = (V, E)$, where V is the set containing the n oil wells and the Treatment Oil Station (TOS), and E is the set of edges (u, v) where u and $v \in V$, which have associated weights $t_{u,v}$ that represent the time spent to travel between wells u and v. The objective is to find a daily route for each BMU for the period of d days, considering that the travel time can not exceed the input parameter T_{max}, and that a well can only be revisited after its recovery day interval is achieved. This formulation is based on [2].

The following notation is used:

Constants

V = set of wells including TOS, $|V| = n + 1$
D = set of days of the planning period, $|D| = d$
B = set of BMUs, $|B| = b$
t_{ij} = travel time between *i* e *j*, $\forall i, j \in V$
p_i = amount of oil that may be extracted from well *i*
s_i = time that a BMU spent lifting the oil from well *i*
r_i = minimum number of days between two consecutive visits to well *i*
T_{max} = maximum allowable time spent by a BMU on a daily route

Variables

$$x_{il} = \begin{cases} 1, & \text{if well } i \text{ is visited in day } l \\ 0, & \text{otherwise} \end{cases}$$

$$z_{ijkl} = \begin{cases} 1, & \text{if well } j \text{ is the next well visited after} \\ & \text{the well } i \text{ by BMU } k \text{ in day } l \\ 0, & \text{otherwise} \end{cases}$$

w_{ijkl}= amount of oil that flows in edge *(i,j)* carried by BMU *k* in day *l*

Using this notation, the PBMURP may be formulated as follows:

$$\max \sum_{l=1}^{d} \sum_{i=1}^{n} p_i * x_{il} \quad (1)$$

Subject to:

$$\sum_{i=0}^{n} \sum_{j=0, j\neq i}^{n} (t_{ij} + s_i) * z_{ijkl} \leq T_{max}; \forall l \in D, \forall k \in B \quad (2)$$

$$\sum_{k=1}^{b} \sum_{i=0; i\neq q}^{n} z_{iqkl} = x_{ql}; \forall q \in \overline{V}, \forall l \in D \quad (3)$$

$$\sum_{k=1}^{b} \sum_{i=0; i\neq q}^{n} z_{qikl} = x_{ql}; \forall q \in \overline{V}, \forall l \in D \quad (4)$$

$$\sum_{i=0; i\neq q}^{n} z_{iqkl} - \sum_{i=0; i\neq q}^{n} z_{qikl} = 0; \forall q \in \overline{V}, \forall l \in D, \forall k \in B \quad (5)$$

$$\sum_{k=1}^{b} z_{ijkl} \leq \frac{x_{il} + x_{jl}}{2}; \forall i \in \overline{V}, j \in \overline{V}, (i \neq j), \forall l \in D \quad (6)$$

$$x_{0l} = 1; \forall l \in D \quad (7)$$

$$\sum_{j=1}^{n} z_{0jkl} = 1; \forall l \in D, \forall k \in B \quad (8)$$

$$\sum_{i=1}^{n} z_{i0kl} = 1; \forall l \in D, \forall k \in B \quad (9)$$

$$\sum_{j=0; j\neq q}^{n} w_{qjkl} = \sum_{i=0; i\neq q}^{n} (w_{iqkl} + v_q \times z_{iqkl}); \forall k \in B, \forall l \in D, \forall q \in \overline{V} \quad (10)$$

$$w_{0jkl} = z_{0jkl}; \forall j \in \overline{V}, \forall k \in B, \forall l \in D \quad (11)$$

$$w_{ijkl} \geq z_{ijkl}; \forall i \in V, \forall j \in V, (i \neq j), \forall k \in B, \forall l \in D \quad (12)$$

$$z_{ijkl} \geq \frac{w_{ijkl}}{\sum_{j=1}^{n} v_j}; \forall i \in V, \forall j \in V, (i \neq j), \forall k \in B, \forall l \in D \quad (13)$$

$$\sum_{l=l'+1}^{l'+r_i} x_{il} \leq (1 - x_{il'}); \forall l' \in D, \forall i \in \overline{V} \quad (14)$$

$$x_{il} \in \{0, 1\}, \forall i \in V, \forall l \in D \quad (15)$$

$$z_{ijkl} \in \{0, 1\}, \forall i \in V, \forall j \in V, \forall k \in B, \forall l \in D \quad (16)$$

$$w_{ijkl} \geq 0, \forall i \in V, \forall j \in V, \forall k \in B, \forall l \in D \quad (17)$$

The objective function (1) maximizes the amount of oil collected by all BMUs during the period D.

Constraints (2) guarantee that the time spent by a BMU on a daily route is less than the limit T_{max}. Constraints (3), (4), and (5) state that each well *i* is serviced by no more than one particular BMU in a specific day. Constraints (6) guarantee that if a well j should be visited after well i in a day l, then just one specific BMU must visit both wells. Constraints (7) determine that the TOS is visited in all days of the period. Constraints (8) and (9) ensure that no empty routes are generated, i.e, each BMU should collect the oil from at least one well in each day of the period. Constraints (10) state that the amount of oil carried by a BMU that

leave a well $q \in V$ has to be the sum of the amount of oil that the BMU has when it arrives at the well q added to the amount of oil lifted from the well q by the BMU. As the Treatment Oil Station (TOS) belongs to V but does not produce any oil, Equations (11) associate to all edges used by the BMUs to leave the TOS an oil flow with unit value. This flow is not computed by the cost function (1), because TOS (node 0) is not included in the computation of this function. Constraints (12) ensure that if a BMU travels by a specific edge on a specific day ($z_{ijkl} = 1$), then there should have a positive and integer flow associated this use ($w_{ijkl} \geq 1$). Constraints (13) guarantee that if there is a flow associated to an edge ($\frac{w_{ijkl}}{\sum_{j=1}^{n} v_j} > 0$), then there should have a BMU that travels by this edge on the same corresponding day ($z_{ijkl} = 1$). Constraints (14) do not allow a well to be visited before its recovery day interval.

3 GRASP heuristics

GRASP is an iterative process, where each iteration consists of two phases: construction and local search. In the construction phase a feasible solution is built, and its neighborhood is explored by a local search. The result is the best solution found over all iterations.

The construction phase of GRASP is an iterative process where, at each iteration, the elements that do not belong to the partial solution are evaluated by a greedy function, which estimates the gain of including it in the partial solution. They are ordered by their estimated value in a list called restricted candidate list (RCL) and one of them is randomly chosen and included in the solution. The size of the RCL is limited by a parameter α. This process stops when a feasible solution is obtained.

The solutions generated by the construction phase are not guaranteed to be locally optimal. Usually a local search is performed to attempt to improve each constructed solution. It works by successively replacing the current solution by a better one from its neighborhood, until no more better solutions are found.

Path-relinking is a technique proposed by Glover [6] to explore possible trajectories connecting high quality solutions, obtained by heuristics like tabu search and scatter search. The pure GRASP metaheuristics is a memoryless method, because all iterations are independent and no information about the solutions is passed from one to iteration to another. The objective of introducing path-relinking to a pure GRASP algorithm is to retain previous good solutions and use them as guides in the search of new good solutions [5].

In the next subsections, we describe the construction and local search heuristics developed for generating GRASP algorithms, and the path-relinking strategy developed to improve GRASP performance.

3.1 Construction Phase

The strategy implemented for the construction phase is to build, for each day of the period D, a route for each of the BMUs, so that the time limit for each route is not exceeded and the largest possible quantity of oil is collected. The Algorithm 1 shows the basic steps for constructing a solution. Let s be the set of $|D| \times |B|$ routes, where D is the set of the days and B is the set of BMUs. In line 2, a candidate list CL is created containing all wells that can have their oil lifted in the specific day i. An oil well can have its oil extracted only if its recovery day interval has already been achieved. Then, a route is generated for each BMU from line 3 to 15 starting from the Treatment Oil Station. The first element w to be inserted in the route is selected in line 4. If the travel time for the route TOS-w-TOS does not exceed the time limit, then a Restricted Candidate List (RCL) is created in line 6, w is inserted in the solution in line 8, and from line 7 to 11, the procedure tries to insert new elements to this route. In line 9, the RCL is updated after a new element is inserted in the partial solution, and in line 10 a well is selected from RCL.

Algorithm 1 Construction Heuristic

```
1:  for i=0 to |D| do
2:      CL = Available(V,i)
3:      for j=0 to |B| do
4:          w ← Select_First_Well(CL)
5:          if TimeLimit_NotExceeded(s,i,j,w) then
6:              Create_RCL(CL-{w})
7:              while TimeLimit_NotExceeded(s,i,j,w) do
8:                  Insert_Well_Route(s,i,j,w)
9:                  Update_RCL
10:                 w ← Choose_Element_RCL
11:             end while
12:         end if
13:         2-optimal(s,i,j)
14:         Insert_New_Elements(s,i,j,V)
15:     end for
16: end for
```

To select the first element of a BMU route, all candidates $c \in CL$ are classified in descending order according to a function f, which favors elements that may produce greater oil amount(*oilprod*(c)) and are closer to TOS (*distance*(c,TOS)), as presented in (18).

$$f(c) = \textit{oilprod}(c)/(\textit{distance}(c, TOS)) \qquad (18)$$

Then, considering that f_{min} and f_{max} are the minimum and maximum values found in evaluating f for all candidates from CL, the elements, which present a value for f greater than lim_{RCL}, are selected, where α is selected from the

interval $[0, 1]$ and:

$$lim_{RCL} = \alpha * f_{min} + (1 - \alpha) * f_{max} \quad (19)$$

Then, an element is randomly selected from this subset.

After the first element is inserted in a route, a RCL is created according to some criterion, a new element is randomly selected from this RCL, and inserted in the route. This process is repeated until the time limit is achieved for the route or there are no more wells to be inserted. We developed two different ways of creating the RCL, which are described below.

The sweep heuristic (SH) is based on the *sweep algorithm* developed by Gillet and Miller [7]. With the TOS as a pivot, a ray, originating from the TOS, and passing through the starting well of the route partially constructed, is rotated in a counter-clockwise direction. As the wells are swept by this ray, they are inserted in the RCL until the maximum number of elements for the RCL is achieved. Then, the elements are sorted according to the function f presented in (18), and one element is randomly selected from RCL. If adding it to the current route does not violate the time limit T_{max}, it is inserted in the partially constructed route. A new ray is defined connecting the TOS and the last inserted well, and this process is repeated until adding a new element would violate the time limit T_{max}.

The second heuristic called Randomized Nearest Insertion Heuristic (RNIH) is based on the Nearest Insertion heuristic developed to find a minimal-weight hamiltonian cycle of a graph $G(V, E)$ [8].

Initially, the first well is selected in the same way described before, and a route is constructed joining the TOS to this well. Then, for each element already inserted in the partially constructed route, we select its k nearest neighbors. All these candidates c are sorted in decrescent order according to (20):

$$fn(c) = \textit{oilprod}(c)/(\textit{distance}(c, nearest\_neighbor) \quad (20)$$

The RCL is built by selecting the candidates, that present a value for fn greater than lim_{RCL}, where α is selected from the interval $[0, 1]$ and:

$$lim_{RCL} = \alpha * fn_{min} + (1 - \alpha) * fn_{max} \quad (21)$$

Then, an element is randomly selected from RCL, and we calculate the increments in the route travel time caused by inserting it before or after its nearest neighbor. The position which brings the little increment is chosen. This process is repeated until adding a new element to the current route would violate the time limit T_{max}.

After a feasible route is determined for day i and for BMU j, two improving functions are applied one after another: 2-optimal and Insert_New_Elements, as shown in lines 13 and 14 of Algorithm 1. The first one tries to find a better visiting order for the wells associated to this route, so that performing the route would take less time. The second tries to insert available wells to this route, which are not yet associated to it.

3.2 Local Search Phase

After a solution is constructed, a local search phase should be executed to attempt to improve the initial solution. We developed two independent procedures for performing the local search. The first one tries to enhance the solution by searching for better routes for each day of the period, and the second one looks for greater oil extraction by visiting the wells more days in the period. We apply these two procedures one after another to implement the local search phase.

3.2.1 Daily Local Search-DLS

The objective of this procedure is to find better routes for the wells allocated by the construction phase for a specific day of the period, and also to try to insert available not yet visited wells in these routes. The Algorithm 2 receives as an input parameter the solution s which contains, for each BMU, the routes that they should perform in each day of the period. In line 4, the neighborhood $N(w)$ is defined by searching for the k wells located nearer to the well w visited in a specific day i by a BMU j. For each neighbor, which is not visited by the same BMU j that visits well w, we verify if taking it from its original route and inserting it in the route containing well w implies in less travel time for these two routes. If it is true, this change is processed in line 6. After verifying the neighborhood of all wells, we try to insert in this route wells that are not visited in day i by any BMU, assuring that the solution remains feasible, as shown in line 10.

Algorithm 2 Daily Local Search (s)

```
1:  for i=0 to |D| do
2:    for j=0 to |B| do
3:      for all well w ∈ s[i, j] do
4:        for v ∈ N(w) do
5:          if Well_Not_In_Route(v, w) and Profit(v, w)
            then
6:            Insert_Well_Route(s, v, w, i, j)
7:          end if
8:        end for
9:      end for
10:     Insert_AvailableWells(s)
11:   end for
12: end for
```

3.2.2 Period Local Search-PLS

This procedure tries to enlarge the amount of oil extracted from the wells. For this purpose, we calculate for each well w, the loss $L(w)$, which corresponds to the amount of oil that could be collected in the period and is not being extracted in the solution s. The loss $L(w)$ is calculated by subtracting the amount of oil extracted in the current solution from the possible largest amount of oil that could be obtained in the same solution, if all wells were visited, respecting their interval times for being revisited.

The algorithm 3 shows each step of PLS. From lines 1 to 3, all oil production wasted in solution s is calculated for each well i. In line 4, a list WL is constructed containing all wells sorted in decrescent order according to its loss. For each of the first δ wells from WL, in line 6, a new visit days combination is created so that the well may be visited all possible days in the period. In line 7, this combination of visiting days is inserted in the current solution by the following way. If the well should be visited in the new combination and is not visited in the current solution, then the well is inserted in a current route that contains its nearest neighbor. In case that the well is in the current solution, but should not be in the new combination, the well is deleted from the current route. This process may create routes that exceed the travel limit time T_{max}. So, in line 9, each unfeasible route is made feasible by removing wells that present the major ratio between the decrease in the travel time by taking it out of the route and its oil production. Due to these modifications, it may be feasible to insert additional wells into the routes of the new solution. This procedure is executed in line 10.

Algorithm 3 Period Local Search(s, δ)

1: **for** i=1 to $|V|$ **do**
2: $L[i] \leftarrow Calculate\_Loss(i)$
3: **end for**
4: $WL \leftarrow Sort\_Wells\_Loss(L)$;
5: **for** i = 1 to δ **do**
6: $New\_Visits \leftarrow Generate\_New\_Visits(WL[i])$
7: $Insert\_New\_Visits(s)$
8: **end for**
9: $Make\_Feasible(s)$
10: $Insert\_Available\_Wells(s)$

3.3 GRASP and Path-relinking

We developed two pure GRASP algorithms by combining one of the two construction procedures with the local search procedure. To improve the pure GRASP procedures, we used path-relinking as an intensification strategy. An elite set S_{Elite} is maintained, in which good solutions found in GRASP iterations are stored to be combined with solutions created by other GRASP iterations. For each solution s generated in a GRASP iteration, the solution e which is more diverse from s is selected from S_{Elite} and a path-relinking is applied between them. A path-relinking is performed by starting from an initial solution s_i, which is the one with better cost between s and e, and gradually incorporating attributes from a guide solution s_g to it, which is the one that presents worst cost, until s_i becomes equal to s_g. This procedure is shown in Algorithm 4. The attributes considered to differ one solution from another are the routes performed by each BMU in each day. In line 1, the most diverse element from the current solution s is selected from S_{elite}. Then, in line 2, the initial and guide solutions are determined based on their solution costs. In line 3, we check the attributes that are equal in the initial and guide solution, i.e, we verify which days the routes for the BMUs are different. From line 4 to line 11, the initial solution is changed to the guide solution, by changing, for each day, the routes of the initial solution to the routes of the guiding one. The changing of the routes of one day may generate an unfeasible solution, because the period for the recovery of a well may be not respected. So, in line 7, a procedure to make this new solution feasible is performed, by taking these wells out of the routes. In line 9, the current best solution of the GRASP procedure is updated if the new solution generated by the path-relinking presents a better cost.

Algorithm 4 Path_Relinking(s, S_{elite}, s^*)

1: $e \leftarrow$ Most_Diverse(S_{elite}, s);
2: $Determine\_Initial\_Guide(s, e, s_i, s_g)$;
3: $\Delta \leftarrow Different\_Attributes(s_i, s_g)$;
4: **while** $\Delta \neq \phi$ **do**
5: $s_{pr} \leftarrow Best\_Attribute\_Change(s_i, s_g, \Delta)$;
6: Update(Δ);
7: Make_Feasible(s_{pr});
8: **if** (s_{pr} presents better cost than the best solution s^* **then**
9: $s^* \leftarrow s_{pr}$;
10: **end if**
11: **end while**

4 Computational Results

We tested two pure GRASP procedures G1 and G2. G1 combines the construction heuristic SH with the local search described in 3.2, and G2 combines the RNIH heuristic with the same local search procedure. We also tested the effect of introducing path-relinking to both strategies (G1PR and G2PR).

GRASP algorithms were implemented in C, compiled with gcc compiler version 3.3.3 and were tested on a Intel Pentium 4 2.80 GHz with 512 Mbytes of RAM.

For the first set of computational experiments, we created twelve small instances for the problem with number of days of the period varying from 1 to 3, number of BMUs equal to 1 or 2, and number of wells in the interval $[5, 15]$. These instances were solved using the mathematical formulation developed by us and implemented using the XPRESS 2005 tool. These tests were performed on a AMD Athlon 1.6 GHz with 256 Mbytes of RAM. In Table 1, the first column identifies the instance: the first number is the number of days of the period, the second corresponds to the number of BMUs and the third one to the number of wells. The second column shows the optimal values found by the exact algorithm using the software XPRESS, and the third column (XP) presents the computational time in seconds spent to find these values. The next four columns show the average computational times in seconds spent by the four proposed heuristics to find these optimal values. Each instance was executed by each heuristic ten times using different seeds. All heuristic algorithms found the optimal values in much less time than the exact procedure, and all present similar computational times.

		XP	G1	G2	G1PR	G2PR
Inst.	Opt.	T(s)	T(s)	T(s)	T(s)	T(s)
I[1-1-05]	114	0.1	0.00	0.00	0.00	0.00
I[1-1-10]	450	0.4	0.01	0.01	0.01	0.01
I[2-1-10]	900	1.5	0.02	0.03	0.03	0.03
I[2-2-10]	1300	216.4	0.03	0.03	0.04	0.03
I[1-1-13]	600	2.7	0.02	0.02	0.03	0.03
I[2-1-13]	1110	148.5	0.04	0.04	0.04	0.04
I[2-2-13]	1610	823.8	0.05	0.05	0.06	0.06
I[1-1-14]	300	0.2	0.02	0.01	0.02	0.02
I[1-2-14]	550	3.1	0.03	0.03	0.03	0.03
I[2-2-14]	1050	123.9	0.05	0.05	0.05	0.05
I[3-2-14]	1350	3294.1	0.05	0.07	0.05	0.04
I[3-1-15]	620	4366.1	0.02	0.02	0.02	0.06

Table 1. Results obtained by XPRESS and GRASP heuristics

For the second set of experimental tests, twelve larger instances for the problem were created with number of days of the period chosen from the interval $[5, 14]$, number of BMUs equal to 2 or 3, and number of wells in the interval $[50, 250]$.

Each GRASP heuristic performed 200 iterations. Each instance was executed ten times using different random seeds. Tables 2, 3 and 4 present the best value found and the average value of the solution cost. The first column in all tables identifies the instance and bold values indicate best values.

In Table 2, the results obtained for the pure GRASP heuristics G1 and G2 are shown. We can see that G2 found better solutions than G1 for all instances. Table 3 shows the results obtained for the pure GRASP algorithm G1 and with path-relinking (G1PR). We can see that path-relinking can substantially improves the performance of G1. For all instances, except the instance I [10-03-100], G1 with path-relinking found better results than G1.

Inst.	G1		G2	
	Average	Best	Average	Best
I [10-02-050]	2616.7	2778.0	**2934.2**	**3075.0**
I [07-02-050]	3727.5	3906.0	**4454.7**	**4683.0**
I [07-03-050]	4570.0	4850.0	**5330.0**	**5450.0**
I [10-02-050]	5875.0	6400.0	**7165.0**	**7400.0**
I [05-02-050]	1982.5	2057.0	**2149.8**	**2276.0**
I [15-03-100]	29380.5	30015.0	**33277.3**	**33587.0**
I [10-03-100]	17392.4	17751.0	**19460.8**	**19884.0**
I [14-02-150]	14991.2	15556.0	**18973.1**	**19663.0**
I [07-02-150]	7062.2	7279.0	**9072.1**	**9516.0**
I [10-02-200]	13646.6	14560.0	**17438.6**	**17862.0**
I [10-02-250]	28688.0	29760.0	**39376.0**	**39680.0**
I [10-03-250]	23599.0	24438.0	**31525.2**	**31765.0**

Table 2. Solution costs obtained for pure GRASP heuristics G1 and G2

Inst.	G1		G1PR	
	Average	Best	Average	Best
I [10-02-050]	2616.7	2778.0	**2967.9**	**3007.0**
I [07-02-050]	3727.5	3906.0	**4378.6**	**4455.0**
I [07-03-050]	4570.0	4850.0	**4960.0**	**5000.0**
I [10-02-050]	5875.0	6400.0	**6325.0**	**6450.0**
I [05-02-050]	1982.5	2057.0	**2198.8**	**2238.0**
I [15-03-100]	29380.5	30015.0	**29981.5**	**30116.0**
I [10-03-100]	17392.4	**17751.0**	**17477.8**	**17751.0**
I [14-02-150]	14991.2	15556.0	**17706.4**	**17813.0**
I [07-02-150]	7062.2	7279.0	**8721.0**	**8909.0**
I [10-02-200]	13646.6	14560.0	**16041.0**	**16221.0**
I [10-02-250]	28688.0	29760.0	**33888.0**	**34480.0**
I [10-03-250]	23599.0	24438.0	**26908.4**	**27726.0**

Table 3. Comparison of pure GRASP G1 with G1 with path-relinking

In Table 4, we show the results obtained for G2 algorithm and with path-relinking (G2PR). We can see that path-relinking improves the average values for 11 instances out of 12 and was able to achieve the best value in 8 instances. In Table 5, we compare the average results obtained by the four algorithms. For each instance, we selected the best average result achieved and calculated the percentage deviation of the results obtained by the other algorithms with respect to this best value. We can see that the algorithm G2 with path-relinking provides the best results for all instances. In Table 6, we show for each instance the average computational time achieved. We can observe that path-

Inst.	G2		G2PR	
	Average	Best	Average	Best
I [10-02-050]	2934.2	3075.0	**3131.7**	**3182.0**
I [07-02-050]	4454.7	4683.0	**4820.6**	**4882.0**
I [07-03-050]	5330.0	5450.0	**5495.0**	**5550.0**
I [10-02-050]	7165.0	**7400.0**	**7360.0**	**7400.0**
I [05-02-050]	2149.8	2276.0	**2262.7**	**2284.0**
I [15-03-100]	33277.3	33587.0	**33542.3**	**33780.0**
I [10-03-100]	19460.8	**19884.0**	**19576.8**	**19884.0**
I [14-02-150]	18973.1	19663.0	**19630.5**	**19749.0**
I [07-02-150]	9072.1	9516.0	**9475.10**	**9596.0**
I [10-02-200]	17438.6	17862.0	**17780.6**	**17970.0**
I [10-02-250]	**39376.0**	**39680.0**	**39376.0**	**39680.0**
I [10-03-250]	31525.2	**31765.0**	**31570.0**	**31765.0**

Table 4. Comparison of pure GRASP G2 with G2 with path-relinking

Inst.	G1	G2	G1PR	G2PR
I [10-02-050]	-0.164	-0.063	-0.052	0.00
I [07-02-050]	-0.227	-0.076	-0.092	0.00
I [07-03-050]	-0.168	-0.030	-0.097	0.00
I [10-02-050]	-0.22	-0.026	-0.141	0.00
I [05-02-050]	-0.124	-0.050	-0.028	0.00
I [15-03-100]	-0.124	-0.08	-0.106	0.00
I [10-03-100]	-0.112	-0.06	-0.17	0.00
I [14-02-150]	-0.236	-0.033	-0.098	0.00
I [07-02-150]	-0.255	-0.043	-0.080	0.00
I [10-02-200]	-0.233	-0.019	-0.098	0.00
I [10-02-250]	-0.271	0.00	-0.139	0.00
I [10-03-250]	-0.252	-0.01	-0.148	0.00

Table 5. Percentage deviation of average results

relinking increases very little the computational time, while increasing significantly the quality of the solutions.

Inst.	G1	G2	G1PR	G2PR
I [10-02-050]	2.58	3.22	3.02	3.64
I [07-02-050]	2.61	3.75	2.79	3.95
I [07-03-050]	3.05	4.22	3.17	4.34
I [10-02-050]	3.07	4.26	3.38	4.54
I [05-02-050]	1.41	1.69	1.52	1.78
I [15-03-100]	29.97	38.05	32.12	40.18
I [10-03-100]	10.09	13.77	10.76	14.47
I [14-02-150]	63.21	71.15	65.77	74.05
I [07-02-150]	21.42	30.9	22.04	31.57
I [10-02-200]	86.67	104.61	88.37	106.48
I [10-02-250]	119.84	165.68	121.67	167.60
I [10-03-250]	99.30	152.93	101.36	155.26

Table 6. Average Computational Time (seconds)

5 Conclusions

This paper presented some proposals to solve approximately a particular period vehicle routing problem. We developed a mathematical formulation for this problem, and construction and local search heuristics to implement GRASP algorithms. We also applied path-relinking to the pure GRASP heuristics to improve the quality results.

Experimental results showed that, for small instances of the problem, the GRASP heuristics were able to find the same results obtained by solving the mixed integer linear formulation in much less computational time.

For larger instances, the version which uses Randomized Nearest Insertion construction algorithm showed better results than the Sweep Heuristic. The introduction of path-relinking improved substantially both pure GRASP algorithms, and did not cause a significant increment in the computational time.

References

[1] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, pp. 345–358, 1992.

[2] N. Christofides and J. Beasley, "The period routing problem," *Networks*, vol. 14, pp. 237–256, 1984.

[3] D. J. Aloise, L. Moura, B. Assmann, C. Barros, and J. Neves, "Optimization of employment of oil retrieval system in the exploration of oil wells fields without force to flow to the surface," in *Rio Oil & Gas Expo and Conference*, 2000.

[4] T. Feo and M. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, pp. 109–133, 1995.

[5] M. Resende and C. Ribeiro, "Grasp with path-relinking: Recent advances and applications." T. Ibaraki, K. Nonobe and M. Yagiura, 2005, pp. 29–63.

[6] F. Glover, M. Laguna, and R. Martí, "Fundamentals of scatter search and path relinking," *Control and Cybernetics*, vol. 39, pp. 653–684, 2000.

[7] B. Gillet and L. R. Miller, "A heuristic algorithm for the vehicle-dispatch problem," *Operations Research*, vol. 22, pp. 340–349, 1974.

[8] E. Lawler, J. K. Lenstra, A. R. Kan, and D. B. Shmoys, *The traveling salesman problem: A guided tour of combinatorial optimization*. Wiley Series in Discrete Mathematics and Optimization, 1995.