Hybrid Adaptive Memory Programming using GRASP and Path Relinking for the Scheduling Workover Rigs for Onshore Oil Production

Viviane de Aragão Trindade
Instituto de Computação
Universidade Federal Fluminense
Niterói - RJ - Brasil
vtrindade@ic.uff.br

Luiz Satoru Ochi
Instituto de Computação
Universidade Federal Fluminense
Niterói - RJ - Brasil
satoru@ic.uff.br

Abstract

There are many oil wells in onshore fields that need artificial lift methods in Brazil. These wells periodically need some maintenance services that are performed by workover rigs, avaliable on a limited number. The problem consists in optimizing the workover rigs schedule by minimizing the production loss associated with the wells that are waiting for service. We propose some GRASP algorithms including hybrid and adaptive versions to solve this problem. Experimental computational results illustrate the effectiveness of the hybrid and adaptive versions to be above standard GRASP in terms of solution quality.
Key words: *hybrid Metaheuristics, intelligence heuristic, adaptive algorithm.*

1. Introduction

The metaheuristics or intelligent heuristics have been shown to be one of the most efficient alternative to approximately solve NP-Complete and NP-Hard problems. Among the different metaheuristics found in literature, some have been detached like: Tabu Search (TS) [5, 6], Greedy Randomized Adaptive Search Procedure (GRASP) [1, 3, 4, 8], Variable Neighborhood Search (VNS) [7] and Hybrid Genetic Algorithms (HGA), also known as Evolutionary Algorithms (EAs) [10]. This papper has as its objective, to experimentally develop and analyze different versions of GRASP heuristics to solve a workover rig schedule for an onshore oil production problem (SWRP). This is an existing problem in the northeast area of Brazil, where there are oil wells that rely on artificial lift methods to make the oil surface. Oil can be lifted by different techniques, which require specialized equipment. As time passes maintenance service is necessary because of equipment failure, which is essential to the exploitation of the wells [2]. This service is performed by workover rigs, that are avaliable in a limited number because of the high cost of that equipment. The company in charge of managing these workover rigs receives service orders for some wells. The workover rigs schedule for the onshore oil production problem consists in finding for each period the best schedule for avaliable workover rigs to attend all wells demanding maintanance service, therefore minimizing the oil production loss. The production loss is evaluated as the mean daily yield of the well under regular operation, multiplied by the number of days its production is interrupted [2]. The SWRP is a NP-Hard problem [2], which justifies the development of heuristic algorithms. To our knowledge there is only one proposal heuristic to solve this problem, which uses VNS concepts [2]. However, the instances used in [2] are not available, making it difficult to compare with other proposals. In this paper we propose the development and experimental analyzis of GRASP algorithms to solve the SWRP. Nowadays this metaheuristic is classified as one of the best heuristics techniques to solve hard combinatorial problems. GRASP is an iterative multi-start method, that means that its iterations are totally independent, or that it does not use any kind of memory [3]. Another important aspect shown in literature is that the best metaheuristics versions normally are the hybrid ones where different method concepts are combined in one single algorithm [4, 5, 8, 9, 11]. The main goal of this paper is to present hybrid GRASP versions including path relinking concepts, making them adaptive by using memory between its iterations. GRASP algorithms proposals are presented in Section 2, computational results including an empirical probability distribution of time-to-target value are reported in Section 3. Concluding remarks are drawn in Section 4.

2. Proposal Algorithms

The Greedy Randomized Adaptive Search Procedure - GRASP [3] is an iterative multi-start procedure, which each

iteration consists of two phases: the construction phase and the local search phase. The construction phase builds a feasible solution, where the neighborhood is explored by local search. The best solution for all GRASP iterations is returned as the result. In the construction phase, a feasible solution is built, one element at a time. At each construction iteration, the next element to be added is determined by ordering all elements in a candidate list with respect to a greedy function that measures the benefits of selecting each element. The adaptive component of the heuristic arises from the fact that the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous elements. The probabilistic component of GRASP is characterized by randomly choosing one of the best candidates from the list, but usually not the top candidate. This way of making the choice allows for different solutions to be obtained at each GRASP iteration. The solution found by the construction phase of the GRASP metaheuristic is not guaranteed to be locally optimal. Therefore, the application of a local search algorithm in order to obtain an improvement of this solution is recomended. In this paper we propose different versions of GRASP heuristics to solve the SWRP. The propose algorithms are two constructive algorithms and three local search algorithms. Each combination: construction + local search generates one GRASP version, resulting in six different versions. In a second stage we propose a hybrid GRASP including a path relinking procedure for the best of the six versions from the stage before. This hybrid version uses a memory to keep some information from the previous iterations and uses those in the next ones, that makes the GRASP become adaptive. Our main objective is to evaluate the impact of each proposal in GRASP performance. So we simulated some different possible configurations, analyzing experimental results in different ways. Next, we describe each construction, local search and path relinking proposal algorithm.

2.1. Constructive Algorithm 1 (C1)

This construction algorithm is based on the nearest neighborhood insertion [11]. We define A greedy function as: consider R_k a partial route that is under construction to a workover rig k. The source is always the well where the workover rig is located at the beginning of the planning. Consider p_i, the newest well added to R_k. So, for every well p_j not yet selected, the p_j priority level for the R_k route is given by $PRI(p_j, k) = [v_j]/[tp_i, p_j + te_j]$; where v_j represents the daily production of the well p_j, tp_i, p_j represents the travel time spent to go from p_i (the newest well added to R_k) to p_j, and te_j is the expected time to fix p_j. The wells are arranged in descending priority order. Each workover rig k has a well candidate order list CL_k that is updated after each insertion. Thus, for each list (workover rig) and each insertion of well, we create a restricted candidate list (RCL_k) coposed of the best candidates, where the size of this restricted list is input data [3]. To create different solutions, instead of selecting best candidate from RCL_k all the time, a well from RCL_k will be selected randomly and inserted in R_k. After that, each workover rig list is updated (including R_k) and then we select the next well to a next workover rig. By alternating the workover rigs, we espect to get in the end balanced routes.

2.2. Constructive Algorithm 2 (C2)

In this algorithm, the priority is only the daily oil production of each well, so, to each construction phase iteration, only one Candidate List (CL) is created with all wells that were not added to a route yet. The wells of CL are arranged in descending daily production order. So, in each inserction of a new well, a new RCL is formed and then one well from RCL is picked randomly, just like in C1. But in C2, unlike C1, the picked well is added to the workover rig that can arrive the fastest to its, the objective being to minimize this well's production loss, knowing that this well is still one of the wells that is not included with the biggest daily production wells. This procedure is repeated until all wells that are waiting for service are placed, or until all workover rigs are not able to have another well added to its route in the planning time considered.

2.3. Local Search Algorithm 1 (LS1)

This Local Search (LS) executes two neighborhood procedures, that are repeated one after the other until the solution does not get any better in any of the neightbohood procedures. First of all, a structure with the nearby r neighbors of each well p_i is built ($N(p_i)$), then we pass this structure to the two neighborhood procedures (where r is an input parameter). Given a initial (base) solution, the first neighborhood consists in exchanging each well p_i that is in route R_k with each well from $N(p_i)$ that is in the same route as p_i. After each exchange, the best solution obtained will be the new base solution and we repeat the procedure for all other wells. This procedure is repeated for all routes (workover rigs). The second neighborhood is like the first one, but the exchanges will be done with the wells of $N(p_i)$ that are in a different route than the one that is p_i.

2.4. Local Search Algorithm 2 (LS2)

Considering a well p_k that is in the route R_k (workover rig k), this local search procedure works as follows: the well

p_k has its location tested in every possible position in all other workover rigs that are different from k. The best location (that produces the least production loss) will be considered to be the new base. If there is no improvement, this well remains in the original position. This analyzis is done for each well in every route.

2.5. Local Search Algorithm 3 (LS3)

This local search has also two neighborhood procedures that are excecuted sequentially one after the other, until there is no improvement. The first one (N1) checks for each well p_j that is in route R_k the best position on that route. If a change improves a solution, we update the current solution and analyze the next well in the route. This procedure is done for every route. Then, with the solution returned by $N1$ we begin the second neighborhood procedure, which is exactly the LS2.

Having the construction and the local search algorithms, the initial six GRASP versions are generated, they are formed by the following combinations: **G1** = C1 + LS1, **G2** = C2 + LS1, **G3** = C1 + LS2, **G4** = C2 + LS2, **G5** = C1 + LS3 and **G6** = C2 + LS3.

3. Adaptive Memory GRASP

One limitation of the GRASP, is the fact that it does not have any memory between its iterations. In fact, at each GRASP iteration, a new solution is generated without considering any information from the previous ones. Because of that, it is called a multi-start heuristic. This paper proposes an adaptive GRASP, using information about a set of the best generated solutions found so far, called the elite set (ES). The resulting solution of each GRASP iteration is then compared with the worse solution of ES, updating this set whenever it is possible. This way, the GRASP heuristic begins to use a memory that saves relevant information from previous iterations. The ES is used in the proposal GRASP to make an intensive local search called Path Relinking. The Path Relinking (PR) method was proposed originaly for Tabu Search and Scatter Search methods [5, 6]. The procedure consists in analyzing all intermediate solutions between two good solutions, looking for a third one that is better than the both extreme solutions. Normally a PR is used as follows. First a ES is saved, formed by the best c distinct solutions found so far. After each r iterations GRASP (where c and r are input parameters) a solution from a GRASP iteration is choosen, and it is called the base solution (bs). Each solution from ES is called the target solution (ts). The PR analyzes all intermediate solutions between a bs and a ts in one or both ways. The path relinking procedure starts with one of these solutions (say bs) and gradually transforms it into the other (ts) by swapping in elements that are in the ts solution but do not belong in the bs solution and swapping out elements that are in the bs solutions but not belong in the ts solution. For each intermediate solution is we applied a local search algorithm (LS1 or LS2 or LS3) generating an improved intermediate solution (iis). We compare the iis with the worse solution from current ES updating this set, if it is the case. This procedure is repeated until a new swapping results in the target solution (ts). The PR analyzes then the search in the other way, by just changing the two extreme solutions status (base and target). The PR can be done for all solutions from ES or for one of them randomly choosen. We propose two versions using PR, called G7 and G8. Those consists in G6 plus the PR method (G6 + PR), differentiated only by the number of times that the PR is actived. In G7 the path relinking is activated when the current GRASP iteration is a multiple of 50 or when 50% of ES has been changed since the last execution of PR. In G8, the PR is executed only once at the end of all GRASP executions.

4. Computacional Results

Although the literature presents contribution of the SWRP [2] there is not an available set of test problems. We randomly generated two sets of tests, varying the following parameters: the number of wells; the daily production of each well; the travel time between two wells and the planning time. In this work, to simplificate, we considered the number of workover rigs fixed in 3 ($k = 3$). The number of wells is equal to: 50, 100, 200, 300, 400, 500, 700 and 1000. We generated two sets of instance (A and B) having for each, all the cited instances above. In the fist set (A), the mean value of travel time between two wells is considered bigger than the mean value of wells production. And in the second one (B) the oposite occours. All GRASP versions, described in this work, were implemented in C, compiled with gcc and tested in an Athlon XP 2.4 GHz with 512 Mbytes of RAM. For each instance, each GRASP algorithm was initially executed three times using the following set of parameters:
$a = 0.1$ (RCL size in construction fase).
The GRASP stop criteria: Maximun Number of Iterations : 200.
$r \leq 20$ (referring to the nearby r neighbors from LS1).
At each GRASP execution we used the seeds: 3, 7 and 11.
Size of Elite Set: ES = 3 (to the versions G7 and G8).

4.1. Set of Tests A: Travel Time $>$ Production

In the SWRP problem there are two factors that influence the choice of the next well to be added in a partial route: the

total travel time of the current route until the next candidate well and the candidate well's dialy production. The table 1 presents the computational results of G1 to G6 versions, to the instances of A. As the metaheuristics, including GRASP, can generate diferent solutions at each execution, even using the same input parameters, each instance was executed three times for each algorithm. In every table that has the algorithm performance analyzed based on solution quality (mean solution quality), the showed values represent the percentile deviation-mean between the *best(t)* and the mean solution from the associated algorithm (presented by the equation: *(mean solution - best)divided by the best*; where *best(t)* = *best solution* of the instance t considering all algorithms.

Inst.	G1	G2	G3	G4	G5	G6
50	14,33	17,05	1,95	1,27	0,48	**0**
100	27,20	42,35	3,55	1,05	0,16	**0**
200	48,63	64,90	6,08	1,59	2,73	**0**
300	65,90	81,67	4,74	1,53	2,56	**0**
400	86,63	94,75	6,38	2,40	3,29	**0**
500	99,80	110,72	6,01	2,04	1,65	**0**
700	116,06	123,73	4,32	1,40	1,95	**0**
1000	141,59	149,62	5,67	1,39	3,16	**0**

Table 1. Average solution from each heuristic

From the results showed in table 1 we can verify that the G6 version (using the constructive algorithm (C2) and the local search (LS3)) has presented the best mean solution quality results followed by the versions G4, G5 and G3 that had similar results. The G1 and G2 versions presented a very poor performance compared to the others, indicating that the LS1 is less efficient then the others local search algorithms. Concerning the computational times, besides this datas are not ilustrated in this work, it was observed that the average results are similar, and there isn't any discrepancy between them. In general terms, the pair (G1 and G2) was faster, followed by the pairs (G3 and G4) and (G5 and G6).

4.2. Set of Tests B: Production $>$ Travel Time

The second set of test average results can be observed in the table 2.

The table 2 results show, again, a superiority of the G6 version based on the solutions quality. A ordered ranking based on mean performance considering all sets of test (*A* and *B*) and the six version is showed in the table 3.

Inst.	G1	G2	G3	G4	G5	G6
50	11,26	16,18	1,40	4,41	**0**	0,78
100	30,94	27,39	4,37	3,88	**0**	1,12
200	62,99	62,13	4,49	4,02	0,70	**0**
300	74,66	83,58	2,98	2,78	**0**	0,80
400	108,51	105,34	5,73	3,25	2,26	**0**
500	124,89	113,39	4,36	0,72	2,28	**0**
700	161,79	164,42	6,52	4,10	3,39	**0**
1000	208,19	221,56	4,44	2,56	3,20	**0**

Table 2. Average solution from each heuristic

Ranking	GRASP	Deviation-Mean
1	G6	0,17%
2	G5	1,74%
3	G4	2,40%
4	G3	4,56%
5	G1	86,46%
6	G2	95,42%

Table 3. Mean ranking from GRASP algorithms based on the both sets of test (A and B).

4.3. Compared results from GRASP and Path Relinking

In this section we include a Path Relinking (PR) procedure in the best GRASP version generated until now (G6), proposing the version G7 = G6 + PR and G8 = G6 + PR2, where the differences have been explained before. Because of the results similaritie obtained and the increase of the new versions execution times using the current stop criteria, we ilustrated only the results that have 500 wells or less. The impact of the PR inclusion in GRASP can be analyzed in tables 4 and 5.

Inst.	Deviation-Mean			Time-Mean		
	G6	G7	G8	G6	G7	G8
50	0,74	**0**	0,52	0	0	0
100	0,16	**0**	0,13	0	78	17
200	0,23	**0**	0,004	163	2.804	468
300	0,82	**0**	0,47	819	11.244	2.707
400	0,49	**0**	0,34	2.089	37.403	8.619
500	0,12	**0**	0,12	4.342	91.468	20.757

Table 4. Set of Tests A: Average solution value from G6, G7 and G8 based in deviation-mean related to the *best* (in %) and the mean execution time (in seconds).

Inst.	Deviation-Mean			Time-Mean		
	G6	G7	G8	G6	G7	G8
50	1,00	**0**	0,89	0	0	0
100	**0**	**0**	**0**	0	77	13
200	1,86	**0**	0,23	133	2.155	465
300	0,14	**0**	0,02	733	10.284	2.602
400	0,31	0,07	**0**	2.080	37.599	7.889
500	0,06	**0**	0,04	4.278	91.826	20.381

Table 5. Set of Tests B: Average solution from G6, G7 adn G8 based in deviation-mean related to the *best* (in %) and the mean execution time (in seconds).

The average results presented in tables 4 and 5 show that the PR inclusion brings significant improvements in final GRASP solution quality, indicating a clear superiority of the versions with Path Relinking. This proves, once again, that hybrid versions of GRASP metaheuristics can have better performance than traditional GRASP versions, as G6. In relation to the G7 and G8 computational times, as we expected, there is an increse in both of them. However, the average time analysis must be made carefully, because of the current stop criteria (maximum iteration number) can harms some versions where there are more work for each iteration, but demands a least number of iterations to get a target value, as we believe that is the case of G7 and G8. This analysis are presented in next section of this paper.

4.4. Empirical probability distributions of time-to-target value of GRASP algorithms

In this section, the main objective was to verify the empirical distribution of the random variable *time to target solution value* (i.e. find a solution as well as the target or better in fuction of time) in different instances. To evaluate the performance of the proposed algorithms, we fixed a solution target value and ran each algorithm 100 times, recording the running time when a solution with the quality as good as the target value or better is found, or a time limit was hit (that was calculated near three times the mayor execution time considering all GRASP versions for the instance that the analysis involve). This target value was choosen as sub-optimum value or a value near the optimum soltuion (in our case near the *best*) for each instance. To plot the empirical distribution for each instance, we associated the execution time t_i of the $i-th$ sorted running time to a probability $pi = (i - 0.5)/100$ and the points $z_i = (t_i; p_i)$ were plotted for $i = 1; ...; 100$ [1]. This analysis was made for every instance here analyzed, but because the behavior was very similar between them, we only show the graphs of the $100A$ and $100B$ instance. In the figure 1, each graphic line represents the computational time demanded to each algorithm hit the target value. A simple analysis of the results can be done considering the alignment of the curves: left most aligned curves indicate faster convergence of the algorithm while right most aligned curves indicate an algorithm with slower convergence. In this figure, it can be observed that the most sofisticated versions (G6 and G7) demand a similar computational time, or even minor than the other versions. In the first graphic from figure 1 the versions G1 and G2 weren't presented because those versions couldn't converge in most of the cases of the 100 executions with the time limit equal to 86 seconds. In the second graphic the version G8 was not considered because this version only uses the PR procedure in the final of G6. The second graphic shows that the G7 version always hits the target value, in the other hand, G6 only hits it in about 80% of the executions.

5. Conclusions

This work presented some proposals to improve the GRASP performance applied to a solution of a real problem that generates optimized routes to a scheduling workover rigs for onshore oil production problem in the northeast area of Brazil. The main goal was, inicialy, to analyze the influence in the final GRASP performance of the following algorithms: construction phase, local search phase and path relinking method. It shows that GRASP strongly depends especially on the local search phase and that a good construction and local search combination (G4, G5 and G6) can produces very competitive aproximated solutions. In a second phase, the paper showed the performance of hybrid versions integrating path relinking concepts inside a GRASP structure. The empirical results obtained show that this union can improves significantly the final GRASP performance. Beyond the Path Relinking module is considered a way to effects intensive searches between two extreme solutions good quality, a good performance of the G7 and G8 adaptive versions can in part be justified by the use of a memory to keep relevant informations obtained in previously iterations through elite set (ES). Finally, other contribution of this paper is the empirical probability distributions where the main goals are to verify the robustness of these heuristics and also to show an alternative stop criteria to the GRASP.

References

[1] R. M. Aiex, M. G. Resende, and C. C. Ribeiro. Probability distribution of solution time in grasp: an experimental investigation. *Journal of Heuristics*, 8(3):343–373, 2002.

[2] D. Aloise, C. T. M. Rocha, J. C. R. Filho, L. S. S. Moura, and C. C. Ribeiro. Scheduling workover rigs for onshore oil production. *To appear in Discrete Applied Mathematics*, 2005.

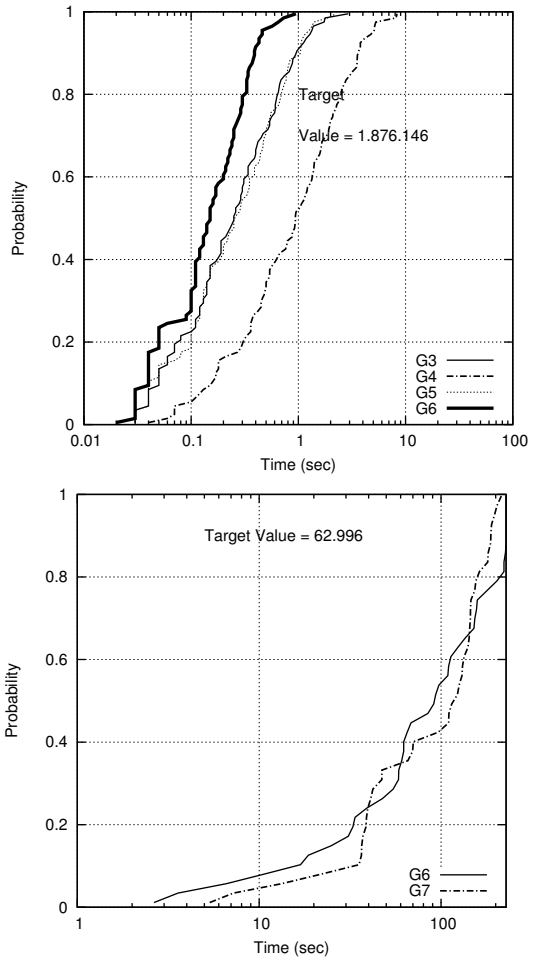

Figure 1. Empirical convergence of GRASP heuristics for the instance 100B and 100A, respectively.

[3] T. Feo and M. G. Resende. Greedy randomized adaptive search procedure. *Journal of Global Optimization*, 6:109–133, 1995.

[4] P. Festa and M. G. Resende. Grasp: An annotated bibliography. *In C.C. Ribeiro and P. Hansen, editors. Essays and surveys in metaheuristics, Kluwer*, pages 325–367, 2002.

[5] F. Glover and M. Laguna. Tabu search. *Kluwer Academic Publishers*, 1998.

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

[7] P. Hansen and N. Mladenovic. Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100, 1997.

[8] G. C. Silva, S. L. Martins, and L. S. Ochi. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. *In Lecture Notes on Computer Science (LNCS)*, 3059:498–512, 2004.

[9] M. J. F. Souza, N. Maculan, and L. S. Ochi. A grasp-tabu search algorithm for solving school timetabling problems. *In Combinatorial Optimization Book Series, Metaheuristics: Computer Decision-Making, Kluwer*, 15(31):659–672, 2003.

[10] E. D. Taillard, L. M. Gambardella, M. Gendreau, and J. Y. Potvan. Adaptive memory programming : A unified view of metaheuristics. *European Journal of Operational Research*, 135:1–16, 2001.

[11] E. G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8:541–564, 2002.