

Journal of Heuristics  
Manuscript Draft

Manuscript Number: HEUR314R1

Title: GRASP STRATEGIES FOR SCHEDULING ACTIVITIES AT OIL WELLS WITH RESOURCE DISPLACEMENT

Article Type: Regular

Section/Category:

Keywords: GRASP, constraint programming, petroleum, offshore, scheduling

Manuscript Region of Origin:

# On the Comments from the Second Reviewer

We comment on the suggestions put forward by the Second Reviewer. As can be seen below, we have followed most of the specific suggestions offered by the Reviewer. In doing so, we believe we have much improved the presentation.

We separate our comments in two parts: one part discusses the Reviewer detailed suggestions and another part replies to his more general, or philosophical, comments.

## 1 On the Reviewer's "minor comments"

We start with his "minor comments".

**page 2:** "*The first sentence on page 2 can safely be removed.*"

We agree. It has been removed.

**page 2:** "*On page 2, "developed for maximizing" instead of "developed maximizing".*"

Corrected.

**page 2:** "*On page 2, I would de-emphasize the advantage of home-grown GRASP code over ILOG Solver/Scheduler; if you're saving tens of millions of dollars, nobody will blink over the 30K ILOG expenditure. Plus, I would argue that relying on non-commercial libraries isn't always a good thing, e.g., in terms of reliability and speed.*"

The ILOG license price was not really the main point in that paragraph. The last phrase was eliminated. But we do have to mention the ILOG Solver code developed by the company, since the cited gain of our GRASP implementation is measured against that particular code.

**page 2:** "*How does the '830M' USD figure on page 2 correlate with the results shown in Table 5? This wasn't clear at all.*"

Table 5, Criterion 2, shows that G14 generated 346.814.967 barrels of oil, while ORCA generated 331.043.827. The difference is 15.771.140 barrels of oil, which we rounded to 16 million. Assuming that a barrel goes for 52 USD (in fact, much more nowadays), we reach the cited gain of 832M USD.

In order to make this point clearer, near the '830M' USD claim, on page 2 of the original text, we inserted an explicit reference to subsection 3.4, where the reader can find the above reasoning.

**page 4:** "*On page 4, 'depending on' instead of 'depending to'.*"

Corrected.

**page 6:** "*You discussion on page 6 seems to imply that GRASP is generally good at scheduling problems. This is definitely not the case; I can't think of any GRASP implementations that are competitive with, for example, tabu search, on any of the standard academic scheduling problems (flow/job/open shop). As a scheduling researcher, I found the selection of GRASP rather strange.*"

At the beginning of Section 2 (on the original manuscript) there is an introductory discussion where tabu search is also mentioned. We did not mean to propose GRASP as a *generally* good scheduling algorithm, or that GRASP is *in general* better than tabu search. In fact, depending on the particular sub-strategies embodied in GRASP or tabu – such as long term memory, path-relinking, and many others – any one of the two heuristics can be the winner over a great number of search problems.

Our mention of tabu search at that point in the text was, for us, somewhat natural. We had tried some versions of a tabu search heuristics on a simpler version of the WDRDP. A first version was a more direct and simpler approach. In a second version, we implemented tabu search together with a very large neighborhood (VLN) technique, still on the simpler version of the problem. This VLN implementation was interesting, since the search through the large neighborhood was done using constraint programming techniques in order to efficiently find adequate neighbours where to move next. This implementation produced good results (see references in the original text) on the simpler problem. From the tabu search implementation it was just natural to try a GRASP-like approach heuristics on the same problem. But since the GRASP approach is based on a great number of starts at different points in the search space, it is advisable to use neighborhoods that can be visited very efficiently. So, in the GRASP approach we changed our concept of neighborhood, and the final implementation produced very good results, still on the simpler version of the WDRDP (see references in the text). Given that, we decided to start with a GRASP strategy on the full and more complex WDRDP that is described in the present manuscript.

The results, as reported, were very good when compared with the original, in house, development of ORCA. Note that we are not comparing the GRASP approach described in the text with any tabu search that is nearly as sophisticated. The ORCA implementation addresses the same problem, but uses constraint programming techniques. And our earlier implementations of tabu search were focused on a different, simpler, version of the WDRDP. So, a comparison between these tabu implementations and our GRASP algorithm for the full WDRDP would not be in order. We simply, at this point, did not try a careful implementation of a tabu search heuristics to solve the full WDRDP. And we certainly do not discard that one such implementation could produce equally good, or better, results.

Also, note that the WDRDP is not exactly a standard academic scheduling problem. It has many specific constraints that stem from the real operation of an oil industry in the field. And GRASP approaches do have performed well in a variety of scheduling problems (see the references in the text) related to industrial operations.

In the light of the above arguments, we feel that our choice to start with a GRASP implementation for the WDRDP was a natural one, that could not be ruled out before tested on real instances.

In any case, it was important to us to hear from the Reviewer that the original text led to confusion, and even conflicting, impressions regarding these issues. We have rewritten the introductory part of Section 2 (of the original manuscript), and now feel that the new text clearly avoids these problems.

**page 6:** “On a related note, you indicate on page 6 that your tabu search approach to solving the WDRDP failed due to the lack of an adequate neighborhood definition. But you obviously found a tolerable one for local search in GRASP. This seems to be a contradiction.”

It could lead to that impression.

But, as commented on the previous answer, we were dealing with different problems. Moreover, when implementing a tabu search heuristics on a simplified version of the WDRDP, our central idea was to study its behavior when dealing with very large neighborhoods. These large neighborhoods are inappropriate for a GRASP implementation, given the very nature of a GRASP approach. Of course, it *is possible* to test a tabu search approach using the neighborhood we defined for the GRASP implementation.

But this is not the focus of the present article, whose purpose is to describe the basic GRASP heuristics implemented and, moreover, to report on the extensions that made that implementation clearly outperform the already competitive ORCA algorithm, developed by the company over four years, using constraint programming techniques.

As already mentioned, we rewrote the introductory part of Section 2 (of the original manuscript) in order to make these points clear.

**page 6:** “*GRASP is in no way ‘adaptive’, at least in any sense of the word as used within the meta-heuristics community.*”

See, for example, Binato et al. (2002). We have seen the word ‘adaptive’ used in association with GRASP in several documents in the open literature.

The ‘adaptive’ adjective stems from the fact that, during the construction phase, the method ‘adapts itself’ at each iteration. That is, after each new piece of the solution being constructed is chosen, the cost of bringing in the next piece is recalculated according to a greedy function. This procedure ‘adapts’ the cost of the new candidates to enter the solution in the next iteration, so as to reflect the insertion of the last element in the partial solution at hand.

We did not consider it necessary to alter the original text to further clarify this point. First, the text of Section 2.1 of the original manuscript was just a very short overview of the basics of the GRASP method. And, further, by the end of page 6, there was already in place a comment on the ‘adaptive’ nature of the method.

**On page 8:** “*The phrase ‘ironed out’ is a bit too colloquial.*”

Changed to ‘removed’.

**On page 8:** “*maintaining a pool of elite solutions is only \*one\* form of long-term memory, not the only implementation of the general idea (e.g., frequency-based memory is a long-term memory).*”

We did not intend to imply that. A couple of phrases were rewritten in the text in order to remove such an impression on the reader.

**On page 8:** “*Ant Colony Optimization is another obvious way to induce dependence among the various constructive probes of GRASP.*”

Of course. A phrase was added to the text, stating that the strategies we proposed are not the only possibilities.

**Toward the bottom of page 9:** “‘*C3 and C4’ instead of ‘C3 e C4’. Also, ‘of which one’ instead of ‘of whose one’.”*

Corrected.

**page 6:** “I think you can eliminate the pseudo-code in Tables 1 and 2; citation of a standard GRASP reference would suffice. However, I do believe detailed pseudo-code would be greatly beneficial when introducing the algorithms documented in Sections 2.4/2.5. This would improve significantly over the prose description, and facilitate reproducibility; as it stands, I don’t believe the results - assuming you had the problem instances - would be easily replicatable.”

Indeed. We agree that pseudo-code for the algorithms documented in Sections 2.4/2.5 would be useful, and we have incorporated several of them: Figures 3–8. Also, throughout the text, we have inserted comments referring to those figures, when appropriate.

But we kept Tables 1 and 2, now more appropriately cited as figures, since we use them for comparison purposes when explaining the new construction phase of our GRASP implementation. This material was originally in Section 2.4, “The Construction Phase”.

**page 11:** “It sounds like real-world instances of the WDRDP, or at least the ones available to you, are not all that difficult; greedy construction can find the best-known. Further, ORCA and GRASP perform the same on these instances. So what is it about the other instances that makes GRASP outperform ORCA?”

On only six out of seventeen real instances the greedy construction could find the best-known solution. On the remaining eleven instances, the greedy construction did not find the best-known solution at all. In fact, in those cases, the greedy construction found only very poor solutions. We may say that we had six simple instances, and eleven more complex ones. In any case, we inserted an observation at that point in page 11 in order to make this point clearer.

It is not quite true that “ORCA and GRASP perform the same on these instances”. On page 16 (Section 3.4), we read that in only *one instance* ORCA was competitive with our GRASP implementation. And, even in this particular instance, GRASP outperforms ORCA after 1134 seconds. Since this comment about ORCA’s performance was not in the text, we hope that the explanation above is sufficient to clarify the Reviewer’s comment.

**no page:** “There is no reference to ORCA anywhere in the paper. If this is a standard tool in some subset of the petroleum industry, then there should be some citable source.”

ORCA is an in-house development by Petrobras. Its results and algorithms were not discussed outside the company. It is not clear that Petrobras has some interest in making public data about ORCA’s performance.

We added a reference item where the in-house development of ORCA is made explicit, although we could not cite any documents about it that are available in the open literature. The reference appears at the first spot where ORCA is mentioned, in (the original) Section 1.3.

**page 16:** “The phrase ‘Not only it generates solutions with higher oil production, but often it outputs solutions with the same oil production’ is self-contradictory.”

What we meant here is that the GRASP approach not only generates solutions with higher oil production than the corresponding ORCA solution but, moreover, the GRASP solutions are obtained in much less time. We replaced the offending phrase in the revised text.

**page 16:** “*Relating again to the ‘advantage’ of GRASP over ILOG, you may have avoided the cost of commercial libraries, but the labor costs involved in the GRASP implementation could easily outstrip the 30K cost of commercial licenses.*”

We agree, but one also has to take into account the costs of creating a constraint satisfaction model for the problem, which often is a time consuming task, with its own costs that will add up. In any case, cost was not the issue we intended to focus on here. Rather, we wanted to bring up the contrast between using a set of professional libraries against programming from the ground up. That phrase has been rewritten.

**references:** “*You have numbers for some of the ATT technical reports, but not all. I haven’t seen ‘IIE Transactions’ printed before as ‘I.I.E. Transactions’. Three references to ILOG’s Solver manuals is unnecessary.*”

The reference to the “Transactions” was corrected, and the new text cites only the ILOG Reference Manual. The ATT technical reports had their numbers inserted. One of the ATT technical reports was removed and a journal reference to the same article was used.

**table 5:** “*Please use American (as opposed to European standard) currency punctuation, e.g., ‘131,200.5’ instead of ‘131.200,5’.*”

Done.

**figures 1 and 2:** “*Figure 1 didn’t add anything to the discussion, and the utility of Figure 2 is marginal.*”

We still consider that those figures may help the less familiar reader to more easily understand the issues involved. Since their utility was marginal but not negative to the Reviewer, we decided to maintain them in the text.

**figures:** “*All of the figures were very blurred, basically rendering them unreadable. Consequently, any interpretation of results that refers to them is very hard to decipher.*”

We have remade them all. The graphics is now of a much better quality.

## 2 On the Reviewer’s “more general, or philosophical observations”

In this part, we comment on his more general observations about the original text.

— “*I have a general concern that the significance of the paper hinges completely on the comparison with the ORCA algorithm, which in turn was extended by the authors to handle resource displacement constraints. If the ORCA algorithm were extended in a non-intelligent fashion - and I’m not claiming this is true - then of course it isn’t that interesting that a GRASP-based algorithm can outperform it. Assessment of this nuance would require a CP expert, which I am not. Given ILOG’s broad experience with CP for industrial problems and similar experiences echoed by academics, I find it difficult to believe that CP techniques and/or ”trick of the trade” haven’t been developed to handle such a benign-looking constraint such as resource displacement. At the very least, the authors need to devote more discussion to this point (it currently stands at only a paragraph), given its centrality. Right now, the paper reads like*

*"we spent a few weeks working on a CP extension of ORCA to handle resource displacement, but it didn't work too well. Then we spent a year working on a GRASP implementation, and it outperformed our ORCA extensions". Not surprising, right? I am also concerned that the comparison is based on an ancient version of ILOG solver, circa 1999. There have been significant performance advances in their tool suite in the last five years (if you were doing a comparison in 2005 between CPLEX 6.0 instead of 10.0, that would be an issue in the mathematical programming community). I'm also concerned that you didn't reference ILOG Scheduler, but rather only Solver. Is ORCA only based on the latter? If so, this raises a host of other issues. "*

First of all, we would like to state clearly how ORCA was used in this work. We reckon that we have not been successful in the original version of the paper to explain to the reader our relation with this code.

As we say now in the paper, during our research, Petrobras allowed us to use ORCA's executable files for testing. However, we had no permission to access the source files. Therefore, we are unaware of ORCA's implementation details other than those we gathered from the developers and, of course, that were cleared to be made public. These information are spread throughout the text.

In light of that, our paper should be viewed as a successful application of GRASP to a relevant real-world problem. The computational comparison we made between GRASP heuristics and the constraint programming tool ORCA is, by no means, intended to show that one technique is superior to the other. That would not be fair since there is certainly some room for improvement in ORCA that, as explained above, we could not exploit. Nevertheless, since the technicians from Petrobras are very confident on the quality of ORCA's solutions, it is legitimate to use them as a measure to assess the performance of our heuristics.

Concerning the version of the ILOG suite, from the remarks above it is clear that there was nothing we can do about that. Petrobras uses the 1999 version and, as of today, did not acquire of a newer one. Finally, we confirmed with the developers of ORCA that they also used the ILOG Scheduler library and this is now mentioned in the text.

- “*The empirical evaluation is problematic. First, considering just the GRASP variants, there is ample possibility for spurious correlations and the associated false identification of performance differences among variants. Specifically, the authors use a very small number of trials per instance (five), don't do any statistical comparisons to determine whether the observed differences are statistically significant, and don't control for differences across individual instances (all comparisons are done at the level of all problem instances). A proper experimental analysis would use paired rank-based tests of the Wilcoxon variety, for example, with controls at the overall experimental level to avoid identification of spurious statistically significant correlations. Such analysis is especially critical given small differences in both relative and absolute performance, as shown in Table 5, for all of the GRASP variants. I'm not doubting that some variants perform better, rather that the authors have not conclusively demonstrated this fact or identified the superior variants correctly. ”*

We agree with the referee that several statistical tests could possibly be done to evaluate the GRASP variants. However, more consistent testing was virtually impossible for us to carry out. With 14 variants, 17 instances, 5 executions per instance and a CPU limit of 1800 seconds, we already reached almost 600 hours of CPU running time.

It is important to notice that our experiments were conducted at Petrobras. This is because we wanted to have our heuristics run on the same machine as the ORCA tool did. Of course, the use of hundreds of hours of CPU time had to be negotiated with the company. It does not seem plausible that we would be able to obtain substantial additional machine time for further experimenting with the algorithms.

Besides this practical aspect, we also have to have in mind that we are dealing with a real-world application in which a tenth of a hundredth of percentage gain represents millions of dollars. Further statistical testing might bring more light on whether an algorithm is better than another or not. If we do so, it is possible, due to the small percentage differences between the oil productions attained by the many GRASP variants, that we would end up concluding that, “basically, all the variants perform equally well”. However, this will not change the fact that, for the engineers, the best algorithm is the one that reaches the higher production in an acceptable amount of computation time. That was precisely what we tried to capture with the statistics we computed.

For these reasons we decided not to pursue with a more detailed statistical analysis of the results.

- “*On a philosophical note, there is a general “fuzziness” regarding where and why the authors have decided to use the start-time representation of a schedule instead of a disjunctive graph representation, and vice versa. In the scheduling literature, there is fairly broad evidence that disjunctive graph schedulers typically outperform, often by large margins, start-time-based approaches. This is one of the major tenants of modern CP-based scheduling, and is embodied in ILOG scheduler. Why did the authors decide on a start-time-based approach for the constructive phase, and then switch over to a disjunctive graph representation in the local search phase ? Some discussion on this point should be introduced. Focus on the start-time-based representation also compounds my concerns on the lack of an ILOG Scheduler-based (and therefore \*true\* CP-based scheduling) comparative baseline.”*

During our work, we had no strong reason to support the usage of a start-time-based or of a disjunctive graph approach. Our main focus was on the the project goals: produce solutions of high quality in an acceptable amount of time. It turned out that the start-time-based approach performed very well in the constructive phase by generating good solutions quite rapidly. This simply drove us away from considering the usage of a different approach.

Now, concerning the local search phase, as we explain in the text, we already had faced difficulties with the local search on a simplified version of the problem. Also, according to the literature, the disjunctive graph approach had been successfully applied to job shop scheduling problems, which are closely related to ours. This motivated us to implement the same approach in our heuristics.

Therefore, one can say that our decisions were based on experimentation. It is also worth noting that our objective function, which considers the production over a given horizon, strongly depends on the start-time of the activities. Not all scheduling problems are defined with objectives so intimately related with starting times. This indicated that start-time-based approaches should not be discard *a priori*.

As for the ILOG tools, again we emphasize that we had no control on the implementation choices made for the constraint programming solver developed by Petrobras.

### **3 Conclusions**

Finally, we express our appreciation for all the referee's detailed comments and more philosophical remarks. They were all very helpful when revising the text. We hope that the new version of the text does meet his demands for improvements.

# On the Comments from the First Reviewer

We comment on the suggestions put forward by the First Reviewer. As can be seen below, we have followed most of the specific suggestions offered by the Reviewer. In doing so, we believe we have much improved the presentation.

We separate our comments in two parts: one part discusses the Reviewer detailed suggestions and another part replies to his more general, or philosophical, comments.

## 1 On the Reviewer's "more detailed comments"

We start with his "more detailed comments".

**p4, constraint C4:** "*does the model only contains disjunctive resources and no cumulative ones? May an activity require several resources at the same time?*"

The model only allows for disjunctive resources. We have added a sentence to make this point clear.

**p4, constraint C7:** "*this constraint boils down to some simple precedence constraints, right?*"

Exactly. We thought that it was not necessary to be more explicit in the text.

**p5, section 1.4:** "*I personally find it very disturbing to have all the figures and tables in the end of the paper as one always have to navigate between them and the text.*"

This particular way of placing figures and tables is part of the instruction that JH requires authors to follow at this phase of manuscript preparation. We believe that when the final formating is done and the manuscript is ready for printing, figures and tables will be placed in more appropriate places along the text.

**p8, in the description of the PO:** "*it is not clear how LS is applied at several points in the constructive phase. There seems to be many redundant moves when LS is successively applied on a schedule that is growing. How is this handled?*"

In our coding, the routine that implemented the POP idea was a fast one, compared to the other parts of the code. So, it was not necessary to worry about redundant moves here. Nonetheless, we felt that the text at the end of the paragraph that explained the POP was a bit confusing and we rewrote it. We also added a few lines about the nature of the LS at this stage.

**p8, in the description of Intensification:** "*it is not clear if and how solutions are removed from the elite set. Starting from the initial solution and following the description, I do not see how the elite set can be something else than all the solution explored so far with the best cost so far. Is it possible to have an idea of the size of the elite solutions set? A naive implementation of the computation of  $Int(\alpha)$  has a linear complexity with respect to the number of elite solutions. Is it an issue or is this negligible? Furthermore, the value of  $Int(\alpha)$  should increase roughly linearly with the number of elite solutions, thus this number seems to play an important role for balancing the bias function, basically, if all elite solutions were duplicated in the elite solution set, it has a big impact on the bias whereas, I do not think it should change anything.*"

After we run some tests, the size of the elite set was fixed at 5 elements. This number produced the best results, keeping the elite set small, thereby rendering  $Int(\alpha)$  a very fast procedure. At the start, the first 5 solutions populate the elite set. From this point on, in order to include a solution in the elite set, we followed the procedure described in the **Intensification** subsection, on page 8 of the original manuscript. We agree with the referee, though, that it is interesting to include the new information about the set size and its initial formation, and we did so.

We also agree with the referee that the text was not clear about removing elements from the elite set. To make the point clearer, we do remove one element each time we insert a new one. We tried two strategies for removing elements from the elite set. In the first trial, the element removed was the one with the smallest oil production, among all elements in the elite set. In the second trial, we removed the element with fewer differences in its scheduling, when compared with the new element. Again, tests run on real instances proved that the first approach gave superior results. We inserted a few lines in the original text in order to make these points clear.

As for the computation of  $Int(\alpha)$ , as already mentioned, it was very fast and posed no issue. Remember that the size of the elite set was limited to 5 elements. The value of  $Int(\alpha)$  does play an important role in balancing the bias function, as the referee noted. But elite solutions are never duplicated in the elite set. This is precluded by the protocol for including solutions in the elite set, which forces any solution that enters the set to be different from all other solutions that are in the set at the moment of inclusion.

**p11, item 3.** “*The RCL is built with those wells that, together with their successors offer the highest yields of oil. How is this estimated? By using the earliest starting time for the activities and their successors or are resource conflicts also considered in this estimation?*”

In order to choose the best wells to enter the RCL, we order them by the value of their oil yields. Initially, only the well’s own yield is considered. In order to estimate the production of a well, we simulate the scheduling of this well, i.e., we insert this well in the solution and obtain its production, returning the solution to its original state afterwards. If there are wells which are constrained to succeed the current well, we also estimate the production of these successors. As this process advances, we keep the maximum among all these yields, i.e., the well’s own yield and the yields of all its constrained successors. Actually, we tried many other possibilities like, for example, summing up the yield of the well and the yields of all its successors. The best schema we found, according to our tests, was the one that retained the maximum yield.

We also agree that the text of item 3 does not make these issues clear. The text of item 3 was expanded in order to clarify how the RCLs were constructed.

**p11.** “*As mentioned before, it would be interesting to see which heuristic was used in ORCA. It seems that it was a different (and worse) heuristics.*”

Yes, ORCA uses a different heuristics. But ORCA is an internal development by Petrobras, and we did not have access to its code. We treated ORCA as a “pre-existent black box scheduling tool” that was already fully developed by an internal team at Petrobras, to their satisfaction. Even if we did have access to its code, however, we feel that such a full description of the behavior of ORCA would take us too much aside from our central goal of presenting the

strategies we used in our GRASP approach to the same problem. We used ORCA mainly as a yardstick against which we could compare the efficiency and solution quality of our GRASP implementation.

As a quick summary, we can mention that ORCA was developed using the ILOG Solver and ILOG Scheduler libraries. After the constraints are imposed, the mechanism of constraint propagation is handled directly by the ILOG libraries. The search procedure is based on backtracking, coupled with a branch-and-bound strategy. At each step in the search tree, the Solver chooses the best open node with respect to a given evaluation function and moves to that node in order to expand it. The evaluation function used in ORCA is the Limited Discrepancy Search (LDS), which was first discussed in (Harvey & Ginsberg 1995).

We did not include a full description of ORCA's algorithms in our text. But we added a couple of lines, after item 6, page 12, of the original manuscript, in order to give a high level synthesis of the flavor of its internal workings.

**p16, end of section 3.3.** “*First, I want to say that I really appreciate this section (although it is sometimes a bit long) whose objective is to identify which advanced techniques contribute to the success of the approach. The experimental study makes it clear that POP is the best advanced technique on top of GRASP to solve the WDRDP. Given this, it could have been interesting to individually test the other advanced techniques on top of GRASP+POP, that is GRASP+POP+Bias (this was done: G10), GRASP+POP+Int, GRASP+POP+P-R, GRASP+POP+SLS, GRASP+POP+BIR.”*

We are glad that section 3.3 contributed in a decisive way to make the text valuable and clear. We thank the referee for his comments.

We note that GRASP+POP+SLS was done (this was G8), and that GRASP+POP+BIR was also done (this was G9). It is true that we could have tried a number of other possibilities. In fact, there are many possible combinations. On the other hand, we had to run each test several times, in order to get an average behavior for each combination tried, since the GRASP routine is stochastic by nature. And obtaining the complete data for each possibility took a number of hours of CPU time. So, trying to exhaust all possibilities would be prohibitively expensive, in terms of CPU time. We feel that the combinations tried already make the case for the superiority of the GRASP approach, against the original ORCA approach. In any case, in the concluding section, page 16 of the original text, we added a few remarks noting that this line of research could be pursued as a possible extension of this work.

— “*It would be very useful if some "synthetic" instances that captures most of the complexity of this very challenging scheduling problem were created and made available to the scheduling community.”*

We agree. In order to make this happen we would have to convince Petrobras that some such instances could be made public, even if they have to be disguised in order to protect the real data from the competition, but without disturbing the instances to a point where they would lose their value. Maybe the older instances are not so critical to them. In any case, if they agree to it, we will make some instances available in public places such as the OR Lib.

— “*As a final comment, I think the work described in this article is very relevant and the proposed approach seems to obtain very good results on an important industrial scheduling problem. I*

*recommend its publication in the Journal of Heuristics provided the positioning with respect to the existing system ORCA is improved in order to clearly separate (1) the benefits/drawbacks of using a CP framework from (2) the implementation choices on how to use such a framework that were made in ORCA (constraint model, type of search, heuristics)."*

We thank the referee again for his kind comments on the manuscript as a whole. We hope that the new version of the text does meet his demands for improvements, so as to warrant publication of the new version of the manuscript.

## 2 On the Reviewer's "more general, or philosophical observations"

In this part, we comment on his more general observations about the original text.

— “*Still, I find that the article is not really fair with the usage of CP in general (and ILOG CP product line in particular) with respect to solving this type of scheduling problem. ORCA is implemented on top of the ILOG CP optimization suite which is a library for solving feasibility and optimization problems using CP.*”

We agree. We now state very clearly at the beginning of the paper how our results, especially the comparisons with the constraint programming solver, should be interpreted (see second paragraph, page 3).

— “*The present paper makes some claims that do not seem well justified:*

- “*In the last paragraph of page 5, it is said that the constraint programming approach had many difficulties when treating the setup times introduced by resource displacement. This is strange as there is, in ILOG Scheduler, the concept of transition time which exactly covers this need which is very common in scheduling problems. It allows to represent and efficiently propagate setup times on resources. Instead, in the ORCA implementation, a quadratic number of meta-constraints is introduced which results in a dramatical slow down of the engine (and a very weak propagation).*”

As we now say in the text, we had no control on how the ILOG tools were applied. The difficulties faced by ORCA were reported to us by the team of developers from Petrobras.

The reviewer is correct about the concept of transition time in ILOG Scheduler. Actually, as the Petrobras' team informed us, this concept was used in ORCA.

In the first version of the paper we tried to explain in a succinct way how this concept works for those not familiar with the ILOG Scheduler. The creation of a “quadratic number of meta-constraints” is implied by the information provided in the manual of this library: “Given two activities A1 and A2 the transition time between A1 and A2 is an amount of time that must elapse ...”. In other words, there is one transition time between each pair of activities. Thus, the number of transition times is quadratic in the number of activities.

Anyway, since we have no further information about this topic, we decided to shorten this discussion in the paper (see second paragraph, page 4).

- “It is not said which type of search was used in ORCA. Was it a pure deterministic tree search? A randomized search? Was local search used at some point? Pure deterministic tree search is known, especially in scheduling, not to be very efficient for dealing with optimization problems where at least some randomization is necessary. This is by no mean inconsistent with using a CP framework.”

Again, we emphasize that we were not allowed to alter the source files of ORCA. However, we were informed by Petrobras that ORCA used some pre-defined search techniques available in the ILOG Solver: DFS (Depth First Search), BFS (Best First Search), LDS (Limited Discrepancy Search), DDS (Depth Bounded Discrepancy Search), IDFS (Interleaved Depth First Search). The best performance was attained with the LDS approach. A comment about this choice has been added to the text (see the last paragraph before Section 3, on page 13).

We want to stress here our belief that there is probably a better way to use Constraint Programming (CP) in this problem. We agree that it would be unfair to make a statement that GRASP is better than CP for this problem. This was not our intention and we hope that this is clear in the new version of the paper.

- “It is mentioned on page 11 (item 1) that the basic greedy heuristic implemented in the present approach finds the best solution in 6 out of 17 real instances. This basic heuristic clearly makes sense and could have easily been implemented in ORCA on top of CP. This doesn’t seem to be the case as ORCA only finds the best solution in 2 instances (c.f. Table 5).”

We agree with the referee. This is certainly something that the team of developers of ORCA should had considered. But again, we could not change ORCA’s source code, since it was not made available to us.

— “I strongly suggest the authors to clearly separate (1) the benefits/drawbacks of using a CP framework from (2) the implementation choices on how to use it that were made in ORCA (constraint model, type of search, heuristics).”

We have, to some extent, followed this suggestion. We state clearly from the beginning that we had no control on the implementation of ORCA and we also establish the limits of the comparisons involving the results attained by ORCA and by our GRASP heuristics.

### 3 Conclusions

Finally, we express our appreciation for all the referee’s detailed comments and more philosophical remarks. They were all very helpful when revising the text.

### References

Harvey, W. D. & Ginsberg, M. L. (1995), ‘Limited discrepancy search’, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* **1**, 607–613.

# On the Comments from the Third Reviewer

We comment on the suggestions put forward by the Third Reviewer. As can be seen below, we have followed most of the specific suggestions offered by the Reviewer. In doing so, we believe we have much improved the presentation.

We separate our comments in two parts: one part discusses the Reviewer detailed suggestions and another part replies to his more general, or philosophical, comments.

## 1 On the Reviewer's "other comments"

We start with his "minor comments".

**page 2 (fourth paragraph):** "*Some results are commented even though the results were not presented.*"

OK, the results that were commented upon were moved to the concluding section.

**section 1.2:** "*In a mathematical model for the problem, C1 and C7 would be comprised in the same set of constraints.*"

True. We separated them because they have very distinct meanings to field engineers who plan the development of activities.

**section 1.3:** "*Section 1.3 is useless. We have the impression that the authors will explain the constraint programming method, but in fact we face a raw description of the tools used (CPLEX solver and scheduler). Besides, the results presented are not referenced.*"

The Reviewer certainly meant the ILOG Solver and Scheduler, not CPLEX. In this section we introduced a search tool, named ORCA, developed by Petrobras to solve the WDRDP. Our intention was only to describe the basic tools used by ORCA, nowadays a successful application in use at Petrobras. Clearly, our focus in that section was not to explain constraint programming modeling, nor constraint programming tools (like the Solver and the Scheduler) that can be used to search for solutions.

But the Reviewer is right about the misleading title of Section 1.3, which leaves the impression that the text will deal with constraint programming methods and tools. We have changed the title of the section to a more appropriate one.

**page 5 (first paragraph of section 1.4):** "*Replace square by rectangle.*"

Done. We did the same also for the first paragraph of Section 2.5.

**(Figures 1 and 2):** "*Do the light gray rectangles represent different oil wells or different activities at the same oil well?*

- *It seems that dark gray rectangles are the resources and not group of activities.*
- *I did not understand the statement below, perhaps due to the quality of the figure: 'Each square in the "Schedule" area represents a different well. A square in dark gray represents a group of activities of the same well which are executed by the resource, the light gray square, which is in the same horizontal line.'*

The Reviewer stands correct.

The light gray rectangles are activities, or group of activities associated with a same well. The dark gray rectangles represent resources. Each resource executes the sequence of activities that appear in the corresponding horizontal line.

We have rewritten that part of the text in order to clarify these points.

**page 5 (third paragraph of section 1.4):** “*Thus, a good selection of which resource will execute the activity is even more important when resource displacement is considered.’ A good criterion for selection is always important, regardless of the problem or variant considered.”*

Of course it is. What we meant is that now the criterion for selection is *even more important*, as it will cause higher impact in the solution quality.

We changed the text to make this point clear.

**page 6 (first paragraph):** “*The argument for not trying tabu search is insufficient since the authors provide a local search algorithm from which a tabu search algorithm could be devised.”*

It is true we could have tried a tabu search with the neighborhood we defined for GRASP. Given that it is a small neighborhood (quadratic), it seemed to us that applying a fast local search over many greedy random solutions was a very promising start. And this is precisely the philosophy behind GRASP.

But, as the Reviewer pointed out, the text could be read to imply that tabu search should not be tried at all. We removed the misleading sentence.

**page 8 (Intensification):** “*In our problem, a solution is considered sufficiently different from another if the number of activities that have different start dates in both, or the number of different resources allocated to both is higher than the number of activities divided by the number of resources. As the number of resources is usually smaller than five, two solutions will be sufficiently different if at least 20% of their activities have a different start date or resource allocated to both.’*

*This criterion is not clear. What is the relationship between the number of activities and the value (number of activities/number of resources)? It’s even contradictory since if we have fewer resources, the probability that the activities are assigned to different resources is smaller. However, the authors state, accordingly to the criterion, that more different assignments (activities - resources) are required to consider that two solutions are different. Besides, two solutions can be quite similar, although having different start times for the activities. There is no test in the article to evaluate the established criteria.”*

We agree with the Reviewer that the criterion was not clear, and we have rewritten the text.

The threshold value  $V$  that was used to tell when two solutions were sufficiently different was computed as the number of activities divided by number of resources,  $NA/NR$ . If the number of activities increases, we force the difference between two solutions to be higher, in order for them to be considered different. As for  $NR$  we followed a different strategy. With fewer resources, the probability to generate similar schedules increases. In this case, we wanted the threshold value also to increase, so that two similar solutions would not be considered different.

We did test other thresholds that could be used to ascertain when two solutions were considered different. This was done during the ‘parameter setting phase’ in the development of

our application. The value used,  $NA/NR$ , proved to be the best among all other thresholds tested. Also, since this value is in the range of 20%-25%, and gave good consistent results, we did not spend too much time in this issue. Further, note that  $NA/NR$  is in this interval, for almost all instances.

In any case, we followed the Reviewer's advice and inserted a few sentences in the text indicating that this value for  $V$  was selected because it produced the best results when tested against several other possibilities.

**page 9 (first paragraph of section 2.3):** “*What is the sampling of the integer variables representing the times?*”

There are as many time variables as there are activities, a number which is typically around 110. Values in the domain of each time variable represent days, and the domain of these variables ranges from 0 to the time horizon, usually from 1000 to 3000 days.

Some of these numbers are explained in Section 3.1. We added a few more comments in the first paragraph of Section 2.3, in order to make the range of time variables more precise to the reader.

**page 11 (last paragraph):** “*'In case of violations, and this can be tested after each activity is scheduled, the construction of this solution is aborted and a new one is started. We could, instead, backtrack a few steps, but this would slow down this phase, especially if the first steps were not appropriate.'*”

*In which frequency it happens? In a so constrained problem, it should happen a lot of times.”*

In the construction phase almost all solutions are constructed until they are complete. The instance with fewer solutions constructed until completion had less than 10% of them interrupted before completion. We used some special strategies to help completion, though. For example, we created a routine that, during the partial construction of solutions, searches for activities that have a very short domain, and that have not yet been scheduled. If such activities are found, we try to schedule them at this point. We do this to prevent other activities to be scheduled before the problematic ones, thus obliterating the short domains and rendering invalid the solution being constructed. The problem being so constrained makes the work of this special routines yet more important.

We inserted a few lines in the text that comment further on this point.

**page 12 (first paragraph):** “*It should be a continuation of the previous paragraph.”*

We followed the suggestion.

**page 12 (Local search):** “*The article could be enriched if a graphic illustration is created to show the advantages and shortcomings of each option considered for the local search phase.”*

We agree that it could. However, the present manuscript shows some length already. This is due to all the rewriting and clarifications introduced when following most of this Reviewer's suggestions, and also adhering to most of two other set suggestions from other Reviewers. At this point, we feel that the inclusion of illustrations, together with some accompanying text would render the manuscript too lengthy, without introducing that much precision to the ideas, besides what is already in the text.

**table 3:** “*The lower part of Table 3 shows the ten instances where C7 constraints were present.*”

Yes, the Reviewer is correct here. We have not changed the table, since this is clear from the table as it is depicted now.

**table 3:** “*In Table 3, the authors should also provide information about the average outflows and average times for performing the activities.*”

We followed the suggestion and inserted two lines in Table 3, presenting the requested information.

**page 13 (second paragraph of section 3.1):** “*Thus, any gains in production, when comparing two solutions, happen before all wells are scheduled in both.*”

Yes, the Reviewer is also correct here. We did not feel that any text ought to be added to the manuscript in order to clarify this point any further.

**page 14 (first paragraph):** “*Make a label for the instances used in the graphics.*”

All the graphics have been relabeled. Now the labels also indicate the instance that was used as data in each case.

**page 14 (first paragraph):** “*How many times the size of the RCL was changed in each one of the instances tested?*”

Typically, 4 times. We inserted a sentence, commenting on this point.

**page 14 (section 3.3):** “*The criteria 1 and 2 are not good since the sum of average and best results are absolute values. To solve it, the authors can try to normalize the results.*”

We used these criteria in order to have an absolute yardstick that could be used to order the solvers. On the other hand, we could also have used normalized results, as suggested by the Reviewer. In fact, we have recomputed the figures in (the original) Table 5, normalizing the results for the ease of visualization.

**sections 3.4 and 4:** “*The sections 3.4 and 4 could be condensed.*”

We followed the suggestion. Section 3.4 was merged into Section 4.

**page 16:** “*The paragraph below*

*It is worth mentioning that ORCA is built using the ILOG Constraint Programming suite, a set of expensive and sophisticated libraries with years of development. Using GRASPW, which was entirely programmed from the ground up, these costs could be averted. In opposition, the ILOG suite favors easiness of development, of maintenance and of understanding of the source code.*

*is not appropriated for a scientific article. It is more suited for a presentation inside the company that sponsored the work.*”

We followed the suggestion and removed the offending text from the manuscript.

**tables and figures:** “*Why the article has a section for figures and tables? Even worse, the algorithms are classified as tables!*”

A separate section for figures and tables is a recommendation from the Journal of Heuristics style for authors, at least in this stage of manuscript revision. We believe that figures and tables will be properly interspersed with text, in the final printed form of the article.

That notwithstanding, we reclassified the algorithms as figures, not tables, as suggested by the Reviewer.

## 2 On the Reviewer's "major reasons"

In this part, we comment on the Reviewer's major complaints about the original text.

**organization:** "*There is no organization to present the problem (WDRDP). Many parts of section 1 and 2 could be condensed in just one introductory section.*"

We reckon that the Reviewer is referring to the introductory section and Section 1. Section 2 is specialized about the GRASP heuristic, and there is where some advanced techniques that we implemented were discussed.

But we recognize that the initial organization of the manuscript could also be improved if we followed the Reviewer's advice. We have considerably reorganized the introductory section and Section 1.

**formal definition:** "*It also lacks a formal description for the problem. The constraints and objective function are literally explained. Just to exemplify the benefits of a mathematical model, if it was provided, the resource displacement feature would not be a constraint, as said in the text (in order to verify the importance of this feature in the paper, it is enough to look to its title).*"

We did not provide a formal problem description because the WDRDP is similar to other traditional scheduling problems. It has its particularities, and some different constraints and objective function, but that did not seem to demand a formal description of the whole problem. Besides, there were no need to write a mathematical description of the problem, as would be necessary if we were to resort to some form of linear programming model.

As for the resource displacement feature not being a constrain, the ILOG Scheduler uses transition times in order to treat this feature. According to the ILOG Scheduler Reference Manual: “Transition times are taken into account when the disjunctive constraint or the type timetable constraint on a resource are posted”. That is, resource displacement is just a transition time constrained to happen between activities using the same resource. The importance of resource displacement was that ORCA could not deal well with this feature. Actually, it was successful, but Petrobras engineers noticed that the solutions were not as good as they were before this feature was introduced in the model they were developing. Since we had already developed a better tool than ORCA without considering the resource displacement (Pereira et al (2005)), it was natural to conjecture that we could also address the resource displacement constraint, thereby making our solutions much more realistic.

Computational experiments confirmed that supposition.

**graphics:** "*The graphic quality of the figures is horrible!*"

Acknowledged. We have completely redone all graphics. They are now much cleaner and clearer.

**pseudo-codes:** “*The authors neglected to provide the pseudo-codes of the proposed algorithms for the problem. Instead, they provide less relevant generic algorithms (e.g. constructive heuristic and GRASP) while the reader would prefer to have a better understanding of how these algorithms were instanced to solve the WDRDP.*”

Indeed. We agree that pseudo-code for the new algorithms would be useful, and we have incorporated several of them: Figures 3–8. Also, throughout the text, we have inserted comments referring to those figures, when appropriate.

But we kept Tables 1 and 2, now more appropriately cited as figures, since we use them for comparison purposes when explaining the new construction phase of our GRASP implementation.

**heuristic description:** “*It seems that the authors are not familiar with a heuristic description. They do not provide a meaningful algorithmic description of GRASPW, focusing on constraint programming terms to explain it.*”

We have provided algorithmic description for the major six blocks of GRASPW, in the form of pseudo-codes. See the new Figures 3–8. Of course, the text was also modified, at appropriate places, in order to explain the pseudo-codes and to call the reader’s attention to them.

**POP:** “*The authors conclude that Proximate Optimality Principle is the best advanced technique to be used with the GRASP proposed, but do not explain or reference anything related to how:*

- a. the Path-relinking method was designed;*
- b. the Intensification was controlled together with the chosen bias function;*
- c. the Selective Local Search was used in terms of parameters.*

*Therefore, the arguments for this conclusion are insufficient.”*

Several parts of the text, related to the issues above, have been rewritten. See the new Section 2.2. Pseudo-codes have been provided, giving much more details on the implementation. See Figures 3–8. In particular,

- *Path-relinking:* see Figure 6 for a pseudo-code for the path-relinking routine. See also Figure 3, to obtain information as to when the routine is activated.
- *Intensification:* the text has been considerably rewritten, giving more precise details on how the elite set is created and maintained. The bias is used in the search procedure, as now can be seen from Figure 3.
- *Selective Local Search:* see Figure 7 for a pseudo-code for the local search routine. See also Figures 3 and 5 to observe where the local search strategy is used in the global search routine and in the POP routine, respectively.

**references:** “*The references are out of date. Many articles were already published or have more recent versions.”*

We reviewed all our citations, and substituted the one we found to be outdated for more recent journal versions.

### **3 Conclusions**

Finally, we express our appreciation for all the referee's detailed comments and more philosophical remarks. They were all very helpful when revising the text.

# GRASP STRATEGIES FOR SCHEDULING ACTIVITIES AT OIL WELLS WITH RESOURCE DISPLACEMENT

ARNALDO V. MOURA<sup>1</sup>, ROMULO A. PEREIRA<sup>2</sup>, AND CID C. DE SOUZA<sup>3</sup>

*Institute of Computing, University of Campinas*

<sup>1</sup> Instituto de Computao, UNICAMP, Caixa Postal 6176  
13084-971 Campinas, SP  
arnaldo@ic.unicamp.br  
+55 (19) 3788-5859

<sup>2</sup> romulo\_a\_pereira@yahoo.com.br

<sup>3</sup> cid@ic.unicamp.br

## Abstract

Before promising locations at petroliferous basins become productive oil wells, it is necessary to complete development activities at these locations. The scheduling of such activities must satisfy several conflicting constraints and attain a number of goals. Moreover, resource displacements between wells are also important. We describe a Greedy Randomized Adaptive Search Procedure (GRASP) for the scheduling of oil well development activities with resource displacement. The results are compared with schedules produced by a well accepted constraint programming implementation developed by a large oil company. Computational experiments over real instances proved that the GRASP heuristics implementation was competitive, outperforming the constraint programming implementation.

**Key-words:** *Offshore petroleum, GRASP, constraint programming, scheduling.*

Oil and gas are important fossil ingredients for making plastics, dyes, kerosene, gasoline and many other products. A significant amount of these fossil fuels is extracted from oceanic basins, *e.g.*, from the offshore Marlin basin in Rio de Janeiro, Brazil. Petrobras is a company with recognized expertise in oil exploration in deep sea waters, being also one of the twenty biggest oil companies in the world. Usually, Petrobras explores diverse petroliferous basins, each with hundreds of promising spots where productive oil wells could be located. However, before these places are turned into productive wells they must be developed, that is, a sequence of engineering activities must be completed at each promising spot, to render them ready for oil extraction. Oil derricks and ships are used to complete these activities. These resources have to move from one spot to another and such displacements must be considered when sequencing the activities. Furthermore, such resources are limited and expensive, either in acquisition or rent value, and must be used efficiently.

More specifically, when a spot is considered a promising oil well, oil derricks are sent there to accomplish the due *drilling* operations. After a well is drilled, the preparation for oil extraction develops in several stages. First, in the stage called *completion*, oil derricks place Wet Christmas Trees (or WCTs, structures where hydraulic valves are attached) at the mouth of the wells in order to avoid oil leakage. Later, boats connect pipelines between WCTs and manifolds, this stage being called *interconnection*. Manifolds are metallic structures installed by boats at the sea floor. Their use prevents the need for exclusive pipelines connecting each well to the surface, which would be prohibitively expensive. Once this stage is completed, oil extraction can begin. For that, Stationary Units of Production (SUPs) are anchored at specific locations in the surface and boats interconnect manifolds to them. SUPs are used to process, and possibly store, the extracted products. Later, ships fetch the products from SUPs to land storage sites or other processing units. If the oil outflow is very high or a SUP does not have storage capacity, a petroliferous platform may be installed at the surface.

Usually, there are two types of wells that may be developed: (i) *productive wells*, which are those that have an oil yield; and (ii) *injection wells*, where only maintenance activities are executed and oil is not extracted from them.

The oil *well development with resource displacement problem* (WDRDP) can be summarized thus: given a set of promising spots, the activities to be executed at each location, and the available resources, find a scheduling of the activities and resources, fulfilling several conflicting engineering and operational constraints, including the displacements of the resources, in such a way as to optimize some objective criteria. In this work, the specific WDRDP faced by Petrobras is studied. This WDRDP imposes much more realistic constraints than other similar studies (do Nascimento (2002), Pereira et al. (2005)). Actually, Pereira et al. (2005) is an early version of this paper, which did not consider resource displacement. Here, we take this characteristic into account when describing our GRASP heuristic strategies for maximizing oil production within a given time horizon. In addition, we discuss other advanced techniques not explored in Pereira et al. (2005).

A project team from Petrobras addressed the same problem and developed a Constraint Satisfaction (*cf.*, Marriott & Stuckey (1998)) model using ILOG's Solver and Scheduler<sup>1</sup> (ILOG (1999)). After four years of development and testing, the tool, named

---

<sup>1</sup>Registered trade marks of ILOG Inc.

ORCA<sup>2</sup>, became operational and very successful (Petrobras (1999)). Nowadays, the ORCA solver is often used by engineers both to define a good scheduling for the drilling activities and, also, to analyze the need for acquiring or renting new resources. They confirmed that ORCA generates better solutions than those made by humans. In one real instance, ORCA showed that buying a third oil derrick was unnecessary and that it was better to add a new LSV ship instead. As a result, an expenditure of US\$ 15 million was avoided, while anticipating oil production by 26 days. Despite the good performance of ORCA, searching for even better solutions is still important, since a tenth of a percent improvement in the oil production may represent an increase of millions of dollars in the company's revenue.

Before we proceed, we should emphasize that Petrobras allowed us to use only ORCA's executable files. We had no access to the source code. Therefore, we are unaware of ORCA's inner details, other from some general ideas that were cleared to go public. This information is spread throughout this text. In light of that, this work should be viewed as a successful application of a GRASP heuristics to a relevant real-world problem. The comparisons we make between the GRASP heuristics and ORCA's constraint programming strategies is not to be construed upon in order to establish the superiority of one technique over the other. Moreover, that would not be a fair comparison, since there is certainly some room for improvement in ORCA's code that we could not exploit, as explained above. Nevertheless, since the technicians from Petrobras are very confident about the quality of ORCA's solutions, it is legitimate for us to use them as a measure to assess the performance of our GRASP implementation.

Section 2 discusses GRASP implementations for the WDRDP. Section 3 presents our computational results and compares them to other results derived from the constraint programming implementation presently running at Petrobras. Finally, some concluding remarks are offered in the last section.

## 1 The Well Development Problem

As already mentioned, here we will deal with a new facet of the problem, the resource displacement. This constraint forces a period of inactivity for the resource, so that it can unanchor, travel between two wells and anchor in the new spot. We call this period the set-up time. It is present between activities of different wells which are scheduled consecutively. In Figure 9.a we see a typical schedule with no resource displacement. In the figure, the light gray rectangles are activities, or group of activities associated with a same well. The dark gray rectangles represent resources. Each resource executes the sequence of activities that appear in the corresponding horizontal line. In Figure 9.b, we present a schedule with resource displacement. Note that between each pair of wells there is a void representing the set-up time.

The set-up time is a constant for each resource. It is calculated by engineers, based on the average speed of the resource and on the average distance between spots in the field, plus the average time to anchor and unanchor. As engineers noted when scheduling the activities at wells of an oil basin, the average distance between these wells does not differ much from the average distance traveled by the resources. The weather, which influences

---

<sup>2</sup>Portuguese acronym for “Optimization of Critical Resources in the Production Activity”.

the speed of the resource, was also considered in the average speed.

However, different resources may have different set-up times. Consider, for example, resources  $X$  and  $Y$  in Figure 9.b. Note that the set-up time of  $Y$  is twice longer than the set-up time of  $X$ . This may alter the selection of which resource will execute activities at a well. For example, wells  $X2$  and  $Y2$  have their “end of execution” activities scheduled at the same instant. This would not be the case if both resources  $X$  and  $Y$  had the same set-up times. When resource displacement is being considered, a good criterion to select which resource will execute which activity may be even more important in the search process, given that the set-up times may postpone the schedule of activities.

According to its developers, ORCA faced difficulties when treating the resource displacement constraints. Several constraint programming techniques available in the ILOG Solver and Scheduler, like the usage of transition times, were tried by the Petrobras team when dealing with these constraints. A first approach created virtual activities to represent the set-up time. But, since it is not known in advance if an activity from another well will be scheduled consecutively, it would be necessary to create virtual activities dynamically during the search for solutions. Of course, they would have to be destroyed when backtracking. This was a complex process and slowed down the search considerably. Instead, an *if-then constraint* was set, imposing that if the previous activity executed by the resource was from another well, then the current activity was shifted by the corresponding set-up time. As this constraint was imposed between each pair of activities, the number of constraints to be added to the model was quadratic in the number of activities. The effect was a slow down in the execution of ORCA, causing it to generate poor solutions for many instances. As a general remark, the use of the readily available techniques to treat the resource displacement constraints slowed down the execution of ORCA, reducing its ability to generate good solutions in an satisfactory amount of time.

## 1.1 The WDRDP Constraints

Any solution for the WDRDP must obey the following constraints involving the oil well development activities:

- C1. *Technological Precedence*: sets an order between pairs of certain activities. When considering the precedence between the start and finish of the activities in each pair, any of the four possible combinations can be present.
- C2. *Mark-Activity*: an activity must finish before or initiate after a fixed date, or *mark*, with or without lag time. This date is often related to some external event, *e.g.*, the installation of a petroliferous platform.
- C3. *Baseline*: sets the start date of the activities.
- C4. *Use of Resources*: to execute an activity, due to its intrinsic nature, the resource used must match some operational characteristics. For a boat, it must be verified if the on-board equipments can operate at the specified depth. For an oil derrick, its type and capabilities must be verified, as well as its maximum and minimum depth of operation and drilling. All resources are disjunctive, that is, an activity can require at most one resource at any given time.

- C5. *Concurrence*: two activities at the same well, or executed by the same resource, can not be simultaneous.
- C6. *Unavailability*: resources may be unavailable for a period of time, either for maintenance reasons or due to contract expiration.
- C7. *User Defined Sequences*: the user can specify a sequence for the drilling activities or for the “start production” activities of different wells, depending on the type of this sequence. These sequences are specified by engineers in order to avoid loss of pressure in the oil field. If a well *A* appears before well *B* in the sequence, then the activity of well *A* must terminate before the start of activity of well *B* can be scheduled.
- C8. *Surface Constraints*: represented by a polygonal security area defined around a well. When the center of a well is inside the restricted area of another well, activities executed at both wells cannot be simultaneous. These constraints must be verified between pairs of mobile and pairs of mobile and anchored oil derricks.
- C9. *Cluster Constraints*: an activity may be part of a cluster, which is a set of activities that must use the same resource.
- C10. *Oil Derrick Displacements*: when an oil derrick moves between two wells, a set-up time will be considered. Therefore, unnecessary displacements must be avoided, for example, by making the same oil derrick execute as much activities at a well as possible. For more details, see Section 1.

The oil yield is calculated as follows. Each well has an associated outflow and an activity that marks the beginning of its production. When this last activity is concluded, the well is considered in production. The yield is obtained by multiplying the oil outflow by the period between that instant and the established time horizon of production. If the start production activity is set for a time after the horizon, the corresponding yield is disregarded. The objective is to obtain a scheduling of all activities, satisfying all constraints, while maximizing the oil yield.

Other goals to be attained by automating the scheduling of the activities are:

1. *Faster solutions*. Human made solutions take many hours, even days, to be constructed. A faster method would permit the analysis of different scenarios for the same problem, for example, by adding or removing resources. Furthermore, modifications in already committed plans would not result in new hours, or days, spent in rescheduling.
2. *Better resource allocation*. With an automated scheduling, all highly skilled engineers responsible for the manual scheduling can receive other duties.
3. *Savings*. The automated schedule will, usually, result in a better use of boats and derricks, thereby saving considerable operational costs.

As can be seen, the WDRDP is a difficult combinatorial optimization problem. In fact, it is simple to devise a polynomial-time reduction from the classical *Job Shop Scheduling Problem (JSP)* to the WDRDP, thus showing that the WDRDP is NP-hard.

## 2 GRASP Strategies for the WDRDP

Despite its conceptual simplicity, GRASP (Feo & Resende (1995)) is a well studied meta-heuristics which has been successfully applied to a wide variety of optimization problems (Glover & Laguna (1997), Festa & Resende (2002)). In particular, GRASP implementations have dealt well with scheduling problems (Bard & Feo (1989), Feo & Bard (1989), Feo et al. (1995), Bard et al. (1996), Binato et al. (2002)). After some initial investigation (do Nascimento (2002)), a GRASP approach seemed appropriate to solve the WDRDP. In fact, a GRASP heuristics had already proved itself valuable in dealing with a simpler version of the WDRDP, where resource displacement was not considered (Pereira et al. (2005)).

Subsection 2.1 reviews some GRASP basics, while subsection 2.2 presents some more advanced GRASP techniques. The remaining subsections 2.3–2.5 describe our specific implementation, named GRASP-WDRDP (or GRASPW, for short), which was designed to solve the full WDRDP, with resource displacement.

### 2.1 Greedy Randomized Adaptive Search Procedure (GRASP)

In the GRASP methodology, each iteration consists of two phases: a *construction phase* followed by a *local search phase* (Feo & Resende (1995)). See Figure 1.

In the construction phase, a feasible solution is built one element at a time. See Figure 2. At each step of the construction phase a candidate list is obtained by ordering all elements that can be added to the solution at this point. The ordering follows a greedy function that measures the, maybe myopic, benefit of inserting each candidate element into the partial solution being constructed. After the candidate list is constructed, an initial segment is selected, forming the *Restricted Candidate List* (RCL). The construction step is completed by randomly selecting an element from the RCL and effectively inserting it into the partial solution. The construction cycle repeats itself until a complete solution is obtained. The adaptive component of the heuristic arises from the fact that the benefits computed by the greedy function are updated at each step, in order to reflect the changes brought on by the selection of the candidate in the previous step. The probabilistic component stems from the random choice of one of the candidates in the RCL, not necessarily the best one, to be added to the partial solution. This mechanism allows for different solutions to be obtained at each iteration, while not necessarily jeopardizing the adaptive greedy component.

The solutions generated by the construction phase are not guaranteed to be locally optimal. Hence, in the local search phase, a local search procedure is applied to improve the constructed solution. The search phase is a standard deterministic local search algorithm that seeks to optimize the solution built in the previous construction phase.

### 2.2 GRASP Advanced Techniques

As the problem increased in complexity, compared to the problem with no resource displacement studied in Pereira et al. (2005), we decided to consider some improvements and alternative techniques to be introduced in the basic GRASP procedure. They are discussed in the sequel.

**Bias Function:** In the construction procedure of the basic GRASP, the next element to be introduced in the solution is chosen at random from the candidates in the RCL, with the elements of the RCL being assigned equal probabilities. However, any probability distribution can be used to bias the selection toward particular candidates. Bresina (1996) proposed a construction mechanism based on the *rank*  $r(\alpha)$  assigned to each candidate element  $\alpha$ , according to its value, measured by the greedy function,  $v(\alpha)$ . However, as the sorting of elements to obtain the *rank* has time complexity  $O(n \log n)$ , where  $n$  is the number of elements in the RCL, and since the sorting must be repeated at each choice of an element, this process could slow down the implementation. Thus, we decided to use directly the value  $v(\alpha)$  of the candidates to create the following bias functions:

- uniform:  $bias(\alpha) = 1$
- linear:  $bias(\alpha) = v(\alpha)$
- log:  $bias(\alpha) = \ln v(\alpha)$
- exponential:  $bias(\alpha) = e^{v(\alpha)}$
- quadratic:  $bias(\alpha) = v(\alpha)^2$
- square root:  $bias(\alpha) = \sqrt{v(\alpha)}$

Once the value of the bias function is evaluated for all elements of the RCL, the probability of the candidate  $\alpha$  being chosen is:

$$\frac{bias(\alpha)}{\sum_{\alpha' \in RCL} bias(\alpha')}$$

The bias function was inserted into the new construction phase, see Figure 4.

**Proximate Optimality Principle (POP):** This technique is based in the idea that “good solutions at one level are likely to be found ‘close to’ good solutions at an adjacent level” (Glover & Laguna (1997)). Fleurent & Glover (1999) provided a GRASP interpretation of this principle. They suggested that imperfections introduced during steps of the GRASP construction phase can be removed by applying local search during (and not only at the end of) the GRASP construction phase. A practical use of the POP in a GRASP implementation would be to apply a local search during a few intermediate points in the construction phase, and not at the end of each construction iteration. Binato et al. (2002), when dealing with a JSP, applied a local search when 40% and 80% of the construction moves had been taken, as well as at the end of the construction phase. We tested this idea by running a local search procedure at several intermediate points during the construction phase. The best results were obtained when the same 40% and 80% of the construction moves had been taken. Given the closeness between the JSP and the WDRDP (see Section 1.1), this was not surprising.

The local search strategy we used in order to test the POP was a usual well swapping process, with the proviso that we only swap wells that have already been scheduled. A pseudo-code for the implementation of POP in the new GRASPW procedure is illustrated in Figure 5. The routine is activated during the construction of a greedy solution, see Figure 4.

One possible shortcoming of the basic GRASP method is the independence of its iterations, *i.e.*, the fact that one iteration does not learn from the history of solutions found in previous iterations. This is so because the standard algorithm discards information about any solution encountered that does not improve the incumbent solution. But it is known that information gathered from good solutions can be used to implement memory-based procedures. We show some of these strategies below. There are, of course, a number of other possibilities to induce dependence among constructive phases of GRASP, such as ant colony optimization.

**Intensification:** Fleurent & Glover (1999) proposed a scheme to introduce long term memory in a GRASP heuristics using a set of elite solutions.

In our case, in order to be included in this elite set, a solution must be better than all elite solutions already in the set, according to their objective function values. Or, else, it must be better than the worst solution of the set, while still being sufficiently different from all elite solutions in the set. A solution is considered sufficiently different from another one if the number of activities that have different start dates in both, or that have a different resource allocated to both, is higher than the number of activities divided by the number of resources. As the number of resources is usually smaller than five, two solutions will be sufficiently different if at least 20% of their activities have a different start date, or a different resource allocated to both. We did test other thresholds values that could be used to ascertain when two solutions were considered different. This was done during the parameter setting phase in the development of our application. The value we retained proved to be the best among all other thresholds tested.

The size of the elite set, fixed at 5 elements, was also obtained by testing different values over real instances. A five element set produced the best results, while keeping the elite set quite small. At the start, the first 5 solutions populate the elite set. From this point on, an element is always removed from the elite set each time we insert a new one. We tried two strategies for removing elements from the set. In the first one, we removed the element with the smaller oil production, among all elements in the elite set. In the second trial, we removed the element with fewer differences in its scheduling, when compared with the new element. Again, tests run on real instances proved that the first approach gave superior results.

Next, we show how the elite solutions biased the selection of the candidates in the GRASP construction phase. To each candidate  $\alpha$  we evaluate an intensity function,  $Int(\alpha)$ , defined as:

$$Int(\alpha) = \frac{\sum_{e_i \in S} Prod(e_i)}{Max_{e_j \in E} Prod(e_j)},$$

where  $S$  is the set of elite solutions in which the element  $\alpha$  has the same start date and resource allocated to as the solution being built,  $Prod()$  is the oil yield associated with a solution,  $E$  is the set of elite solutions, and  $Max$  returns the maximum value in the indicated set. With the size of the elite set limited to 5 elements, the computation of  $Int(\alpha)$  was very fast.

Let  $iter$  be the number of iterations executed and let  $k$  be a parameter. We use the

intensity function to define the bias function,  $biasInt(\alpha)$ , as:

$$biasInt(\alpha) = bias(\alpha) + Int(\alpha) \times (iter/k).$$

The fraction  $iter/k$  is used to reinforce the role of the intensity function as the number of iterations increases and, thus, the quality of the elite solutions in which the intensity function is based also possibly increases.

**Path-Relinking (P-R):** This technique was originally proposed in Glover (1996) as an intensification strategy exploring trajectories connecting elite solutions obtained by tabu search or scatter search methods (Glover (2000), Glover & Laguna (1997), Glover et al. (2000)). Starting from one or more elite solutions, paths in the solution space leading towards other elite solutions are generated and explored in the search for better solutions. This is accomplished by selecting moves that introduce, in the moving solution, attributes contained in the target solution. Path-relinking may be viewed as a strategy that seeks to incorporate attributes of high quality solutions, by favoring these attributes in the selected moves.

The use of path-relinking within a GRASP procedure was first proposed in Glover et al. (2000). It was followed by several extensions, improvements and successful applications (Aiex et al. (2005), Canuto et al. (2001), Resende & Ribeiro (2003)). Two basic strategies are:

- path-relinking is applied as a post-optimization step to all pairs of elite solutions.
- path-relinking is applied as an intensification strategy to each local optimum obtained after the local search phase.

According to Resende & Ribeiro (2002) the second strategy is more promising. Another consideration, also from Resende & Ribeiro (2002), is that exploring the two trajectories, in both directions, between two solutions results in small gains and demands twice the time. Thus, it is usually more appropriate to explore only one path, the one from the best solution to the other, as better solutions are more often found near the neighborhood of the best solutions (Resende & Ribeiro (2002)).

An illustration, in pseudo-code, for the path-relinking strategy appears in Figure 6.

**Selective Local Search (SLS):** Another approach that can be used within the standard GRASP procedure is to apply local search only in those solutions sufficiently different from the elite solutions, or that have a good quality, *i.e.*, better than the worst of the elite solutions. In other words, we apply local search only when there are higher chances of obtaining better solutions, improving the efficiency of the algorithm. This idea was also used in our new implementation of the basic GRASP procedure, see Figures 3 and 7.

## 2.3 The New GRASP Solver Implementation: GRASPW

The GRASPW implementation was constructed using the C/C++ programming language. Our model uses two types of integer variables. One represents the beginning of

execution of each activity in the corresponding well. Values in the domain of these variables represent days, and the range of each domain varies from zero to the time horizon. The latter depends on the instance being treated, but varies usually between 1000 and 3000 days. Of course, during the search for solutions, as activities are scheduled, other activities have the domain size of the corresponding variables diminished. The second type of variables represents which resource will execute each activity in a well. Their domains are characterized by a set of the possible resources, of which one must be chosen to execute the corresponding activity.

All the constraints described in Section 1 were enforced. Three constraints, namely, C2, C3 and C4, were set while reading the problem data, before the search begins. Note that, in these cases, all information needed to check the constraints are already present. The other constraints were dealt with during the search for solutions, the variables involved being assigned single values.

Constraint C10, that deals with resource displacement and which was responsible for a loss of performance of the original ORCA search algorithm, was treated here in a simple way. We displace the current activity by at least the amount of the set-up time, if the previous activity scheduled in the same resource is from a different well.

A pseudo-code for the GRASPW implementation appears in Figure 3, and can be compared with Figure 1.

## 2.4 The Construction Phase

The following two adaptations were made to the procedure illustrated in Figure 1:

- The search procedure was interrupted by a time limit instead of by the number of iterations; and
- During a complete run of the GRASPW meta-heuristic, the value of *ListSize* (see Figure 1) can be monotonically incremented by a fixed amount when a predefined interval of time is reached with no improvement on the best solution. This allows the algorithm to explore larger regions of the search space. Alternatively, during a run of GRASPW, the value of *ListSize* can be monotonically decremented between iterations, thus focusing into a greedier heuristic. With this scheme we obtain a *dynamic sized RCL*, in opposition to the original *static sized RCL*. Note that, as GRASP iterations are independent, one could imagine that there is no difference between increasing and decreasing the RCL size. However, as we do not know in advance the amount of time the algorithm will execute in each run, or when the RCL size will be altered, we can not anticipate the result of a GRASPW run, when increasing or decreasing the RCL size. Our implementation proved that, for the WDRDP, the best strategy was to increase the RCL size, see Subsection 3.2 and Figure 3.

A pseudo-code for the new construction phase implementation is illustrated in Figure 4. As in the ORCA implementation, we seek solutions with the highest oil yield. To this end, the construction phase illustrated in Figure 2 was modified thus:

1. The first time the construction phase is initiated, we set *ListSize* to one, and the algorithm behaves like a greedy heuristic. If there are few constraints obstructing

the greedy heuristic, it tends to generate good or even very good solutions. For example, in six out of seventeen real instances, the best solution was found in the first pass of the construction phase. However, in the other eleven cases the greedy solutions were much poorer than the final corresponding GRASP solutions.

2. The candidates are defined by the production wells that are available (meaning that there are no wells yet not scheduled which must precede them), or the injection wells that have activities of production wells succeeding them. The activities of injection wells that do not have activities of production wells succeeding them are left to be scheduled after all others. Note that injection wells are not productive and therefore must not be scheduled before production wells, unless there are technological constraints forcing such a schedule. We schedule the wells and not single activities because otherwise many solutions created would have a huge number of displacements of resources between wells, which would affect the oil production negatively.
3. The evaluation of incremental costs (line 3 of Figure 2) assesses how much oil a well can offer until the end of the time horizon. In order to choose the best wells to enter the RCL, we order them by the value of their oil yields.

Initially, only the well's own yield is estimated. In order to estimate the production of a well, we simulate the scheduling of this well, i.e., we insert this well in the solution and obtain its production, returning the solution to its original state afterwards. If there are wells which are constrained to succeed the current well, we also estimate the production of these successors. As this process advances, we keep the maximum among all these yields, i.e., the well's own yield and the yields of all its constrained successors. This is the production value used to order the elements that enter the RCL.

Actually, we tried many other possibilities like, for example, summing up the yield of the well and the yields of all its successors. The best schema we found, according to our tests, was the one that retained the maximum yield.

4. In the construction phase, the next element to be introduced in the solution is chosen uniformly from the candidates in the RCL (line 5 of Figure 2). However, any probability distribution can be used to bias the selection. See Section 2.2 and Figure 4. We tried some bias functions in the selection of the candidates, as will be discussed later.
5. To schedule the candidate well (line 7 of Figure 2), the routine is as follows. See also Figure 8.

As long as there are activities not yet scheduled in the well:

- (a) choose any activity available in the well, *i.e.*, one not yet scheduled and such that there is no other activity not yet scheduled in the wells that must precede it;
- (b) choose a resource for this activity that can execute it, and that can complete the activity the earliest;

- (c) set the start time of the activity at the earliest possible time, *i.e.*, the maximum between the earliest time the resource is available to execute the activity (considering the constraints, including the set-up time) and the minimum start time of the activity; and
- (d) all activities that are constrained to succeed the chosen one must have their minimum start times updated so that all constraints are satisfied.

The scheduling of a well is done so as to satisfy all constraints, including the seven ones not yet enforced while reading the problem data. In case of violations, and this can be tested after each activity is scheduled, the construction of this solution is aborted and a new one is started. We could, instead, backtrack a few steps, but this would slow down this phase, especially if the first steps were not appropriate. We note that, in the construction phase, almost all solutions are constructed until they are complete. We used some special strategies to help completion. Specifically, we created a routine that, during the partial construction of solutions, it searches for activities that have a very short domain, and that have not yet been scheduled. If such activities are found, we try to schedule them at this point. We do this to prevent other activities to be scheduled before the problematic ones, thus obliterating the short domains and rendering invalid the solution being constructed. After a well is scheduled, any activities that must succeed it have their minimum start times updated to satisfy all constraints. If that is not possible, the construction of this solution is also aborted.

## 2.5 The Local Search Phase

As we have no guarantees that the solution found in the construction phase is locally optimal, a local search is used to improve the solution. To this end, an appropriate neighborhood was defined, so as to permit explorations that quickly lead to better solutions. The 2-exchange local search algorithm based on the disjunctive graph model of Roy & Sussmann (1964) was used. The same neighborhood was used in Binato et al. (2002) for a Job Shop Scheduling problem. In order to apply the 2-exchange local search to the WDRDP, we swap two elements in the scheduling. For example, consider the schedule presented in Figure 10.a. Groups of activities (light gray rectangles) are executed by the resource (dark gray rectangles) which is depicted in the same horizontal line. The size of the rectangles represents the execution time of the corresponding elements. We swap elements *A* and *B*, which results in the schedule shown in Figure 10.b. Since the execution time of elements *A* and *B* can be different, all activities after them may have their start times updated.

We need to decide, of course, what an element stands for. Some options are:

1. *An activity.* Very small granularity, giving rise to huge neighborhoods (do Nascimento (2002)) and, worse, moving an activity to another position would possibly force its predecessors and successors in the same well to be moved as well, in order to avoid the displacements that may result from constraints of type C10;
2. *A well.* With higher granularity. But since the sequence of activities in a well may be splitted in the present schedule due to constraints of type C1, moving all

activities takes time to verify all constraints, and exchanging the whole well may not be possible even though exchanging only part of it could be;

3. *Part of a well.* That means a maximal set of activities of the same well scheduled consecutively in the same resource. With medium granularity, it avoids the displacements that may result from constraints of type C10.

In our implementation, we chose the last alternative, where the local search algorithm exchanges all pairs of parts of wells, no matter on what resource they have been scheduled. That neighborhood is of size  $O(n^2)$ , where  $n$  is the number of parts of wells. For practical instances, this is one order of magnitude smaller than the neighborhood that uses activities as the moving elements. See Figure 7.

To fully specify the local search phase we need a rule that defines how the neighborhood is searched and which solution replaces the current one. This rule is called the *pivoting rule* (Yannakakis (1997)), and examples of it are the *first improvement rule* (FIR) and the *best improvement rule* (BIR). In the first case, the algorithm moves to a neighboring solution as soon as it finds a better solution; in the second case, all neighbors are checked and the best one is chosen. In either case, the worst case running time of each iteration is bounded by  $O(n^2)$ , where  $n$  is the number of elements in the neighborhood. In the next section we present a comparison between these two alternatives.

Besides the improvements proposed above to the standard GRASP procedure, all the advanced techniques presented in Section 2.2 were implemented and tested.

It is also worth saying a few words about ORCA's innerworkings. The ORCA algorithm uses the basic ILOG Solver and ILOG Scheduler libraries. As usual, after all constraints are imposed, the mechanism of constraint propagation is handled directly by the ILOG libraries. The search procedure is based on backtracking coupled with a branch-and-bound strategy. When traversing the search tree, the Solver always chooses the best open nodes with respect to a given evaluation function, before moving to that node in order to expand it. The evaluation function used in ORCA is the Limited Discrepancy Search (LDS), which was first defined in (Harvey & Ginsberg 1995).

### 3 Computational Results

In this section, computational results for the GRASPW implementation are discussed. They are also compared with results obtained with the ORCA implementation over the same real instances.

Beforehand, it is important to mention that we treated the ORCA algorithm as a black-box, for comparison purposes. That is, we did not tinker with its code. On the other hand, ORCA is an application that took Petrobras, a very large and competitive oil company, more than four years to develop. It has been in use for a number of years, producing results that are sensibly superior than manual solutions. This success attests to the quality of the solutions produced by the ORCA algorithm.

All tests were run on a platform equipped with a Sun SPARC Ultra 60 processor, running a Solaris 9 operating system at 450 MHz and with 1024 MB of RAM. Both the GRASPW and the ORCA implementations were allowed to run for 1800 seconds on each instance.

### 3.1 Typical Instances

Twenty two real instances provided by Petrobras were used in our tests. Table 1 summarizes the dataset. Columns with the same numerical data, like columns 8 and 9, refer to distinct instances that differ in the number of other constraints not shown in the table, like C9. The first part of that table displays the instances where no C7 constraints were found. In order to reduce the amount of time spent in testing, in some experiments we used only 7 of these instances, eliminating instances that differed only by a few constraints. The lower part of Table 1 shows the ten instances where C7 constraints were present. It is worth mentioning that, as GRASP makes use of randomization, each instance was tested five times, and the results being reported always reflect the average of the tests.

The time horizon ranges from a thousand to three thousand days. Note that, after every well has been scheduled, the oil yield is the same between all solutions until the time horizon is reached. Thus, any gains in production, when comparing two solutions, happen before all wells are scheduled. If we considered the full horizon to compute yields, these gains, in percent, would be smaller. Therefore, for each instance we use as the horizon of production the end date of the last activity of the well scheduled the latest, among all solutions of the solvers.

### 3.2 Setting GRASPW Parameters

In Section 2 we presented the idea of a *dynamic sized RCL*. There are at least two ways we could exploit this idea: we may decrease monotonically the number of candidates using a greedier heuristic; or we may increase monotonically the number of candidates in order to drive the search away from a local optimum into new regions of the search tree. In the first case, the initial RCL size was set to  $\max(13, w)$ ,  $w$  being the number of wells, and was decreased by one every 300 seconds without improvement. In the second case, we started with  $\max(5, w)$  for the initial RCL size and increased it by one every 300 seconds without improvement. The first approach did not yield good results when applied to the WDRDP, generating the same or worse solutions than those found by GRASPW with a static sized RCL. However, the second approach proved promising. Figure 11.a shows the algorithm with dynamic sized RCL generating better solutions after 150 thousand iterations, when the RCL is increased. The same happens in Figure 11.b after 50 thousand iterations, when another real instance is tested. Amongst twelve scenarios tested, four had better solutions with the dynamic sized RCL, summing up an increase of around 261 thousand barrels of oil. In the other eight scenarios, the same solutions were found. The size of the RCL changed around four times in a typical run.

We also considered two options for searching the neighborhood and selecting a new neighbor: the *first improvement rule* (FIR) and the *best improvement rule* (BIR). Tests were executed and FIR proved to be the better. On the other hand, the BIR heuristics found solutions whose production was equal to a predefined target value with the least number of iterations (see Figure 12.a). On average, to find a solution with a predefined production, the BIR strategy used about 60% of the number of iterations of the FIR strategy. On the other hand, the FIR strategy was faster in most instances (see Figure 12.b). On the average, to find a solution with a predefined production, the FIR strategy used 66% of the time used by the BIR strategy. That is because, on the average, a FIR

iteration was almost seven times faster than a BIR iteration. Note that, to users, running time is deemed important. Note also that the FIR solver built solutions with a slightly higher oil production, as can be seen in Figures 12 and 13. Summing up all instances, the gain was almost 80 thousand barrels of oil. We concluded that the FIR strategy was the one that better suited this problem.

Another technique tested was to bias the selection towards some particular candidates, those with the highest oil yield. Five probability distributions, besides the uniform distribution, were considered, as indicated in Section 2.2. Comparative experiments showed that the exponential and quadratic bias functions generated much worse solutions, with production of more than 2 millions barrels of oil smaller than the other approaches, summing up over all instances. Among the other bias functions, the one based on the square root proved to be the best one for this problem, as can be seen in Figures 14.a and 15.a. For a better view we compared the square root with the others bias functions. See Figures 14.(b,c,d) and 15.(b,c,d).

### 3.3 The GRASPW Implementations

In Section 2.2, we presented some advanced techniques that can be included in the basic GRASP procedure. Many combinations of these were tested, but not all possible combinations. With 6 distinct techniques (five advanced plus the pivoting rule), all possible combinations would number 64 distinct solvers. It was simply too time consuming to execute broad tests with each and every one of these solvers. We choose the best combinations based on faster tests, selecting a pool of 27 solvers. To simplify the visualization and data analysis, we show only 14 solvers, besides the ORCA solver. See Table 2.

As there are many solvers and in many tests there were ties between them, we elaborated five criteria to determine which would be the best solver:

1. Sum of the oil production of the best solutions for all instances.
2. Sum of the oil production of the average solutions for all instances.
3. Average of the rank obtained sorting decreasingly the solver solutions of each instance using the *Ranking1 Rule* (R1), explained below. The sorting was according to the average oil production and average time of execution to find the best solution. We ranked all solvers.
4. Average of the rank obtained sorting decreasingly the solver solutions of each instance using the *Ranking2 Rule* (R2), explained below. The sorting was according to the average oil production and average time of execution to find the best solution. We ranked all solvers.
5. Number of instances for which the solver obtained the best solution known to that instance.

*Ranking1 Rule* is a rank where, if  $s$  solvers are drawn in rank  $r$ , the next solvers would come in rank  $r + 1$ . *Ranking2 Rule*, in the same example, would put the next solvers in rank  $r + s$ .

Table 3 presents the values of the criteria achieved by the 15 solvers created for the WDRDP. Values in boldface are the best results among the solvers for each criteria. For

the ease of visualization, criteria 1 and 2 have been normalized from 0.0%, the worse result in each column, to 100%, the best result in each column. In terms of absolute values, solver G13 reached a production of 354,932,391 barrels, according to criteria 1. Solver G14 reached 346,814,967 barrels, according to criteria 2.

Note that G14 stands out as being the best solver according to three criteria, followed closely by G7 which was the best in two criteria. However we can not say yet that G14 is the best overall solver, because if it behaves poorly in the other two criteria another solver could prove to be more appropriate for the WDRDP. Given these values, we elaborated Table 4, where we show the rank of the solvers in each criterion. Note that there are significant variations in the rank of the solvers when using different criteria.

In order to make a decision about which would be the best solver, we again ranked the solvers according to the average rank they obtained in each criterion. We made this rank using both R1 and R2 type rules, as can be seen in Table 5. Using this table, we obtained the rank of the best solvers, presented in Table 6. We conclude that G14 is actually the best GRASP solver that we developed for the WDRDP.

This solver makes use of many advanced techniques for GRASP heuristics, like POP, bias function based on the square root, intensification, path-relinking and SLS (see Section 2.2). Our second best solver for the problem was G7, which makes use only of the POP technique. As can be seen in Table 3, and by the average rank in Table 5, G7 is almost as good as G14. Hence, the other advanced techniques used in solver G14, besides POP, gave rise only to marginal gains. POP, on the other hand, improves drastically the performance of a solver for this problem. This can be seen comparing G1 (basic GRASP) to G7 (basic GRASP with POP) in the same tables mentioned above, or comparing the group of solvers G1 until G6 (solvers without POP) to the group of G7 and the other solvers, that is, the group of solvers that implemented POP. When POP is applied, a great leap in quality can be seen. Note, for example, that G1 is just the eleventh best solver, while G7 is the second best.

Note also that G7 is better than G10, G11, G12 and G13, solvers which combine POP and other advanced techniques. This happened because these advanced techniques required longer processing times, and the gains were not sufficient to justify their use. Solver G7 could only be overcome when all the advanced techniques were combined in G14, and even so by a small margin. With the above facts in mind, we conclude that POP is the best advanced technique that we applied to solve the WDRDP.

## 4 Conclusions

Scheduling activities efficiently is of paramount importance to the industry, in general. Petrobras, a leading company in deep water oil exploration, presented us the WDRDP, a scheduling problem related to oil well development. Here we contrast two approaches to the WDRDP: the constraint programming tool ORCA and a GRASP implementation, dubbed GRASPW.

As can be seen in Section 3.3, the G14 solver was considered the best GRASPW implementation. Note that the constraint programming solver, ORCA, was much worse than all the GRASPW solvers, even when compared to the standard GRASP procedure, G1. Compared to ORCA, the G14 solver generated a net surplus of more than 16 million

barrels of oil, summing up over the average solutions of all instances. This increase of 4,5% in oil production would generate gains of almost US\$ 1.14 billion, quoting each barrel at US\$ 71.25, when using the G14 solver, instead of the ORCA solver. Besides, even generating better solutions, G14 takes, to find its best solution, 95% of the time ORCA takes to find its best solution. And to find better solutions than the ORCA's best one, G14 takes only 45% of the time taken by ORCA. Among the 17 instances tested, G14 generates, within the first second of execution, better solutions than ORCA's best solution in 14 instances. Two examples of that can be seen in Figure 16. In Figure 17, we show two instances where G14 does not generate, within the first second of execution, solutions better than the ORCA best solution.

However, there is an instance where ORCA was competitive compared to G14, as can be seen in Figure 18. Here, ORCA's best solution was found in 5 seconds and G14 only generated a better solution than that after 1134 seconds. Nevertheless, after 1522 seconds, G14 generated a solution with an oil production almost 63 thousand barrels superior to ORCA's best solution.

The GRASP implementations were built and then tested over several real instances provided by Petrobras. In fact, Section 3.3 could be made much longer, by trying some of the many other possible combinations over the basic GRASP heuristics that were not tried here, such as GRASP+POP+Int and GRASP+POP+P-R, for example. We leave these possibilities as extensions of this work. The combinations studied, however, already support the conclusion that the GRASPW heuristics greatly outperforms the original ORCA approach.

## References

- Aiex, R. M., Resende, M. G. C. & Toraldo, G. (2005), 'GRASP with path-relinking for the three-index assignment problem', *INFORMS J. on Computing* **17**, 224–247.
- Bard, J., Feo, T. & Holland, S. (1996), 'A GRASP for scheduling printed wiring board assembly', *IIE Transactions* **28**, 155–165.
- Bard, J. F. & Feo, T. A. (1989), 'Operations sequencing in discrete parts manufacturing', *Management Science* **35**, 249–255.
- Binato, S., Hery, W. J., Loewenstern, D. & Resende, M. G. C. (2002), A greedy randomized adaptive search procedure for job shop scheduling, in P. Hansen & C. C. Ribeiro, eds, 'Essays and Surveys on Metaheuristics', Kluwer Academic Publishers, pp. 59–79.
- Bresina, J. L. (1996), Heuristic-biased stochastic sampling, in 'Proceedings of the Thirteenth National Conference on Artificial Intelligence', pp. 271–278.
- Canuto, S. A., Resende, M. G. C. & Ribeiro, C. C. (2001), 'Local search with perturbations for the prize-collecting steiner tree problem in graphs', *Networks* **38**, 50–58.
- do Nascimento, J. M. (2002), Hybrid computational tools for the optimization of the production of petroleum in deep waters, Master's thesis, Institute of Computing, University of Campinas.

- Feo, T. A. & Bard, J. F. (1989), 'Flight scheduling and maintenance base planning', *Management Science* **35**, 1415–1432.
- Feo, T. A., Bard, J. & Holland, S. (1995), 'Facility-wide planning and scheduling of printed wiring board assembly', *Operations Research* **43**, 219–230.
- Feo, T. A. & Resende, M. G. C. (1995), 'Greedy randomized adaptive search procedures', *Journal of Global Optimization* **6**, 109–133.
- Festa, P. & Resende, M. (2002), 'GRASP: An annotated bibliography', *Essays And Surveys In Metaheuristics - Kluwer Academic Publishers*.
- Fleurent, C. & Glover, F. (1999), 'Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory', *INFORMS Journal on Computing* **11**, 198–204.
- Glover, F., Laguna, M. & Mart, R. (2000), Fundamentals of scatter search and path relinking, in M. Laguna & J. L. Gonzales-Velarde, eds, 'Control and Cybernetics', Vol. 39, Kluwer, pp. 653–684.
- Glover, F. & Laguna, M. (1997), *Tabu Search*, Kluwer Academic Publishers, Norwell, Massachusetts.
- Glover, F. (1996), Tabu search and adaptive memory programming — advances, applications and challenges, in R. S. Barr, R. V. Helgason & J. L. Kennington, eds, 'Interfaces in Computer Science and Operations Research', Kluwer, pp. 1–75.
- Glover, F. (2000), Multi-start and strategic oscillation methods — principles to exploit adaptive memory, in M. Laguna & J. L. Gonzales-Velarde, eds, 'Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research', Kluwer, pp. 1–24.
- Harvey, W. D. & Ginsberg, M. L. (1995), 'Limited discrepancy search', *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* **1**, 607–613.
- ILOG (1999), *ILOG Solver 4.4 Reference Manual*, ILOG.
- Marriott, K. & Stuckey, P. J. (1998), *Programming with Constraints: An introduction*, MIT Press, Cambridge, Massachusetts.
- Pereira, R. A., Moura, A. V. & de Souza, C. C. (2005), Comparative experiments with GRASP and constraint programming for the oil well drilling problem, in S. E. Nikoletseas, ed., 'Experimental and Efficient Algorithms - WEA 2005, LNCS 3503', Springer, pp. 328–340.
- Petrobras (1999), 'ORCA'. A constraint programming solution to the WDRDP developed in-house by Petrobras. No known publications in the open literature.
- Resende, M. G. C. & Ribeiro, C. C. (2002), Greedy randomized adaptive search procedure, in F. Glover & G. Kochenberger, eds, 'Handbook of Metaheuristics', Kluwer, pp. 219–249.

- Resende, M. G. C. & Ribeiro, C. C. (2003), ‘GRASP with path-relinking for private virtual circuit routing’, *Networks* **41**, 104–114.
- Roy, B. & Sussmann, B. (1964), Les problmes d’ordonnancement avec contraintes disjonctives, in ‘Note DS No 9 bis’, SEMA, Paris.
- Yannakakis, M. (1997), Computational complexity, in E. H. L. Aarts & J. K. Lenstra, eds, ‘Local Search in Combinatorial Optimization’, John Wiley & Sons, Chichester, pp. 19–55.

## 5 Tables

Instance	1	2	3	4	5	6	7	8	9	10	11	12
# wells	29	22	29	29	17	22	22	29	29	22	29	22
# activities	98	107	98	98	111	107	128	98	98	107	98	107
# boats	1	1	2	1	1	2	1	1	1	2	1	2
# derricks	3	2	3	3	2	2	3	3	3	2	3	2
# C7	0	0	0	0	0	0	0	0	0	0	0	0
Avg. Outf. (barrels)	2321	338	3026	3026	2320	3026	3026	2671	3026	3026	2671	3026
Avg. Act. Dur. (days)	27	21	19	21	27	21	27	27	21	27	27	21

Instance	13	14	15	16	17	18	19	20	21	22
# wells	22	22	22	22	22	22	22	22	29	65
# activities	107	107	107	107	107	107	107	107	98	338
# boats	2	2	2	1	2	2	2	2	1	1
# derricks	2	2	2	2	2	2	2	2	3	2
# C7	1	1	1	1	1	1	1	1	2	2
Avg. Outf.(barrels)	3026	3026	3026	3026	3026	3026	3026	3026	2321	4028
Avg. Act. Dur. (days)	21	21	21	21	21	21	21	21	27	12

Table 1: Tested Instances.

Techniques	C.P.	GRASP	POP	Bias	Int	P-R	SLS	BIR
ORCA	•							
G1		•						
G2		•		•				
G3		•	•	•	•			
G4		•				•		
G5		•					•	
G6		•						•
G7		•	•					
G8		•	•				•	
G9		•	•					•
G10		•	•	•				
G11		•	•	•			•	
G12		•	•	•			•	
G13		•	•	•	•		•	
G14		•	•	•	•	•	•	

C.P.	Constraint Programming
GRASP	Greedy Randomized Adaptive Search Procedure
POP	Proximate Optimality Principle
Bias	Bias Function (Square Root)
Int	Intensification
P-R	Path-Re-linking
SLS	Selective Local Search
BIR	BIR

Table 2: Solvers

Criterion	1	2	3	4	5
ORCA	0,00	0,00	9,588235	23,47059	2
G1	97,89	96,90	7,682353	10,47059	7
G2	97,86	96,46	8,035294	11,11765	6
G3	98,31	96,85	7,470588	10,23529	6
G4	97,69	96,45	8	11,41176	6
G5	97,89	97,01	6,823529	9,176471	7
G6	97,91	97,23	4,623529	8,235294	7
G7	99,61	99,95	<b>2,941176</b>	<b>4,352941</b>	8
G8	99,61	99,55	4,352941	5,764706	8
G9	99,61	99,92	3,470588	5,294118	8
G10	99,99	99,79	4,411765	6,588235	7
G11	99,99	99,67	4,294118	6,411765	<b>9</b>
G12	99,99	99,66	4,588235	6,941176	7
G13	<b>100,00</b>	99,69	4,058824	6	7
G14	99,81	<b>100,00</b>	<b>2,941176</b>	4,764706	<b>9</b>

Table 3: Values obtained by the Solvers in the Comparative Criteria

Criterion	1	2	3	4	5
	G13	G14	G7-G14	G7	G11-G14
G10-G11-G12	G7	G9	G14		G7-G8-G9
G14	G9	G13	G9	G1-G5-G6-G10-G12-G13	
G7-G8-G9	G10	G11	G8		G2-G3-G4
G3	G13	G8	G13		ORCA
G6	G11	G10	G11		
G1-G5	G12	G12	G10		
G2	G8	G6	G12		
G4	G6	G5	G6		
ORCA	G5	G3	G5		
	G1	G1	G3		
	G3	G4	G1		
	G2	G2	G2		
	G4	ORCA	G4		
	ORCA		ORCA		

Table 4: Rank of the Solvers

Analysis of Criteria	R1	R2	Average R1-R2
ORCA	11,8	15	13,4
G1	8,8	10,4	9,6
G2	10,2	13	11,6
G3	8,4	11	9,7
G4	10,6	13,4	12
G5	7,8	9,4	8,6
G6	7	8,6	7,8
G7	2	2,6	2,3
G8	4,6	5,4	5
G9	2,8	3,6	3,2
G10	4,4	5,2	4,8
G11	3,8	4	3,9
G12	5,4	6,2	5,8
G13	3,4	4,2	3,8
G14	<b>1,6</b>	<b>2</b>	<b>1,8</b>

Table 5: Analysis of the Rank of the Solvers

Solver
G14
G7
G9
G13
G11
G10
G8
G12
G6
G5
G1
G3
G2
G4
ORCA

Table 6: Rank of the Best Solvers

## 6 Figures

```
1: procedure GRASP(ListSize, MaxIter, Seed)
2: for k = 1 to MaxIter do
3:   Solution  $\leftarrow$  Construct_Solution(ListSize, Seed);
4:   Solution  $\leftarrow$  Local_Search(Solution);
5:   Update_Solution(Solution, Best_Solution_Found);
6: end for
7: return Best_Solution_Found;
8: end GRASP
```

Figure 1: The GRASP Meta-heuristics.

```

1: procedure Construct_Solution(ListSize, Seed)
2:   Solution  $\leftarrow$  0;
3:   Evaluate the incremental costs of the candidate elements;
4:   while Solution is not a complete solution do
5:     Build the restricted candidate list, RCL(ListSize);
6:     Select an element s from the RCL at random;
7:     Solution  $\leftarrow$  Solution  $\cup$  {s};
8:     Reevaluate the incremental costs;
9:   end while
10:  return Solution;
11: end Construct_Solution

```

Figure 2: The Construction Phase of GRASP.

```

1: procedure searchForSolutions()
2: while Time limit is not reached do
3:   if Time limit with no improvements is reached then
4:     RCLSize  $\leftarrow$  RCLSize + 1;
5:   end if
6:   Solution  $\leftarrow$  greedyRandomSolution();
7:   if Production of Solution  $\geq$  least production of the elite solutions OR
      Solution differs from elite solutions then
8:     doTwoExchangeLocalSearch(Solution);
9:   end if
10:  doPathRelinking(Solution);
11:  if Solution is an elite solution then
12:    Insert Solution into the elite set;
13:  end if
14:  if Production of Solution is the highest so far then
15:    BestSolution  $\leftarrow$  Solution;
16:  end if
17: end while
18: return BestSolution;
19: end searchForSolutions

```

Figure 3: The new GRASP solver implementation.

```

1: procedure greedyRandomSolution()
2: //Schedule first production wells and injection wells that precede production wells;
3: while Not all production wells were scheduled in Solution do
4:   doPOP(Solution);
5:   for Well in List of wells availables do
6:     if A start date of any activity of Well has very small domain then
7:       schedule(Solution, Well);
8:     end if
9:   end for
10:  WellsToSchedule  $\leftarrow$  Wells availables that produces or with producer successors;
11:  Sort WellsToSchedule decreasingly by estimated production;
12:  RCL  $\leftarrow$  First 'RCLSize' wells from WellsToSchedule;
13:  WellChoosen  $\leftarrow$  Choose well from RCL randomly with bias;
14:  schedule(Solution, WellChoosen);
15:  if Solution does not satisfy constraints then
16:    return empty solution;
17:  end if
18: end while
19: Schedule the remaining wells, the injection wells;
20: if Solution does not satisfy constraints then
21:   return empty solution;
22: else
23:   return Solution;
24: end if
25: end greedyRandomSolution

```

Figure 4: The new Greedy Randomized procedure.

```
1: procedure doPOP(Solution)
2: if % of activities scheduled in Solution is a multiple of the POP frequency
   then
3:   doTwoExchangeLocalSearch(Solution);
4: end if
5: end doPOP
```

Figure 5: The POP procedure.

```

1: procedure doPathRelinking(Solution)
2: EliteSolution  $\leftarrow$  Get a random solution from the elite set;
3: Current  $\leftarrow$  Best solution between Solution and EliteSolution;
4: Guide  $\leftarrow$  Worst solution between Solution and EliteSolution;
5: for Resource in List of resources do
6:   for Well in List of wells consecutively scheduled in Resource in Guide do
7:     //Puts Well in Current in the same position it appears in Guide
8:     Position  $\leftarrow$  Position where Well is scheduled in Resource in Guide;
9:     Well2  $\leftarrow$  Get well scheduled in position Position in Resource in Current;
10:    Swap wells Well and Well2 in Current;
11:    if Production of Current is the highest so far then
12:      BestSolution  $\leftarrow$  Current;
13:    end if
14:  end for
15: end for
16: end doPathRelinking

```

Figure 6: The Path-Relinking procedure.

```

1: procedure doTwoExchangeLocalSearch(Solution)
2: for Well in List of wells scheduled in Solution do
3:   for Well2 in List of wells scheduled in Solution continuing from Well do
4:     Swap wells Well and Well2 in Solution;
5:     if Production of Solution is the highest so far then
6:       BestSolution  $\leftarrow$  Solution;
7:     end if
8:   end for
9: end for
10: end doTwoExchangeLocalSearch

```

Figure 7: The Two-Exchange Local Search procedure.

```
1: procedure schedule(Solution, Well)
2: for Activity in List of activities availables of Well ordered by constraints do
3:   Resource  $\leftarrow$  Resource able to do Activity the earliest, preferring last used in Well;
4:   scheduleInResource(Solution, Activity, Resource);
5:   if Well produces then
6:     Adds the production of Well to the production of Solution;
7:   end if
8: end for
9: end schedule
```

Figure 8: The Schedule procedure.

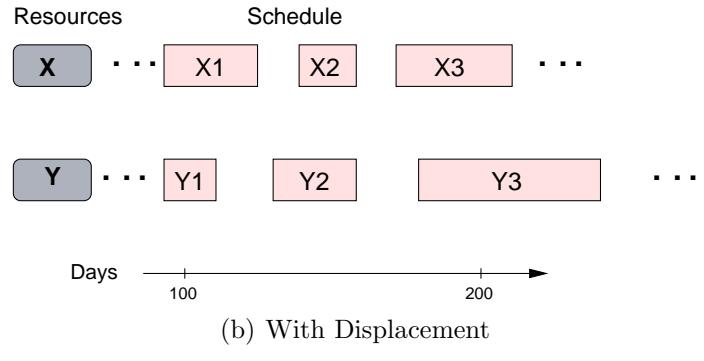
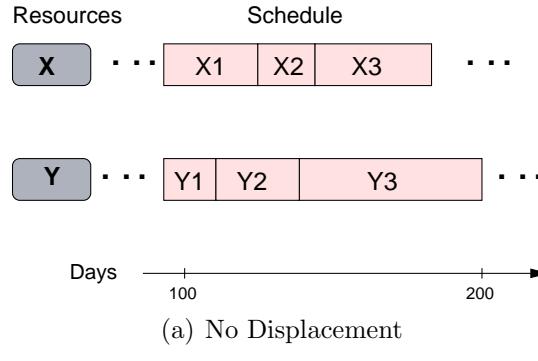


Figure 9: Schedules without and with resource displacement.

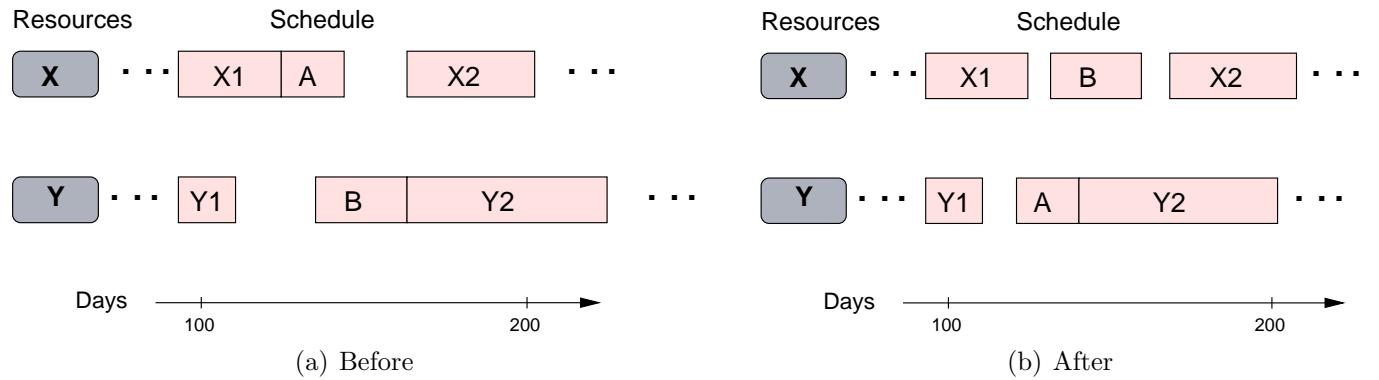
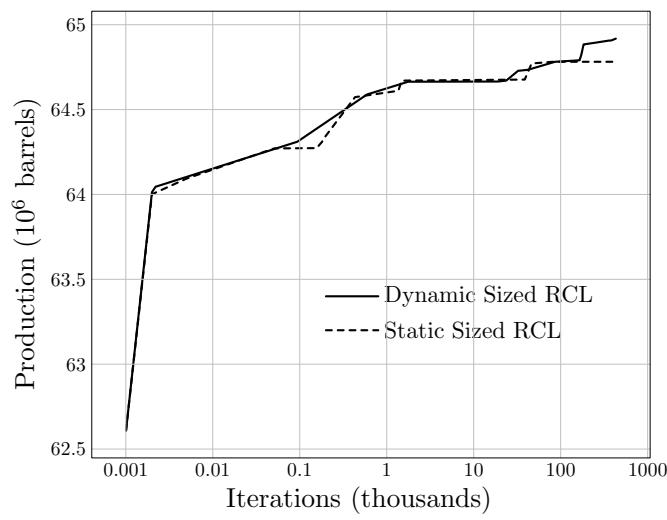
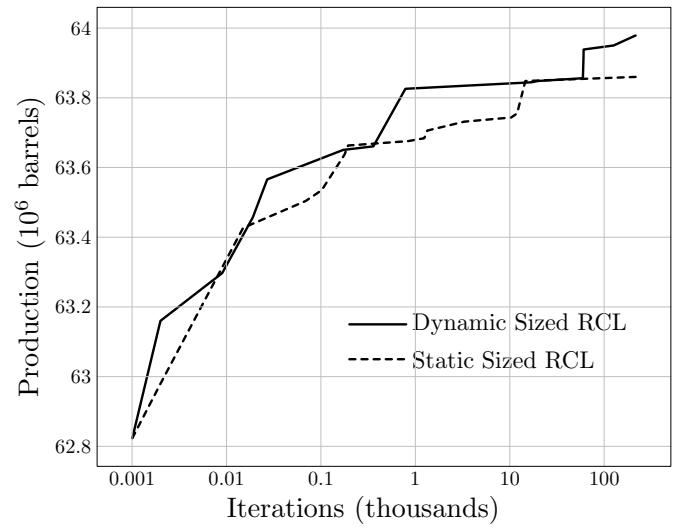


Figure 10: 2-exchange swap.

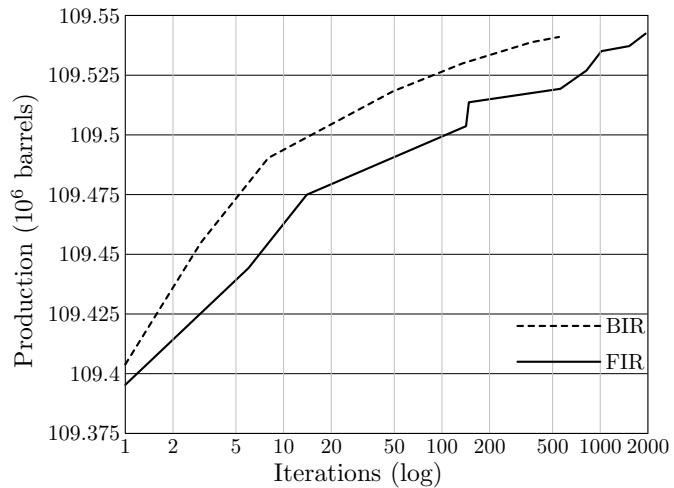


(a) Example 1 – Instance 4

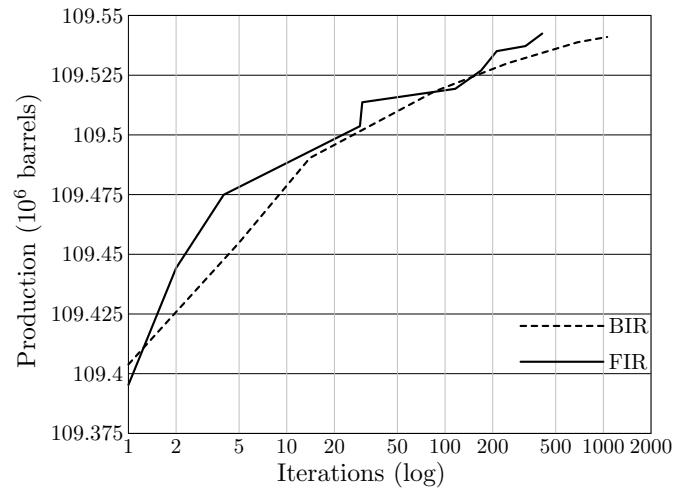


(b) Example 2 – Instance 6

Figure 11: Static Sized RCL Solver x Dynamic Sized RCL Solver.

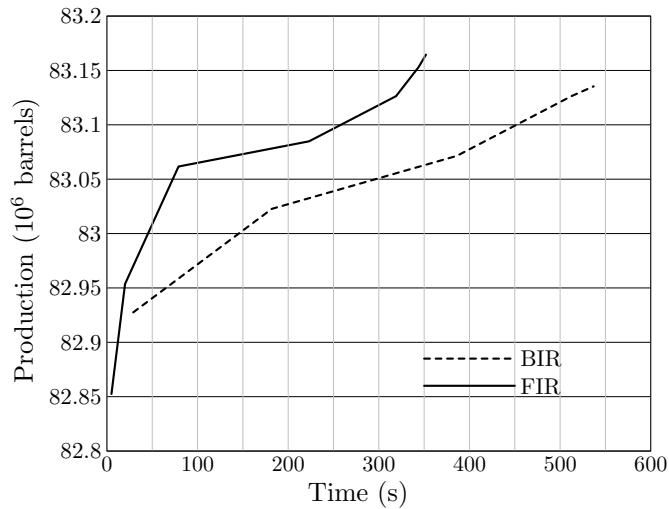


(a) Using instance 21

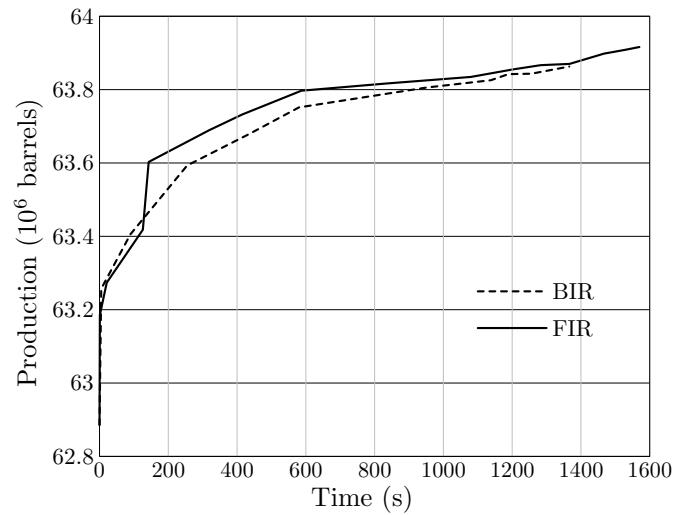


(b) Using instance 21

Figure 12: BIR x FIR: Example 1

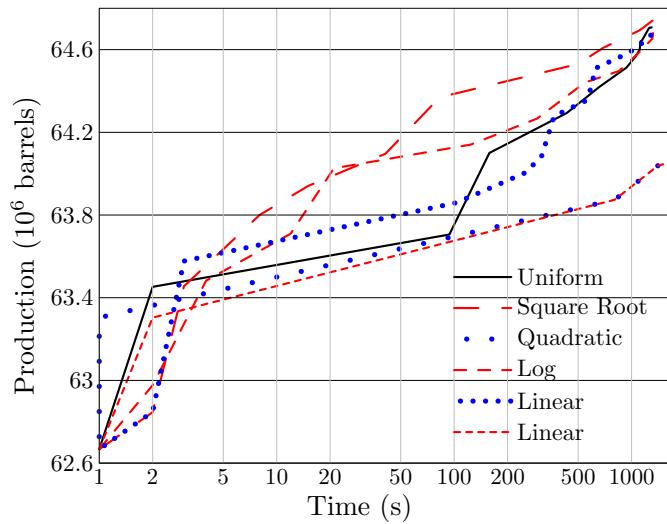


(a) Example 2 – Instance 22

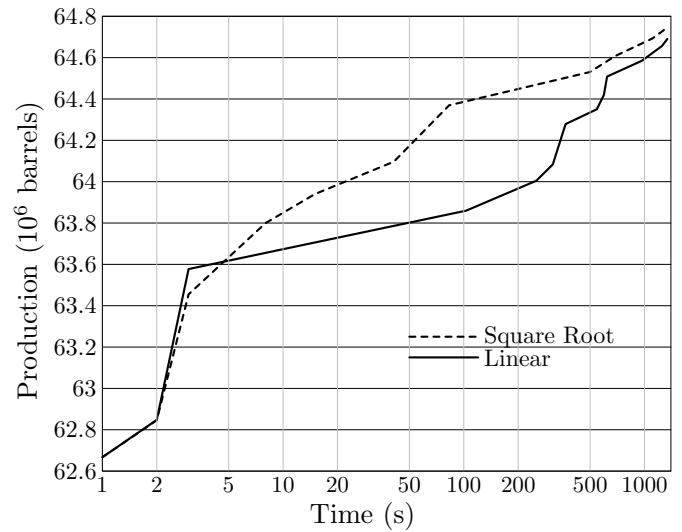


(b) Example 3 – Instance 4

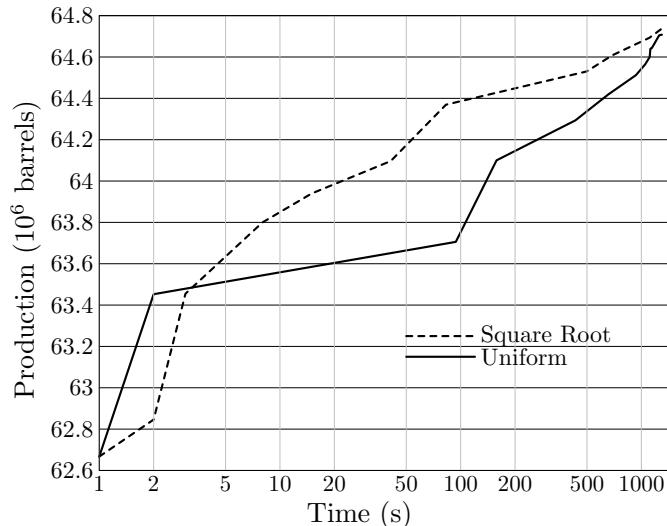
Figure 13: BIR x FIR.



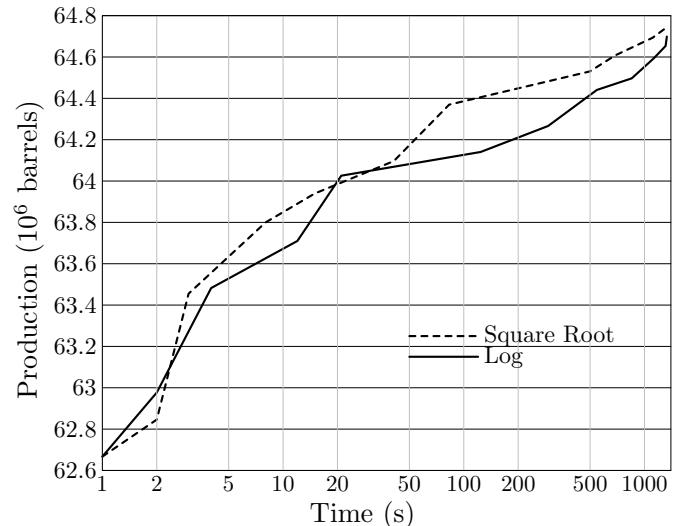
(a) Example 1– Instance 3



(b) Square root X Linear– Instance 3

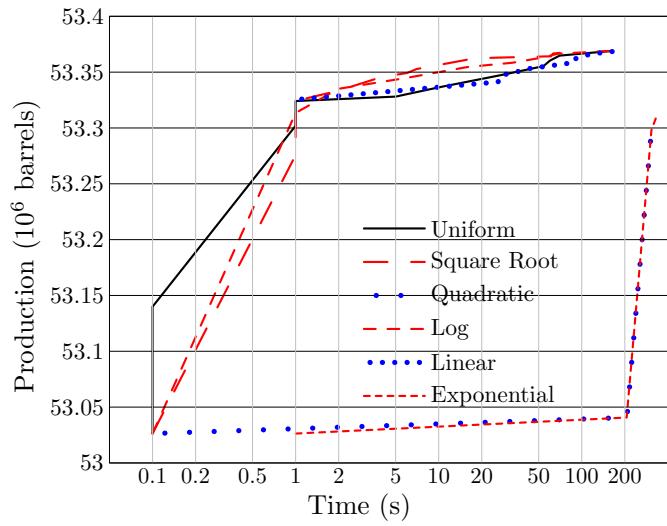


(c) Square root X Uniform – Instance 3

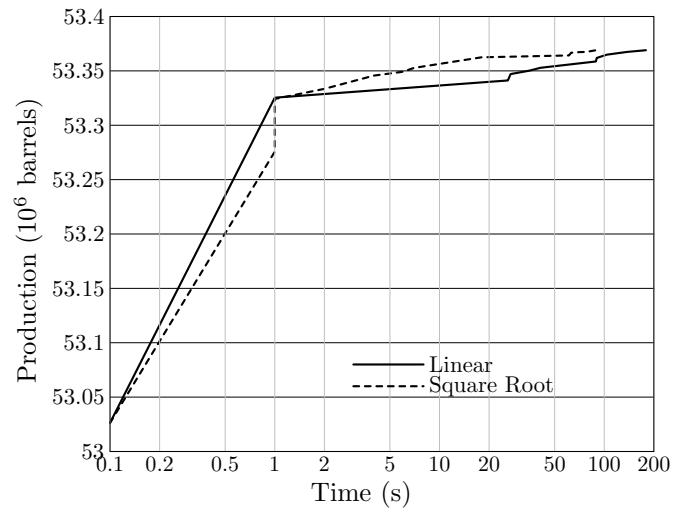


(d) Square root X Log – Instance 3

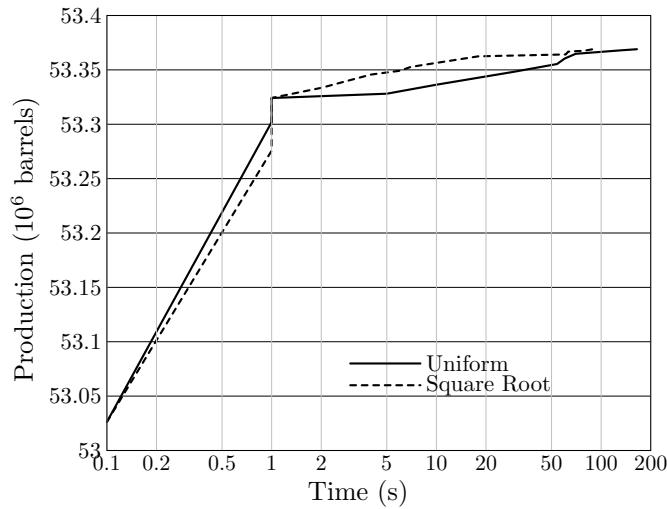
Figure 14: Bias Functions, I.



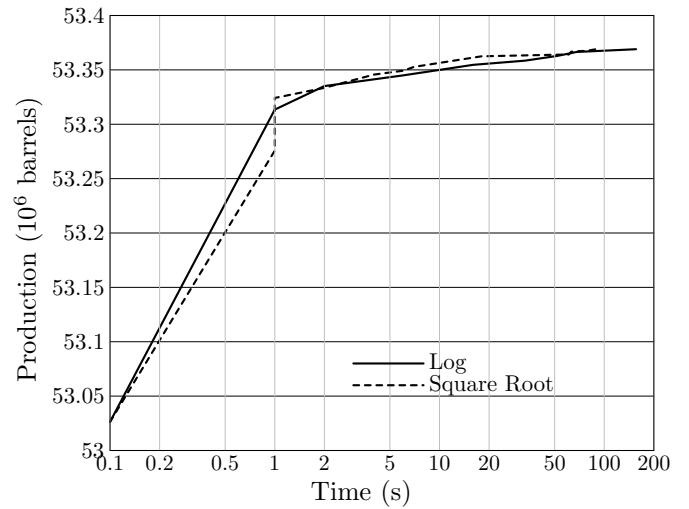
(a) Example 2 – Instance 8



(b) Square root X Linear – Instance 8

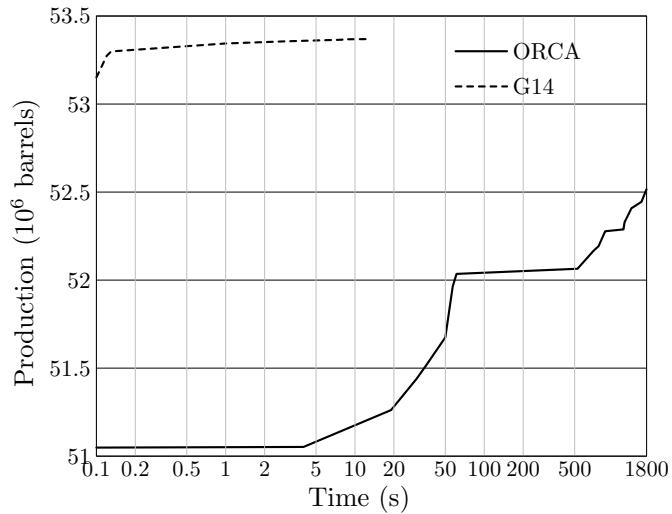


(c) Square root X Uniform – Instance 8

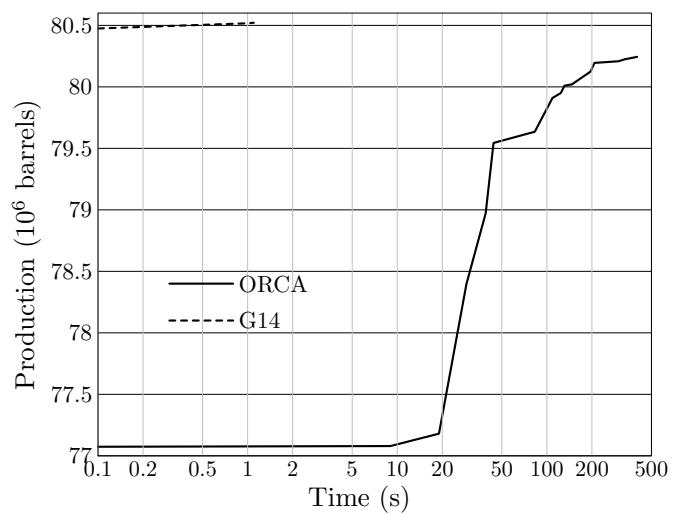


(d) Square root X Log – Instance 8

Figure 15: Bias Functions, II

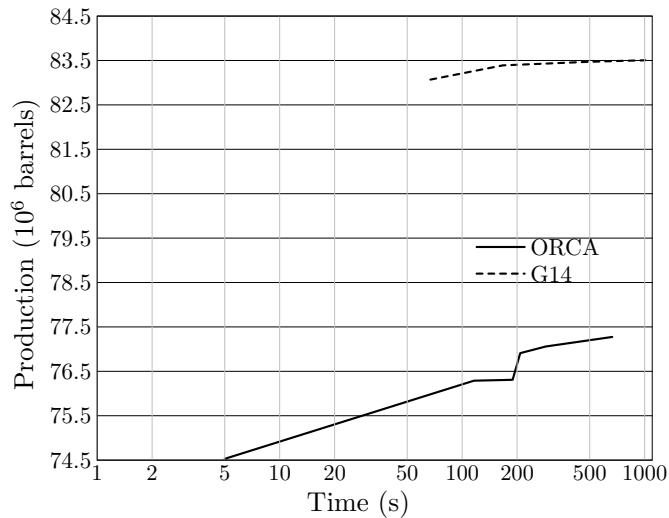


(a) Example 1 – Instance 8

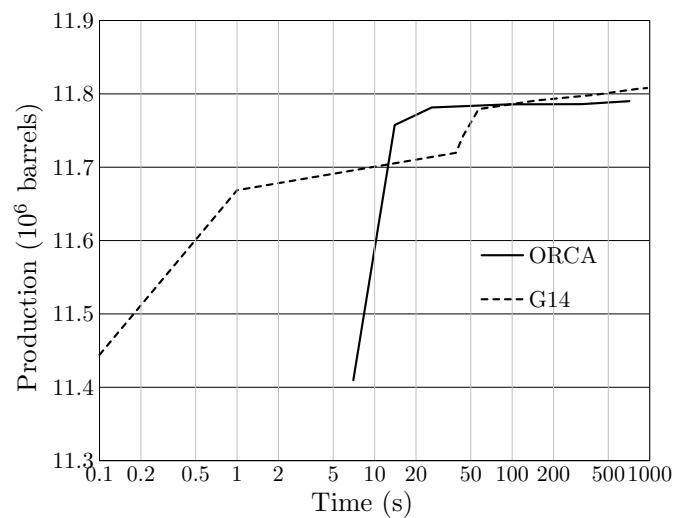


(b) Example 2 – Instance 11

Figure 16: ORCA  $\times$  G14, I



(a) Example 3 – Instance 22



(b) Example 4 – Instance 15

Figure 17: ORCA  $\times$  G14, II

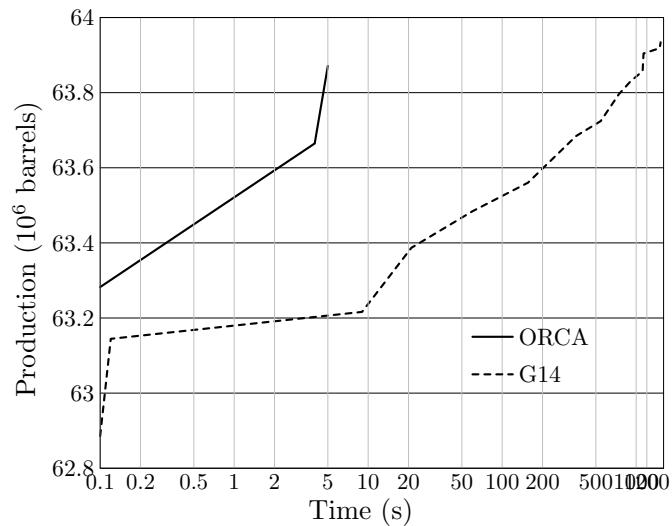


Figure 18: ORCA×G14 – Instance 4