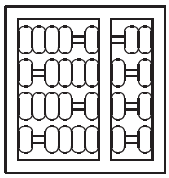


INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Hybrid method to optimize petroleum extration in deep sea waters

J. M. Nascimento *A. V. Moura*
C. C. de Souza

Technical Report - IC-02-08 - Relatório Técnico

September - 2002 - Setembro

Hybrid method to optimize petroleum extration in deep sea waters

Juliana Martins do Nascimento* Arnaldo Vieira Moura†
Cid Carvalho de Souza‡

Abstract

Bacia de Campos is a large area in the sea where Petrobras explores petroleum in deep waters. There are a lot of specific locations in this site that have been determined as promising oil wells. Before the extraction begins, these locations must be fully developed. The objective is to construct a schedule maximize the oil production in a given amount of time, subject to a number of restrictions such as a given precedence relation among the activities, the proper match between resources and activities, and resource routing, among others . We propose a hybrid approach that combines constraint programming (CP) techniques and tabu search in order to solve the problem. At each neighbor, a scheduling problem and a first feasibility test are performed initially, without using CP. Next, CP is used to assign the start time of the activities. Up to 500 activities and 130 oil wells are considered in the instances tested. We used integer linear models to prove that the solutions obtained are less than 9% from an global upper bound. Finally, to estabilish the robuteness of our approach, a sensibility analysis was performed indicating that the technique performs well when solving similar instances.

1 Introduction

Petrobras is one of the world most efficient companies concerning the extraction of petroleum in deep sea waters. *Bacia de Campos* is a large sea area where Petrobras explores petroleum. There are a lot of specific locations in this site that have been determined as promising oil wells. Before the extraction begins, these locations must be fully developed.

*Institute of Computing, University of Campinas, 13081-970 Campinas, SP. Research supported by FAPESP — Fundação de Amparo a Pesquisa do Estado de São Paulo, grant 00/14120-8
†Institute of Computing, University of Campinas, 13081-970 Campinas, SP.
‡Institute of Computing, University of Campinas, 13081-970 Campinas, SP.

Roughly, the development process begins when the well is drilled. After that, a huge metal structure, named *ANM*, must be placed on its top. This structure avoids the spill of oil and has special connections where equipments and pumps can be attached to receive the extracted oil. Next, an oil pipe connects the *ANM* to a *manifold* or directly to a platform on the surface. A *manifold* is a structure that interconnects several oil pipes at the sea bottom. This structure connects to the surface by a single pipe. After these activities are executed, the real oil extraction can begin.

Petrobras is interested in routing the resources and scheduling the activities involved in the development process of wells at *Bacia de Campos*. The objective is to maximize the oil production in a given amount of time. In a typical problem instance, up to 500 activities and 130 promising oil wells are considered. Clearly this is a very large and real important combinatorial optimization problem.

In this paper, we propose a hybrid method that combines constraint programming (CP) techniques and a tabu search heuristic that explores a very large scale neighborhood to solve the problem. The aim in developing hybrid techniques to solve combinatorial optimization problems is to strength the good features of the methods that are being combined to compensate for their weakness, since these problems are generally NP-hard [14].

Recently, a great deal of research has been focused on the integration of CP and metaheuristics [15]. This can bring promising results when solving combinatorial optimization problems [28, 29, 23]. Much effort has also been concentrated on the study of local search algorithms that explore efficiently very large scale neighborhoods, yielding very good results in several problems [13, 1, 2]. These algorithms can visit large neighborhoods in polynomial time. When the exploration of a neighborhood is NP-hard or when a polynomial algorithm is not known, a heuristic is used. The integration cited earlier makes possible the exploration of large neighborhoods in competitive time. As a result, the possibilities to obtain real improvement in the solutions are very high, because with better quality and bigger sizes in a neighborhood, the more efficient metaheuritics tend to perform.

We use CP to help the exploration of a very large scale neighborhood in a tabu search framework. In our approach, CP is not used just to generate the initial solution to the tabu search [27], nor just to verify the feasibility and cost of the neighbors like in [25, 11, 8]. It is also not used to control the entire process needed to visit the neighbors [23]. At each neighbor, the routing problem and a first feasibility test are performed initially, without using CP. Next, CP is used to assign the start time of the activities. A similar approach was used in [7] to solve the job shop problem.

The main objective of this work is to obtain the best solution to the problem under consideration, but comparisons between a hybrid technique and a pure tabu search approach is also appropriate. We want to investigate how well the tabu approaches adapt to this problem. So we also use a pure abu search with a polynomial

neighborhood to solve the problem.

As there are no previous results for the instances used in this paper, an effort to calculate strong bounds is undertaken. Finally, we also want to establish the robustness of the methods, so a sensibility analysis is performed.

The remainder of this paper is organized as follows. Section 2 describes the problem and section 3 describes the instance provided by Petrobras and an instance generator. Section 4 presents the methods applied to solve the problem. Section 5 discusses the calculation of upper bounds while section 6 shows the computational results obtained. Section 7 describes the sensibility analysis. Finally, a conclusion is presented in section 8.

2 The problem

Bacia de Campos is a large sea area (oil field) where Petrobras explores petroleum. There are a lot of specific locations there that have been determined as promising oil wells. Before extraction begins, these locations must be fully developed. This process involves a number of different engineering activities, such as drilling activities, connection activities and extraction activities. Some of the wells may be in different stages of the development process.

Given a set of wells and the correspondent activities to be performed in each well and a set of resources, like boats and derricks, the objective is to determine a scheduling and a routing of the activities into the resources, satisfying certain constraints and maximizing the oil production in a given period of time. This period of time is called the *horizon*. It is important to note that activities can be scheduled after the horizon. Preemption in not allowed.

The most relevant constraints to this problem are:

1. **Technological Precedence**: this constraint defines a partial order between activities. If activity A must be performed before activity B, there is a technological precedence from A to B. This kind of precedence applies between activities that belong to the same oil well.

2. **Date Constraint**: an activity may have a fixed **date** to begin and to end. Another date constraint is that an activity must end before or begin later than a specified date with or without *lag time*.

3. **Activity Features**: the execution of an activity may require an specific type of resource. The resource must be able to operate at the appropriate depth and it should also have the type of equipments required to execute the activity.

4. **Resource Availability**: resources can only perform one activity at a time. Resources can also become unavailable during a certain period due to maintenance or due to contract constraints.

5. **Oil Well Availability**: each well can have only one of its associated activity executed at a time. Even if there is no precedence constraint between two of its activities.

6. **Area Constraint**: On some wells that are close to each other, for safety reasons and depending on the type of the resources needed, overlapping execution of activities should be avoided.

7. **Efficiency factor**: the processing time of an activity may change due to an efficiency factor associated to the resource that will execute it.

In this paper, only constraints $1, 3, 4, 5$ are taken into account. These are the main constraints. The real data provided by Petrobras only cover these constraints. But other constraints, can be incorporated in the hybrid model without much effort, since one of the features of CP is to allow the addition of new constraints easily. In the tabu search approach, there is a stronger connection between the constraints and the neighborhood, which makes it harder to introduce modifications in the model.

The production of petroleum is calculated as follows: each oil well has an associated outflow per day and a last activity that is responsible to turn it into a productor well. When this activity is finished, the well is ready to produce oil. The total oil production is given adding the values obtained by multiplying the daily outflow of each well by the number of days between the beginning of its production and the horizon. Wells that begin their production after the horizon are not considered in the calculation.

3 The real instance and the instance generator

Petrobras has provided one real instance of the problem, which we call the real instance. In order to access the robustness of our techniques, an instance generator was implemented. The instance generator introduces small random changes in the petroleum instances.

In this section, the features of the real instance and how the instance maker was created will be presented.

The partial order of the activities associated with each well of the real instance follows a pattern. The pattern is determined by the type of the activities and by the precedence relationship between them. These patterns are depicted in figure 1. In this figure, the nodes are the activities and the arcs are the precedence relationship between them. There are different patterns with the same graph. This happens due

to the fact that these patterns have the same precedence relationship between their activities, but the type of their activities differs. For example, patterns 17, 19 and 20 present a total order between their activities, but the activities type of these patterns are different. Figure 1 also shows that the order of the activities on the wells is not always total. This fact will be important to establish that our neighborhoods are *very large*.

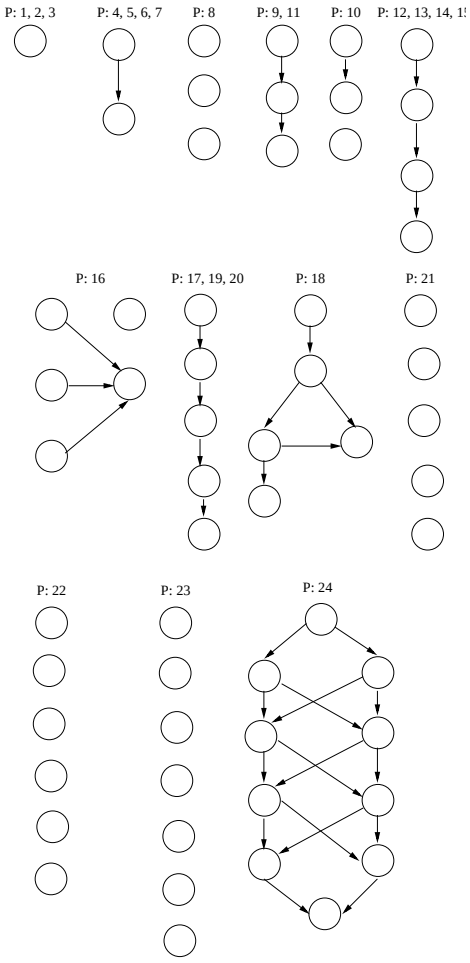


Figure 1: Precedence relationship between the activities of each pattern

Table 1 presents some numerical data associated with each pattern. Note that the frequency of occurrence of each pattern is not uniform. For example, just a well presents pattern 3, while 42 wells present pattern 19.

There are two type of resources: boats and derricks. In the real instance, there are 3 boats and 5 derricks available. Whenever an activity requires a resource, any of its kind can be allocated to it. That is, there is no distinction or restriction to

Pattern	#Oil Wells	%Oil Wells	#Activities	Pattern	#Oil Wells	%Oil Wells	#Activities
1	1	0,8772	1	13	8	7,0175	4
2	1	0,8772	1	14	2	1,7544	4
3	1	0,8772	1	15	1	0,8772	4
4	5	4,3860	2	16	1	0,8772	5
5	1	0,8772	2	17	12	10,5263	5
6	1	0,8772	2	18	4	3,5088	5
7	1	0,8772	2	19	42	36,8421	5
8	1	0,8772	3	20	1	0,8772	5
9	21	18,4211	3	21	2	1,7544	5
10	1	0,8772	3	22	1	0,8772	6
11	1	0,8772	3	23	1	0,8772	7
12	3	2,6316	4	24	1	0,8772	12

Table 1: Pattern data - Real Instance

use any of the 3 boats, when an activity requires a boat. The same is true for the derricks. The type of each activity determines the kind of resource needed to execute it.

Table 2 shows the number of oil wells, activities, boats, derricks and patterns in the real instance. Table 3 shows some numerical data about the activities processing time in the real instance.

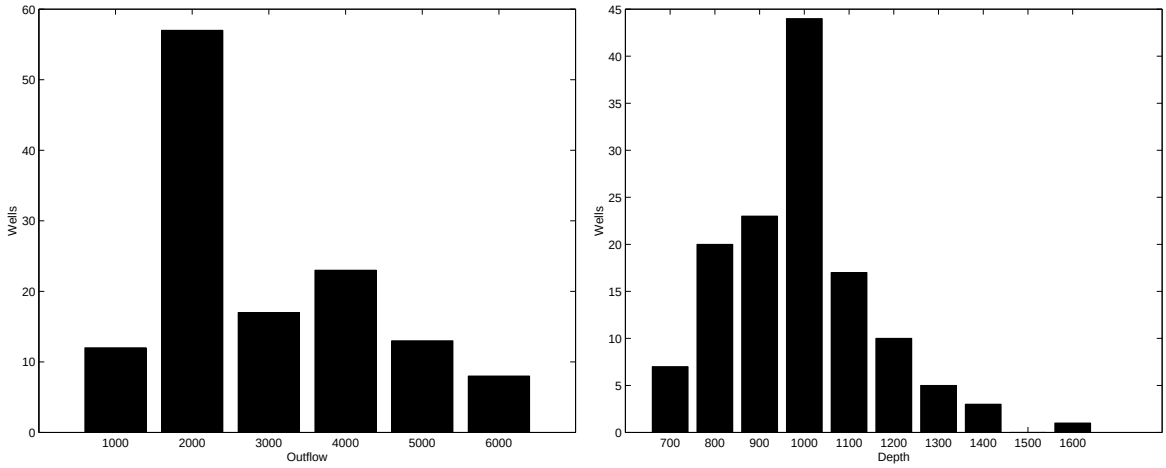
#Oil Wells	#Activities	#Boats	#Derricks	#Patterns
130	482	3	5	24

Table 2: Quantitative data - Real instance

Mean	Median	Mode	Interval	Min	Max	Sum
17	12	1	129	1	130	8195

Table 3: Numerical data about the activities processing time - Real instance

Figure 2 shows the histograms for the outflow and the depth of the wells. The unity of measure for the depth is meters and for the outflow it is an internal measure unit used by Petrobras. We will always use this internal unity of measure in all figures and tables. This figure also shows that values of the outflows vary a lot. This feature makes our problem harder, because there are two factors to be considered and to be balanced: the outflow of each well and the minimum time required to execute all of its activities until the well can produce oil. Another relevant information that figure 2 brings is the fact that the maximum depth is 1600 meters. As all resources of the real instance are able to operate on this depth, we conclude that the depth of the wells do not restrict the choice of the resources needed to perform the activities.



(a) Outflow histogram (b) Depth histogram

Figure 2: Histograms - Real instance

The Instance Generator

The pattern of the wells of the generated instances was determined by the frequency of occurrence of each pattern on the real instance. The processing time (duration) of each generated activity was randomly chosen among the activities processing time that have the same type as the generated activity on the real instance. The outflow and the depth associated with each generated well were established following a uniform distribution between the minimum and maximum values of the outflow and depth, respectively, of the real instance wells.

As mentioned before, the resources of the real instance have all the necessary features and are able to operate on depths greater than the depths associated to the wells. So the generated resources also have all the possible features and are able to operate in depths greater than the generated ones.

The number of oil wells, boats and derricks are the input parameters for the program that generates the instances. The number of activities of each well is fixed once its pattern is determined.

The notation used for the instance names are standardized as follows: the name contains an identifier, the number of oil wells, derricks and boats. For example, the name 2W112S5B3 means that instance 2 has 112 oil wells, 5 derricks and 3 boats.

When the instance generator was applied to produce an instance with 112 wells, 5 derricks and 3 boats, the result obtained appears in tables 4, 5 and 6 and in figure 3. This instance was named 2W112S5B3. Comparing with tables 2, 3 and 1,

respectively, it can be seen that the results are very similar, except for some random variations introduced by the instance generator. In tables 1 and 6 pattern 19 is the most common one representing about 36% of the wells. Also, tables 3 and 5, show that the mean processing time is about 17 days.

#Oil Wells	#Activities	#Boats	#Derricks	#Patterns
112	464	3	5	24

Table 4: Quantitative data - Instance 2W112S5B3

Mean	Median	Mode	Interval	Min	Max	Sum
16.65	12	1	129	1	130	7726

Table 5: Numerical data about the activities processing time - Instance 2W112S5B3

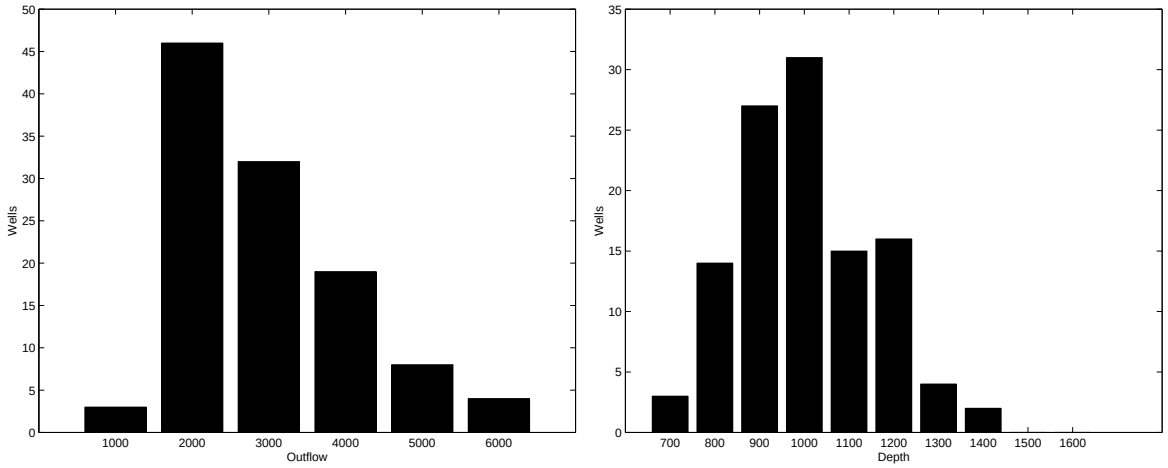
Pattern	#Oil Wells	%Oil Wells	#Activities	Pattern	#Oil Wells	%Oil Wells	#Activities
1	1	0,892857143	1	13	3	2,678571429	4
2	0	0	1	14	5	4,464285714	4
3	0	0	1	15	0	0	4
4	5	4,464285714	2	16	0	0	5
5	0	0	2	17	7	6,25	5
6	4	3,571428571	2	18	3	2,678571429	5
7	0	0	2	19	40	35,71428571	5
8	0	0	3	20	4	3,571428571	5
9	28	25	3	21	1	0,892857143	5
10	3	2,678571429	3	22	2	1,785714286	6
11	2	1,785714286	3	23	1	0,892857143	7
12	2	1,785714286	4	24	1	0,892857143	12

Table 6: Pattern data - Instance 2W112S5B3

In this paper, four instances will be considered. The real instance 1W130S5B3 and instances 2W112S4B3, 3W95S5B3 and 4W130S5B3 generated by the instance generator. The first generated instance shows a small number of wells and resources. The second one maintains the number of resources and considers a smaller number of wells. The last one was generated using the same number of wells and resources as the original instance.

4 Techniques

We propose a hybrid method that combines constraint programming techniques and a tabu search heuristic that explores a very large scale neighborhood. We use CP to



(a) Outflow histogram (b) Depth histogram

Figure 3: Histograms - Instance 2W112S5B3

help the exploration of a very large scale neighborhood in a tabu search framework. At each neighbor, the routing problem and a first feasibility test are performed initially. Next, CP is used to assign the start time of the activities.

This section describes, in details, how each of these techniques are applied and combined to solve the oil problem. Subsection 4.1 shows how the initial solutions were generated. Subsection 4.2 describes the neighborhoods used in this paper. It also presents a proof that finding the start time of the activities in each neighbor, while maximizing the total production is an NP-hard problem . Subsection 4.3 details how CP is used to assign start times. Finally, subsection 4.4 describes some particularities of tabu search when each neighborhood of section 4.2 is used. It also describes a pure tabu search approach.

4.1 Initial Solution

The tabu search method requires a feasible initial solution. This subsection will describe four techniques that were used to generate initial solutions to the problem.

The first two techniques are based on constraint programming and were implemented using the *ILOG Scheduler* and the *ILOG Solver*[1]. We will call them **CP1** and **CP2**, respectively.

The variables and constraints used on **CP1** and **CP2** are the same. A set of variables is associated to the resources and another set is associated to the start

[1]Softwares from the ILOG suite. http://www.ilog.com

times. As described in section 2, only constraints $1, 3, 4, 5$ were enforced. We did not spend much effort on refining the constraint programming models, since the initial solution were not our main objective.

The last two techniques are greedy heuristics. We will call them **H1** and **H2**. These two heuristics proved much faster than the previous techniques and the oil production obtained using **H1** proved to be the highest among all techniques for all considered instances, as will be discussed later.

- **CP1**

  The relevant point about this technique is the strategy used to instantiate the variables. The variables that represent the resources are instantiated first and then the values of the start time variables are determined.

  The order used for selecting the next variable for labelling is the same for the two sets of variables. It is based on the following rule: the variable that represents the activity with the greatest number of direct successors, greatest total number of successors and longest duration is chosen first.

  The values assigned are chosen by internal algorithms of the *ILOG Scheduler*. Due to limitations of the software, in this model, all activities must be executed before the horizon.

- **CP2**

  As in the previous technique, the relevant information is the instantiation strategy. And again the resources are instantiated before the start time and the order is the same for both sets of variables.

  The selection order is based on the following priority rule: the estimated yield of the correspondent well of each activity is added to the number of its direct successors. The variable that represents the activity with the greatest sum is selected first. The yield of each well is estimated considering that all of its activities are executed sequentially without interruptions, and its production starts when the last activity of the sequence is finished. As the yield values are either equal or differ by a great amount, the number of direct successors is used to break ties.

  The value chosen depends on the type of the variable. The value designated to variables that represent resources corresponds to that resource that matches the activity type and which has the least number of activities assigned to it. The value assigned to the start time variables is the least value of their present domain.

- **H1**

  This heuristic tries to finish first the wells that have the greatest outflow or the wells that have the least remaining time.

  To reach this objective, an available resource is immediately allocated to an activity that is feasible to be executed by this resource at this moment. If there is more than one such feasible activity, the activity chosen is:

  1. The activity that has the best value for

     $$(horizon - (actualTime + remainingTime)) \times outflow$$

     where $horizon$ is the time limit specified, $actualTime$ is the start time for the activity, $remainingTime$ is the total processing time for all remaining activities in the correspondent well, and $outflow$ is the outflow of the associated well;
  2. The activity with the greatest number of total successors;
  3. The activity with the longest duration.

  The second and third rules are used to break ties.

  Initially, the algorithm follows these rules and assigns activities to resources until all of them are busy. Every time an activity is terminated, its resource is released. At this moment all its successor activities become available for scheduling. Another activity is selected, following the priority rules described earlier, and the cycle repeats. The algorithm terminates when all start times are assigned.

- **H2**

  This heuristic, as in the first two techniques, first allocates the resources and then assigns the start times.

  For each activity, the heuristic verifies if there are activities on the same well that are already allocated to resources. If this is true and if among the resources allocated to the activities of the same well, there are resources able to execute the considered activity, the heuristic chooses randomly one of these resources. If one of the last two conditions is false the heuristic chooses randomly a resource among all possible resources.

  To determine the start times, a data structure is used to store all available activities, i.e., the activities whose predecessors have already been scheduled. Initially, this data structure contains all activities without predecessors. At

each iteration, an activity is chosen from this data structure, following the priority rule of **H1**. The *actualTime* of each activity is the earliest possible time permitted by the actual scheduled resource and the other executing activities that belong to the same well. The algorithm finishes when all activities are scheduled. i.e., the data structure is empty.

Table 7 shows the production obtained for the initial solution of the real and generated instances that are considered. The horizon was 1500 days for all instances. The computational time of techniques **CP1** and **CP2** was less than 200 seconds and less than 60 seconds, respectively, for all four instances. The computational time of both **H1** and **H2** was less than a second. Cells without value means that no solution was found in 200 seconds[2].

Instance	CP1	CP2	H1	H2
1W130S5B3(real)	163.6	207.4	246.2	206.5
2W112S4B3	-	184.3	220.3	173.3
3W95S5B3	155.6	187.3	225.4	185.7
4W130S5B3	-	218.9	276.5	218.3

Table 7: Initial solutions

As **H1** obtained the best results for all instances, it will be used to generate the initial solution of our instances. In order to test the behavior of our techniques when a poorer initial solution is considered, we will also use **H2** as an algorithm to obtain initial solutions, since it is much faster then **CP1** and **CP2**. Whenever **H2** is used for initial solutions, this fact will be clearly stated.

4.2 Neighborhoods

It is easy to note that our problem and the Flexible Job Shop problem (FJS) [20] and the Job Shop Schedule with Multi Purpose Machines problem (MPM) [6] are very similar. In each case, an oil well corresponds to a job and the set of activities to be performed in each well corresponds to the set of operations associated with each job. The precedence constraints between the activities corresponds to the order between the operations of each job. If the FJS is considered, there is a total order among the operations of the same job. In our problem, this is not necessarily true. In some wells, the precedence restrictions only give a partial order. The boats and derricks are the machines that execute the operations. Given these similarities, our neighborhoods are inspired by ideas already used to solve these problems [10, 9, 17, 20].

[2]All computational times refer to a PC plataform with 1 GHz processor and 1 Gb of memory

Disjunctive Graphs

We use the disjunctive graph model (DG) to represent our neighbors. Disjunctive graphs were created by Roy and Sussmann [26] to model and solve job shop problems (JSP). Balas [5] was another author to extensively explore the proprieties of this graph.

Figure 4 shows a disjunctive graph for a JSP with 4 jobs and 3 machines. In this figure, complete lines are called *conjunctive arcs*. They represent the precedence constraint among the activities. The dashed lines are the *disjunctive arcs*. They represent the order of the activities in the resources. Every time two activities are assigned to the same machine, a new bi-directed disjunctive arc is added between these two activities. Whenever the order between two activities is established, the correspondent bi-directed disjunctive arc is oriented, representing this order. It is important to note that when there is no precedence constraint between two operations of the same job, or in our case, two activities of the same well, bi-directed disjunctive arcs are added between these activities, since the order of execution is not known.

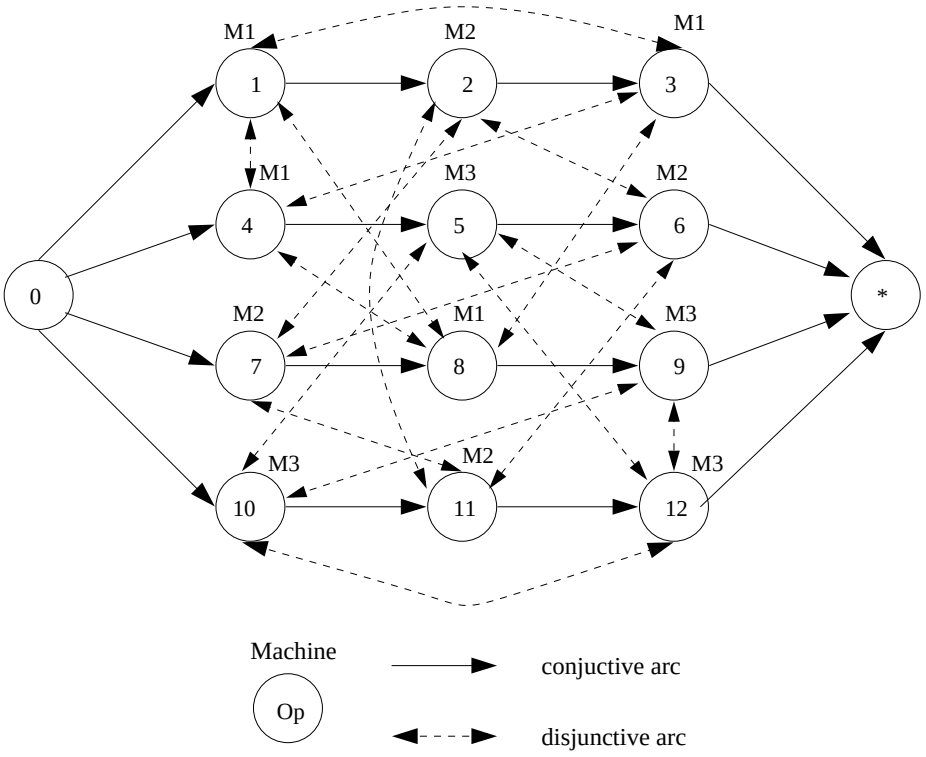


Figure 4: Disjunctive Graph

The solutions and neighbors of this paper are represented using this graph. If after the addition and orientation of the disjunctive arcs an acyclic graph is obtained, the

solution/neighbor is considered feasible and the activities start times can be assigned. If the arcs establish a total order among the activities, the start times can be obtained in polynomial time by running a topological sort algorithm [12]. This situation will be addressed on the following paragraphs. If a total order is not established, the problem of assigning the start times to the activities maximizing the production is NP-Hard. Subsection 4.2.2 presents a reduction that proves this fact.

Regular Measure of Performance - Dominant Sets - Types of Schedules

Baker [4] defines regular measure of performance and dominant sets for it. He also defines the types of schedule: *semiactive, active* and *nondelay*. These definitions are important because they explain why only the earliest possible start time of each activity should be considered when the activities are totally ordered.

A schedule is called *semiactive* if given the order of the operations on the machines and the total order of the operations on the jobs, no operation can be started earlier without violating the established order. A schedule is *active* if no operation can be started earlier without delaying other operations. Finally, a schedule is *nondelay* if no machine is kept idle when there is an operation available for processing.

A performance measure Z is regular if the scheduling objective is to minimize (maximize) Z, and Z can increase (decrease) only if at least one of the completion times in the schedule increases.

According to [4], a set D is a dominant set of schedules for regular measures of performances, if only solutions that are in this set are needed to be considered when searching for the optimal solution.

In the case of job shop problems, *active* and *semiactive* schedules dominate the set of all schedules if the objective is a regular measure of performance. In our problem, the *oil production* is a regular measure of performance, so we can concentrate our attention to active and semiactive schedules.

The following subsections describe the neighborhoods used in this paper.

4.2.1 Neighborhood 1 - Insertion

Given a solution represented by a disjunctive graph, this neighborhood is obtained by choosing each activity and inserting it in all possible positions of all resources that are able to execute it. The feasibility of each neighbor is tested by verifying if the new disjunctive graph obtained is acyclic. Figure 5 shows the disjunctive graph for a solution and the disjunctive graphs for the neighbors obtained when activity 1 is inserted in the second position of resource 3 and when this activity is inserted in the third position of resource 3. The latter is infeasible, since its graph contains a cycle, while the former is feasible. Note that the conjunctive arcs are the same for all neighbors, due to the fact they represent the precedence constraint between the

activities that belong to the same well. The disjunctive arcs are the ones that change for each neighbor, since they represent the position of the activities on the resources. In this figure, redundant disjunctive arcs are not shown. Clearly, this neighborhood has $O(n^2)$ neighbors.

It is important to observe that on each neighbor, all existing disjunctive arcs determine a total order of the activities on each resource. Clearly, in face of the conjunctive arcs that may be, this is not sufficient to determine an optimal schedule for all activities. Even when considering, in addition, the conjunctive arcs, the ordering of the activities in a given well may not be known. This fact makes the optimal assignment of the activities start time an NP-Hard problem. We use a reduction to prove it. Therefore, this neighborhood is a very large scale neighborhood, because despite the polynomial number of neighbors, the exploration of each neighbor is still an NP-hard problem. On the other hand, if the conjunctive arcs define a total order in each well, an optimal schedule can be found in $O(n)$, as the *semiactive* schedules are a dominant set.

4.2.2 Reduction

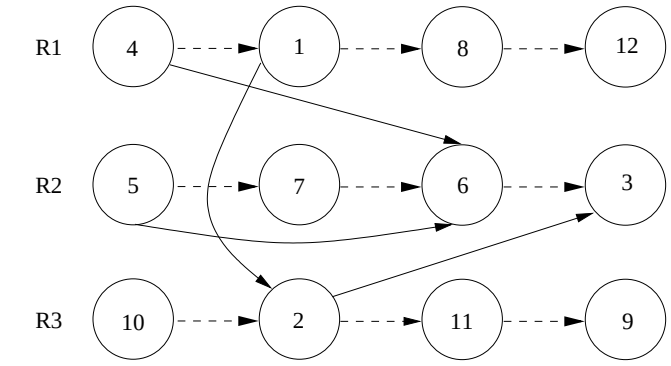
We consider two problems.

1. **Assignment of optimal start times (Neighborhood 1) - Decision Problem**

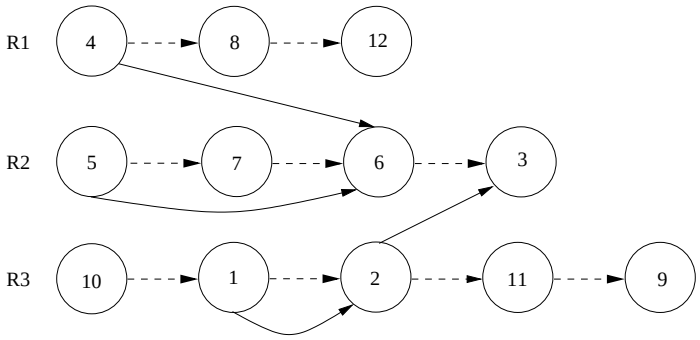
   - *Instance:* Total oil production $V \in Z^+$, horizon $H \in Z^+$, set P of oil wells, set R of resources. For each $r \in R$ there is an associated ordered collection of activities $a_k[r]$, $1 \leq k \leq n_r$, i.e., the resource r executes activity $a_k[r]$. For each such activity a (notation a is an abbreviation for $a_k[r]$) there is a processing time $l(a) \in Z_0^+$, an oil well $p(a) \in \{1, 2, \cdots, P\}$, a type $t(a)$ and an outflow $v(a)$ associated to it. The activity type defines a partial order between the activities of the same well.
   - *Question:* Is there a time instantiation $\sigma(a)$ for each activity a, such that the order of the activities on each well and the order of the activities on each resource is respected, and the total oil production obtained within the horizon H is equal or greater than V? Activities from the same well may not overlap. The oil production is given by:

$$\sum_a \max(0, H - \sigma(a) - l(a)) \times v(a).$$

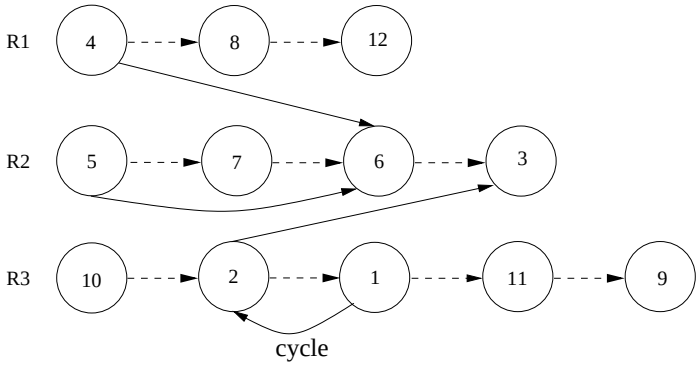
2. **Job shop scheduling - Decision Problem**



(a) Current Solution



(b) Feasible neighbor



(c) Infeasible neighbor

Figure 5: Current solution and neighbors obtained when activity 1 is moved to the second and third positions of resource 3

- *Instance:* A deadline $H \in Z^+$, number $m \in Z^+$ of processors, set J of jobs, each $j \in J$ consisting of an ordered collection of operations $t_k[j], 1 \leq k \leq n_j$. For each such operation t (the notation t is an abbreviation for $t_k[j]$) there is an associated length $l(t) \in Z_0^+$ and processor $p \in \{1, 2, \cdots, m\}$, where $p(t_k[j]) \neq p(t_{k+1}[j])$ for all $j \in J$ and $1 \leq k < n_j$.
- *Question:* Is there a time instantiation $\sigma(t)$ for each operation t, such that the order of the operations on each job is respected, two operations designated to the same processor do not overlap, and $\sigma(t) + l(t) \leq H$?

Theorem 1: Problem *Assignment of optimal start times* is NP-hard.

Proof: It is easy to see that problem 1 belongs to NP. As problem 2 is NP-hard [14], a reduction that transforms problem 2 to problem 1, in polynomial time, proves that the latter is NP-hard. We will consider that the answer obtained to problem 1 will be the answer given to problem 2.

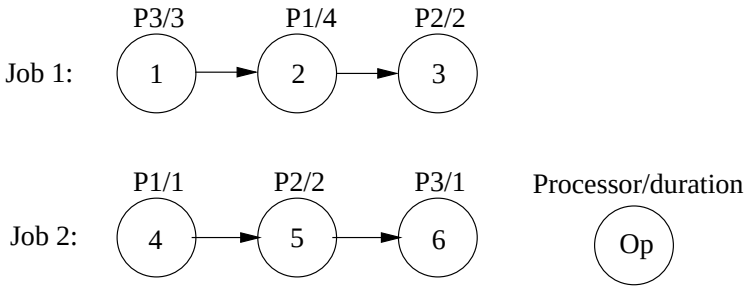
Let an arbitrary instance of problem 2 be given by the deadline H, m processors, the set J of jobs and the ordered sequence of operations associated with each job. In order to map this problem into problem 1, each job $j \in J$ turns into a resource $r \in R$. The set of operations associated with each job in problem 2 is the ordered collection of activities of each resource. The length of the operations $l(t)$ is the processing time of the activities $l(a)$ and the processor of each operation $p(t)$ is now the well associated to each activity $p(a)$. The activities type in problem 1 is chosen in such a way that it does not have precedence relationship with any other type. The horizon of problem 1 is set to $H + 1$ and the total production V is set to 1.

In addition, a dummy oil well is created and $|J| + 1$ dummy activities are created. The processing time of these activities will be zero and their associated well will be the dummy well. The resource designated to the first dummy activity is resource 1, the resource associated to the second dummy activity is resource 2, and so on. Each dummy activity will be placed at the last position of the corresponding resource. It is feasible to associate to each dummy activity a unique resource and a position on this resource, because an instance for problem 1 provides an ordered colletion of activities to each resource. The dummy activity $|J| + 1$ does not need a resource. The type of the first $|J|$ activities is **t1**, a new type. The type of the last dummy activity is **production**, also a new type. The precedence relationship between these two types is: **t1** $\rightarrow$ **production** .

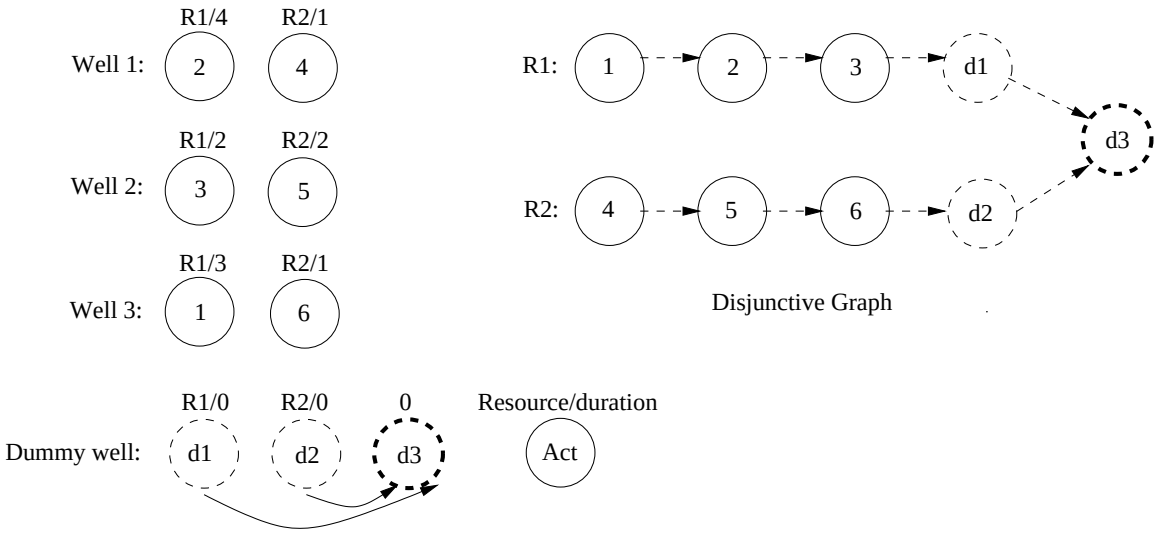
Therefore, the instance of problem 1, that is created, will have $|J|$ resources, $m+1$ oil wells and $(\sum_{j \in J} n_j) + |J| + 1$ activities.

The outflow associated with each activity a is set to zero, except when a is the last dummy activity, in which case its outflow is set to 1.

Figure 6 shows an instance of problem 2 with 2 jobs and 3 processors, and the instance of problem 1 obtained using the transformation described above.



(a) Instance problem 2



(b) Corresponding instance of problem 1

Figure 6: Example of reduction

Since the dummy activity associated with each resource is the last to be executed in that resource, and given the precedence relationship between these dummy activities and dummy activity $|J|+1$, we can conclude that dummy activity $|J|+1$ will be the last one to be concluded among all activities. So, by construction, the expression that gives the total oil production reduces to

$$\max(0, H+1-\sigma(d)),$$

where d is the dummy activity $|J|+1$. Therefore, the total oil production is equal or greater than V, if and only if, $\sigma(d) \leq H$. Since $\sigma(a)+l(a) \leq \sigma(d)$ for all activities a, it implies that $\sigma(a)+l(a) \leq H$ for all activities a. As the activities of problem 1 corresponds to the operations of problem 2, we also have that $\sigma(t)+l(t) \leq H$ for all operations t. So if the answer to the question of problem 1 is *yes*, the answer to the question of problem 2 must also be *yes*.

On the other hand, the total oil production is less than V, if and only if, $\sigma(d) > H$. This means that it was not possible to terminate all the activities before the horizon, i. e., $\sigma(a)+l(a) > H$ for an activity a. Due to the mapping between the two problems, it is known that the operation t that corresponds to the activity a is not finished before the deadline H either. So if the answer to the question of problem 1 is *no*, the answer to the question of problem 2 must also be *no*.

Clearly, the mapping that transforms problem 2 to problem 1 can be done in polynomial time. This establishes that problem 1 is NP-hard.

4.2.3 Neighborhood 2 - Window

This neighborhood is a generalization of the former one. The motivation is to generate neighbors more able to perform big modifications on the current solution than the neighbors of the previous neighborhood. The intention is to obtain a greater improvement of the total oil production. To reach this objective, this neighborhood will consider two sets of activities called windows and a *main activity*. The windows are choosing according to this activity. The activities in the windows will be freed from their current position, but their resources will remain the same. Their new positions will be determined later, when the start times are assigned.

Now the neighborhood will be explained in details. The main idea is not only to insert a single operation on every position of each resource that is able to execute it, but also to free the current positions on the resources of the activities that are inside a window. The considered activity, that from now on will be called *main activity* is on the center of the window and a parameter sets its length. The parameter indicates how many consecutive activities will be freed on each side of the main activity. If the parameter is zero, only the main activity will be considered. Activities that are scheduled earlier on the same resource than the main activity are on its left and activities scheduled later are on its right. If there are less activities on a side than

the parameter, only these activities will be considered on that side. The order of the activities outside the window is preserved. In other words, the disjunctive arcs of the activities in the window are not oriented in the neighbors. The disjunctive arcs that were related with the main activity are deleted, and new disjunctive bi-directed arcs are inserted between the main activity and the activities that are allocated on the new resource of the main activity.

A window is also considered in the destiny resource of the main activity. Just like the other window, a parameter will determine its length. The center of this window be the activity that occupies the same position as the main activity on its original resource. If the position of the main activity on its original resource is greater than the highest position of the destiny resource, then the highest position will be considered as the center of the window. In that case, the right side will be null. The activities within this window will be freed of their positions, like the activities of the first window. This means that their disjunctives arcs will turn from oriented to bi-directed. If the parameter is zero, this window will not be considered.

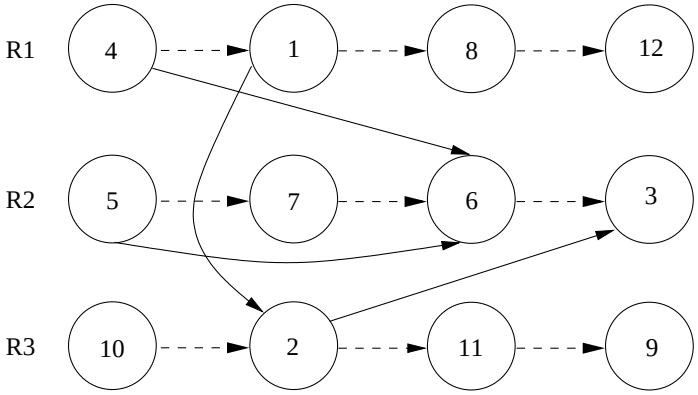
Figure 7 illustrates the disjunctive graph of a current solution and the disjunctive graph obtained when activity 1 is the main activity and the new resource allocated to it is resource 2. The parameters of both windows are 1.

Note that as in neighborhood 1, each activity will be assigned to all resources that are able to execute it. But unlike neighborhood 1, a fixed position will not be assigned to the activity. This will be established implicitly when the start times are determined. The other difference is that the positions of the activities within the windows are not known either. Clearly, if both parameters are zero the only difference between neighborhood 1 and this neighborhood is that on the former, the position of the main activity is fixed on the new resource, and this does not occur on the latter.

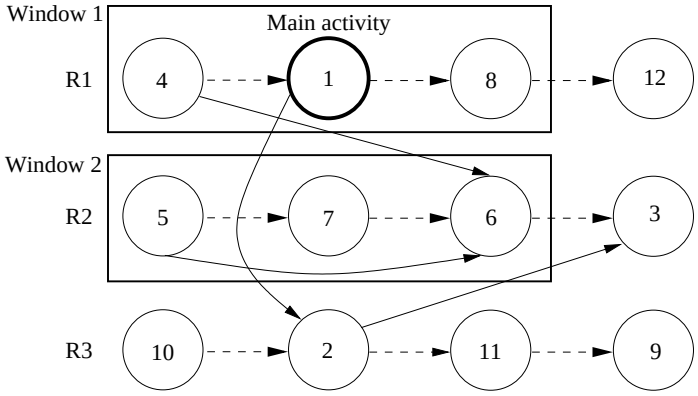
It is easy easy to see that the neighbors are defined by the main activities and their destiny resource. It occurs because the windows are determined depending just on the choice of the main activity and its destiny resource, since the length of the windows is a constant parameter. So, there are $O(nm)$ neighbors in this neighborhood, where n is the number of activities and m the number of resources. On the previous neighborhood, the position of the activities on each resource was known before the start times were assigned. Although, even considering this fact, the problem of assigning the optimal start times proved NP-hard. In this neighborhood, the position of the activities is not known for certain resources. So, the problem of assigning the optimal start times for the neighbors of this neighborhood is at least as hard as the corresponding problem of the previous neighborhood.

4.2.4 Neighborhood 3 - Well

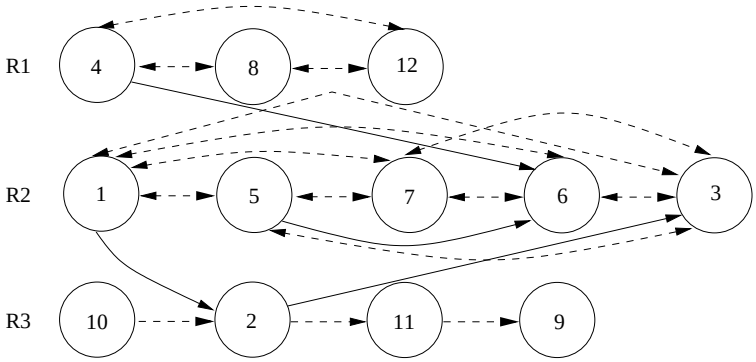
The idea behind this neighborhood is the intuition that activities of the same well should stay close to each other in order to maximize the production of the oil field.



(a) Current Solution



(b) Main activity and windows when parameter of both windows is 1



(c) Disjunctive graph representing the neighbor

Figure 7: Example of neighborhood 2 - Window

Therefore, this neighborhood is slightly different from the previous one. There is also a *main activity* that is the activity that may change from its current resource to all possible resources. But instead of freeing the position of activities that are within a window, this neighborhood frees the position of the activities that belongs to same oil well of the main activity, preserving their resources.

The activities of other wells may also be freed. There is a parameter that indicates how many wells will be freed. When this parameter is set to one, only the well of the main activity is considered. If the parameter is greater than 1 the priority rule to choose the wells is: on the current solution, for each well, determine the maximum distance (difference of start times) between two consecutive activities of the well. Then, multiply this distance by the corresponding outflow of the wells. The higher this value, the higher the priority. Using this priority rule, we will select wells which activities are more spread and that have a bigger outflow. So there will be the chance to put together these activities and maybe improve the production.

Figure 8 illustrates the disjunctive graph of a current solution and the disjunctive graph obtained when activity 1 is the main activity and the new resource allocated to it is resource 2. The parameter is set to 1. Activities $2, 3$ belongs to the same well as activity 1, so they are freed of their positions.

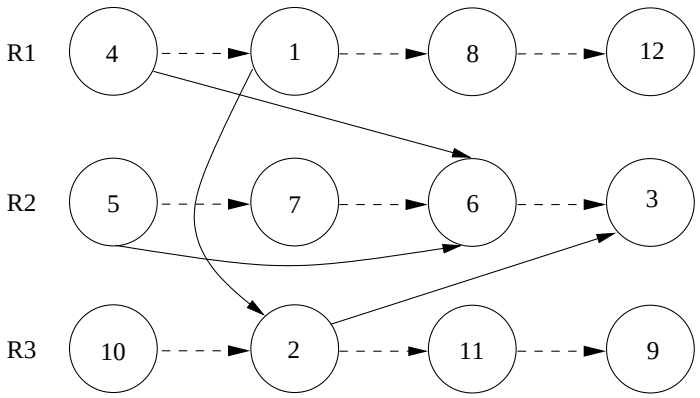
As neighborhood *Window*, this one has $O(nm)$ neighbors, and again the optimal start time assignment to each activity, given the disjunctive graph, is an NP-hard problem, as this problem is at least as hard as the corresponding problem of neighborhood *Insertion*, due to the fact that the activities position on certain resources in this neighborhood is not known before the start times instantiation.

4.3 Assigning the start times

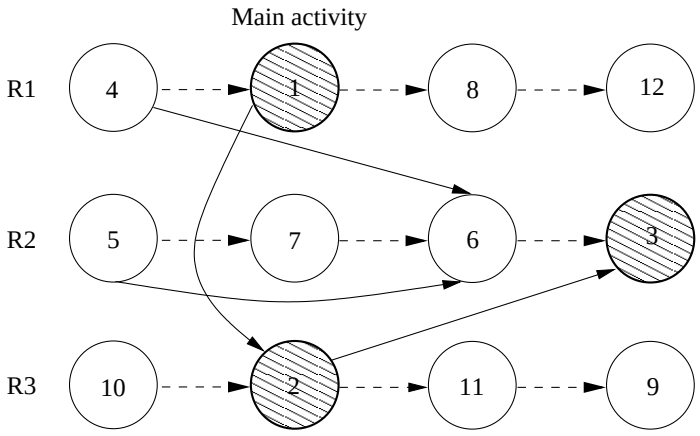
Two techniques were used to instantiate the start times of the activities on each neighbor. One will be called *Greedy* and the other one will be called *Optimized*. The first is a greedy strategy and the latter uses CP to perform the instantiation.

- **Greedy:** This technique was implemented in C++ without using CP. A *heap* S is used to store the activities that are ready to be scheduled, as all their predecessors have been scheduled. The order on the heap is based on the earliest possible start time of the activities. The earlier the start time, the higher the priority on the heap. An algorithm that generates nondelay schedules [4] is used and the top activity of the heap is the one selected on each iteration.

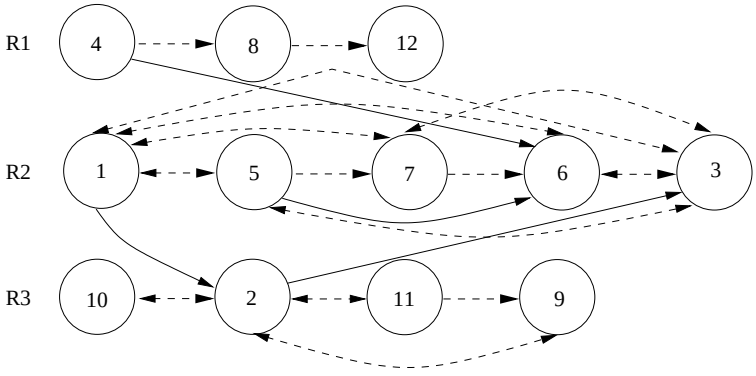
  Initially, heap S contains all activities without predecessors. On each iteration, the top activity of S, called a, is removed and its start time is set to the earliest time allowed by its actual resource and the other executing activities that belong to the same well. After that, all successors of activity a are inserted in S. When



(a) Current Solution



(b) Main activity and the other activities that will be freed



(c) Disjunctive graph representing the neighbor

Figure 8: Example of neighborhood 3 - Well

S gets empty, the algorithm terminates. When this happens all activities are already scheduled.

- **Optimized:** This technique is based on CP and was implemented using the *ILOG Solver*. The constraints of the model are:

  1. technological precedence constraints;
  2. the order of the activities on the resources;
  3. constraints that state that activities on the same resource or of the same well can not overlap.

  One of the most common branching strategy for scheduling problems that have the *makespan* as the objective function is called *Settling Essential Conflicts (SEC)* [18, 3]. This strategy determines a scheduling order for all activities, i.e., it determines the orientation of the disjunctive arcs until all of them are directed. Then, given the total order that was constructed, a simple topological sort algorithm determines the activities start times, generating an semiactive schedule.

  This strategy performs well when the objective is the *makespan* due to certain proprieties of the underlying graph [17] and due to an efficient calculation of some lower bounds [17, 20, 19] that enable, on each node of the tree, the orientation of several disjunctive arcs at a time. Since our objective function is not related to the *makespan*, this branching strategy can not be applicated to our problem. So we considered a different strategy.

  In our strategy, constraints 1 and 2 are imposed upon the model at the beginning, as is natural in any CP model. At this moment, these constraints are given by the disjunctive arcs that are already oriented and by the set of conjunctive arcs. These restrictions are used to prune the domain of the variables. The variables of this model represent the start time of each activity. At the *labelling* phase, the CP solver establishes the start times.

  At first, the default *labelling* mechanism present in the *ILOG Solver* was used. As required by the CP solver, parameters indicating the objective function and the maximum execution time were also indicated. Furthermore, a priority rule to choose the order of instantiation of the variables and a rule to choose a value of the domain were also specified. The priority rule was: among the activities not yet instantiated, the one that had the earliest possible start time was selected. Ties were broken arbitrarily. The value chosen was the smallest of the current domain.

  Note that using this default procedure we are not taking into account the knowledge that the active schedules are a dominant set for this problem. But there is

a more serious problem with this procedure. The default *labelling* mechanism acts like this: if a variable a is instantiated before a variable b, in order to backtrack to a new value for a, all the domain of b must be exhausted.

A simple example shows how this can be bad to the performance of this strategy. The instance is: two oil wells, a set of 5 activities and 2 resources. Activities $1, 2$ and 3 belong to well 1 and activities 4 and 5 to well 2. All resources are able to execute all activities and the precedence constraints are represented in figure 9. Activities 3 and 5 turn the respective wells into production. The outflow of both wells is 10 and the horizon is also 10.

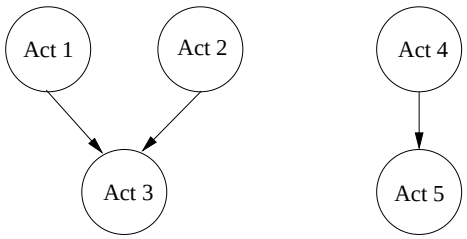


Figure 9: Technological Precedence Constraint

Suppose that on a neighbor the order of the activities on the resources is: Resource 1: $1, 4, 5$ and Resource 2: $2, 3$. If the instantiation order, according to the earliest possible start time, is $2, 1, 4, 3, 5$, the start time of each activity will be set to values as shown on table 8. The total production is 650. But if the start times were selected as in table 9, the production would be 700.

Activity	Start time
1	5
2	0
3	10
4	10
5	15

Table 8: First variable instantiation

The *labelling* algorithm would have to instantiate activity 2 with time values $1, 2, 3, 4, 5$ before it reached the same result. Worse yet, since activity 2 is the first activity on the instantiation order, the domain of all the other variables would have to be exhausted before a new value is assigned to it.

On the other hand, if, by some means, after reaching the solution shown in table 8, the *labelling* mechanism could be forced to immediately return to the

root of the search tree, and be given a new instantiation order as $1, 2, 4, 3, 5$, the improved solution of table 9 would be readily reached.

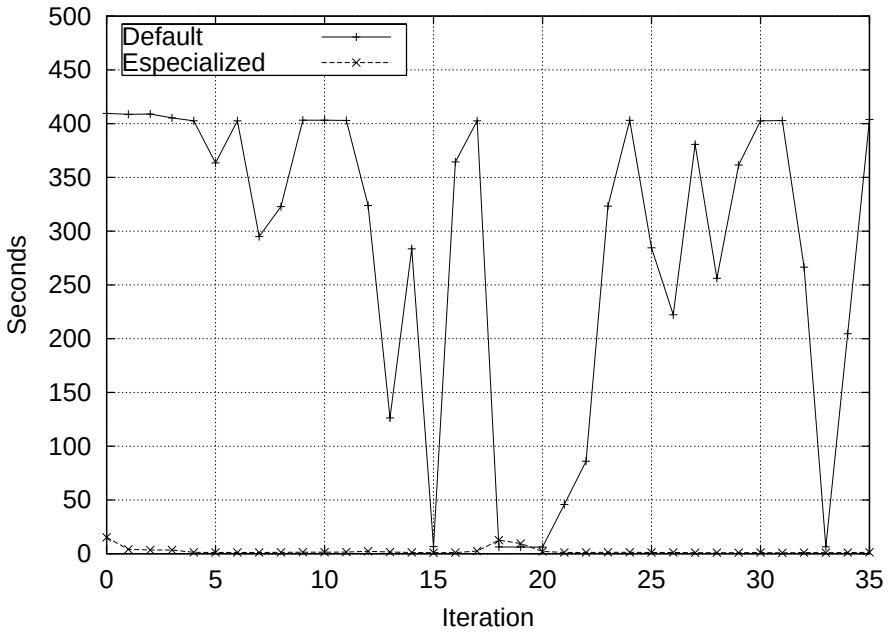
Activity	Start Time
1	0
2	5
3	5
4	10
5	10

Table 9: Final variable instantiation

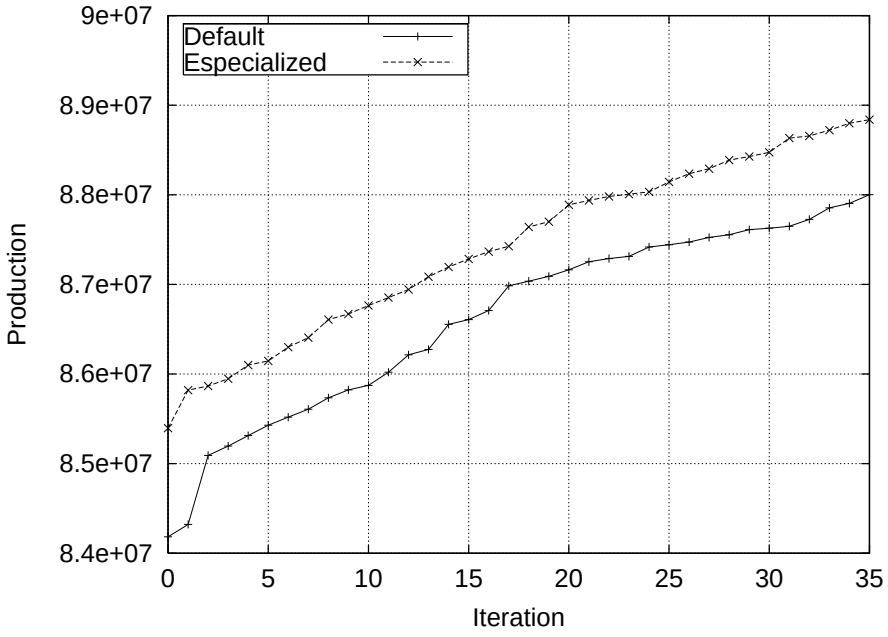
In our strategy, the default *labelling* mechanism was replaced by a new specialized mechanism. Our mechanism changes the instantiation order, considering only active schedules. To accomplish this, the variable instantiation is made in two phases. At each level of the search tree, a variable is created to represent the activities that can be instantiated on that level. These variables are called *level variables*. The feasible activities of each level (those that are on the domain of the level variable), are the activities with indegree zero and that obey the active schedule rule. The indegree of an activity is the number of unscheduled predecessors of the activity. The indegree is dynamically updated during the whole search process. The start indegree is given by the orientated arcs of the disjunctive graph that represents each neighbor. Each step of the search process consists in choosing a value of the domain of the corresponding level variable and assigning the earliest possible time to the variable that represents the start time of the activity indicated be the level variable. So, when a solution is found or when an fail occurs, the *backtracking* is done on the level variable, not on the variable that represents the start time, allowing changes on the instantiation order of the start time variables to occur. Furthermore, to improve the Solver pruning mechanism, whenever a solution is found, a constraint that states the following solutions must have a production higher than the production of the current solution is added.

Figure 10 shows the time and the total oil production obtained per iteration of a hybrid tabu search algorithm applying the default and the specialized *labelling* mechanism to the same problem instance and the initial solution. The neighborhood used was neighborhood *Insertion*. The tabu search algorithm will be explained in detail on the next subsection. The maximum execution time per iteration allowed was 410 seconds.

Analysing this figure, it is easy to see that the specialized mechanism is about 100 times faster than the default mechanism, on most iterations. Furthermore,



(a) Time X Iteration



(b) Production X Iteration

Figure 10: Comparison between the default and the specialized mechanism

the specialized mechanism was able to find an optimal solution. On most iterations, the default mechanism terminated when it reached the maximum allowed execution time, without finding an optimal solution.

For this reason, only the specialized version was considered when using the optimized technique to assign the start times.

4.4 The tabu search metaheuristic

In subsection 4.2 the structure of the neighborhoods was explored and subsection 4.3 explained how the start times were selected. This subsection gives details of the tabu search method for each neighborhood. These details include the *stop rule*, the *tabu list*, the *aspiration rule* and the *selection of the neighbor*. This subsection also discusses a pure tabu search approach and the neighborhood used in this approach.

- **Stop Rule**

  The stop rule for all neighborhoods is the execution time. This makes the comparison among the neighborhoods fair, since no matter how many iterations or neighbors each one has, after the same amount of time all of them are terminated and the results are compared. In this paper the maximum execution time is 3600 seconds.

- **Tabu List**

  For neighborhoods *Insertion* and *Window* the *main activity* of the selected neighbor is considered tabu. This means that this activity can not be the *main activity* while it is considered tabu, unless it satisfies the aspiration rule. A neighbor is considered tabu if its *main activity* is tabu. In this paper an activity is considered tabu for 50 iterations.

  For neighborhood *Well*, the well of the *main activity* is considered tabu. This means that the activities of this well can not be freed while the corresponding well is considered tabu, i.e., they can not be the *main activities* nor the well can be among the ones whose activities can be freed. In this neighborhood, we do not have tabu neighbors. It is not allowed to join a tabu well to the wells that will be freed. For this reason, the aspiration rule does not apply for this neighborhood. So in section 6, whenever the number of tabu neighbors is mentioned for this neighborhood, it means the number of times a tabu well was forbidden to join the wells to be freed. In this paper a well is considered tabu for 10 iterations.

- **Aspiration Rule**

  The aspiration rule for the neighborhoods *Insertion* and *Window* is: whenever the production of a tabu neighbor is better than the best production found so far, this neighbor is considered.

- **Selection of the neighbor**

  The selection of the neighbor is quite different for the neighborhood *Insertion* and the other two neighborhoods. So they will be explained separately.

  Insertion neighborhood

  Several approaches were tested to select the neighbor of this neighborhood. At first, the whole neighborhood was explored using the **Optimized** technique to instantiate the start times, and the best neighbor was selected. As the time required for each neighbor was about 0.2 seconds, this approach turned to be impractical, since there is approximately 250000 neighbors. In the second approach, the whole neighborhood was explored using the **Greedy** technique to instantiate the start times. This took approximately 15 seconds. Next, the best x neighbors were explored again using the **Optimized** technique and the best neighbor was selected. But computational tests showed that exploring only the first y neighbors that improves the best solution was better than exploring the whole neighborhood [22].

  In the third approach, the one used in this paper, the neighborhood is explored using the **Greedy** technique until y neighbors that improve the best solution are found. Then x neighbors, $x \leq y$, are explored using the **Optimized** technique and then the best neighbor is selected. Two different strategies are used to choose the x neighbors. The first chooses the x best neighbors deterministically. This approach was called **ID**. The second chooses x neighbors randomly among the y neighbors. This approach was called **IA**. The values chosen for y and x were $y = 10$ and $x = 10$, in the **ID** approach and $y = 20, x = 5$, in the **IA** approach. These numbers were chosen based on computational results obtained in [22]. It is clear that if there are not y neighbors that improve the best solution, the y best neighbors of the entire neighborhood are considered.

  The Window and Well neighborhoods

  For these two neighborhoods, the rule to select the neighbor is the same. On both neighborhoods it was not practical to use the **Greedy** technique to select neighbors. It was also not practical to explore the whole neighborhood, nor even to select the first y best neighbors. The use of the **Optimized** technique to find the optimal start times of the neighbors proved impractical too. This is

probably due to the fact that obtaining good solutions in these more complex neighborhoods is much harder. Note that the **Greedy** technique was operating on neighborhoods with a large number of disjunctive arcs that were not oriented. This may have caused it to produce inferior solutions when considering neighbors that had a much better solution if a different ordering of the activities was chosen. The **Optimized** technique takes about a second to find a first solution at each neighbor and a great amount of time to prove that such a solution is optimal.

We decided to use a strategy where the neighborhood is explored using the **Optimized** technique until a neighbor that improves the best solution is found, or until the execution time on the neighborhood reaches 20 seconds. Furthermore, the **Optimized** technique has a limited execution time on each neighbor. For neighborhood *Window* this time limit was set to 1 second and for neighborhood *Well* this time limit was set to 3 seconds. This means that the optimality of each neighbor may not be proved. The selected neighbor is the best one found.

In section 6, we will call **WI** the neighborhood *Window* when this approach is used. Similarly, when neighborhood *Well* is used with this approach we will call it **WE**.

After performing some computational tests using neighborhood *Window*, the value chosen for the window length of the main activity was 3 at the original resource, and was set to 4 at the destiny resource. After testing neighborhood *Well*, the number of wells freed was set to 90.

As just the first y best neighbors are explored by all approaches, the order of exploration of the neighbors makes a difference. The priority rule used to choose the main activity, i.e., to choose the neighbor, is the same priority rule used by algorithm **CP2** of subsection 4.1.

The Pure Tabu Search Technique

This tabu search approach was implemented by Vinicius Fortuna in his undergraduate project.

First of all, before defining the neighborhood used, it will be explained how solutions are represented. A solution is represented by an ordered list of activities together with a data structure that indicates which resource is allocated to each activity.

Consider graph G where its nodes represent the activities and its arcs the precedence relationship between them. Using this graph, a schedule is obtained in the following way:

1. Add arc (a_i, a_j) to G if activity a_i is placed before activity a_j in the ordered list and they belong to the same well or are executed by the same resource. Note that graph G provides a total order between the activities.

2. The start time is obtained running a polynomial time topological sort algorithm over graph G.

There are two movements that define the neighborhood. The first consists in changing the resource allocated to an activity to all resources able to execute it. The second consists in removing an activity from its position on the ordered list and inserting it in all positions of the list, preserving the allocation of the resources. Both the feasibility and the schedule of each neighbor is obtained by running a topological sort algorithm over the correspondent graph G. This neighborhood has $O(nm + n^2)$ neighbors, where n is the number of activities and m is the number of resources.

The stop and aspiration rules are the same rules used for the hybrid approaches. Two strategies were adopted to identify tabu movements. The first considers tabu just the movement used to generate the selected neighbor. This strategy will be called **TabuPF**. The second considers tabu groups of movements. If the chosen movement alters the resource of an activity, then all movements that alter the resource of this activity are considered tabu. If the movement changes the position of an activity, all movements that change the position of this activity are considered tabu. This strategy will be called **TabuPR**. A movement or a group of movements are considered tabu for 25 iterations.

The first neighbor that improves the best solution is the selected neighbor.

5 Upper Bounds

As there are no previous computational results for the problem instances used in this paper, the calculation of upper bounds is important to determine the quality of our solutions.

This section explains how four upper bounds were calculated to the problem. In fact, from one approach to another, the intention was to improve the bound. So the first approach, called **Upper0** provides the weakest bounds, while the last approach, called **Upper3** provides the tighter bounds. **Upper0** is based on a combinatorial argument while the other bounds are obtained solving relaxations of the original problems using Integer Linear Programming.

- **Upper0**

  This bound is given by the following equation:

$$\sum_{i=1}^{w}(H - \sum_{j \in Act_i} d_j)) \times v_i$$

  where w is the number of wells, H is the time horizon, Act_i is the set of activities of well i, d_j is the the processing time of activity j and v_i is the outflow of well i.

This equation assumes that there is an unlimited number of resources that are able to execute any of the activities. The number of resources is not being taken into account and, since this number is very limited, the bound obtained with this approach is poor.

- **Upper1**

  In this approach, an integer linear model is formulated to obtain the bound. The corresponding IP model takes into account the fact that there is a limited number of resources. In this model, all available resources can execute all activities, i.e., we have a situation similar to a parallel machine environment [24].

  The model considers that each well has just one activity. The processing time of this activity, denoted by Δ, is the sum of the processing time of all activities of the correspondent well. After this activity is executed, the well is considered apt to produce oil. Since this single activity represents all activities of a well, preemption will be allowed. In this case preemption can occur at any instant during the execution time of an activity. This turns this problem into a relaxed version of the original problem. The objective function is the same as the original problem, that is, to maximize the total oil production. This relaxed version of the problem will be denoted by $P||Production$.

  In the model used in this approach, the value of Δ will be rounded to the highest multiple of 5 less than or equal to Δ, as the time in this model will be discretized in unities of 5, i.e., each time unity in the model represents 5 time unities in the original problem. It will be done to reduce the size of the model, since computational tests showed that it is impractical to consider models where the time is discretized in unities of 1.

  The binary variables of this model are x_{it}, and it is set to 1 if well i is finished at time t, and, it is set to 0 otherwise.

  The objective function is:

$$\max \sum_{i=1}^{w} \sum_{t=0}^{H/5} (H - t) v_i x_{it},$$

  where constants w, H and v_i retain their meaning.

  The constraints of this model are:

  1. $\sum_{t=0}^{H/5} x_{it} \leq 1, \quad i = 1 \cdots w;$

2. $\sum_{k=0}^{t} \sum_{i=1}^{w} x_{ik}\Delta_i/5 \leq t \times m, \quad t = 0 \cdots H/5,$
   where m is the number of resources;
3. $x_{it} = 0, \quad$ for all i and t such that $\Delta_i > t.$

Constraint 1 says that there is at most one termination time for each well. Constraint 2 enforces that there must be enough time, summed over all the resources, to execute all the wells that are terminated by instant t. Note that this constraint allows the execution of a well in more than one resource at the same time. It is important to observe that if the time horizon is large enough for all wells to be terminated within the time horizon, then all of them will be performed, as that the total production is maximized. Restriction 3 tries to reduce the number of possibilities, by forcing a well not to be terminated if there is not enough time by instant t.

- **Upper2**

  This approach improves upon the previous one, by not permitting simultaneous execution of the same well on different resources. The relaxed version of the problem modelled in this approach is the same as the previous one. The difference being that, except for constraint 2, which is replaced for two new constraints and for a new integer variable.

  The replaced constraint reflects the fact that, according to theorem 2 below, preemptions are redundant for problem $P||Production$, i.e., the objective value may not be improved by allowing preemptions.

  Theorem 2: For problem $P||Production$ preemptions are redundant.

  Proof: The proof is divided in two cases. Case 1 occurs when the time horizon is large enough for all wells to be finished within it. Case 2 occurs when this is not true.

  - *Case 1:*
    In this case, solving problem $P||Production$ is equivalent to solve problem $P||\sum w_jC_j$ (the $\alpha|\beta|\gamma$-notation of [16] is used to represent the parallel machine problem of minimizing the weighted sum of completion time).
    A classical result of McNaughton [21] shows that for $P||\sum w_jC_j$ preemption is redundant. Therefore, preemption is also redundant for $P||Production$, since they are equivalent.
  - *Case 2:*
    Consider an optimal solution for $P||Production$, when preemption is allowed and let S be the set of wells which are finished before the horizon. Then, according to *Case 1*, the wells in S can also be optimally scheduled

in an nonpreemptive way. As the wells not in S do not contribute to the objective function, an optimal solution for $P||Production$, when preemption is not allowed, is also an optimal solution when preemption is allowed. So the theorem is established.

The new integer variables of this model are r_t. It counts how many resources are busy at time t. Since the problem has m resources, a straightforward constraint is:

$$r_t \leq m, \quad t = 0 \cdots H/5$$

Constraint 2 of the previous model is replaced by this constraint. In addition, a new constraint must be written so that r_t reflects the number os active wells at instant t. The following constrain is used:

$$r_t = \sum_{k=t}^{H/5} \sum_{i:k<t+\Delta_i} x_{ik}, \quad t = 0 \cdots H/5.$$

It states that if a well i is finished at instant k, a resource is considered busy during instants $k-1, k-2, \cdots, k-(\Delta_i - 1)$, since the well will be executed without interruptions.

In fact, we do not need to consider variables r_t explicitly. In the implementation, these variables will be replaced by the sum they represent, which in known in advance.

Some of the possibilities of the previous model are eliminated, as there must be enough time in each resource to execute all the activities that are allocated to it. This was not true in the previous model, since the time available in all resources was checked at each instant.

- **Upper3**

  All the three previous models did not take into account some relevant informations present in the problem instances considered.

  One such information is that there are two main groups of resources: derricks and boats. In the instances considered, all the activities that require derricks may be executed by any of them. The same occurs with the activities that require boats. But the relevant fact is that there are more derricks than boats and many more activities that require the former than the latter.

  Moreover, the pattern present in all wells uses resources in a sequence

$$\text{Derrick} \rightarrow \text{Boat} \rightarrow \text{Derrick}.$$

Observe that the number of activities being executed in each phase may be different, including zero.

Thus, the version of the problem considered in this approach assumes that each well consists of three activities. The first activity represents all activities of the well that are executed by the first derrick. The processing time of this activity will be the sum of the processing time of the represented activities or it will be zero, if the well does not have this type of activity. The same mechanism is applied for the other two activities. The first activity of well i will be called A_i^1 and its processing time Δ_i^1, the second will be called A_i^2 and its processing time Δ_i^2, and the third will be called A_i^3 and its processing time Δ_i^3. Activities A_i^1 and A_i^3 must be executed by derricks and activities A_i^2 must be executed by boats. Furthermore, the precedence relationship $A_i^1 \rightarrow A_i^2 \rightarrow A_i^3$ must be respected. The production of well i starts when activity A_i^3 finishes. Again, the objective function is the same as the original problem, that is to maximize the total oil production.

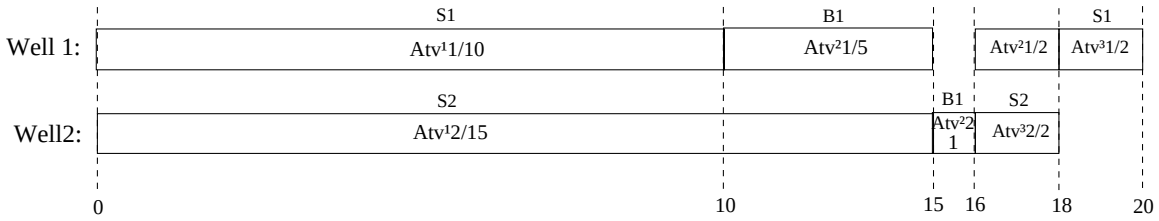
Unfortunately, preemption is nonredundant for this relaxed version of the problem. A simple counter example shows it. Suppose there are two wells, two derricks ($S1$,$S2$) and one boat ($B1$). The horizon considered is 25. The outflow of the wells and the processing time of their activities is given in table 10.

Well	Δ_i^1	Δ_i^2	Δ_i^3	Outflow
1	10	7	2	1
2	15	1	2	2

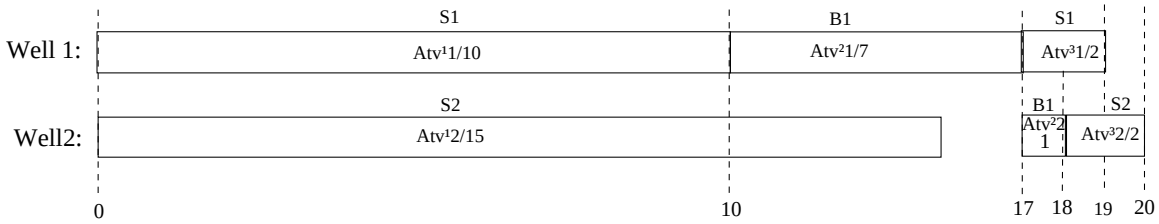
Table 10: Durations and outflow

Figure 11 shows the optimal solution when preemption is allowed and when it is not allowed. In this figure, the values inside the rectangles indicate the activity that is being executed and the corresponding execution time. The values above the rectangles indicate which resource is being used. The production for the first case is 19 units, and it is 16 units in the second case, showing that preemption is important in this example.

The idea behind the new model used in this approach to represent this version of the problem is similar to the one used in approach **Upper1**. Note that in this model there are constraints to represent the precedence relationship between the activities of a well. As in the previous models, the values of Δ_i^1, Δ_i^2 and Δ_i^3 will be rounded to the highest multiple of 5 less than or equal to Δ_i^1, Δ_i^2 and



(a) Optimal preemptive schedule - $Production = 19$



(b) Optimal nonpreemptive schedule - $Production = 16$

Figure 11: Preemption is nonredundant

Δ_i^3, respectively, as the time in this model will be discretized in unities of 5, i.e., each time unity in the model represents 5 time unities in the problem.

This model has three groups of binary variables: x_{it}, y_{it} and z_{it}. Variables x_{it} are set to 1 if activity A_i^1 is finished at instant t and 0 otherwise. Similarly, variable y_{it} assumes value 1 if activity A_i^2 is finished at instant t and 0 otherwise, and z_{it} assumes value 1 if activity A_i^3 is finished at instant t and 0 otherwise .

The objective function is:

$$\max \sum_{i=1}^{w} \sum_{t=0}^{H/5} (H - t) v_i z_{it}$$

where the meaning of the constants w, H and v_i is the same as in the former three approaches.

The constraints for this model are:

1. (a) $\sum_{t=0}^{H/5} x_{it} \leq 1, \quad i = 1 \cdots w;$
   (b) $\sum_{t=0}^{H/5} y_{it} \leq 1, \quad i = 1 \cdots w;$
   (c) $\sum_{t=0}^{H/5} z_{it} \leq 1, \quad i = 1 \cdots w;$

(d) $\sum_{t=0}^{H/5} x_{it} - \sum_{t=0}^{H/5} y_{it} = 0, \quad i = 1 \cdots w;$

(e) $\sum_{t=0}^{H/5} x_{it} - \sum_{t=0}^{H/5} z_{it} = 0, \quad i = 1 \cdots w;$

(f) $\sum_{t=0}^{H/5} y_{it} - \sum_{t=0}^{H/5} z_{it} = 0, \quad i = 1 \cdots w;$

2. (a) $\sum_{k=0}^{t} \sum_{i=1}^{w} (x_{ik}\Delta_i^1/5 + z_{ik}\Delta_i^3/5) \leq t \times m_1, \quad t = 0 \cdots H/5$, where m_1 is the number of derricks;

   (b) $\sum_{k=0}^{t} \sum_{i=1}^{w} y_{ik}\Delta_i^2/5 \leq t \times m_2, \quad t = 0 \cdots H/5$, where m_2 is the number of boats;

3. (a) $x_{it} = 0, \quad \forall i, t$ such that $t < \Delta_i^1/5;$

   (b) $y_{it} = 0, \quad \forall i, t$ such that $t < (\Delta_i^1 + \Delta_i^2)/5;$

   (c) $z_{it} = 0, \quad \forall i, t$ such that $t < (\Delta_i^1 + \Delta_i^2 + \Delta_i^3)/5.$

4. (a) $\sum_{t=0}^{H/5} t(y_{it} - x_{it}) \geq \Delta_i^2/5, \quad i = 1 \cdots w;$

   (b) $\sum_{t=0}^{H/5} t(z_{it} - y_{it}) \geq \Delta_i^3/5, \quad i = 1 \cdots w;$

Constraint 1 says that all phases of a well are terminated exactly once within the time horizon or none of them are. Like in the model used in approach **Upper0**, constraint 2 enforces that there must be enough time in the resources to execute all the activities that are finished before instant t. In addition, in the current model it was necessary to consider the existence of two types of resources. Note that this constraint allows the execution of an activity in more than one resource at the same time. Constraint 3 tries to reduce the number of possibilities. Constraint 4 states the precedence relationship.

Table 11 shows quantitative data for the integer linear models of the last three approaches. The instance considered is the real instance. In this table, column *Constraints* is the number of constraints, column *Variables* is the number of binary variables, column *Const.NZ* is the number of nonzeros coefficients in the constraints and column *Obj.NZ* is the number of nonzeros coefficients in the objective function.

Approach	Constraints			Variables	Const.NZ	Obj.NZ
	Less	Greater	Equal			
Upper1	407	-	1470	31906	4819470	31694
Upper2	407	-	1470	31906	491189	31694
Upper3	920	212	3925	95718	12326123	31694

Table 11: Model quantitative data - Upper Bound - Real Instance

It can be observed from table 11 that the model for approach **Upper3** is much larger than the other two models. On the other hand, the models of approaches

Upper1 and **Upper2** have the same number of constraints and variables. The difference between them lies in the number of nonzeros coefficients.

The order of magnitude of the number of constraints, variables and nonzeros is the same for models that represent the real instance and for the corresponding models for instances 2W112S4B3, 3W95S5B3 and 4W130S5B3. So the data of table 11 is representative for all instances considered in this paper.

Table 12 shows the best bound obtained for the production by each approach. The value between parenthesis indicates the percentage by which approaches **Upper1**, **Upper2** and **Upper3** improved upon the values obtained for approach **Upper0**. The value of the bound is in millions of unities. The models were implemented and solved using the *ILOG CPLEX*. The maximum execution time was set to 3600 seconds. Table 13 shows the number of nodes in the branch tree traversed by each approach and the gap between the best upper and lower bounds.

Instance	Upper0	Upper1	Upper2	Upper3
1W130S5B3(real)	378.6	310.2 (17.3%)	300.6 (20.4%)	287.4 (23.9%)
2W112S4B3	363.8	310.4 (14.6%)	302.4 (16.8%)	270.9 (25.5%)
3W95S5B3	317.8	280.3 (11.8%)	274.9 (13.5%)	257.2 (19.0%)
4W130S5B3	420.7	371.6 (11.6%)	369.6 (12.1%)	334.7 (20.4%)

Table 12: Upper Bounds

Instance	Upper1		Upper2		Upper3	
	Node	Gap	Node	Gap	Node	Gap
1W130S5B3	1600	1.88%	1	0	1	∞
2W112S4B3	1200	1.87%	1	0	1	∞
3W95S5B3	2580	0.55%	1	0	1	∞
4W130S5B3	800	2.27%	1	0	1	∞

Table 13: Computational quantitative data - Upper Bound

These two tables convey important informations. The only model limited by the maximum execution time was **Upper1**. Model **Upper2** is an improvement upon **Upper 1**. This improvement made it possible for model **Upper2** to find the optimal solution for all instances and, after exploring just one node of the search tree. On the other hand, **Upper3** is the approach that provided the best bounds, improving upon approach **Upper0** by more than 19% in all instances. But this approach failed to provide a feasible integer solution, since it execution was terminated because the software ran out of memory after exploiting the first node of the search tree. The

same occurred when some of the execution parameters of the *ILOG CPLEX* were changed in an attempt to avoid this behavior. This fact can be explained by the considerable size of this model.

Since Upper 1 was the only approach to explore more than a node of the branch tree, figure 12 shows the evolution of the upper and the lower bounds on each node for the real instance. Note that the upper bound remains the same in all nodes and the lower bound is changed in just few nodes. A similar behavior was observed on all the other instances.

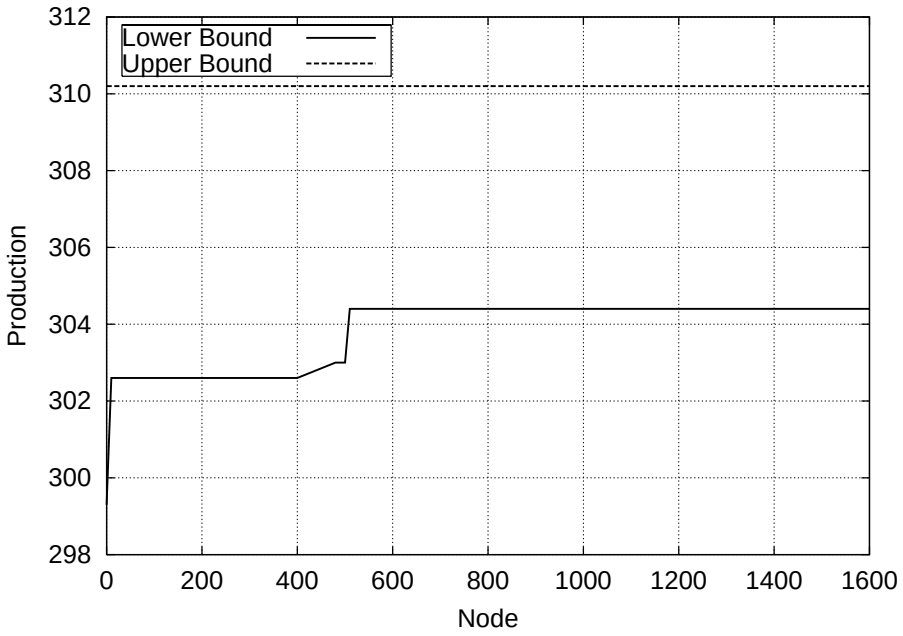


Figure 12: Lower and upper bounds on the nodes of approach Upper1 - Real instance

6 Computational Results

In this section we present the computational results obtained with instances 1W130S5B3 (real instance), 2W112S4B3, 3W95S5B3 and 4W130S5B3 when the techniques described in section 4 are applied. For all these instances, the considered horizon was set to 1500 days and the maximum execution time was set to 3600 seconds.

Table 14 summarizes the computational results for all tested instances. In all cases, the initial solution was obtained using the **H1** technique.

The columns in table 14 have the following meaning:

- Instance: instance identification;

- Approach: approach identification;
- It: total number of iterations;
- Neighbors: total number of feasible neighbors;
- Tabu: total number of tabu neighbors;
- AR: total number of tabu neighbors that satisfied the aspiration rule;
- Production: best value for the production;
- Time: total execution time.

For each instance, the best value obtained for the production is set to bold.

In column *Approach*, the names refer to the various tabu search strategies described in section 4.

For the particular case of the real instance 1W130S5B3, we have more detailed data, describing the behavior of each of the columns *Neighbors*, *Tabu*, *AR*, *Production* and *Time* along the iterations. Figure 13 describes the behavior of the total number of feasible neighbors explored at each iteration for each of the tabu strategies considered. Figure 14 does the same for the total number of tabu neighbors; figure 15 treats the total number of neighbors that satisfied the aspiration rule; figure 16 shows the total oil production; and figure 17 depicts the total time per iteration.

Figure 18 presents more details about the total oil yield when each of the tabu strategies was exercised for the real instance. Figures 19, 20 and 21 do the same for the generated instances, 2W112S4B3, 3W95S5B3 and 4W130S5B3, respectively.

Figure 13 shows that the number of feasible neighbors visited during each iteration varies a lot, since an iteration is finished after y neighbors that improve the best solution are found. The only approach that does not present this pattern is **WE**. In the **ID** and **IA** approaches, it can be observed a tendency for the number of feasible neighbors to increase along with the computation. This indicates that it is easier to improve the best solution on the beginning of the computation, since we search for the first y best neighbors. Comparing figures 13 and 17, it is clear that the time per iteration is roughly proportional to the number of feasible neighbors. Concerning the total number of neighbors, table 14 shows that approaches **TabuPF, TabuPR, ID, IA** can explore a number of neighbors three orders of magnitude larger than approaches **WE, WI**. This was already expected, since these approaches are large scale neighborhoods. Furthermore, techniques **ID** and **IA** use a greedy strategy to select the best feasible neighbors. In the **WE** strategy, we only allow 20 seconds for the search of the best feasible neighbors. Figure 13.f shows that this technique is actually using the allowed time to find 7 such neighbors per iteration.

Instance	Approach	It	Neighbors	Tabu	AR	Production	Time
1W130S5B3	TabuPF	348	5404175	1762	9	260480500	3624
	TabuPR	368	6078448	390383	112	260955920	3619
	ID	216	5097314	482245	65	261797010	3621
	IA	234	5549670	569485	74	**262968780**	3616
	WE	169	1172	1645	0	247388220	3618
	WI	188	1882	174	4	250410270	3606
2W112S4B3	TabuPF	485	6437783	2446	15	250305872	3625
	TabuPR	494	6772949	562549	139	**250448195**	3631
	ID	185	1955786	345472	116	246309923	3639
	IA	153	1462175	262204	198	248792249	3616
	WE	176	708	1715	0	220362248	3609
	WI	200	3361	425	21	224523265	3618
3W95S5B3	TabuPF	307	5453469	1648	2	**238685695**	3644
	TabuPR	335	6999181	309804	68	**238685695**	3604
	ID	128	2247780	165676	109	233766103	3630
	IA	155	2891257	224596	65	238258503	3628
	WE	166	1164	1615	0	225755389	3614
	WI	184	1789	262	3	227267290	3605
4W130S5B3	TabuPF	331	5669485	1760	11	296412436	3622
	TabuPR	307	5232493	441899	98	294203512	3603
	ID	154	1568798	198769	156	297504129	3656
	IA	163	1919470	236697	425	**298814360**	3667
	WE	175	704	1705	0	276423927	3600
	WI	193	1752	316	15	280450182	3607

Table 14: Quantitative Data - H1

Figure 14 shows that the number of tabu neighbors tends to be higher on the final iterations when compared with the initial iterations, when strategies **ID** and **IA** were used. This happens because more neighbors are explored on the final iterations and because the tabu list contains fewer elements on the beginning.

Figure 15 indicates that the number of tabu neighbors that satisfies the aspiration rule does not follow a recognizable pattern. Note that strategy **WE** does not use the aspiration rule. Table 14 indicates that the number of neighbors considered tabu is high, given the total number of feasible neighbors that was visited, except for approach **TabuPF**, because its tabu criteria is the less restrictive among all the strategies. On the other hand, the number of tabu neighbors that satisfies the aspiration rule is very small for all approaches. It is important to recall that for approach **WE**, column *Tabu* indicates how many times a well could not have all its activities freed in an iteration due to the fact that the well was tabu. This explains why the number of tabu wells is greater than the total number of neighbors when technique **WE** was used in all instances. See table 14.

The most interesting results concern the evaluation of the production curve as depicted in figures 18, 19, 20 and 21. Approach **IA** gave the best production for instances 1W130S5B3 (the real instance) and 4W130S5B3, while approach **TabuPR** gave the best production for the 3W95S5B3 and 2W112S4B3 instances. Approaches **WE** and **WI** got into an inferior production plateau and were not able to escape from it in all tested instances. An explanation for this behavior could be that these approaches were not able to explore a large number of neighbors, and the explored neighbors were unable to provide reasonable improvements over the oil production. These figures show a desirable behavior of large scale neighborhoods. Strategies **ID** and **IA** showed a steep rise in the oil production at the very beginning in all instances, to a lesser extent for strategy **ID** in instance 3W95S5B3. This was particularly intense on the real instance, as can be seen from figure 18. We believe that the quality os the neighbors generated by strategies **ID** and **IA** was superior, while still maintaining a competitive computational time. Even though **TabuPR** gave the best production for two instances, the results obtained by the **IA** strategy on those instances were very close to the best ones.

Figure 16 shows that a production plateau was not reached when **TabuPF, TabuPR** and **IA** were used. In order to verify if it was possible to increase the production in these cases, the strategies were allowed to run for 3 more hours. A plateau of 363.5 millions of unities was then reached by all of them and could not be improved.

To observe the behavior of the approaches when a poorer initial solution was used, all techniques were exercised with an initial solution obtained using **H2**. Table 15 and figures 22.b, 23.b, 24.b, 25.b are the correspondents of table 14 and figures 18, 19, 20, 21, respectively, when **H2** was used instead of **H1**. Figures 22.a, 23.a, 24.a,

25.a explore the initial 360 seconds of these computations.

Instance	Approach	It	Neighbors	Tabu	AC	Production	Time
1W130S5B3	TabuPF	435	4096832	1726	13	243419100	3641
	TabuPR	458	5377802	344077	121	243639100	3614
	ID	171	5222193	452063	0	242923630	3603
	IA	181	5549303	452396	0	**244595390**	3602
	WE	172	1166	1675	0	241947730	3602
	WI	218	3618	424	6	243724840	3600
2W112S4B3	TabuPF	475	5272845	2137	11	223398783	3623
	TabuPR	486	6057373	447554	129	223389183	3657
	ID	212	1444141	191602	235	232061191	3651
	IA	207	2066677	306046	217	**230394281**	3622
	WE	174	1164	1695	0	216710680	3615
	WI	250	3544	661	34	225733988	3618
3W95S5B3	TabuPF	428	4776629	2393	22	227393876	3621
	TabuPR	466	6717009	198437	119	227536172	3613
	ID	222	2319646	320605	119	**229981247**	3634
	IA	202	2727995	370660	266	229520098	3634
	WE	181	1172	1765	0	223389186	3612
	WI	222	3941	452	8	227100961	3602
4W130S5B3	TabuPF	445	4133240	1694	21	272924096	3629
	TabuPR	457	4280146	271755	141	274236257	3608
	ID	203	1184770	145761	220	**283119592**	3663
	IA	209	1475233	178982	220	**283119592**	3647
	WE	180	708	1755	0	273515507	3616
	WI	224	3624	343	10	281009822	3609

Table 15: Quantitative Data - H2

Table 15 shows that the number of feasible and tabu neighbors, as well as the number of neighbors that satisfies the aspiration rule is the same as in the previous cases. The production curves deserve closer attention. Approach **IA** gave the best production for instances 1W130S5B3, 3W95S5B3 and 4P130S5B3, while approach **ID** gave the best production for instance 2W112S4B3. Note that the pure tabu search approaches (**TabuPF** and **TabuPR**) did not yield the best production for none of these instances. Furthermore, in the previous case, the gap between the production obtained by the pure tabu search approach and the hybrid approaches was less than 1%. In this case, for instances 2W112S4B3 and 4W130S5B3 the gap was near 4% in favor of the hybrid approach. Another interesting result is, in the previous case, the production obtained with strategy **WI** was not as good when compared to the production obtained with strategies **TabuPF**, **TabuPR**, **ID** and **IA**. In this new

case, the production obtained with approach **WI** was higher than the production obtained by the pure approaches **TabuPF** and **TabuPR** in all instances, except in instance 3W95S5B3. But in this instance, the production obtained with the **WI** approach was less than 1% poorer.

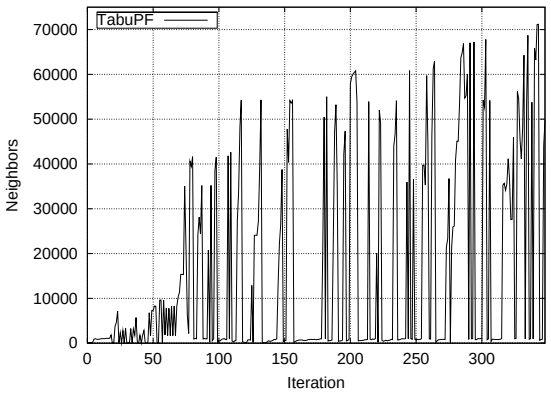
Figures 22.a, 23.a, 24.a, 25.a details the first 360 seconds of computation of each instance. Note how quickly each approach overcomes a poor initial solution. On the former case, when **H1** was used, **IA** improved the initial solution quickly. In this case, strategy **WE** does it faster than any other in all instances. The pure tabu search approaches are only able to reach this level of production after more than 360 seconds. Unfortunately, the **WE** approach gets into an inferior plateau and is not able to escape from it.

Another pattern that can be observed is that when all strategies use initial solutions obtained by the **H2** method, they get stuck in an inferior plateau when compared with the plateau reached when the **H1** was used. This indicates that a poor initial solution probably leads to poorer final results. The relevant aspect of the results obtained by using **H2**, however, is fact that the hybrid approaches improved a poor initial solution more quickly than the pure approaches and gave best production values.

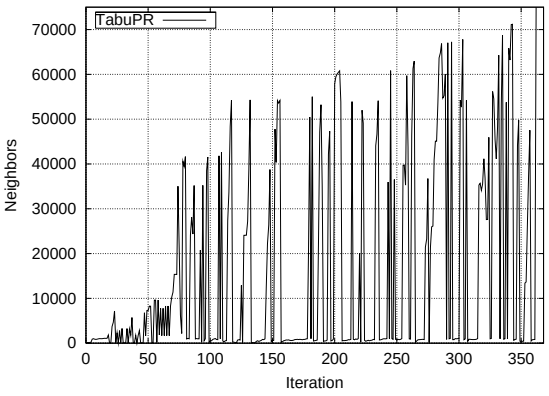
Figures 26.a and 26.b show the initial solution, the best oil production and the upper bound for all four tested instances, when **H1** and **H2** are applied, respectively. Table 16 shows by which percentage the best solution improved the initial one and shows by which percentage the best solution is under the upper bound. Clearly, there was a bigger improvement between the initial and the best solution when **H2** was applied. Note, however, that when considering **H2**, a poor initial solution was used. The gap between the best solution and the upper bound was also bigger for this case. Nevertheless, the most important information conveyed by this table is that the best solutions are less than 9% from the upper bound for instances 1W130S5B3, 2W112S4B3, 3W95S5B3 and is at 10.7% from the upper bound for instance 4W130S5B3. This indicates that the best solutions are already very good ones.

Instance	H1		H2	
	Initial Solution	Upper Bound	Initial Solution	Upper Bound
1W130S5B3	6.8%	8.5%	18.4%	14.9%
2W112S4B3	13.9%	7.55%	28.9%	17.5%
3W95S5B3	5.89%	7.2%	23.84%	10.6%
4W130S5B3	8.06%	10.7%	29.7%	15.4%

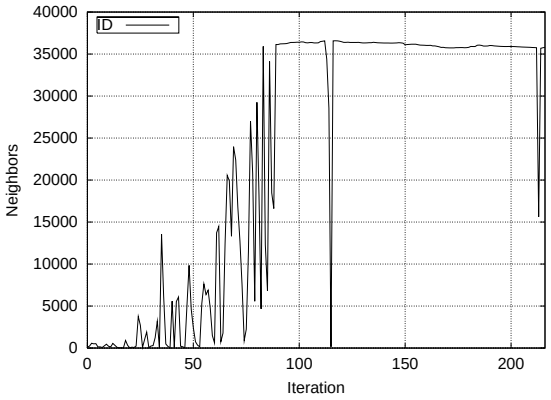
Table 16: Distance among the initial solution, best solution and upper bound



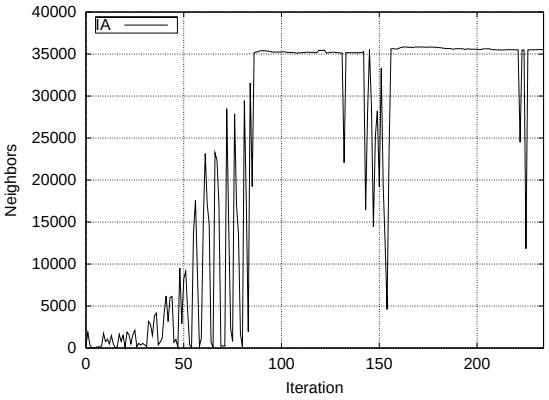
(a) TabuPF



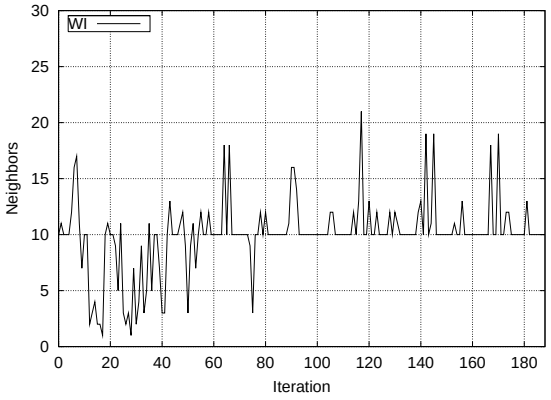
(b) TabuPR



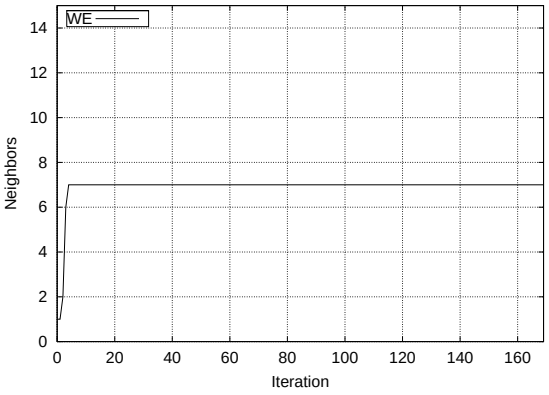
(c) ID



(d) IA

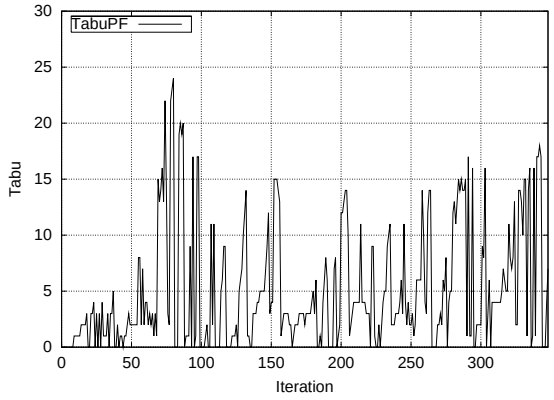


(e) WI

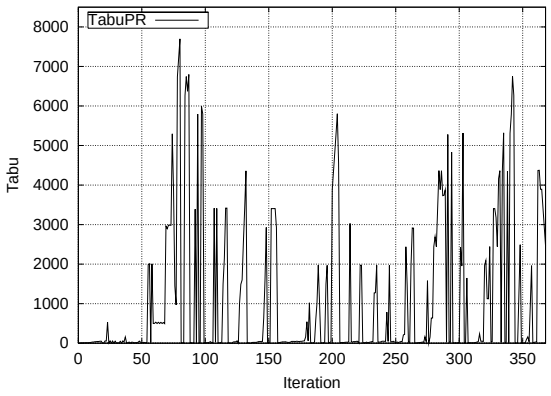


(f) WE

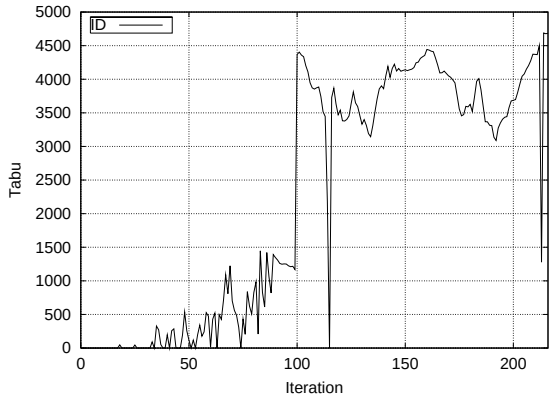
Figure 13: Feasible Neighbors X Iteration - Real Instance



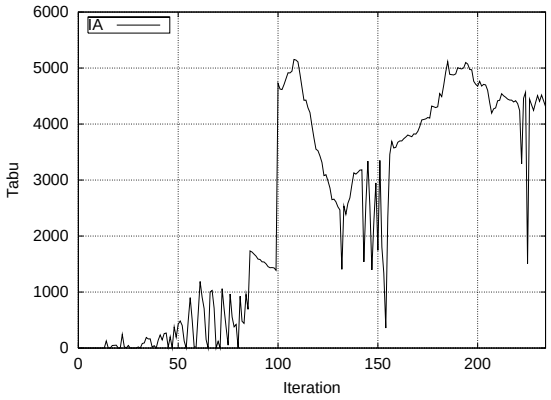
(a) TabuPF



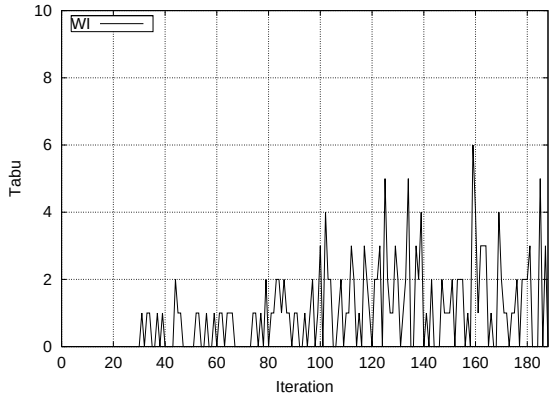
(b) TabuPR



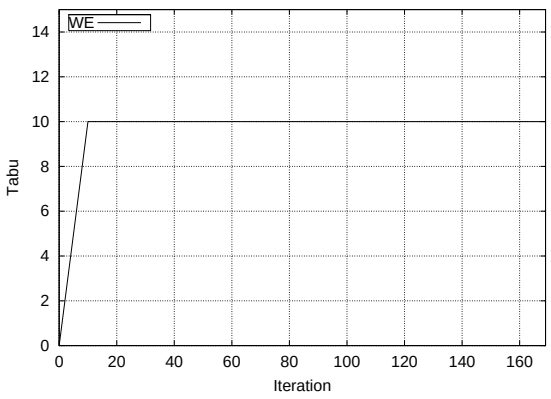
(c) ID



(d) IA

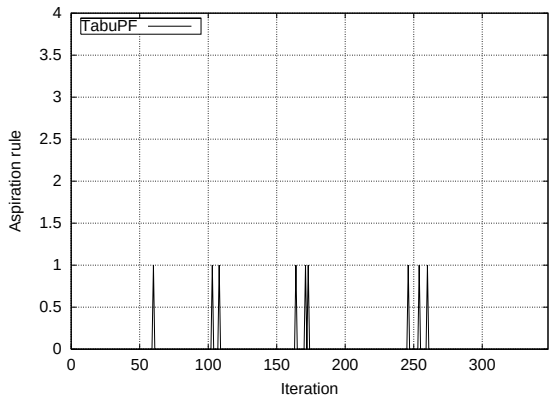


(e) WI

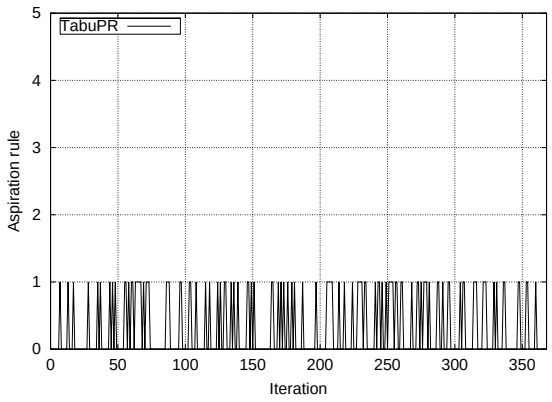


(f) WE

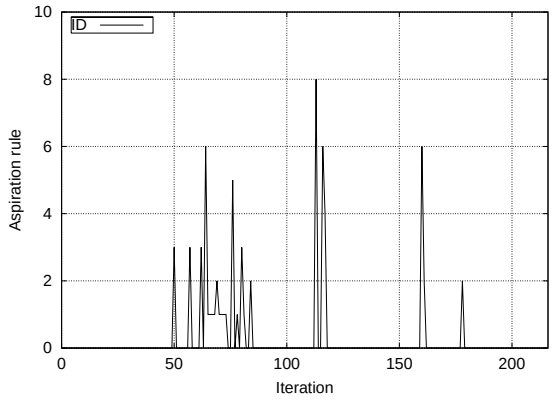
Figure 14: Tabu Neighbors X Iteration - Real Instance



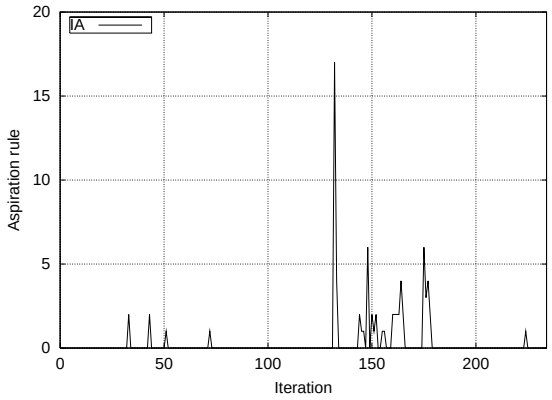
(a) TabuPF



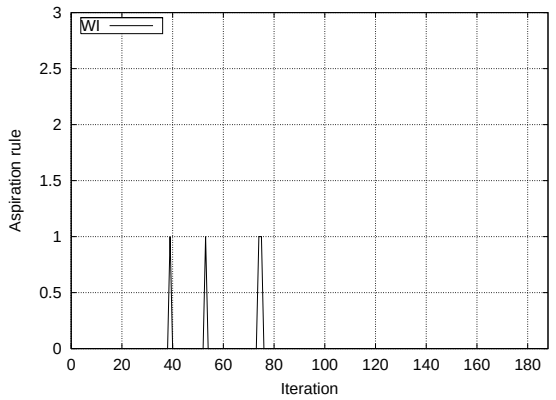
(b) TabuPR



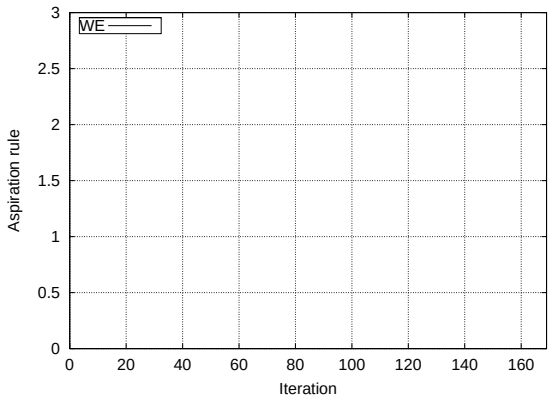
(c) ID



(d) IA

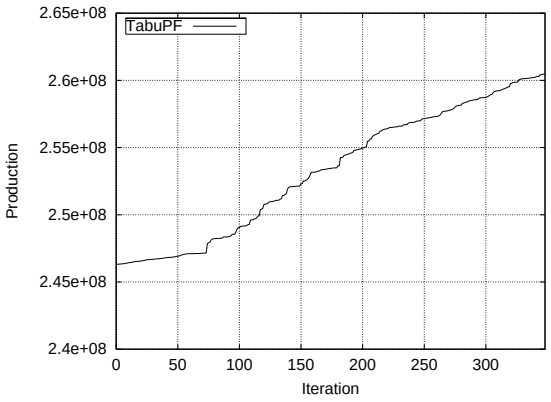


(e) WI

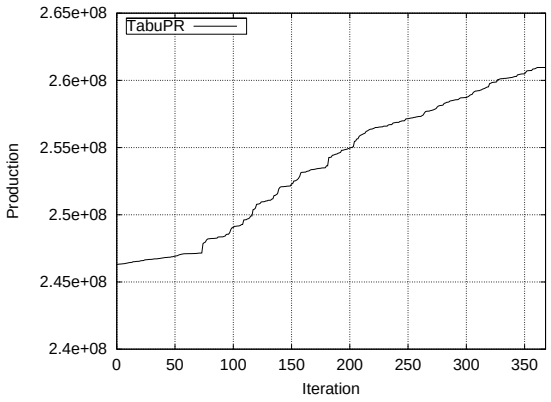


(f) WE

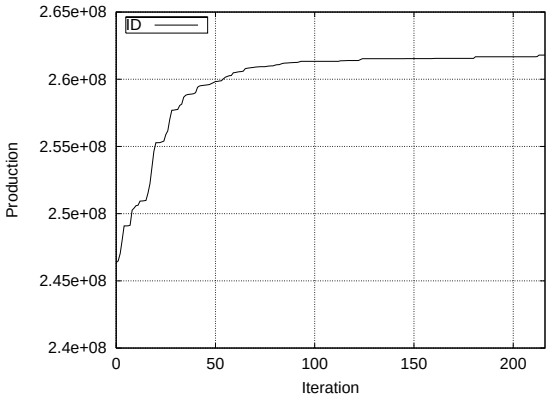
Figure 15: Aspiration Rule Neighbors X Iteration - Real Instance



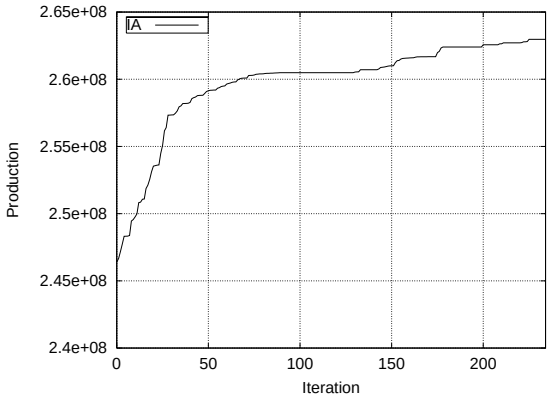
(a) TabuPF



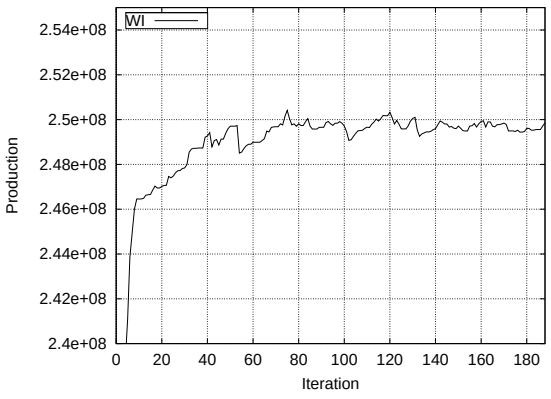
(b) TabuPR



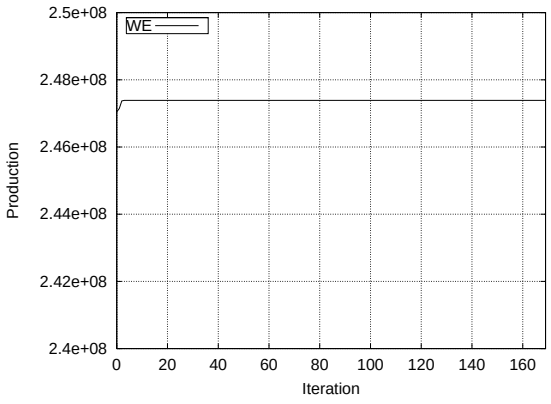
(c) ID



(d) IA

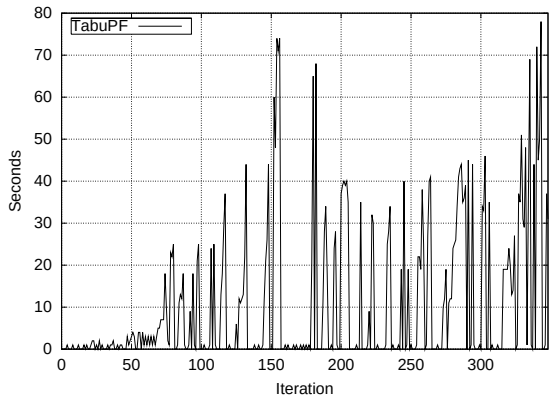


(e) WI

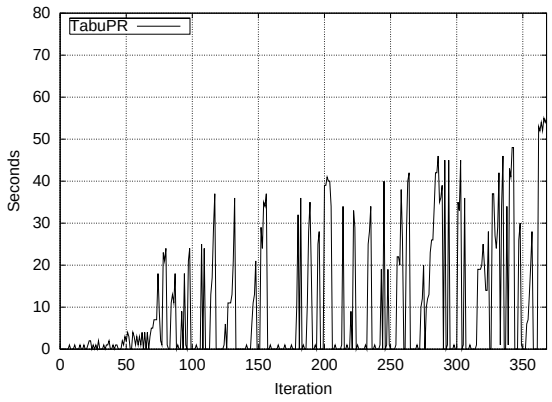


(f) WE

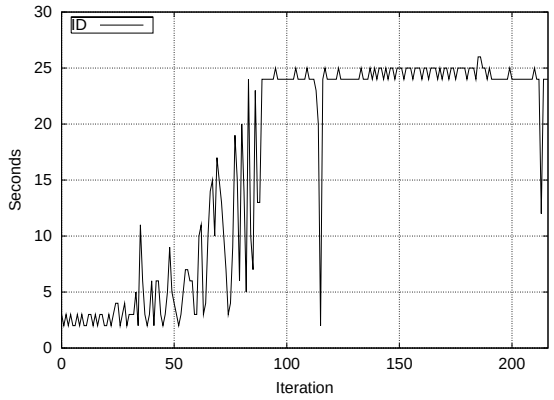
Figure 16: Production X Iteration - Real Instance



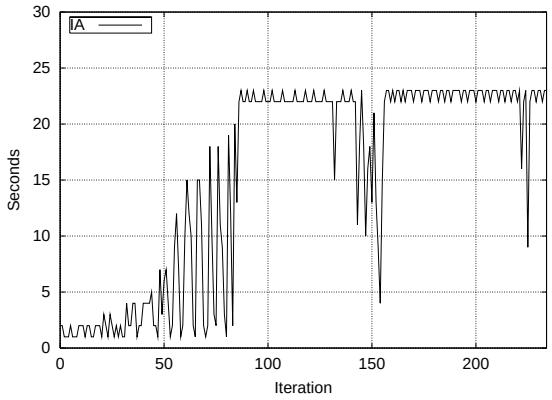
(a) TabuPF



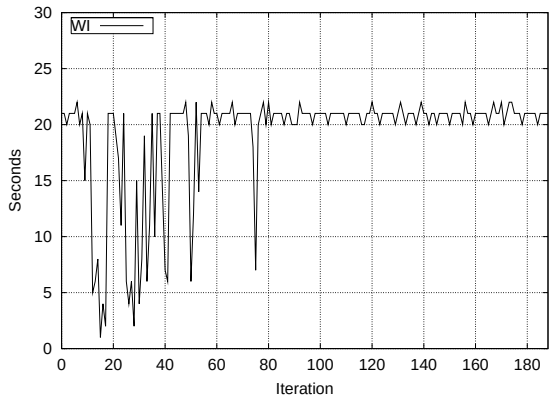
(b) TabuPR



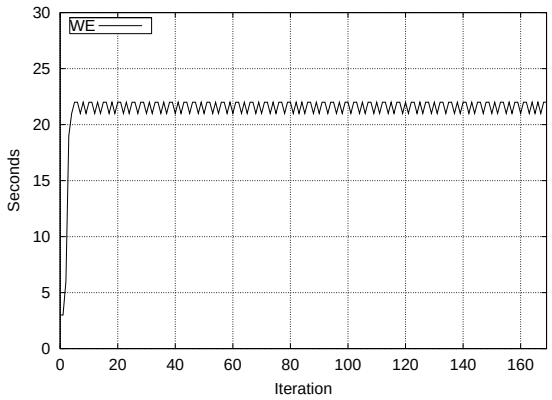
(c) ID



(d) IA



(e) WI



(f) WE

Figure 17: Time X Iteration - Real Instance

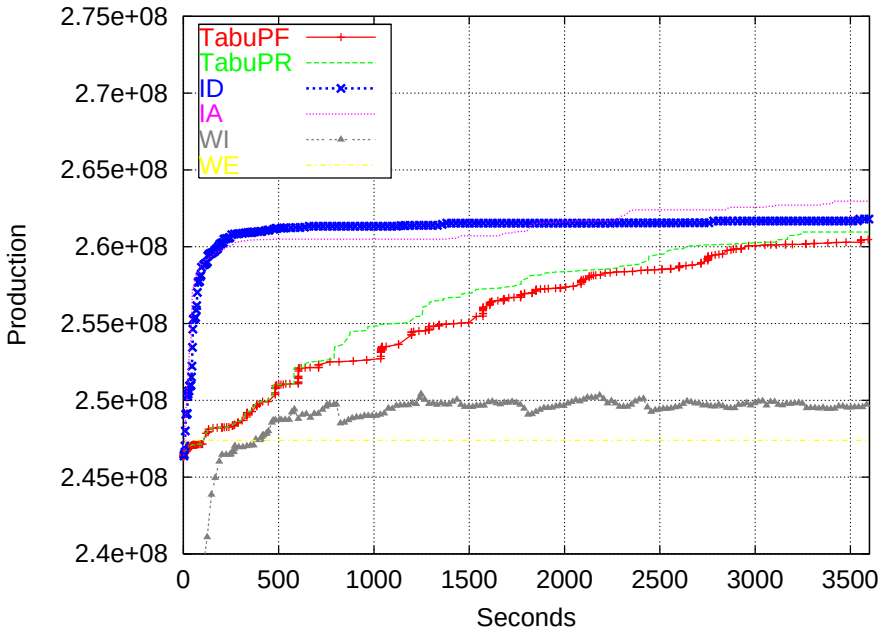


Figure 18: Production X Time - 1W130S5B3 (Real Instance) - H1

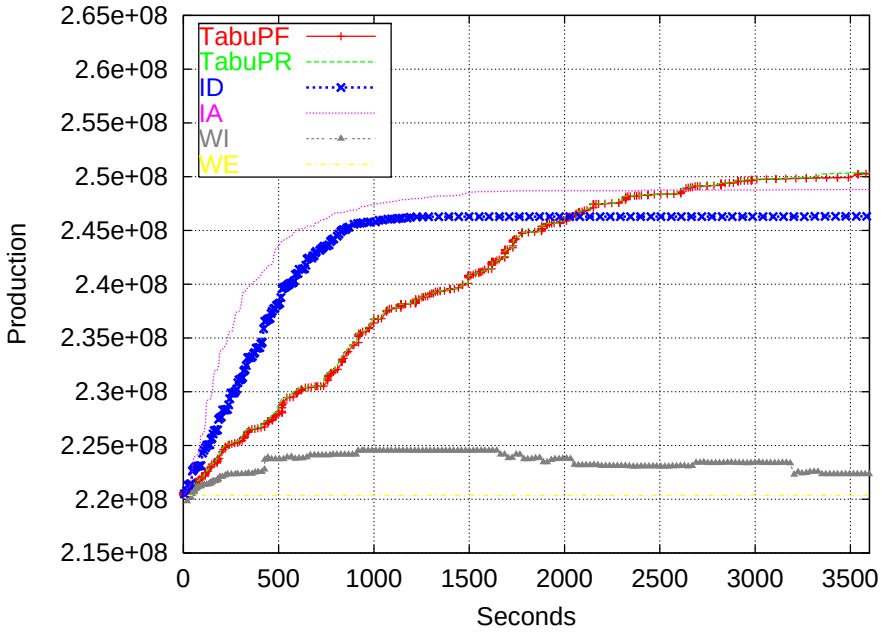


Figure 19: Production X Time - 2W112S4B3 - H1

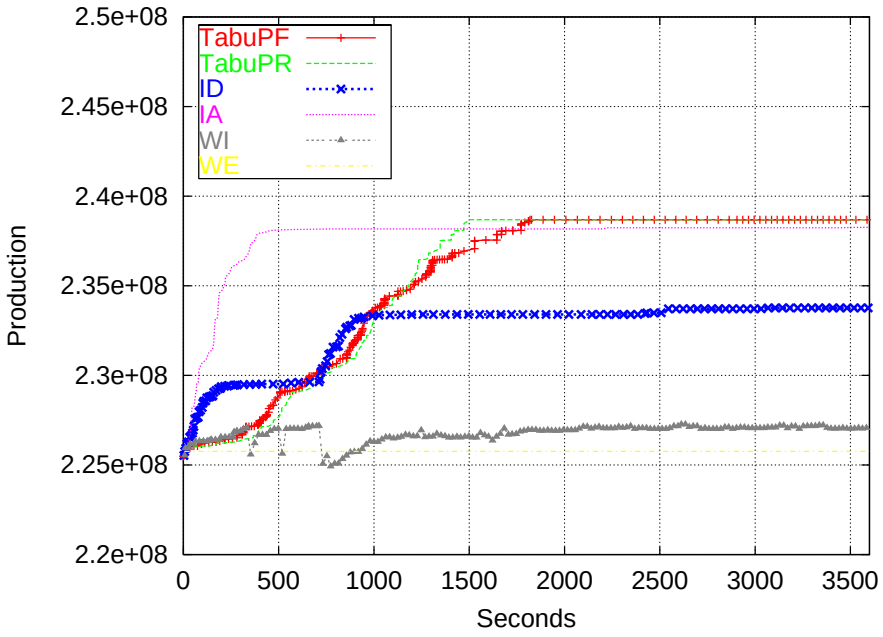


Figure 20: Production X Time - 3W95S5B3 - H1

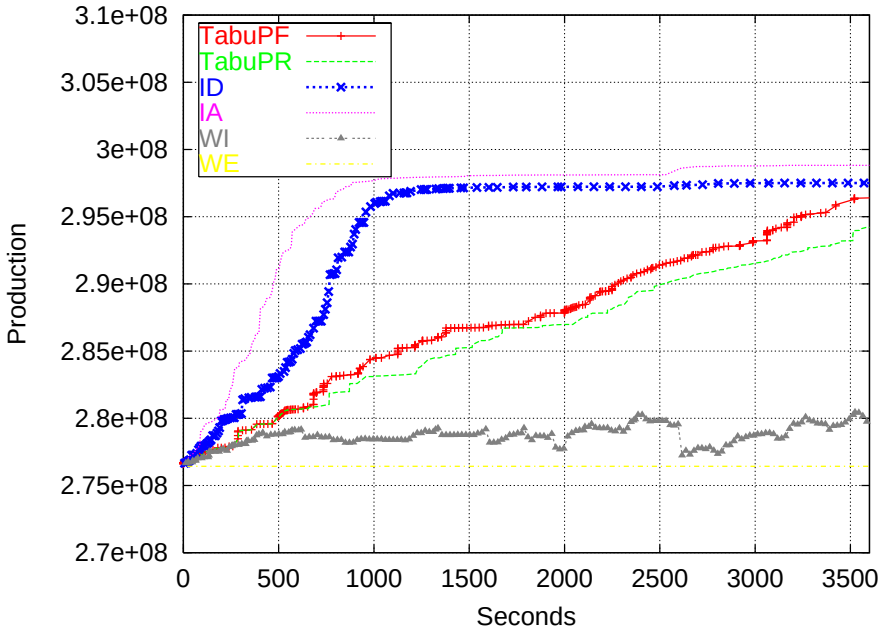
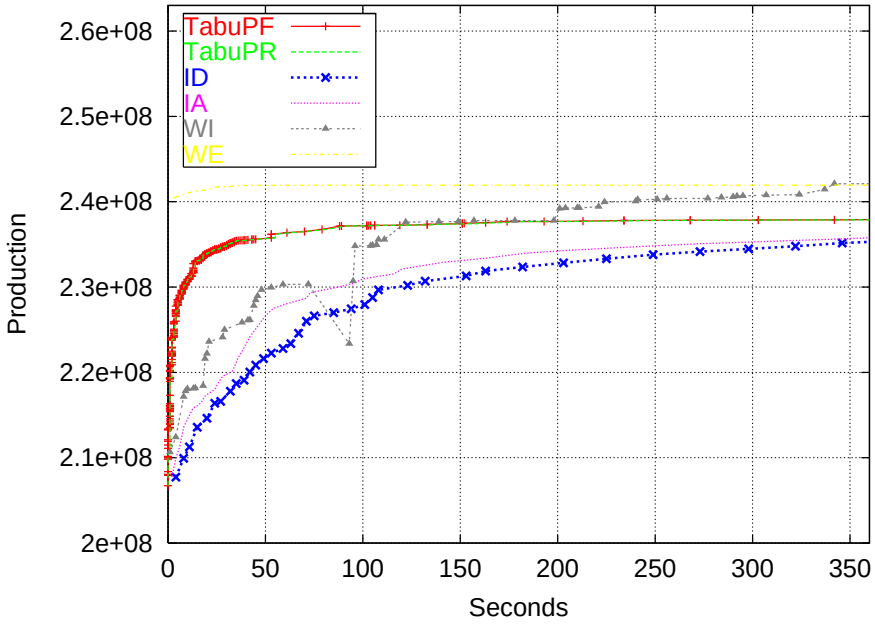
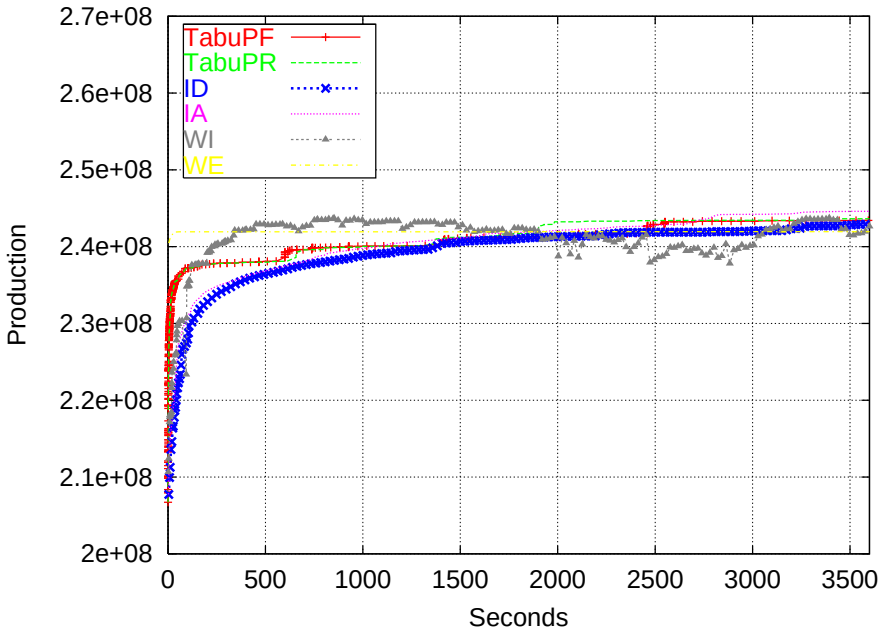


Figure 21: Production X Time - 4W130S5B3 - H1

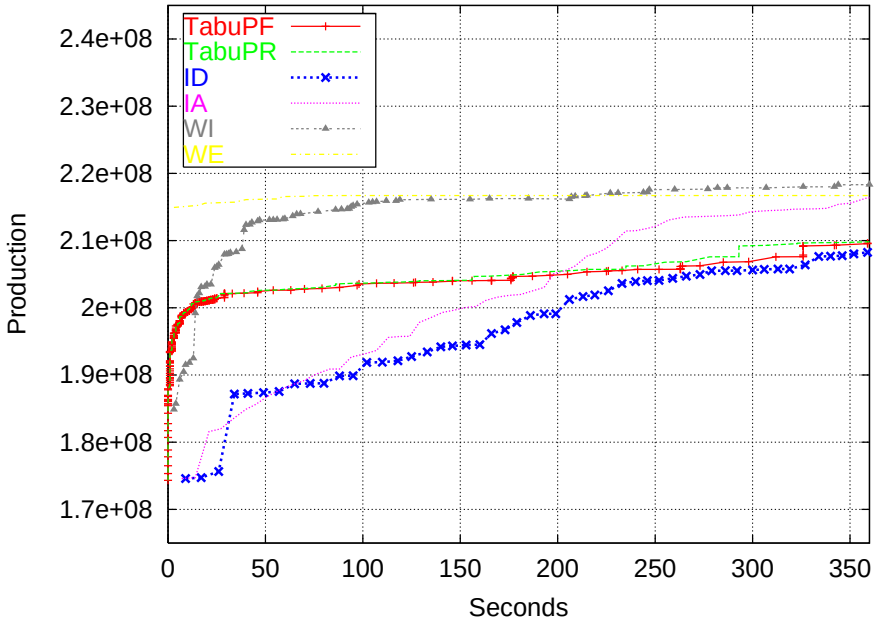


(a) Time=0...360

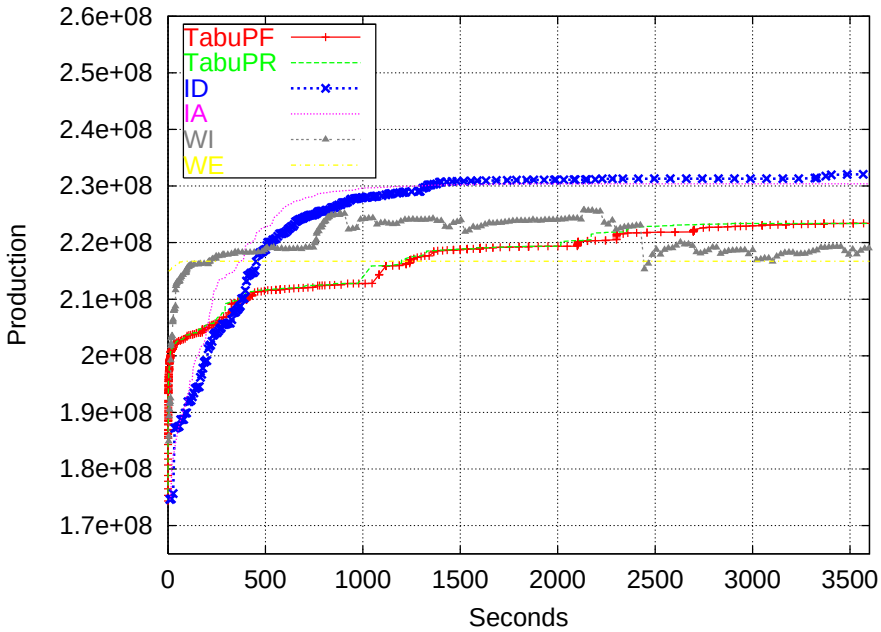


(b) Time=0...3600

Figure 22: Production X Time - 1W130S5B3 (Real Instance) - H2

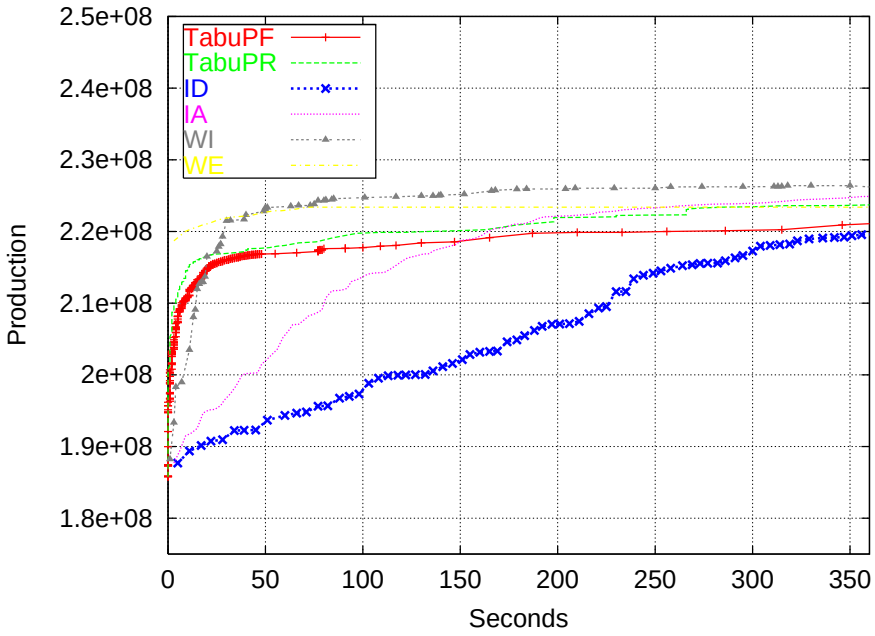


(a) Time=0...360

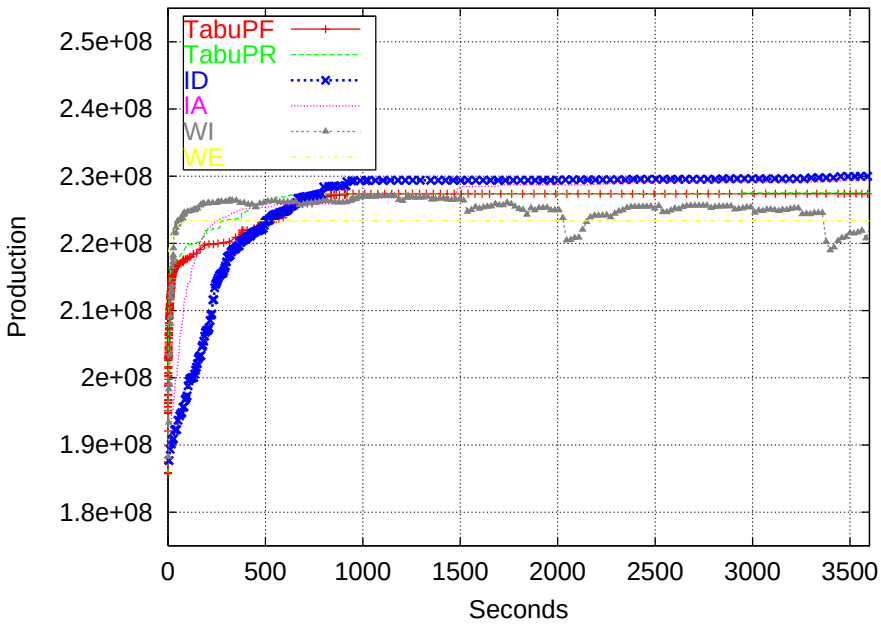


(b) Time=0...3600

Figure 23: Production X Time - 2W112S4B3 - H2

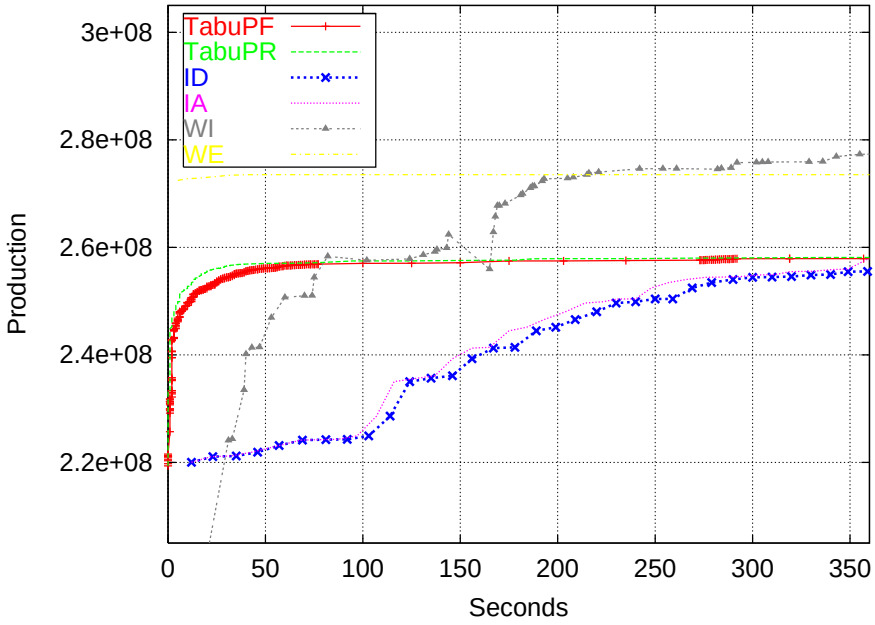


(a) Time=0...360

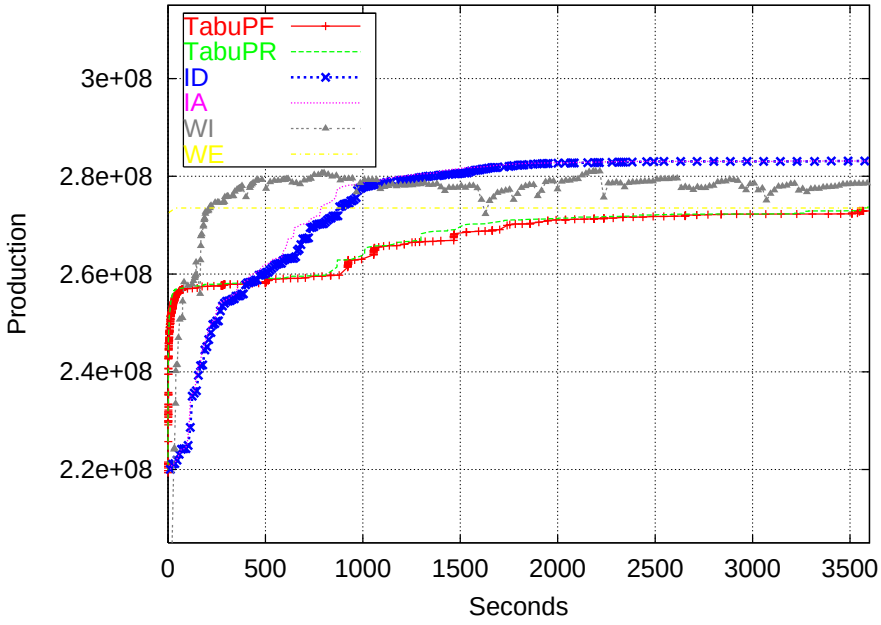


(b) Time=0...3600

Figure 24: Production X Time - 3W95S5B3 - H2

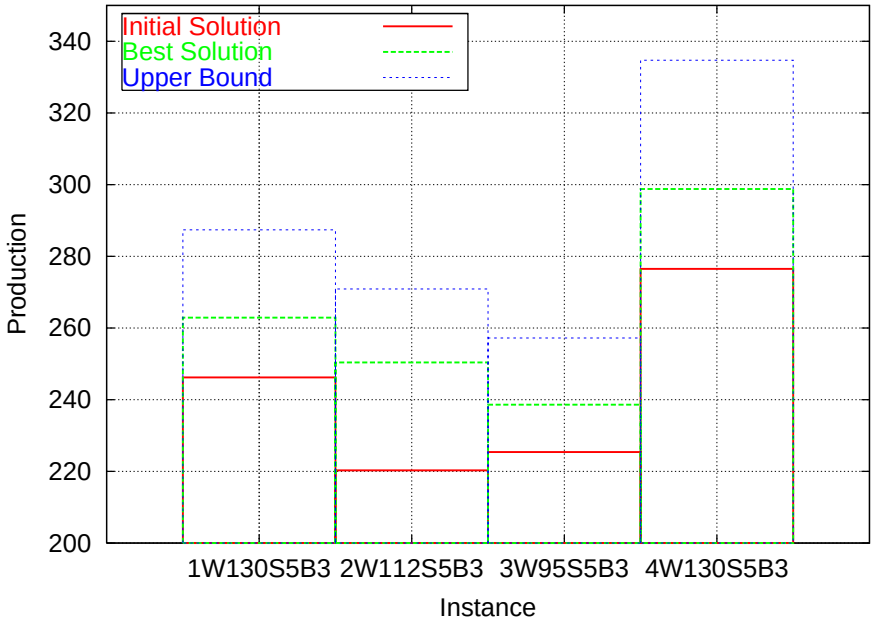


(a) Time=0...360

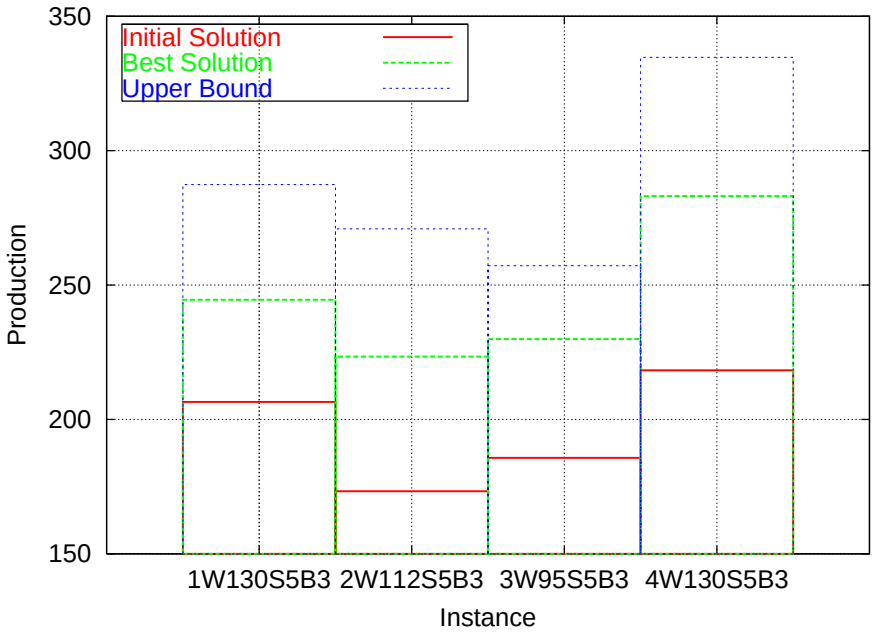


(b) Time=0...3600

Figure 25: Production X Time - 4W130S5B3 - H2



(a) H1



(b) H2

Figure 26: Initial solution, best solution and Upper Bound

7 Sensibility Analysis

In this section a sensibility analysis is presented in order to verify the behavior of our techniques when the instances are perturbed.

Two kinds of perturbation are considered. As discussed in section 5, there are more derricks than boats and many more activities that require the former than the latter resource. In addition, the resources needed by all wells follow the pattern

$$\text{Derrick} \rightarrow \text{Boat} \rightarrow \text{Derrick}.$$

Also, the boats can be considered the bottleneck in the process of developing a well. Moreover, the processing time of all activities which require a boat is in the interval of $[5, 17]$ days. These informations were crucial to the development of approach **Upper3**. The processing time of those activities that require a boat will be perturbed. The original processing time of these activities will be replaced by processing times generated using an uniform distribution between the intervals $[1, 30]$, $[1, 60]$ and $[1, 90]$. This procedure originated the variant instances that we call **Act**, with its three subcases **Act30**, **Act60**, **Act90** that correspond, respectively, to the new three intervals for activities that require a boat.

In all the instances considered in this paper, all derricks can execute all activities that require this type of resource. The same occurs with the boats. Note that constraint 3 of the original problem, described in section 2 says that there may exist instances where not all resources of a certain type are able to perform all the activities that require that type of resource. The other perturbation we considered over the original instances was that each activity was associated with a subset of the resources that are able to perform it. So, in the perturbed instances, to each activity is attributed x specific resources among all the resources that have the type compatible with the activity. The x resources are chosen randomly. This procedure created the variant we call **Res**. This variant also has three subcases, named **Res1**, **Res2**, **Res3**, that correspond to subcases when x assumes the values 1, 2 and 3, respectively.

In this section, only strategy **H1** will be considered to generated the initial solutions. Furthermore, only approaches **TabuPR, IA, WE, WI** will be tested, since strategies **TabuPF** and **ID** displayed a poorer performance when compared to strategies **TabuPR** and **IA**.

Table 17 presents the initial solution to the perturbed instances. The production is in millions of unities.

The upper bound values for the perturbed instances that follow the **Res** variants are the same as the values presented in section 5 for the original instances. Note that the relevant information for these models are the activities processing time and the number of derricks and boats. All these values remained the same on the perturbed instances.

Instance	Variant					
	Res1	Res2	Res3	Act30	Act60	Act90
1W130S5B3	229.8	240.2	243.4	243.9	227.0	168.7
2W112S4B3	206.2	216.7	216.2	216.5	205.2	163.3
3W95S5B3	218.1	222.8	224.0	220.7	205.2	133.6
4W130S5B3	262.2	272.7	273.3	270.9	238.8	186.9

Table 17: Initial solution - H1

On the other hand, as the activities processing time were altered on the perturbed instances that follow the **Act** variant, new bounds were computed for these pertubed instances. Table 18 is similar to table 11. The instance used was 4W130S5B3 - variant **Act90**. As in table 11, column *Constraints* is the number of constraints, column *Variables* is the number of binary variables, column *Const.NZ* is the number of nonzeros coefficients in the constraints and column *Obj.NZ* is the number of nonzeros coefficients in the objective function. The values in this table have the same order of magnitude as the correspondig values, even considering that table 11 was created using the real instance.

Approach	Constraints			Variables	Const.NZ	Obj.NZ
	Less	Greater	Equal			
Upper1	420	-	2463	36000	5321013	35760
Upper2	420	-	2463	36000	777093	35760
Upper3	962	240	6497	10800	13656007	35760

Table 18: Model quantitative data - Upper Bound - Instance $4W130B5S3$ - Variant Act90

Table 19 is similar to table 12, and table 20 is similar to table 13. In the latter case, variant **Act90** was used. Table 19 shows the best bound obtained for the production by each approach. The value between parenthesis indicates the percentage by which approaches **Upper1**, **Upper2** and **Upper3** improved upon the values obtained for approach **Upper0**.

As expected, the bound values shown in table 19 are smaller than the corresponding bounds on table 12. Note that the activities processing time are higher in the pertubed case. As in the original instances, the computation using strategy **Upper1** was limited in one hour and did not find the optimal solution. Strategy **Upper2** found the optimal solution, and strategy **Upper3** ran out of memory and was not able to find a feasible integer solution.

Table 20 shows the number of nodes and the gap between the upper bound and the

Instance	Variant	Upper0	Upper1	Upper2	Upper3
1W130S5B3	Act30	366.6	307.0 (16.2%)	297.4 (18.8%)	284.0 (22.5%)
	Act60	363.7	293.4 (19.3%)	282.5 (22.3%)	280.6 (22.8%)
	Act90	357.6	263.9 (26.2%)	250.3 (30.0%)	253.8 (29.0%)
2W112S4B3	Act30	361.9	296.2 (18.1%)	287.3 (20.6%)	267.9 (25.9%)
	Act60	358.9	284.4 (20.7%)	273.9 (23.6%)	265.7 (25.9%)
	Act90	354.5	255.8 (27.8%)	243.6 (31.3%)	248.5 (29.9%)
3W95S5B3	Act30	316.2	275.9 (12.7%)	268.0 (15.4%)	255.8 (19.1%)
	Act60	358.9	267.1 (25.5%)	258.2 (28.0%)	253.9 (29.2%)
	Act90	308.9	248.3 (19.6%)	237.2 (23.2%)	241.4 (21.8%)
4W130S5B3	Act30	453.2	364.2 (19.5%)	352.3 (22.2%)	332.0 (26.7%)
	Act60	449.3	344.0 (30.2%)	330.2 (26.5%)	326.8 (27.2%)
	Act90	443.8	313.6 (29.3%)	297.8 (32.4%)	302.9 (31.7%)

Table 19: Upper Bound - Activities processing time variant

Instance	Upper 1		Upper 2		Upper 3	
	Node	Gap	Node	Gap	Node	Gap
1W130S5B3	700	2.10%	1	0	1	∞
2W112S4B3	600	3.37%	1	0	1	∞
3W95S5B3	2000	2.57%	1	0	1	∞
4W130S5B3	500	3.27%	1	0	1	∞

Table 20: Computational quantitative data - Upper Bound - Variant Act90

best solution when variant **Act90** was used in all tested instances. The unexpected behavior was that the bound obtained using strategy **Upper2** was tighter than the bound obtained using **Upper3**. There are at least two possible explanations for this fact. This may have occurred because **Upper3** ran out of memory and was unable to improve the bound, while **Upper2** found the optimal solution. Or this may have occurred because with **Upper2** preemptions and parallel activities are not allowed while they are permitted with **Upper3**. Since this was not a problem when the original instances were used, the problem may have arisen due to the large variability of the activities processing time on the perturbed instances.

As for the computational results, table 21 is similar to table 14 and shows some data for all **Act** variants. The columns of this table show the instance identification, the approach identification, the total number of iterations, the total number of feasible neighbors, the total number of tabu neighbors, the total number of tabu neighbors that satisfied the aspiration rule, the best value for the production and the total execution time.

This table shows that the total number of feasible and tabu neighbors as well as the total number of neighbors that satisfies the aspiration rule, are of the same order of magnitude obtained for these values when using the original instances. Except for approach **WI**, where these values were approximately multiplied by 2 in instances 1W130S5B3, 3W95S5B3 and 4W130S5B3.

Figures 27, 28 and 29.b are similar to figure 18 for the real instance. They plot the total oil production when variants **Act30**, **Act60** and **Act90**, respectively, are exercized. Each figure shows the production when each of the **TabuPR**, **IA**, **WI** and **WE** strategies were used. Figures 30, 31 and 32.b do the same for instance 2W112S4B3, the corresponding figure for the unperturbed case being figure 19. Figures 33, 34 and 35.b corresponde to instance 3W95S5B3 and the unperturbed case is shown in figure 20. Finally, figures 36, 37 and 38.b corresponde to instance 4W130S5B3 and are similar to figure 21.

The behavior for variants **Act30** and **Act60** was very similar to their behavior when the unperturbed instances were considered. When variant **Act90** is considered, however, the behavior was different. In order to get more details for the beginning of the computation, figures 29.a, 32.a, 35.a and 38.a depict the first 360 seconds of computation for, respectively, instances 1W130S5B3, 2W112S4B3, 3W95S5B3 and 4W130S5B3 when variant **Act90** is considered.

It can be observed from these figures that the **IA** strategy produced the best solution for all instances and all variants. When variant **Act90** is considered, approach **WI** gave better results than the pure approach **TabuPR** consistently on all instances. In fact, the solution obtained using **TabuPR** was much inferior than the best solution found, showing that this strategy is not robust. As occurred with the original instances, when poor initial solutions were considered, approach **WE** quickly

Instance	Variant	Approach	It	Neighbors	Tabu	AR	Production	Time
1W130S5B3	Act30	TabuPR	248	5848316	352747	63	251490660	3627
		IA	237	5484254	783616	228	**254289300**	3607
		WE	175	880	1705	0	243813630	3604
		WI	199	3519	180	6	248671800	3620
	Act60	TabuPR	334	4640070	430279	59	244399700	3649
		IA	278	5331636	757380	492	**249390940**	3616
		WE	177	879	1725	0	233244810	3603
		WI	213	3383	216	9	240629260	3609
	Act90	TabuPR	907	4641809	440276	224	213631730	3601
		IA	645	5503132	754402	2697	**223935480**	3600
		WE	176	878	1715	0	199547050	3605
		WI	262	3350	412	48	213792920	3601
2W112S4B3	Act30	TabuPR	485	5874813	520058	122	240914193	3607
		IA	324	5633385	883573	487	**242807762**	3600
		WE	177	881	1725	0	217363213	3608
		WI	211	3472	368	14	224353344	3618
	Act60	TabuPR	426	4640689	501234	126	227219755	3621
		IA	310	5664917	891038	468	**228336202**	3613
		WE	177	880	1725	0	213016072	3602
		WI	227	3384	382	32	223265432	3612
	Act90	TabuPR	734	4384954	465444	217	206954958	3604
		IA	509	5734307	911740	1566	**219941934**	3597
		WE	180	879	1755	0	196999164	3606
		WI	269	3382	369	48	207684901	3616
3W95S5B3	Act30	TabuPR	258	4906162	334880	57	229788564	3644
		IA	296	6571118	896624	361	**236134527**	3600
		WE	178	887	1735	0	222951271	3612
		WI	199	3771	177	7	224902175	3613
	Act60	TabuPR	353	6096266	481720	68	223263212	3614
		IA	315	6451718	997282	440	**231285153**	3607
		WE	178	884	1735	0	217123963	3601
		WI	233	3750	318	12	223601108	3615
	Act90	TabuPR	1115	5494002	441047	283	204158392	3619
		IA	619	6003403	974865	1776	**215480227**	3598
		WE	179	886	1745	0	194854023	3617
		WI	258	3422	566	34	206585222	3616
4W130S5B3	Act30	TabuPR	329	4650514	383225	76	287747684	3642
		IA	247	4746251	525334	914	**290475965**	3609
		WE	178	880	1735	0	273238548	3616
		WI	211	3433	206	9	279779931	3619
	Act60	TabuPR	388	4764056	550489	101	267388916	3644
		IA	265	4552634	600832	521	**274690795**	3612
		WE	179	880	1745	0	261125166	3612
		WI	250	3428	245	21	274013147	3604
	Act90	TabuPR	857	4415545	212685	262	237760960	3695
		IA	574	4934794	688041	2275	**265915404**	3608
		WE	176	875	1715	0	241748970	3601
		WI	281	3374	330	48	254766742	3607

Table 21: Quantitative Data - Variant: Activities duration

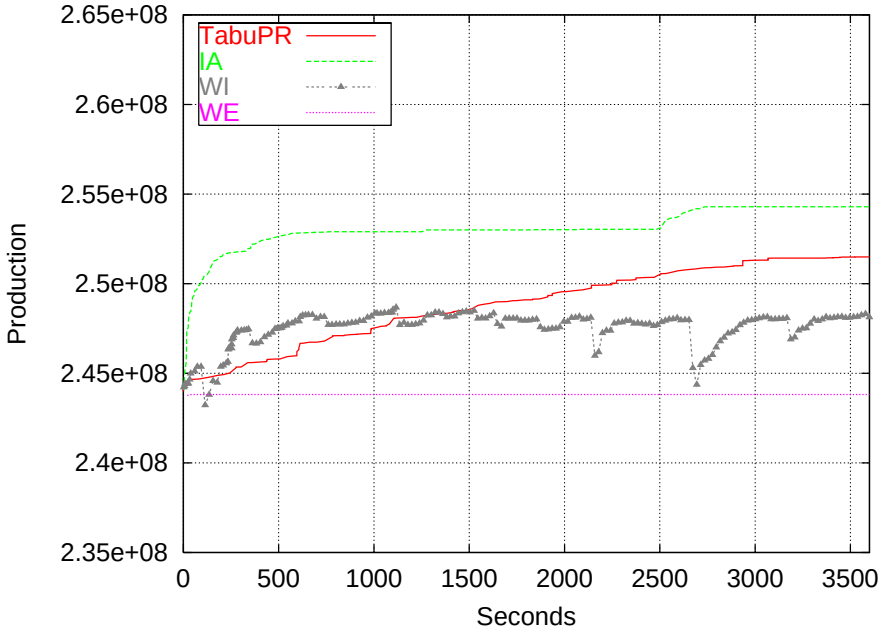


Figure 27: Production X Time - 1W130S5B3 (Real Instance) - Variant Act30

repaired the initial solutions, but, again, got stuck in a production plateau.

Table 22 shows the same kind of data as table 16. The**Act** variants were used to produce the results shown in this table. Observing this table, it can be inferred that the initial solution algorithm could not generate good solutions for the variant **Act90**. But the hybrid approach was able to reach a good solution for all instances, since the gap between the best solution and the upper bound was around 10% for all instances, in this case.

Table 23 repeats table 14 using the pertubed **Res** variants. This table shows that the total number of feasible and tabu neighbors, and the total number of neighbors that satisfied the aspiration rule are of the same order of magnitude as those values obtained with the original instances using strategies **TabuPR** and **WE**. For the **WI** approach, these values are higher in the pertubed case, for all instances. When using approach **IA**, these values are higher in this perturbed case for instances 2W112S4B3 and 3W95S5B3.

As was done for the **Act** variants, we used the **Res** pertubed instances derived from all four original test instances in order to observe the behavior of the **TabuPR**, **IA**, **WI** and **WE** strategies on these perturbed instances. The production obtained with the real perturbed instance is depicted in figures 39.b, 40.a and 41. For the generated 2W112S4B3 instance, the corresponding plots are shown in figures 42.b, 43 and 44. The set of figures 45.b, 46 and 47, shows the data when using instance

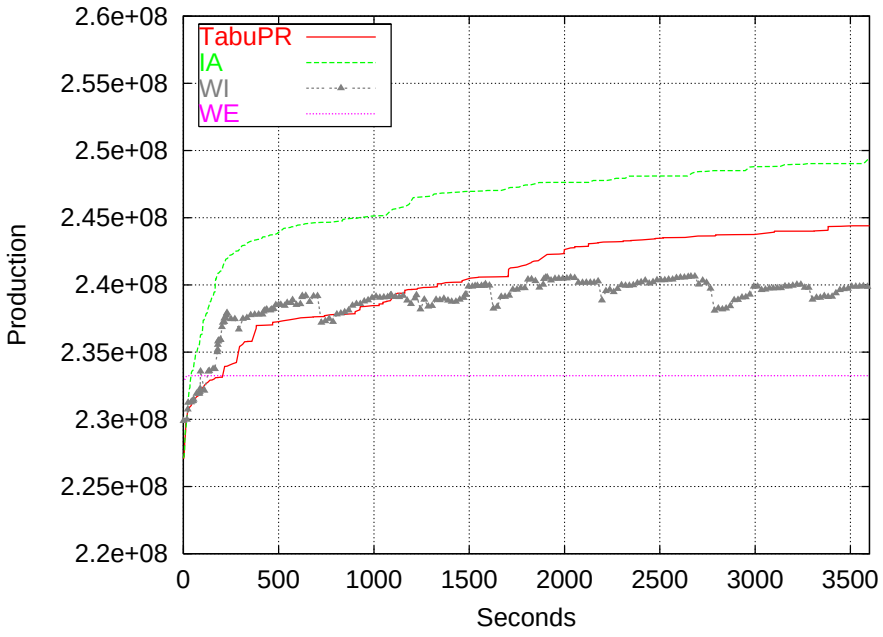
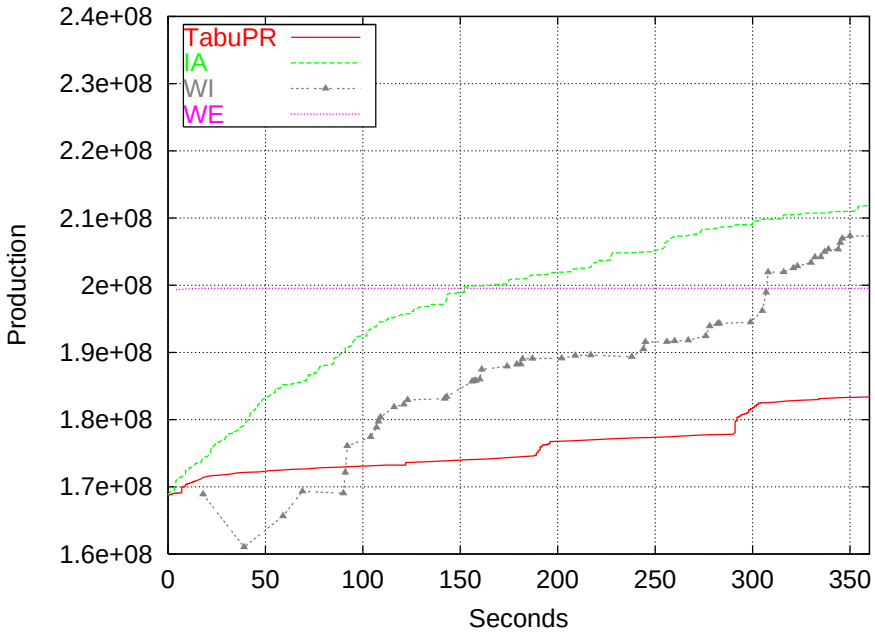


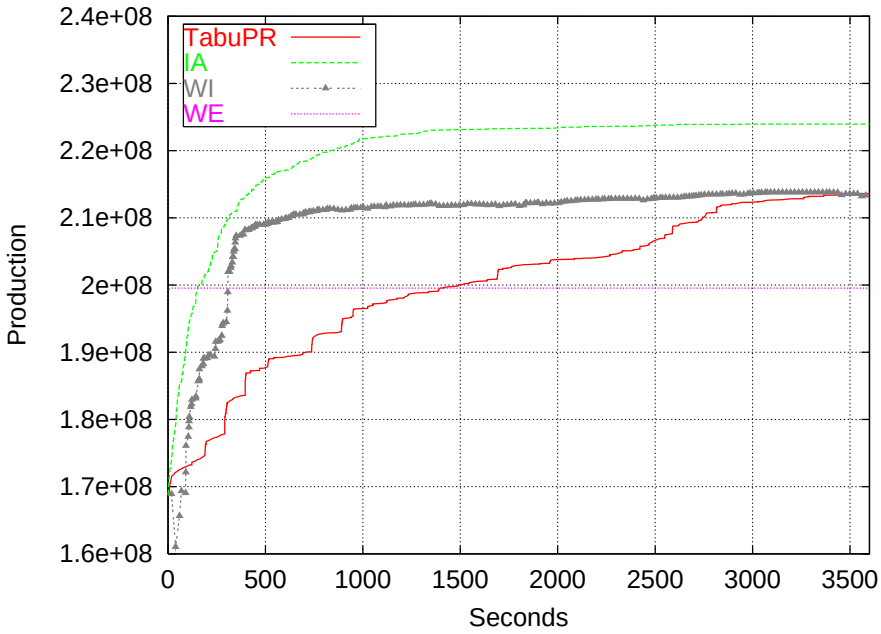
Figure 28: Production X Time - 1W130S5B3 (Real Instance) - Variant Act60

Instance	Variant	H1	
		Initial Solution	Upper Bound
1W130S5B3	Act30	4.2%	10.4%
	Act60	9.8%	11.1%
	Act90	32.7%	10.5%
2W112S4B3	Act30	12.1%	9.3%
	Act60	11.2%	14.0%
	Act90	34.6%	9.7%
3W95S5B3	Act30	7.0%	7.6%
	Act60	12.7%	8.9%
	Act90	61.2%	9.1%
4W130S5B3	Act30	7.2%	12.5%
	Act60	15.0%	15.9%
	Act90	42.2%	10.7%

Table 22: Distance among the initial solution, best solution and upper bound - Variant Act



(a) Time=0...360



(b) Time=0...3600

Figure 29: Production X Time - 1W130S5B3 (Real Instance) - Variant Act90

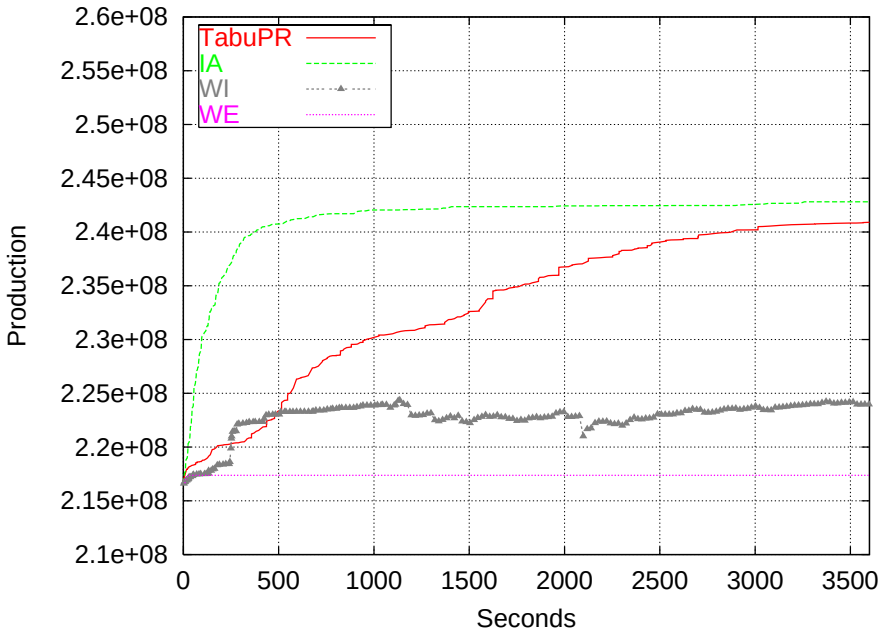


Figure 30: Production X Time - 2W112S4B3 - Variant Act30

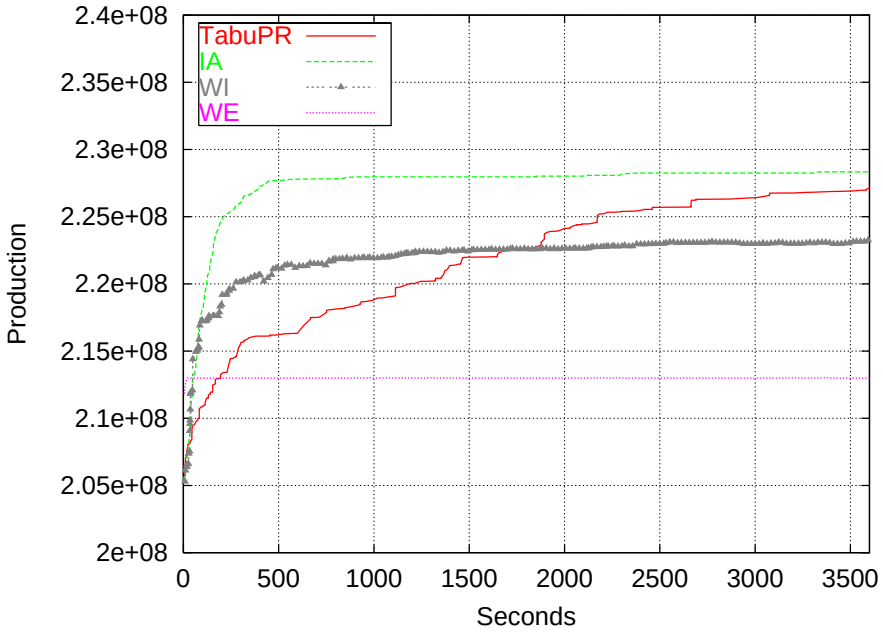
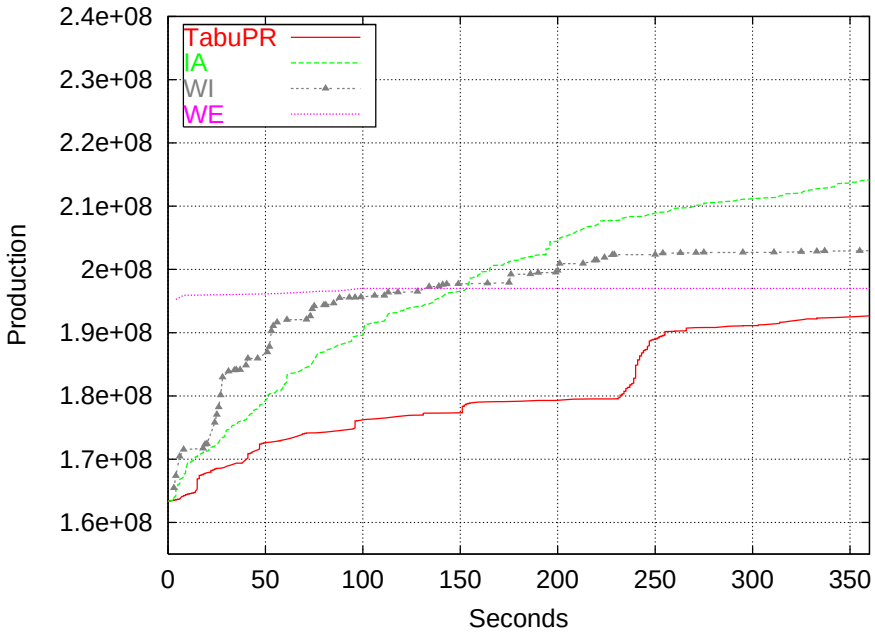
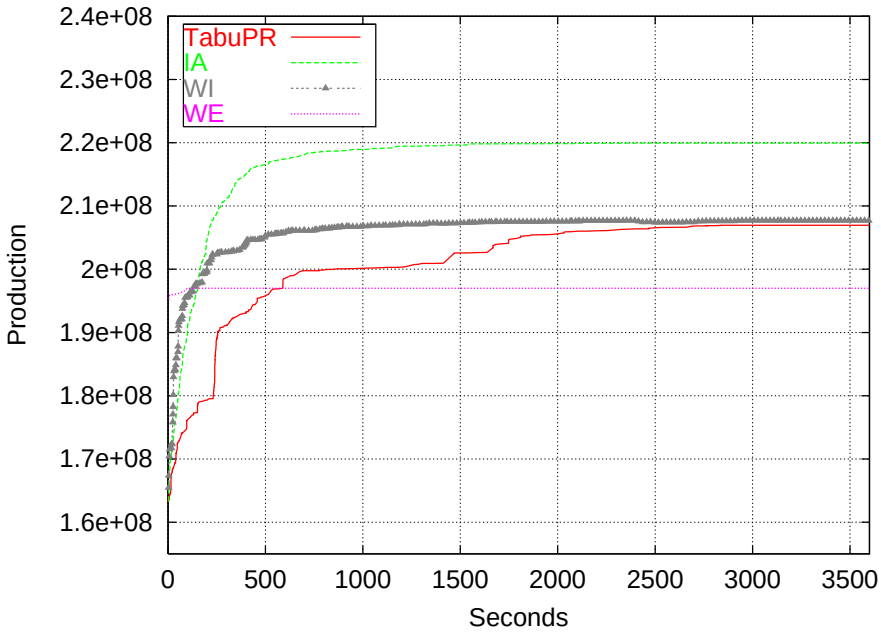


Figure 31: Production X Time - 2W112S4B3 - Variant Act60



(a) Time=0...360



(b) Time=0...3600

Figure 32: Production X Time - 2W112S4B3 - Variant Act90

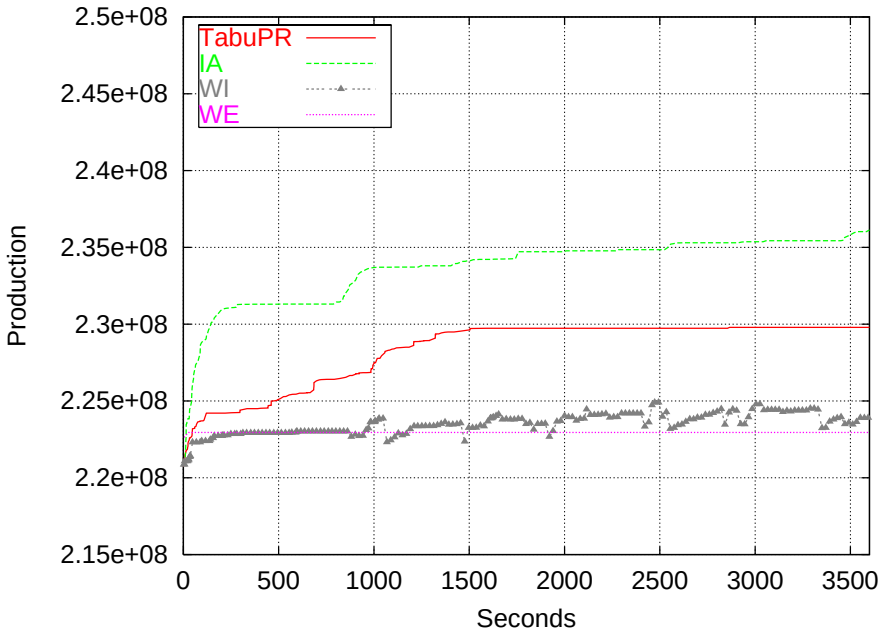


Figure 33: Production X Time - 3W95S5B3 - Variant Act30

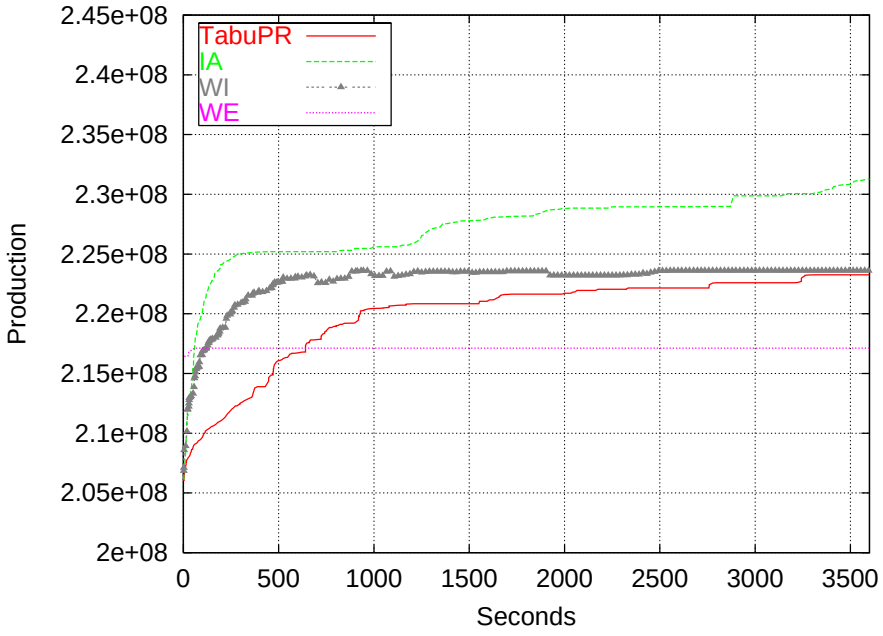
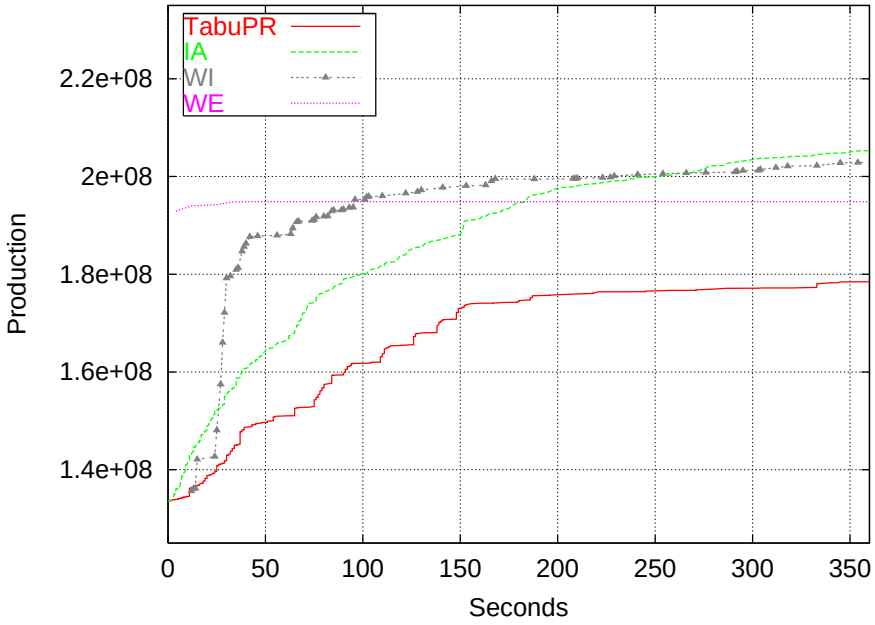
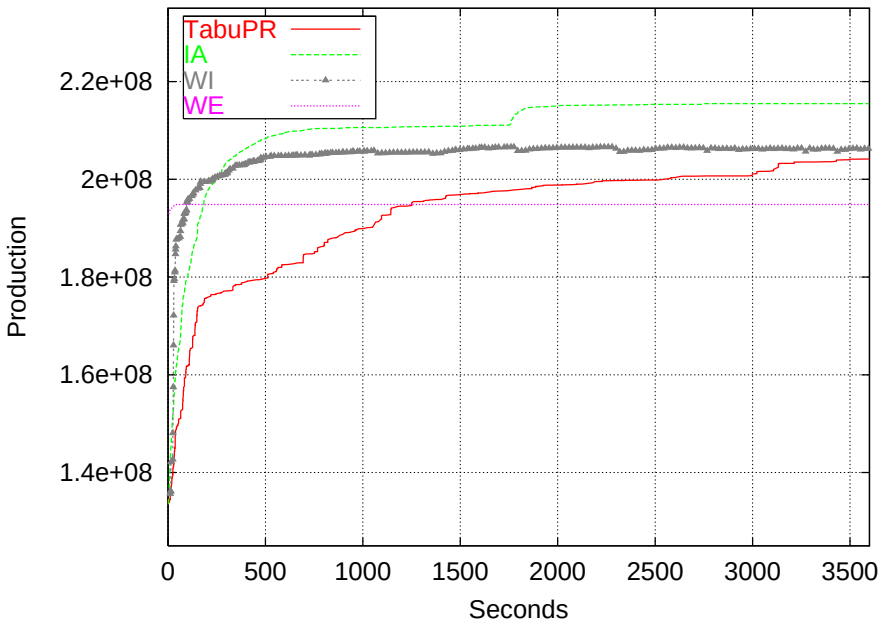


Figure 34: Production X Time - 3W95S5B3 - Variant Act60



(a) Time=0...360



(b) Time=0...3600

Figure 35: Production X Time - 3W95S5B3 - Variant Act90

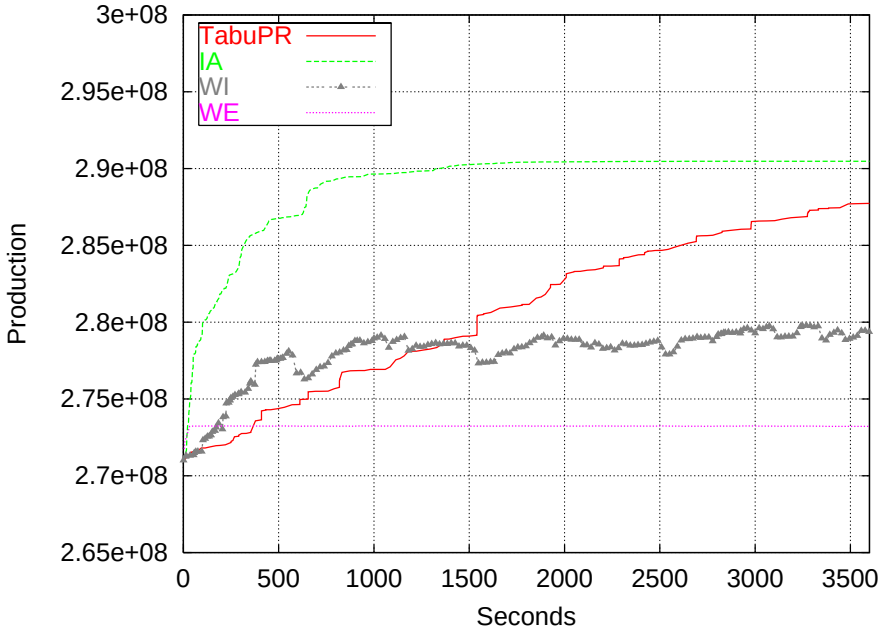


Figure 36: Production X Time - 4W130S5B3 - Variant Act30

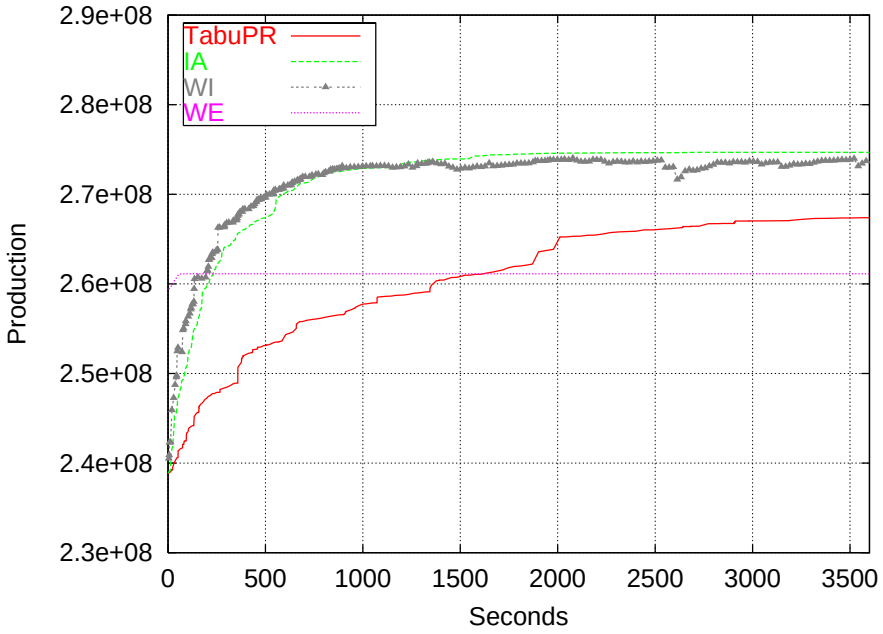
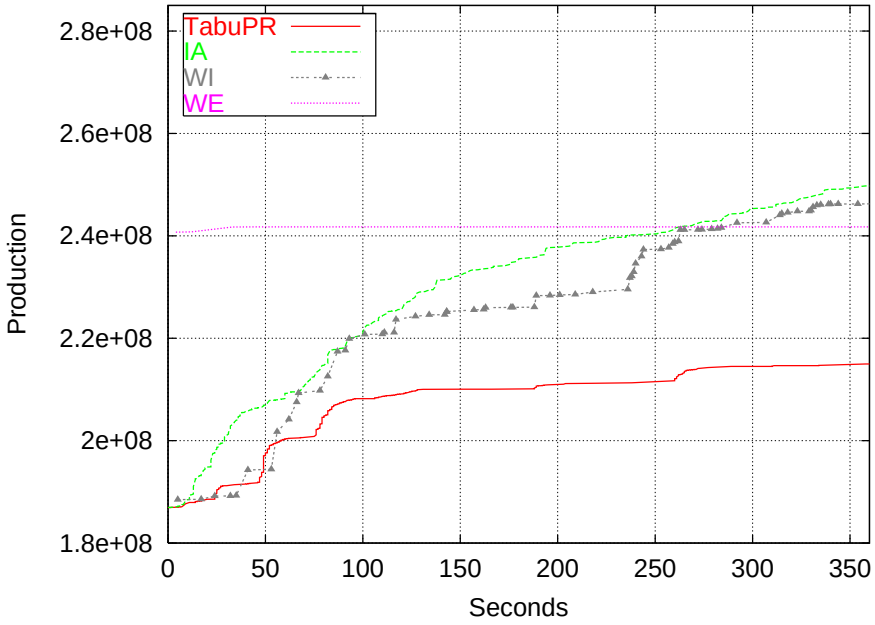
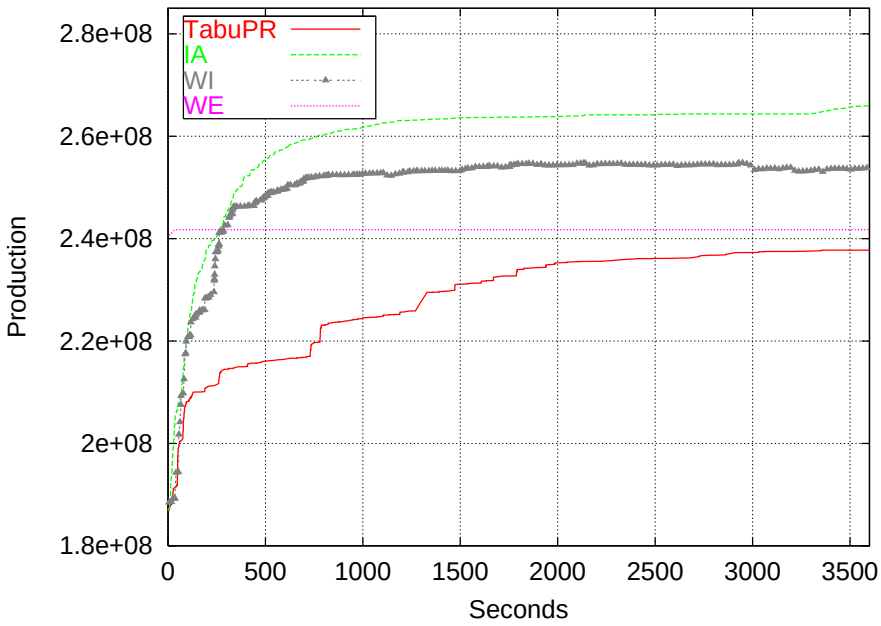


Figure 37: Production X Time - 4W130S5B3 - Variant Act60



(a) Time=0...360



(b) Time=0...3600

Figure 38: Production X Time - 4W130S5B3 - Variant Act90

Instance	Variant	Approach	It	Neighbors	Tabu	AR	Production	Time
1W130S5B3	Res1	TabuPR	166	5653569	763500	53	241408310	3615
		IA	624	4492228	663962	604	**251537690**	3600
		WE	176	881	870	0	234670140	3606
		WI	248	3797	170	6	243599460	3605
	Res2	TabuPR	311	4589353	431117	82	254654730	3640
		IA	376	5122923	766713	218	**255802210**	3609
		WE	177	881	875	0	241946220	3608
		WI	225	4062	168	12	246220480	3612
	Res3	TabuPR	335	6095363	368879	96	257360240	3611
		IA	299	5263970	774131	387	**259247850**	3610
		WE	178	883	880	0	244006440	3618
		WI	208	3935	159	7	248637620	3600
2W112S4B3	Res1	TabuPR	335	7010706	1423290	113	235086600	3605
		IA	634	4640316	695738	994	**239540901**	3597
		WE	177	882	875	0	209531118	3607
		WI	247	4476	231	21	225024965	3604
	Res2	TabuPR	430	6955965	606570	143	243662371	3612
		IA	355	4831703	814819	553	**244840697**	3603
		WE	175	880	865	0	216533315	3600
		WI	206	4640	165	3	220505606	3600
	Res3	TabuPR	504	7110939	840412	155	246225063	3608
		IA	301	5037286	839687	543	**247090041**	3599
		WE	176	881	870	0	216348434	3609
		WI	246	3988	244	26	228074446	3616
3W95S5B3	Res1	TabuPR	217	7343330	1115067	64	231892638	3613
		IA	901	5204502	1004203	423	**233020474**	3602
		WE	177	886	875	0	218454885	3607
		WI	225	5190	207	15	225305466	3615
	Res2	TabuPR	311	7141765	454913	70	237532302	3614
		IA	511	5683934	952828	648	**238780333**	3600
		WE	177	887	875	0	222826848	3613
		WI	219	4998	177	14	228407393	3617
	Res3	TabuPR	223	5895968	393610	42	235592424	3610
		IA	361	5762413	982374	568	**237120503**	3602
		WE	176	885	870	0	223421262	3606
		WI	211	4711	148	6	228464110	3606
4W130S5B3	Res1	TabuPR	197	4823039	915161	58	274715220	3615
		IA	411	2763139	378327	622	**291119199**	3605
		WE	176	877	870	0	264782298	3600
		WI	253	3887	204	17	277047062	3651
	Res2	TabuPR	224	4806321	511323	56	288863831	3605
		IA	266	3405275	478866	354	**293240295**	3624
		WE	176	881	870	0	274430520	3617
		WI	238	4130	145	13	279930174	3602
	Res3	TabuPR	249	4731748	397284	70	292684155	3613
		IA	244	4444513	598703	391	**295370753**	3617
		WE	175	880	865	0	274279895	3619
		WI	211	4387	145	8	279542790	3600

Table 23: Quantitative Data - Variant: number of resources

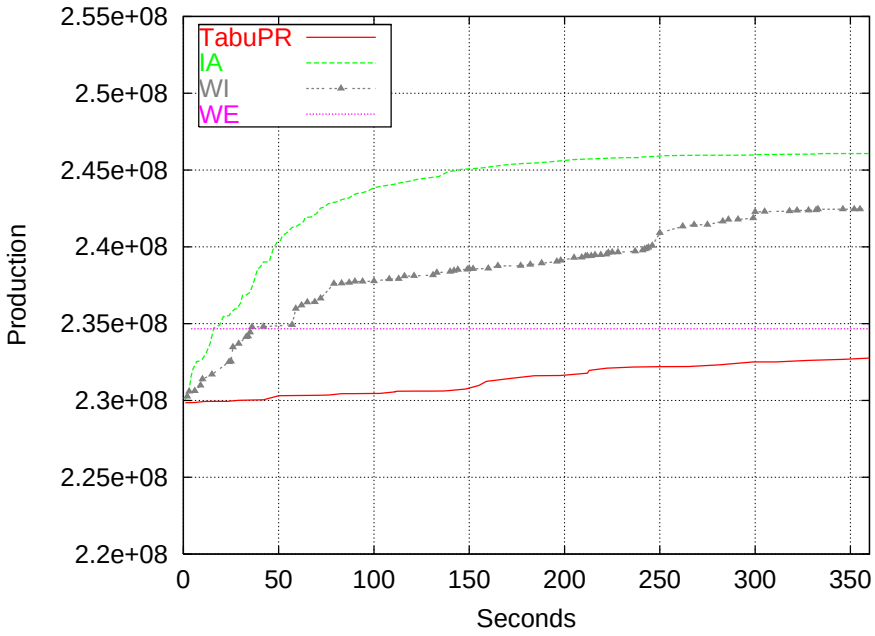
3W95S5B3. Finally, figures 48.b, 49 and 50 treat instance 4W130S5B3.

Variants **Res2** and **Res3** showed a behavior very similar to their corresponding behaviors when the original instances were considered. When variant **Res1** is considered, however, the behavior was different. As the tests using variant **Res1** produced results that differed the most from the results obtained with the original instances, figures 39.a, 42.a, 45.a and 48.a show details of the first 360 seconds of computation of each test instance. It can be observed from these figures that the **IA** strategy provided the best solution over all instances and all variants in this case. Note that approach **WI** gave better results than the pure **TabuPR** approach over instances 1W130S5B3 and 4W130S5B3. In fact, the solution provided by **TabuPR** was inferior than the best solution found, indicating that this strategy is not robust. Approach **WE** did not perform well, even on the beginning of the computation, just like it occurred when variant **Act90** was used.

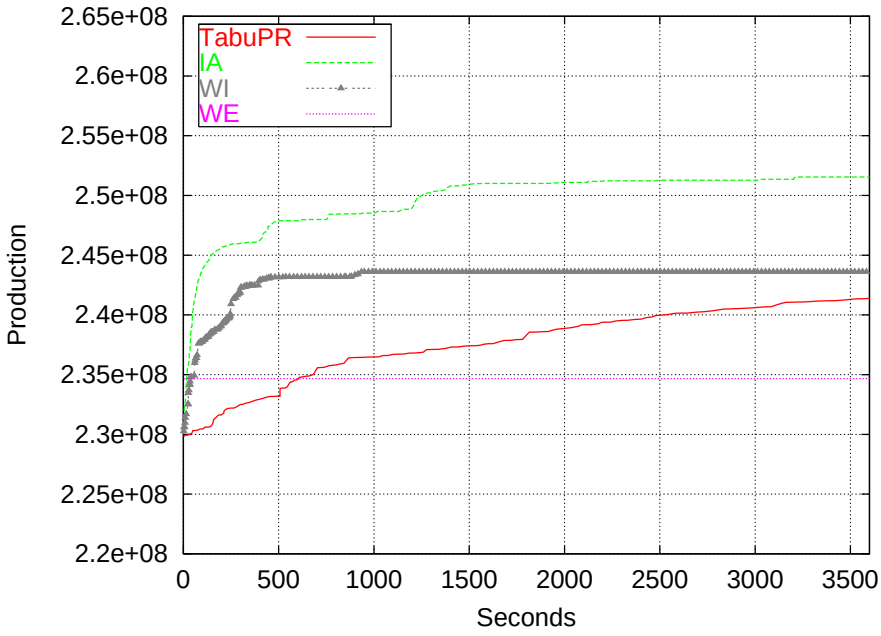
Table 24 shows the same kind of data for the perturbed **Res** variants as does table 16 for all four unperturbed instances. Observing this table, it can be inferred that the initial solution algorithm could generate good solutions for all instances, when these variants were considered. Notice that the maximum improvement obtained was 14.2%. Moreover, the best solutions obtained were good ones, as the gaps between the best solutions and the upper bounds were around 10%.

Instance	Variant	H1	
		Initial Solution	Upper Bound
1W130S5B3	Res30	9.4%	12.5%
	Res60	6.4%	10.9%
	Res90	6.5%	9.7%
2W112S4B3	Res30	16.1%	11.4%
	Res60	12.9%	9.6%
	Res90	14.3%	8.7%
3W95S5B3	Res30	6.8%	9.4%
	Res60	7.2%	7.1%
	Res90	5.8%	7.8%
4W130S5B3	Res30	11.0%	13.0%
	Res60	7.5%	12.3%
	Res90	8.1%	11.7%

Table 24: Distance among the initial solution, best solution and upper bound - Variant Res



(a) Time=0...360



(b) Time=0...3600

Figure 39: Production X Time - 1W130S5B3 (Real Instance) - Variant Res1

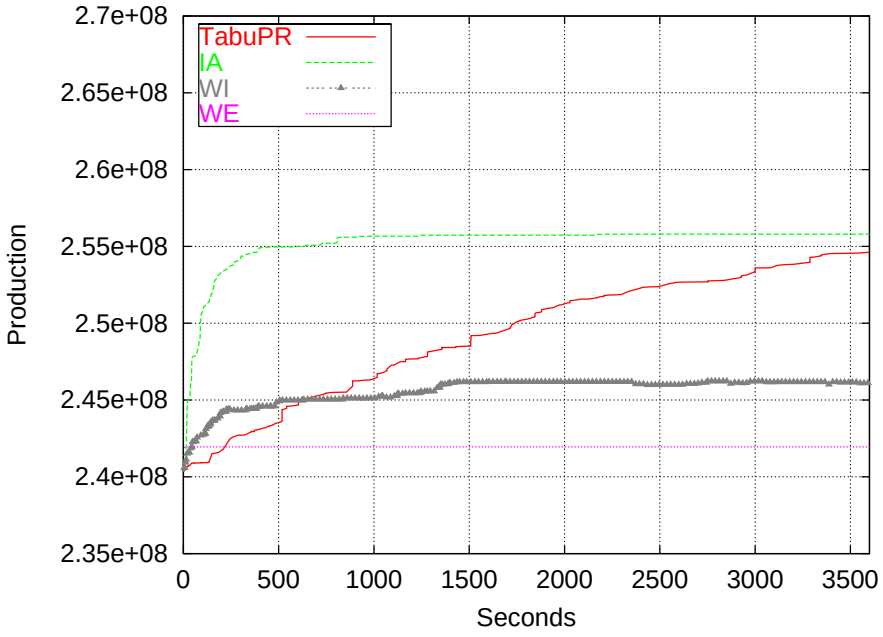


Figure 40: Production X Time - 1W130S5B3 (Real Instance) - Variant Res2

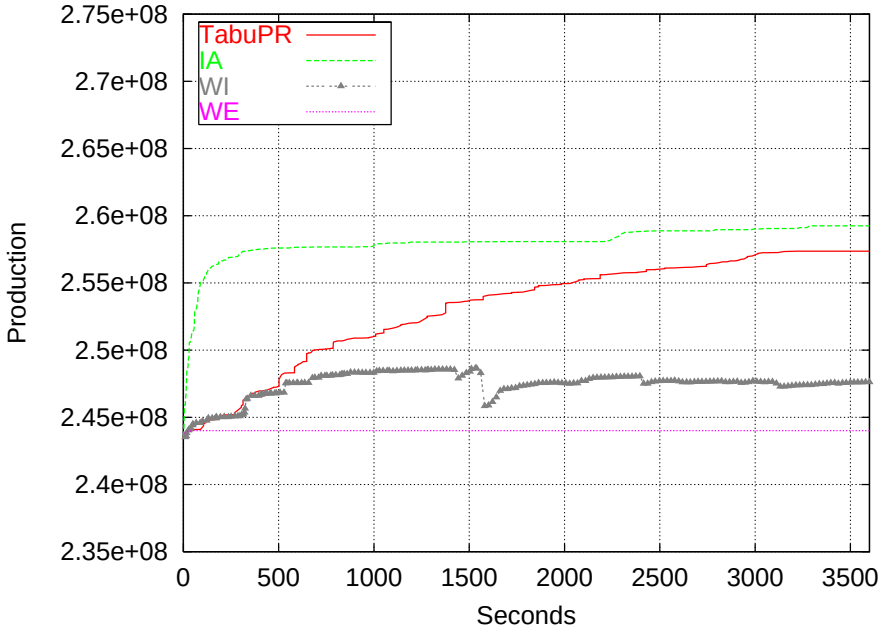
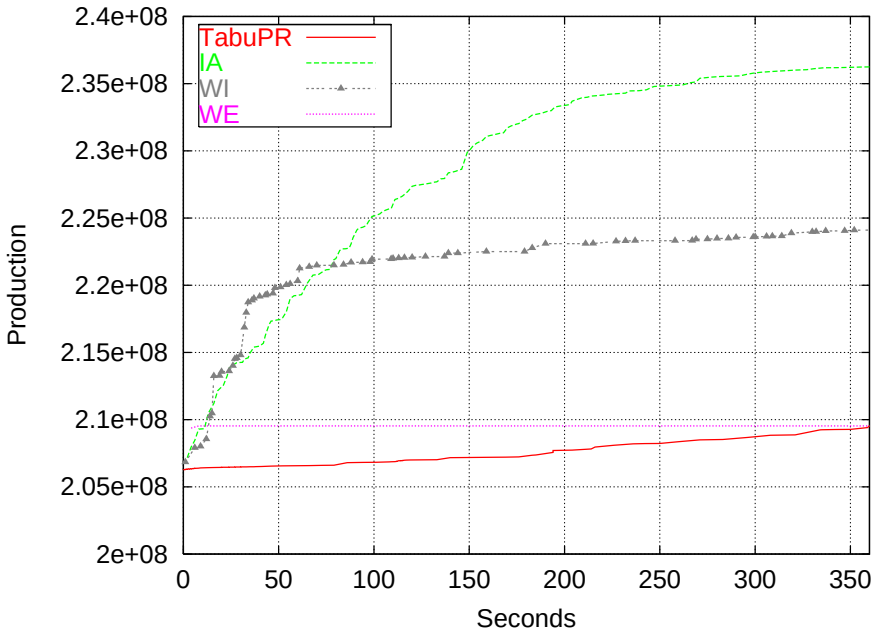
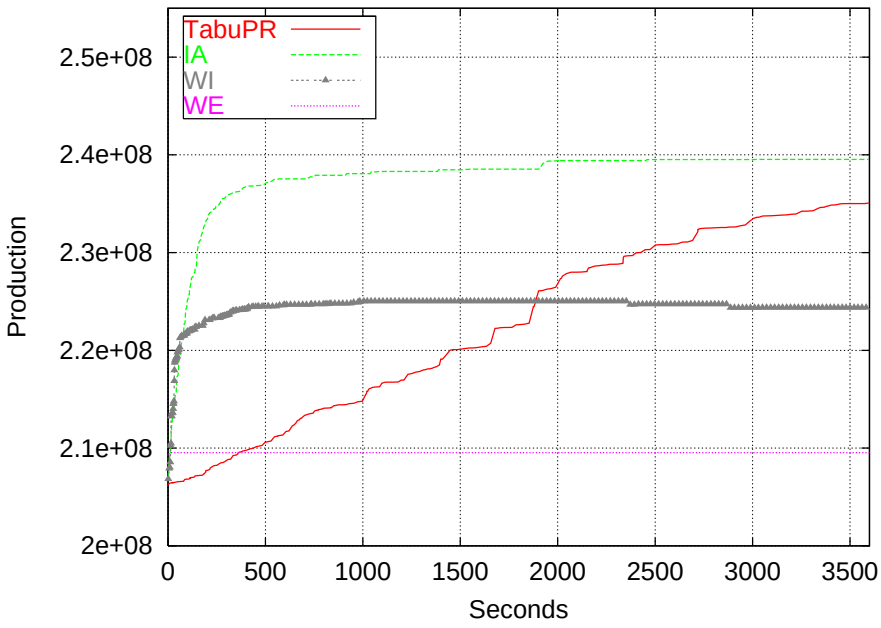


Figure 41: Production X Time - 1W130S5B3 (Real Instance) - Variant Res3



(a) Time=0...360



(b) Time=0...3600

Figure 42: Production X Time - 2W112S4B3 - Variant Res1

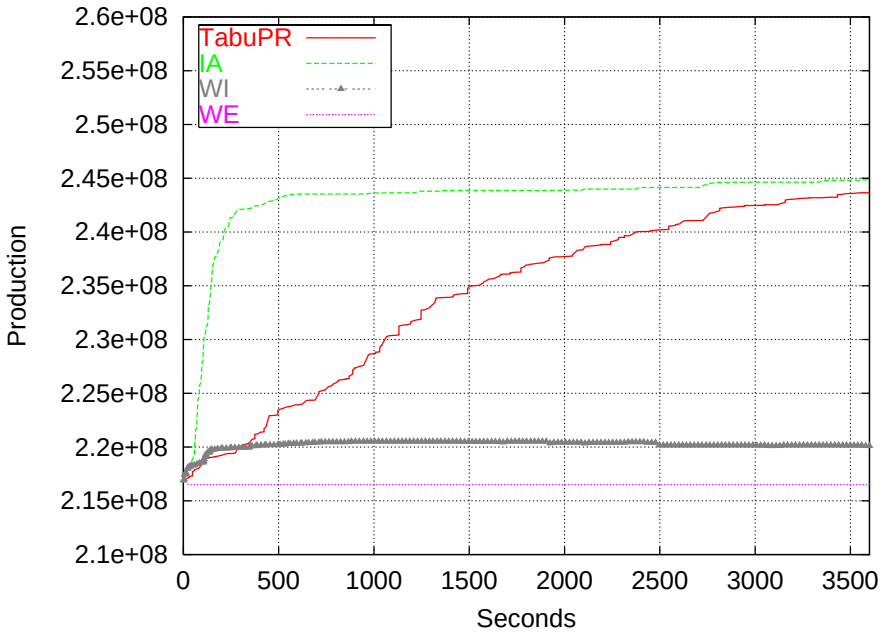


Figure 43: Production X Time - 2W112S4B3 - Variant Res2

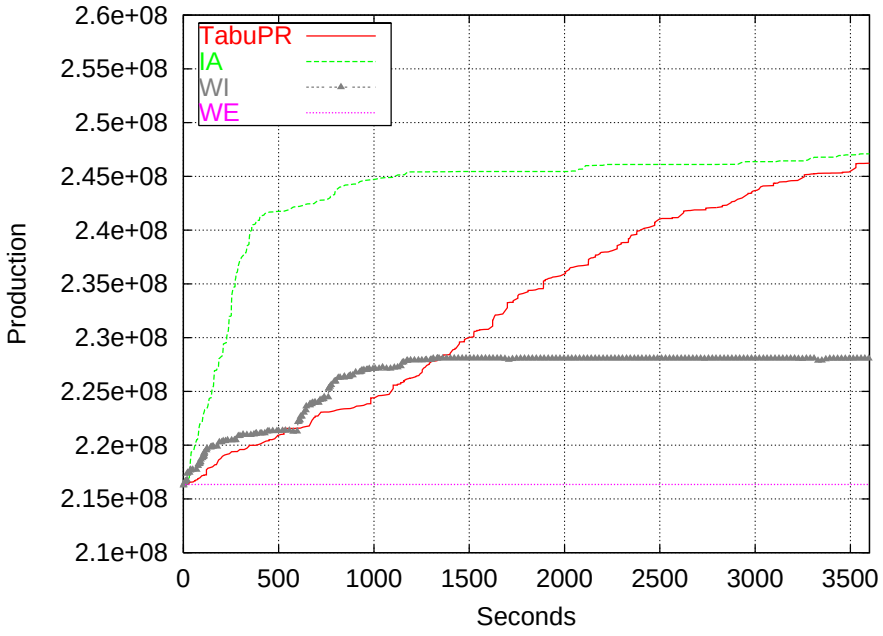
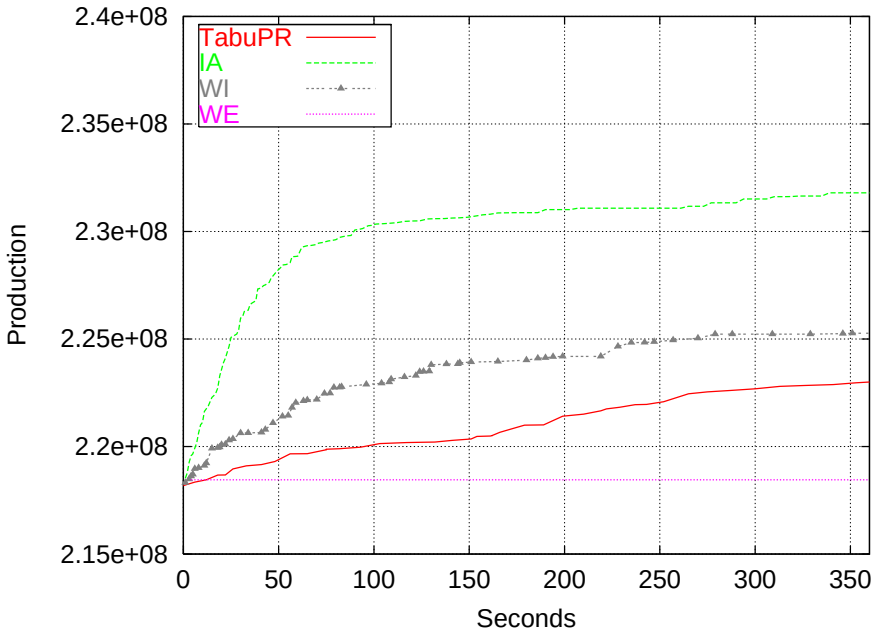
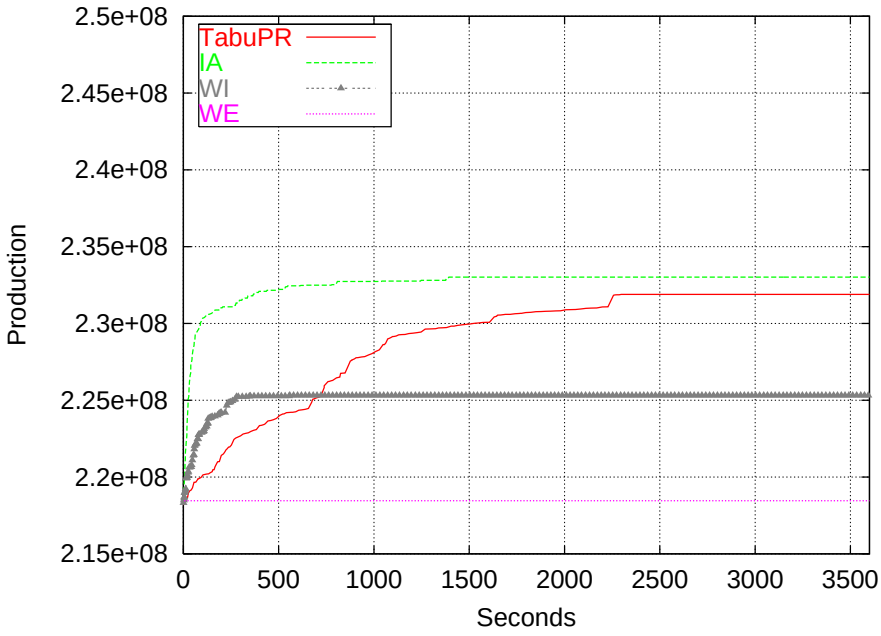


Figure 44: Production X Time - 2W112S4B3 - Variant Res3



(a) Time=0...360



(b) Time=0...3600

Figure 45: Production X Time - 3W95S5B3 - Variant Res1

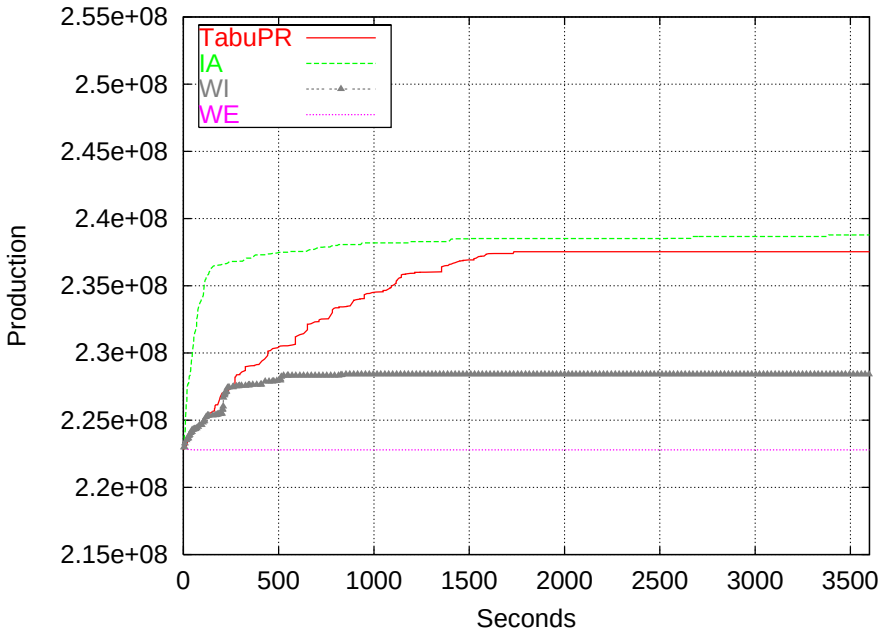


Figure 46: Production X Time - 3W95S5B3 - Variant Res2

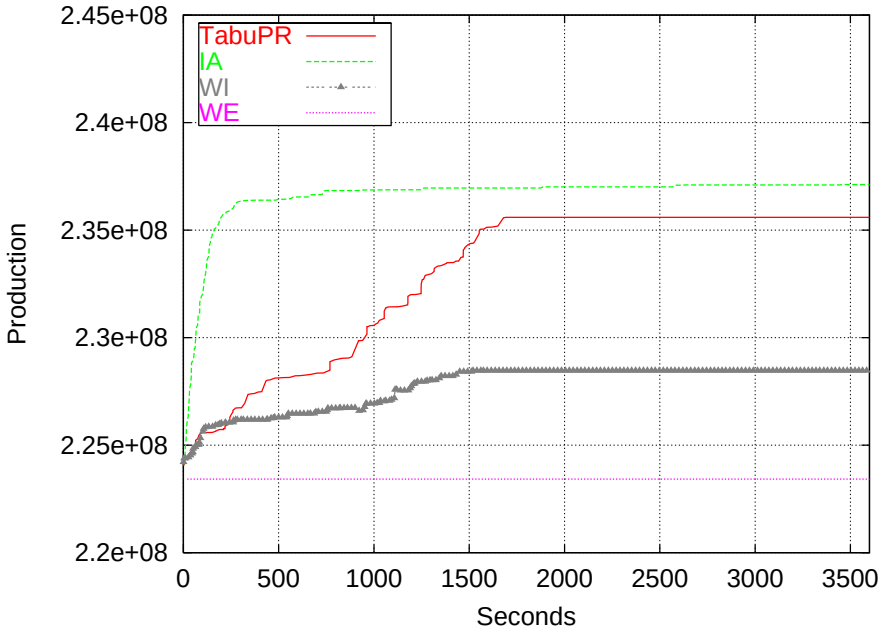
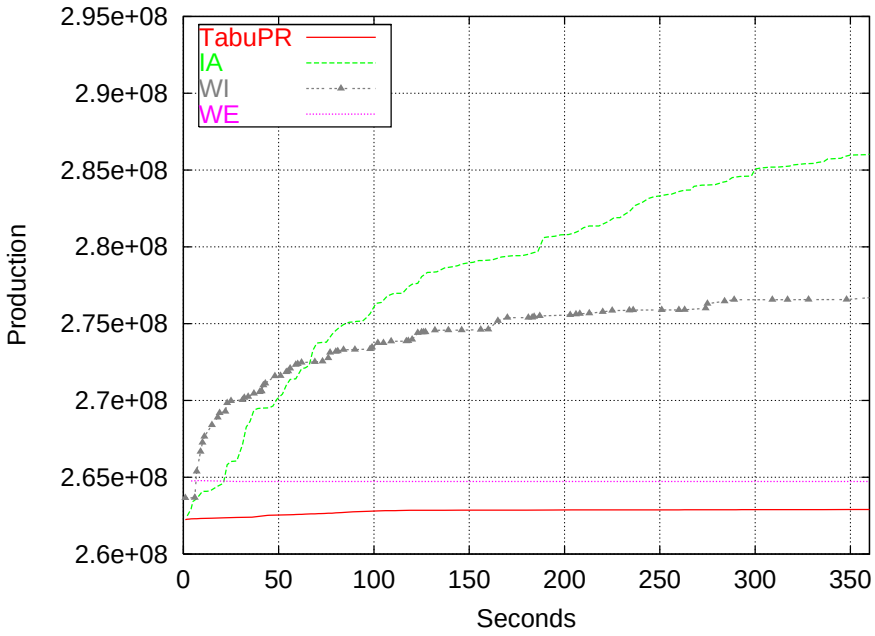
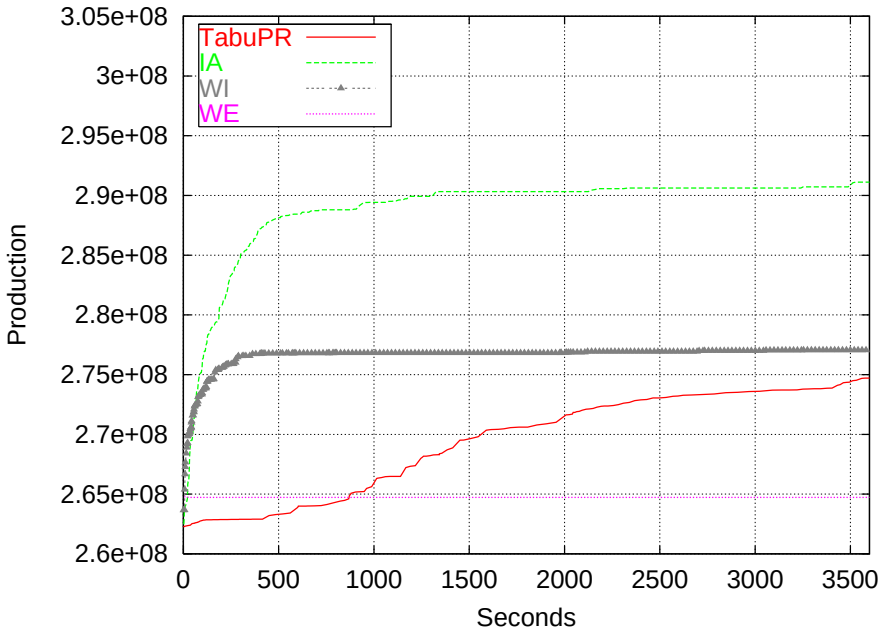


Figure 47: Production X Time - 3W95S5B3 - Variant Res3



(a) Time=0...360



(b) Time=0...3600

Figure 48: Production X Time - 4W130S5B3 - Variant Res1

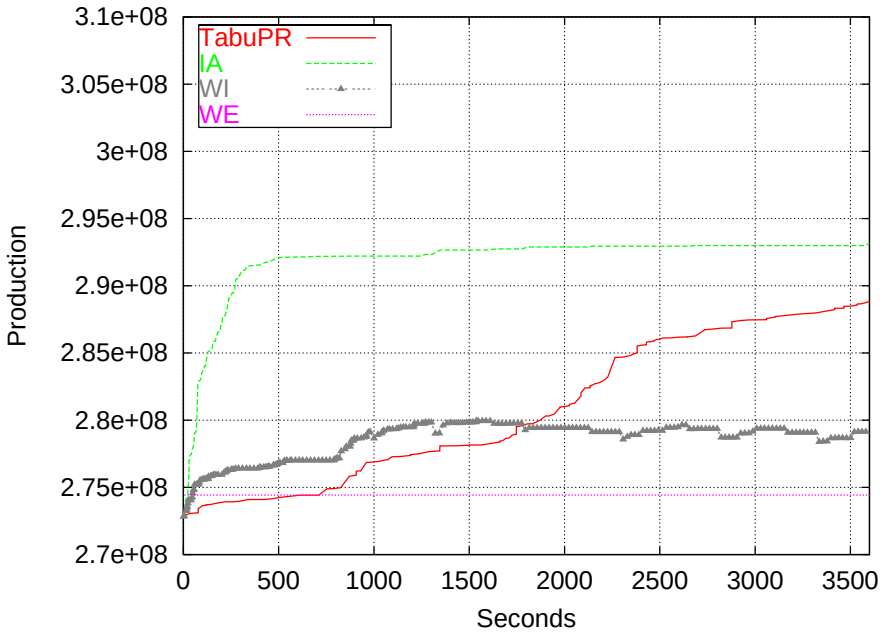


Figure 49: Production X Time - 4W130S5B3 - Variant Res2

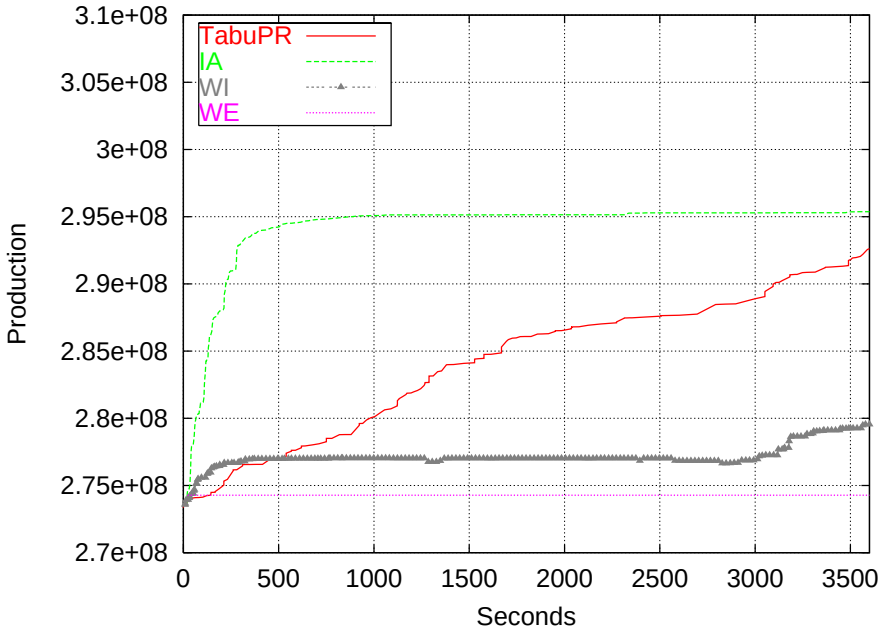


Figure 50: Production X Time - 4W130S5B3 - Variant Res3

8 Conclusions

Several conclusions may be inferred from this work. First of all, approach **IA** can the considered the best and more robust approach, since it provided the best solution for the vast majority of the tested instances and variants considered in this paper. Furthermore, the quality of its solutions are very high, as the gap between its solutions and the correspondent upper bounds were around 10% for all instances and all variants. On the other hand, the solutions obtained with the pure tabu search approach were good when the quality of the initial solutions was also good. Otherwise its performance was not that good, specially if it is considered how quickly this approach can improve a poor solution. Moreover, this approach proved to be weak to solve the variants **Act90** and **Res1**.

The hybrid methods proved very efficient in improving poor solutions, specially approach **WE**. But, unfortunately, this approach was not able to escape from production plateaus after a good solution was found. Also, approach **WI** did not perform well when the initial solutions were already good ones. To improve performance of both **WE** and **WI**, two orthogonal features of these approaches must be considered. First, the neighborhood exploration for these approaches may be improved, making an effort to eliminate intrinsic symmetries. In this way, more promising neighbors could be visited. The other feature that needs to be considered is the tabu list. It should be better studied in order to prevent these approaches from getting stuck in production plateaus.

Regarding the sensibility analysis, variants **Act 30, Act60, Res2** and **Res3** did not impact the behavior of the methods. On the other hand, the behavior of approaches **TabuPR, WE** and **WI** were modified when variants **Act90** and **Res1** were considered.

In summary, the methods implemented provided high quality solutions to the oil production problem. The hybrid methods adapted well for this problem. The next step is to improve approaches **WE** and **WI**, in order to make them comparable to approach **IA**.

References

[1] R. K. Ahuja, O. Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. Taken from the net web site: `web.mit.edu/jorlin/www`, directory `working papers`, July 1999.

[2] R. K. Ahuja, J. Orlin, and D. Sharma. Multi-exchange neiborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming*, (91):71–97, 2001.

[3] J.R. Baker and G.B. McMahon. Scheduling the general job-shop. *Management Science*, 5(31):594–598, May 1985.

[4] K. R. Baker. *Introduction to sequencing and scheduling.* John Wiley and Sons, 1974.

[5] E. Balas. Machine scheduling via disjunctive graphs: An implicit enumeration algorithm. *Operations Research*, 17:941–957, 1969.

[6] P. Bruker. Job shop scheduling with multi-purpose machines. *Computing*, 45:369 – 375, 1990.

[7] Y. Caseau, F. Laburthe, C. L. Pape, and B. Rottmbourg. Combining local and global search in a constraint programming environment. *The Knowledge Engineering Review*, 16(1):41–68, 2001.

[8] Y. Caseau and L. Laburthe. Disjunctive scheduling with task intervals. Technical report, Laboratoire d'informatique de l'École Normale Supérieure, 1995.

[9] J.B. Chambers. Flexible job shop scheduling by tabu search. Taken from the net web site: `citeseer.nj.nec.com/108947.html`, 1996.

[10] J.B. Chambers and J.W. Barnes. Reactive search for flexible job shop scheduling. Taken from the net web site: `citeseer.nj.nec.com/chambers98reactive.html`, 1998.

[11] N. Christodoulou, E. Stefanitis, E. Kaltsas, and V. Assimakopoulos. A constraint logic programming approach to the vehicle-fleet scheduling problem. In *Proceedings of Practical Applications of Prolog*, pages 137–149, 1994.

[12] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[13] V. G. Deineko and G. J. Woeginger. A study of exponential neighborhoods for the travelling salesman problem and for the quadratic assignment problem. *Mathematical Programming*, pages 255–279, February 2000.

[14] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-Completeness.* W. H. Freeman and Company, San Francisco, California, 1979.

[15] C. P. Gomes. On the intersection of AI and OR. *The Knowledge Engineering Review*, 16(1):1–4, 2001.

[16] R.L. Graham, E.L. Lawer, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and squeduling: A survey. In *Annals on Discrete Mathematics*, volume 5, pages 287–326, 1979.

[17] B. Jurish. *Scheduling Jobs in Shops with Multi-purpose machines.* PhD thesis, Fachbereich Mathematik/Informatik, Universitat Osnabruck, 1992.

[18] B.J. Lageweg, K. Lenstra, and A.H.G. Rinnoy Kan. Job-shop scheduling by implicit enumeration. *Management Science*, 4(24):441–450, 1977.

[19] P. Martin and D. B. Shmoys. A new approach to computing optimal schedules for the job-shop scheduling problem. In *Proceedings of the 5th International IPCO Conference*, pages 389–403, 1996.

[20] M. Mastrolilli and L.M. Gambardella. Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling*, 3:3–20, 2000.

[21] R. McNaughton. Scheduling with deadlines and loss functions. *Management Science*, 6:1–12, 1959.

[22] J. M. Nascimento and C. C. de Souza A. V. Moura. Relatorio fapesp. Relatorio 1, Unicamp, 2002.

[23] G. Peasant and M. Gendreau. A constraint programming framework for local search methods. *Journal of Heuristics*, (5):255–279, 1999.

[24] M. Pinedo. *Scheduling: Theory, Algorithms and Systems.* Prentice Hall International Series in Industrial and Systems Engineering. Prentice Hall, 1995.

[25] J.-F. Puget. Object-oriented constraint programming for transportation problems. In *Proceedings of Advanced Software Technology in Air Transportation (ASTAIR)*, 1992.

[26] B. Roy and B. Sussmann. Les problèmes d'ordonnancement avec contraints disjonctives. Note DS 9 bis, SEMA, Paris, 1964.

[27] M. Wallace. Practical applications of constraint programming. *Constraints*, (1):139–168, 1996.

[28] T. H. Yunes, A. V. Moura, and C. C. de Souza. A hybrid approach for solving large scale crew scheduling problems. In *Lecture Notes in Computer Science,* vol. 1753, pages 293–307, Boston, MA, EUA, January 2000. Anais do *Second International Workshop on Practical Aspects of Declarative Languages (PADL'00).*

[29] T. H. Yunes, A. V. Moura, and C. C. de Souza. Solving very large crew scheduling problems to optimality. In *14 th ACM Symposium on Applied Computing (SAC'00)*, pages 446–451, Como, Italy, March 2000.