Constraints-Handling in an Evolutionary Tool for Scheduling in Oil Wells

D. Pandolfi[1], A. Villagra[1], J. Rasjido[1], J. Valdez[1] and G. Leguizamón[2]

[1] Laboratorio de Tecnologías Emergentes
Unidad Académica Caleta Olivia
Universidad Nacional de la Patagonia Austral
Caleta Olivia, Santa Cruz, Argentina
{dpandolfi,avillagra,jrasjido,jcvaldez}@uaco.unpa.edu.ar
[2] Laboratorio de Investigación y Desarrollo en Inteligencia Computacional
Departamento de Informática
Facultad de Ciencias Físico Matemáticas y Naturales
Universidad Nacional de San Luis
San Luis, Argentina
legui@unsl.edu.ar

Abstract. Oil exploitation and transportation are very important activities for the economic development of the industrial modern society. However, these activities are generating risks that are translated in accidental or chronic contaminations that directly affect the ecosystem. It is important that oil companies carry out a correct maintenance of their oil fields. In cases of scheduling maintenance of 200 or more oil wells, our so-called PAE, is a tool able to provide a maintenance visit schedule at the right moment. PAE uses an evolutionary algorithm to produce multiple solutions to this problem. In this work, we study the application of different types of penalty functions to a constrained scheduling in oil wells. Details of implementation, results, and benefits are presented.

1 Introduction

In the last decade, Oil Companies engaged in exploitation, production, and transportation of this natural resource have seen the necessity of implementing preventive measures in order to avoid and/or to minimize the damages caused to the people, environment, and material goods. Accordingly, the best way to attack the problem of oil contamination is to prevent any possible incident.

Usually, problems occur due to equipment or material failure and human mistakes. Equipment or material failure can be corrected by means of periodic inspections and appropriate maintenance tasks; human failure can be corrected through a permanent training of personnel, especially those in charge of maintenance activities. For this reason, it is important for Oil Companies and for the environment surrounding them, to have a suitable maintenance of their fields.

Most problems of optimization include certain kind of constrains that constitute major challenges to appropriately solve them. These constraints are usually

limits imposed to decision variables and in general the constraints are an integral part of the formulation of any problem [1]. Moreover, any situation where a decision should be taken involves constraints. What distinguishes several types of problems is the form of these constraints (rules, data dependencies, algebraic expressions, or other forms).

In the resolution of constrained optimization problems we search for a feasible optimal solution; however, during this search process we usually deal with a small number of feasible solutions and a large number of infeasible ones, depending of the characteristics of the problem constraints. For this reason different techniques exist for handling constraints. Evolutionary Algorithms (EAs) have been quite successful in a wide range of applications [6][13][16]. However, an aspect normally disregarded when using them for optimization (a rather common trend) is that these algorithms are unconstrained optimization procedures, and therefore is necessary to find ways of incorporating the constraints (normally existing in any real-world application) into the fitness function. The selection of an adequate constraint-handling technique for a given EA is an open problem.

EAs with multirecombinative approaches [4] and multiparent [5] for the resolution of diverse types of such problems of planning as scheduling or routing these approaches have turned out to be successful strategies. Particularly in scheduling problems, adding a new variant to the multirecombinative approach called MCMP-SRI (Stud and Random Immigrats) [12].

Our tool, called PAE [18](Planificador basado en un Algoritmo Evolutivo, Evolutionary Algorithm for Planning), is a tool based in an evolutionary approach that aims to schedule the visits of a group of oil fields that: (a) minimizes the total time of visits; that is to say, to find the schedule that visits the fields including the time of intervention in each one, in a shorter time; (b) re-schedules the visits, i.e., provide alternative schedules without significantly diminishing their quality, in case of the occurrence of some events that interrupt the execution of a maintenance schedule (dynamic features of the problem); and (c) obtains solutions that fulfil all the problem constraints.

Particularly, this work shows the obtained results that fulfil an added problem constraint by applying different penalties approaches. The paper continues as follows. Section 2 shows a brief description of the more relevant constraint handling techniques and highlighting the penalty functions applied in this work. Section 3 shows the domain and problem description. Section 4 describes the evolutionary algorithm proposed to solve the problem. Section 5 presents the penalty functions used in our experimental study whereas Section 6 shows experimental tests and results, and finally in Section 7, we give some conclusions and analyze future research directions.

2 Contraints Handling

Coello Coello [1] proposed a useful taxonomy of constraint handling techniques including: (1) Penalty functions, (2) special representations and operators, (3) repair algorithms, (4) separations of objectives and constraints and (5) hybrid

methods. Penalty functions [11] decrease the fitness of infeasible solutions as to prefer feasible solution in the selection process. Special representation and operators are designed to represent only feasible solutions and the operators are able to preserve the feasibility of offspring generated. Repair algorithms aim to transform an infeasible solution into a feasible one. The separation of objectives and constraints consists on using these values as separated criteria in the selection process of an EA; this is opposed to penalty functions, where the values of the objective function and the constraints are mixed into one single value. Finally, hybrid methods are combination of different algorithms and/or mechanisms e.g., fuzzy-logic with EAs, cultural algorithms [9] and immune systems [3].

The most common way of incorporating constraints into an EA have been penalty functions. Penalty functions were originally proposed by Courant in the 1940s [2]. The idea of this method is to transform an optimization problem with constraints in an optimization problem without any constraint. This is achieved by adding (or subtracting) a certain value to the function objective keeping in mind the amount of violation presented in certain solution.

Two types of penalty functions exist: exterior and interior. In the exterior penalty we begin with an infeasible solution and from there we move toward a feasible region. In the interior penalty the term penalty is chosen in such way that its value is small in the points away from the boundaries of the constrains and they will spread to infinite as they come closer to the boundaries of those constraints. Then if we start with a feasible point the subsequent points generated will always be inside a feasible region because the boundaries act as barriers during the process of optimization [15].

Exterior penalty is the most usual method applied in EAs. The main reason for this is because they do not require feasible solutions to proceed with the search. In fact, for many applications where EAs are applied, find at least one initial feasible solution is NP-Hard [17]. The most important type of penalty functions are: (a) Static Penalties, in which the penalty factors do not depend on the current generation number any way, and therefore, remain constant during the entire evolutionary process; (b) Dynamic Penalties, refers to any penalty function in which the current generation number is involved in the computation of the corresponding penalty factors; (c) Annealing Penalties, based in the idea of simulated annealing [8], the penalty coefficients are changed once in many generations, the penalty increased over the time (i.e., the temperature decreases over the time) so that infeasible individuals are heavily penalized in the last generations); (d) Adaptive Penalties, in which a penalty function takes feedback from the search process. (Notice that although the two approaches described above, Annealing and Adaptive Penalties, are also Dynamic Penalties, they where considered separately for a sake of clarity); and finally the simplest approach, (e) Mortal Penalties where no further calculations are necessary to estimate the degree of infeasibility of a solution because no infeasible solution is accepted (also called “death penalty”).

3 Domain and Description of the Problem

Oil Companies carry out maintenance or prevention visits to each of their oil wells (producing wells, injectors, batteries, and collectors). An oil field is formed by batteries, each battery contains about 20 oil wells. Each oil well has different production levels known a priori and they vary in time. The well production defines the category and the number of times it shall be visited in a month. The oil wells can not be visited more than once in the same shift and depending on its type there are some tasks that shall be carried out. Each task has been given the necessary equipment, a frequency of realization and an approximate time for its duration. Currently the route carried out by the team in charge of maintenance visits is scheduled based on their experience. A work day begins in the morning and the oil wells are visited in two shifts of three hours. After a shift is concluded the team in charge should return to the base to carry out certain administrative activities and then continue with the following shift. The demanded time in each oil well will depend on its type. Occasionally, unexpected contingencies may result in an unaccomplished shift maintenance schedule; i.e., some oil wells may not be visited. When this event occurs, it affects the total scheduling and each person held responsible should redefine the new itinerary based on his experience.

3.1 Problem Formulation

The problem can be precisely stated as defined in [14]:

$$1|S_{jk}|C_{max} \tag{1}$$

It denotes a single-machine scheduling problem with n jobs subject to sequence-dependent setup times, where the jobs to be scheduled are the maintenance (or intervention) service in each of the oil fields. The objective is to minimize the makespan (Cmax) subject to the dependent times of preparation of the sequence. It is well-known that this problem is equivalent to the so-called Traveling Salesman Problem (TSP). The makespan can be calculated as:

$$\sum_{k=1}^{n}(S_{jk} + t_k) \tag{2}$$

where S_{jk} represents the cost (in time) of going from oil field j to oil field k, t_k the respective time of maintenance in location k, and n the total number of oil wells in the field. Morover, the above formulation can be extended as follows:

Definition 1 *Let $OW \subset \{1, \ldots, n\}$ be a subset of the all oil wells in the field. OW represents the oil wells that should be visited twice. Furthermore, the oil wells in OW can not be scheduled in the same shift. A solution of a constrained instance of the problem that not fulfill the above condition is considered an infeasible solution.*

As explained in further sections, the introduction of the above constraint will affect the design of the EA as it neccesary to consider a constraint handling technique to solve this problem under the new formulation.

4 PAE: the evolutionary tool

To solve this problem of scheduling the visits to the oil fields we used an evolutionary algorithm. The first step was developing an adequate encoding of the visits to the oil wells that represents a solution to the problem. A schedule of visits was encoded in a chromosome as a sequence of oil wells represented by natural numbers. Therefore, a chromosome will be a permutation $p = (p_1, p_2, \ldots, p_n)$ where n is the quantity of oil wells in the shift. Each element p_i represents the $i-th$ oil well that will be visited according to the sequence of visits. The chromosome gives the sequence order to be followed in order to visit each oil well. The algorithm will devise the best possible permutation so that it finds an optimal schedule and fulfills the problem constraints.

Algorithm 1 EA-MCMP-SRI

```
1: t = 0 {current generation}
2: initialize Stud(t)
3: evaluate Stud(t)
4: while not max_evaluations do
5:    mating_pool = Generate_Random_Immigrant ∪ Select (Stud(t))
6:    while not max_parents do
7:       while max_recombinations do
8:          evolve (mating_pool){recombination and mutation}
9:       end while
10:   end while
11:   evaluate (mating_pool)
12:   Stud(t+1) = select new population from mating_pool
13:   t = t + 1
14: end while
```

In Algorithm 1 is presented a general outline of EA-MCMP-SRI, used for solving our problem and explained in the following. The algorithm creates an initial stud population Stud(0) of solutions to the scheduling problem in a random way, and then these solutions are evaluated. After that, the stud population undergoes a multirecombined process where the algorithm creates a mating pool which contains the stud and random immigrants. The process for creating offspring is performed as follows. The stud mates with each of the parents, then couples undergo crossover and $2 \times n2$ ($n2 \leq max\_parents$) offspring are created. The best of this $2 \times n2$ offspring is stored in a temporary children pool. The crossover operation is repeated $n1$ times ($max\_recombinations$) for different cut points each time, until the children pool is completed. Children may or may not

undergo mutation. Finally, the best offspring created from $n2$ parents and $n1$ crossover is inserted in the new population.

The recombination operator used in this algorithm was PMX (Partial Mapped Crossover). This operator was proposed by Goldberg and Lingle [7]. It can be viewed as an extension of two-cut crossover for binary string to permutation representation. In the mutation operation used, named as Swapping Mutation (SM), we select two random positions and then swap their genes. The selection operator used for selecting an individual was a Proportional Selection.

In our evolutionary tool, a schedule of visits was encoded in a chromosome as a sequence of oil wells represented by natural numbers. The chromosome gives the sequence order to be followed in order to visit each oil well. Also, one keeps in mind that exist oil wells that should be visited more than once (according to OW) which implies that many solutions visited in the search space will be infeasible.

5 Penalty Functions considered

To handling constraints in an EAs with penalty functions, the fitness function $f(p)$ is usually transformed in $F(p) = f(p) + \mathcal{P}(p)$ (for a minimization problem) where $\mathcal{P}(p)$ is called the penalty function. In the present work two penalties functions were defined:

1. A penalty function that calculate the cost of repairing the infeasible solution with respect to the fitness values. It must be noticed that the infeasible solution is only repaired for evaluation porpoises.

$$\mathcal{P}_1(p) =\mid f(p) - f(p') \mid$$

   where p is the infeasible solution, $f(p)$ represents the fitness of this solution, and $f(p')$ the fitness function of p' (the repaired solution).
2. Let us consider an infeasible solution $p = (p_1, .., p_n)$. Accordingly, there will be components that do not satisfy the problem constraint. The corresponding penalty function $\mathcal{P}_2(p)$ will consider each one of these components in the following way:

$$\mathcal{P}_2(p) = 2 \times \sum_{h \in H} s_{hk}$$

   where H is the set of oil wells in solution p that not fulfil the problem constraint, h is a particular oil well, and s_{hk} represents the cost (in kilometers) of going from oil well h to the base of operations k.

According to the above defined basic penalty functions $\mathcal{P}_1$ and $\mathcal{P}_2$, we present the following combinations of Static, Dynamic, and Annealing penalty functions as follows:

A. **Static Penalties**
Two penalty functions are straightforwardly defined for this type:

S1: Consist of applying the first penalty function to the infeasible solution. That is to say, adding to the fitness function the penalty value obtained.

$$F(p) = f(p) + \mathcal{P}_1(p)$$

S2: Consist of applying the second penalty function to the infeasible solution. That is to say, adding to the fitness function the penalty value obtained.

$$F(p) = f(p) + \mathcal{P}_2(p)$$

B. **Dynamic Penalties**
For the dynamic penalties, the first and the second penalty functions ($\mathcal{P}_1$ and $\mathcal{P}_2$) are multiplied by value returned by the following monotonically increasing function:

$$V(g) = \left(\frac{g}{G}\right)^2$$

where g is the current generation, G is the total number of generations, and $0 \leq V(g) \leq 1$.

D1: $F(p) = f(p) + [\mathcal{P}_1(p) \times V(g)]$.
D2: $F(p) = f(p) + [\mathcal{P}_2(p) \times V(g)]$.

C. **Annealing Penalties**
For the annealing penalties, the two static penalty functions ($\mathcal{P}_1$ and $\mathcal{P}_2$) were applied based on the main concepts involved in Simulated Annealing. An acceptation probability value $Prob_A$ is calculated as follows:

$$Prob_A = 1 - \left(\frac{g}{G}\right)^2$$

where g is the current generation, G is the total number of generations.
Using this probability of acceptation ($Prob_A$), infeasible solutions are accepted with a high probability in the first stages. As the population evolves this probability is decreased. Thus, infeasible solutions will be more frequently penalized at the end of the algorithm execution. The two annealing penalty functions considered are:

A1: Consist of applying penalty function S1 to the infeasible solution with a probability of $Prob_A$ and the extended fitness function is defined as:

$$F(p) = \begin{cases} f(p) + \mathcal{P}_1(p) & \text{if } r \leq Prob_A \\ f(p) & \text{otherwise} \end{cases}$$

where $r \in (0..1)$.

A2: Consist of applying penalty function S2 to the infeasible solution with a probability of $Prob_A$ and the extended fitness function is defined as:

$$F(p) = \begin{cases} f(p) + \mathcal{P}_2(p) & \text{if } r \leq Prob_A \\ f(p) & \text{otherwise} \end{cases}$$

where $r \in (0..1)$.

6 Experiments and Results

To solve the problem it was necessary to prepare the data since the original information about the distances among oil wells were not processed. The calculation of the distances among the oil wells based on the roads map and distribution of the oil field was carried out. It is known that the distance between two points that are in any place of the system of coordinated Cartesian, is determined by the relationship denominated Euclidean distance. Nevertheless, in this problem the distance between two points can be calculated, considering the existing routes connecting to the oil wells. For this reason, the oil field road map was used and the distances were scaled down.

We performed 30 runs for the following scenario. We consider 110 oil wells belonging to a set of the exploitation region. The round speed was defined in 12 seconds every 100 meters and the time in the oil well making the maintenance was set in 300 seconds. With respect to the problem constraint we considered six different sets OW consisting respectively of 15, 16, 17, 18, 19, and 20 different oil wells randomly chosen from the set of the 110 oil wells in the field. Based on this, we generate six different instances of the problem determined by set OW. These six instances were used in the experimental study to test the performance of EA-MCMP-SRI under the different penalty functions implemented.

Regarding the EA-MCMP-SRI algorithm, the population size is set to 15 individuals. The initial population was randomly generated. The maximum number of generations is 3000. The recombination operator (PMX) is applied with a probability of 0.65, while the mutation operator (SW) was set with a probability of 0.05. The number $n1$ of recombination and $n2$ of parents were set, respectively, to 16 and 18. Parameters (population size, stop criterion, probabilities, etc.) were not chosen at random, but rather by an examination of values previously used with success (see [10] for example). Also, this algorithm presents schedules that improve the maintenance schedule provided by experts, increasing even a third part of the total time, with the corresponding reduction of costs [18].

Table 1 displays the obtained results for the six instances (according to OW) by the different penalties techniques where the following information is showed in the respective columns: $|OW|$ is the number of constraints, $\mathcal{P}$ is the penalty approach, Median represents the median kilometers traveled, Avg represents the average kilometers traveled, and Evals represents the number of thousands of evaluations made by each approach. It should be particularly noticed that any further reference to S1, S2, D1, D2, A1, and A2, stands for the EA-MCMP-SRI algorithm implementing the respective penalty function.

The first group (left side of the table) belongs to the static approach and it can be observed for all instances the minimum median kilometers traveled is obtained by S2 in almost all of them ($|OW| \in \{16, 17, 18, 19, 20\}$) and also this technique obtained the minimum values for the average kilometers traveled. In regards of the number of evaluations, for $|OW| \in \{18, 19, 20\}$, S1 achieved the minimum values whereas for $|OW| \in \{15, 16, 17\}$ the minimum values are obtained by S2.

Table 1. Results obtained with Static, Dynamic, and Annealing penalties with $|OW| \in \{15, 16, 17, 18, 19, 20\}$.

|OW|	$\mathcal{P}$	Median	Avg	Evals	$\mathcal{P}$	Median	Avg	Evals	$\mathcal{P}$	Median	Avg	Evals
15	S1	444.21	451.66	9073	D1	443.97	444.19	10227	A1	444.90	447.74	10311
15	S2	444.94	447.45	8432	D2	443.93	438.46	9863	A2	444.61	448.73	9923
16	S1	446.03	464.29	10157	D1	445.16	449.98	10305	A1	445.32	452.64	10952
16	S2	445.70	452.85	8507	D2	443.91	446.85	8949	A2	446.79	450.05	9815
17	S1	454.31	465.15	9123	D1	446.34	450.97	10499	A1	446.01	456.45	10752
17	S2	446.84	453.30	9091	D2	444.81	450.34	9186	A2	445.96	458.23	10628
18	S1	466.72	471.13	8189	D1	447.08	464.54	10571	A1	461.70	458.84	10742
18	S2	458.80	463.37	9278	D2	445.44	454.29	9186	A2	446.74	453.12	10791
19	S1	468.22	463.97	8934	D1	463.93	460.30	10356	A1	463.87	462.85	10984
19	S2	457.99	459.01	9145	D2	446.60	449.83	9473	A2	471.58	467.91	11146
20	S1	473.95	471.72	7961	D1	467.11	463.43	10689	A1	462.55	460.09	10097
20	S2	469.20	466.72	8969	D2	446.99	456.44	10050	A2	463.15	464.42	11241

The second group (middle side of the table) belongs to the dynamic approach. It can be observed that D2 obtained the minimum values for all instances in regards of the median kilometers traveled, average kilometers traveled and number of evaluations.

The third group (right side of the table) belong to the annealing approach. Here, the best results for the median kilometers traveled are obtained by A1 in three of the six instances ($|OW| \in \{16, 19, 20\}$), for the average kilometers traveled in four of the six instances ($|OW| \in \{15, 17, 19, 20\}$) A1 obtained the best results, and for the number of evaluations in three of the six instances ($|OW| \in \{18, 19, 20\}$).

To make statistical analysis comparing the performance for the average kilometers traveled, the best of each penalty approach is chosen. For S2 and D2 the differences between S1 and D1 are clear. A1 and A2 show some tiny differences that's why A1 is selected.

In each comparison the ANOVA or Kruskall-Wallis tests were applied as it corresponds. With $|OW| = 15$ there were not significative statistical differences among S2, D2, and A1. For this reason a box-plot diagram is shown for this case (Figure 1(a)) where it can be observed that the median is very similar between the approaches and the values obtained by D2 are more compact and near the median compared with the values obtained by the other approaches. In regards of $|OW| \in \{16, 18, 19\}$, D2 achieved significative statistical differences with respect to S2 and A1 (Figure 1 (b), (d) and (e)). Finally, for $|OW| \in \{17, 20\}$, the results achieved by D2 show significative statistical differences with respect to A1 (Figure 1 (c) and (f)). As a complementary report, in Table 2 we display the computational effort of the approaches. As expected, the static approach requires (for almost all instances) less computational resources than the

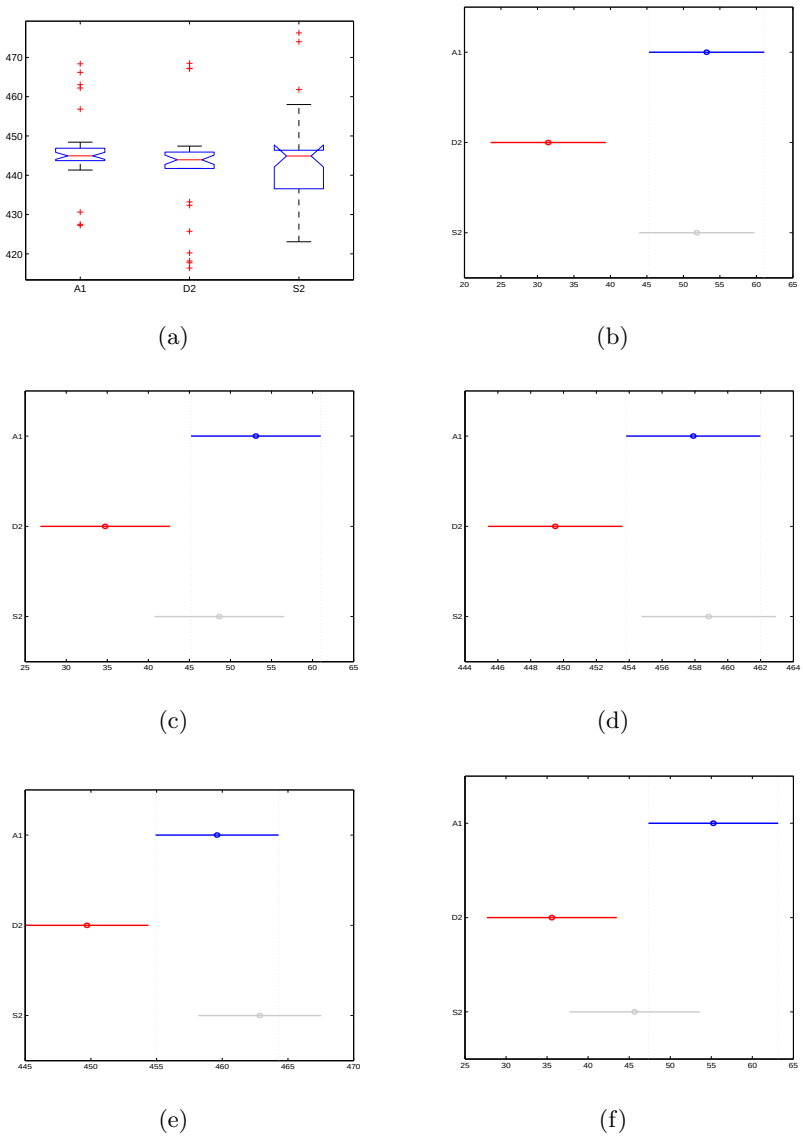

Fig. 1. (a) Box-plots for $|OW| = 15$ and A1, D2, and S2. (b), (c), (d), (e), and (f) displays the statistical differences between the penalty functions according to the Tukey multi-comparison test.

dynamic and annealing approaches. Moreover, the dynamic approach requires less computational effort than the annealing approach for all instances.

Table 2. Best averages of thousands of evaluations made by each penalty approach implemented with $|OW| \in \{15, 16, 17, 18, 19, 20\}$.

|OW|	Static	Dynamic	Annealing
15	8432	9863	10311
16	8507	8949	10952
17	9091	9186	10752
18	9278	9186	10742
19	9145	9473	10984
20	8969	10050	10097

7 Conclusions

PAE is an application built with the objective of providing an effective tool that facilitates the scheduling of maintenance visits to oil fields subject to constraints. Evolutionary Algorithms are metaheuristics that use computational models of evolutionary process. For the constrained scheduling of an oil wells field we used a variant of a multirecombinative approach called EA-MCMP-SRI implementing different penalty approaches for handling the problem constraint. From the carried out experiments we can remark that:

- EA-MCMP-SRI obtained, in general, better results with the dynamic penalty functions.
- D2 approach assures with 95% of confidence the best behavior in two of the six instances tested. For the remaining instances, D2 still shows a better behavior than the other one taking into account the global quality of the variables analyzed.
- In regards of the computational effort of the approaches, the dynamic and annealing approaches consume more computational resources that the static one.

Future works will include the implementation and study of more advanced constraint handling techniques, formulation of different types of constraints and schedules based on multiple maintenance teams.

Acknowledgments We acknowledge the cooperation of the project group of LabTEm and the Universidad Nacional de la Patagonia Austral from which we receive continuous support. The last author also acknowledges the constant support afforded by the Universidad Nacional de San Luis and the ANPCYT that finances his current researches.

References

1. C. Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, 2002.

2. R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society*, 49:1–23, 1943.
3. N. Cruz-Cortes, Trejo-Prez D., and Coello-Coello C. Hangling constraints in global optimization using artifitial immune system. *ICARIS*, 3627:234–247, 2005.
4. A.E. Eiben, C.H. Van Kemenade, and J.N. Kok. Orgy in the computer: Multi-parent reproduction in genetic algorithms. In Springer-Verlag, editor, *3rd European Conference on Artificial Life*, number 929, pages 934–945, 1995.
5. S. Esquivel, H. Leiva, and R. Gallard. Multiple crossovers between multiple parents to improve search in evolutionary algorithms. In *Congress on Evolutionary Computation (IEEE)*, pages 1589–1594, 1999.
6. D. Fogel. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence.* The Institute of Electrical and Electronic Engineers, 1995.
7. D. Goldberg and R. Lingle. Alleles, loci and the traveling salesman problem. In *International Conference on Genetic Algorithms*, pages 154–159, 1987.
8. S. Kirkpatrick, J. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671–6801, 1983.
9. L. Landa-Becerra and C. Coello-Coello. Optimization with constraints using a cultured differential evolution approach. *Genetic and Evolutionary Computation Conference*, 1:27–34, 2005.
10. M. Lasso, D. Pandolfi, M.E. De San Pedro, A. Villagra, and R. Gallard. Solving dynamic tardiness problems in single machine environments. In *Congress on Evolutionary Computation*, volume 1, pages 1143–1149, 2004.
11. Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionaty Computation*, 4(1):1–32, 1996.
12. D. Pandolfi, M. De San Pedro, A. Villagra, G. Vilanova, and R. Gallard. Studs mating immigrants in evolutionary algorithm to solve the earliness-tardiness scheduling problem. *Cybernetics and Systems of Taylor and Francis*, 391-400, 2002.
13. I. Parmee, editor. *The Integration of Evolutionary and Adaptive Computing Technologies with Prod- uct/System Design and Realisation.* Springer-Verlag, 1998.
14. M. Pinedo. *Scheduling: Theory, Algorithms and System.* First edition Prentice Hall, 1995.
15. S. Rao. *Engineering Optimization.* John Wiley and Sons, third edition, 1996.
16. H. P. Schwefel. *Evolution and Optimum Seeking.* John Wiley and Son, 1995.
17. A. Smith and D. Coit. *Handbook of Evolutionary Computation*, chapter Constraint Handling Techniques-Penalty Functions, page C 5.2. Oxford University Press and Institute of Physics, 1997.
18. A. Villagra, E. de San Pedro, M. Lasso, and D. Pandolfi. Algoritmo multirecombinativo para la planificacin dinámica del mantenimiento de locaciones petroleras. *Revista Internacional INFORMACIÓN TECNOLÓGICA*, 19(4):63–70, 2008.