Repair Algorithms and Penalty Functions to Handling Constraints in an Evolutionary Scheduling

A. Villagra[1], D. Pandolfi[1], J. Rasjido[1], C. Montenegro[1], N. Seron[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
{avillagra,dpandolfi,jrasjido,cmontenegro,nseron}@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. Ecosystems are directly and negatively affected by many industrial risky activities such as the oil transportation and exploitation. It is therefore important that oil companies carry out a correct maintenance of their oil fields. In this work we implement in our previously proposed evolutionary optimization tool (called PAE) a set of constraint-handling techniques to solve a constrained version of the maintenance problem. The results and comparisons demonstrated that the proposed repair algorithm required less computational effort (evaluations) with the same quality of solutions for the set of instances used.

1 Introduction

In general, constraints are an integral part of the formulation of any problem. Coello Coello [1] describes 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 [7] 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 combined into one single value.

Finally, hybrid methods are combination of different algorithms and/or mechanisms e.g., fuzzy-logic with EAs, cultural algorithms [6] 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. 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 [10].

As regards repair algorithms, in many combinatorial optimization problems is relatively easy to "repair" an infeasible solution (i.e., to make feasible an infeasible solution). Such a repaired version can be used either for evaluation only, or it can be also replace (with some probability) the original solution. The question of replacing repaired solutions is related with the so-called Lamarckian evolution, which assumes that an individual improves during its lifetime and that the resulting improvements are coded back into the chromosome [11]. We applied this concept to our problem of oil wells maintenance scheduling in [8].

The contribution of this paper is the study of the performance of PAE, our previously proposed evolutionary optimization tool by considering a set of penalty functions and repair algorithms based on set of well known constraint-handling techniques.

The paper continues as follows. Section 2 shows the domain and problem description. Section 3 describes the evolutionary algorithm proposed to solve the problem. Section 4 presents the penalty functions and repair algorithms used in our experimental study whereas Section 5 shows experimental tests and results, and finally in Section 6, we give some conclusions and analyze future research directions.

2 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 that 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, there are oil wells that for diverse causes should be visited more than once (twice or three times) in different shifts, and these are constraints imposed to this problem.

2.1 Problem Formulation

The problem can be precisely stated as defined in [9]:

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

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 one of the oil wells. The objective is to minimize the makespan (Cmax) subject to the dependent times of preparation of the sequence. This model leads to an optimization problem that is NP-Hard [9]. 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 well j to oil well k, t_k the respective time of maintenance in location k, and n the total number of oil wells in the field. Moreover, the above formulation can be extended as follows:

Definition 1 *Let $OW1 \subset \{1, \ldots, N\}$ and $OW2 \subset \{1, \ldots, M\}$ be two subsets of the all oil wells in the field where $OW1 \cap OW2 = \emptyset$. OW1 represents the oil wells that should be visited twice and OW2 represents the oil wells that should be three times. Furthermore, the oil wells in $OW1$ or in $OW2$ can not be scheduled in the same shift. A solution of a constrained instance of the problem that not fulfill the above conditions is considered an infeasible solution.*

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

3 PAE: the evolutionary tool

To solve the problem under the evolutionary approach, 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

to be visited[3]. Each element p_i represents the $i-th$ oil well that will be visited according to the sequence of visits.

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 [4]. It can be viewed as an extension of two-cut crossover for binary string to permutation representation. For the mutation operation used, named 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

[3] It must be noticed that n will varies according the possible elements in $OW1$ and $OW2$ as explained further in this section.

(according to $OW1$ and $OW2$) which implies that many solutions visited in the search space will be infeasible. Moreover, for the constrained version of our problem, the chromosome length will be $n = m + |OW1| + |OW2|$, i.e., $p = (p_1, \ldots, p_m, p_{m+1}, \ldots, p_n)$ where m is the total number of oil wells and $|OW1| + |OW2|$ is the number of the oil wells that must be visited twice or three times. Thus, a $p_i \in \{m+1, \ldots, n\}$ decodes in an integer number that belongs either to $OW1$ or $OW2$. When a solution is evaluated a penalty function or a repair algorithm is applied to infeasible solutions depending on the approach that is being used.

4 Penalty Functions and Repair algorithms considered

To handling constraints in the maintenance scheduling of oil wells we proposed different techniques: three penalty functions and three repair algorithms.

4.1 Penalty Functions

To handling constraints 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 the penalty function was defined in the following way:

– 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 $\mathcal{P}(p)$ will consider each one of these components as follows:

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

where H is the set of oil wells in solution p that not fulfill 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 function $\mathcal{P}$, we present the following combinations of Static, Dynamic and Adaptive penalty functions as follows:

A. **Static Penalty (STT):** Consists of applying the penalty function $\mathcal{P}(p)$ to the infeasible solution. That is to say, adding to the fitness function the penalty value obtained.

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

B. **Dynamic Penalties (DYN):** Consist of multiplied the penalty function $\mathcal{P}(p)$ by the 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 $0 \leq V(g) \leq 1$.

$$F(p) = f(p) + [\mathcal{P}(p) \times V(g)]$$

C. **Adaptive Penalty (ADP):**
For the adaptive approach we propose an adaptive coefficient (ρ) that modifies the penalty function. ρ is updated keeping in mind the degree of constraints violation. To achieve this objective a generations window (fixed in 10 generations) is analyzed and the penalty value imposed is increased or decreased as corresponds. The fitness function is calculated as:

$$F(p) = f(p) + [\mathcal{P}(p) \times \rho]$$

where $\rho \in (0, 2)$ and is calculated analyzing the violations average in the generations window called AV_w and the violations average in the currently generation called AV_c as stated bellow:

$$V_g = \frac{AV_w - AV_c}{AV_w} \times 100$$

Taking in mind the violation grade V_g, ρ is updated as:

$$\rho = \begin{cases} \rho + k & \text{if } V_g < -0.25 \\ \rho - k & \text{if } V_g > 0.25 \\ \rho & \text{otherwise} \end{cases}$$

where k is set to 0.025.

4.2 Repair Algorithms

Three repair algorithms were implemented and they are described in the following:

1. **Lamarckian Approach (LMCK):** Consist of genetically modify an infeasible solution and the transformed infeasible solution, i.e., an feasible one, replaces the infeasible one in the population for further evolution.
2. **Baldwinian Approach (BLDW):** A less destructive approach of the infeasible solutions allows combine learning and evolution. In this approach the solutions are repaired only for their evaluation. Analytic and empiric studies indicate that this technique reduces the speed of convergence of the evolutionary algorithm and allows converging to global optimum [12].
3. **Annealing Approach (ANNL):** This approach is based on the main concepts involved in Simulated Annealing [5]. The infeasible solutions are accepted with certain probability. At first stages the infeasible solutions are accepted then this probability of acceptation is decreasing. When a infeasible solution is not accepted it is repaired.

In all the above approaches, the process of repairing a solution consists in swapping the oil well that is located in an incorrect shift with an randomly selected oil well located in a different shift. This process is repeated to every oil well that not fulfill the problem constraint.

5 Experiments and Results

In order to test the performance of the approaches applied to this problem we performed three experiments considering an oil field with 110 oils wells and two different sets OW1 and OW2 respectively. We defined six instances described in Table 1, where column Ins represents the instance's name, and $|OW1|$ and $|OW2|$ represents respectively the quantity of oil wells that should be visited twice and three times.

Table 1. Instances used in the experiments determined by OW1 and OW2.

Ins	$\|OW1\|$	$\|OW2\|$
I_1	15	0
I_2	17	0
I_3	19	0
I_4	8	8
I_5	9	9
I_6	10	10

In the first experiment we compared the results obtained with the three penalty functions using instances I_1, I_2, and I_3. In the second experiment, we compared the results obtained with the three repair algorithms using the same set of instances as before. And finally, in the third experiment the best technique of the previous experiments is compared using instances I_4, I_5, and I_6. Table 2 displays the obtained results for the first and second experiments where the following information is showed in the respective columns: Ins is the instance's name, $\mathcal{A}$ is the used constraint-handling technique, 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 STT, DYN, and ADP, stands for the EA-MCMP-SRI algorithm implementing the respective penalty function and LMCK, BLDW and ANNL stands for the EA-MCMP-SRI algorithm implementing the respective repair algorithm. In Table 2 for the first group (the first three rows belonging to penalty functions approaches) it can be observed that for all instances DYN obtained the minimum values for the median kilometers traveled and for two instances (I_1 and I_3) obtained minimum values for the average kilometers traveled. As regards the evaluations for two instances (I_2 and I_3) ADP obtained the minimum values. For the second experiment (the last three rows) it can be observed for all instances the minimum values for median and average of kilometers traveled were obtained by ANNL. Nevertheless, the minimum number of evaluations were obtained by LCMK for all instances.

As all samples obtained follow a normal distribution. Thus, we conducted a statistical analysis (ANOVA test) to compare the performance for the average

Table 2. Results obtained by STT, DYN, ADP, LMCK, BLDW, and ANNL on instances I_1, I_2 and I_3.

Ins	$\mathcal{A}$	Median	Avg	Evals	$\mathcal{A}$	Median	Avg	Evals	$\mathcal{A}$	Median	Avg	Evals
I_1	STT	444.94	447.45	**8432**	DYN	**443.93**	**438.46**	9863	ADP	444.73	441.99	9043
I_2	STT	446.84	453.30	9091	DYN	**444.81**	450.34	9186	ADP	445.66	**448.39**	**8001**
I_3	STT	457.99	459.01	9145	DYN	**446.60**	**449.83**	9473	ADP	447.23	453.74	**8919**
I_1	LMCK	448.24	454.93	**8955**	BLDW	448.73	455.03	9093	ANNL	**444.55**	**453.51**	9088
I_2	LMCK	453.44	468.18	**8587**	BLDW	460.20	476.70	9414	ANNL	**447.50**	**455.02**	8692
I_3	LMCK	468.98	468.14	**8702**	BLDW	479.32	477.87	9335	ANNL	**447.92**	**464.96**	8980

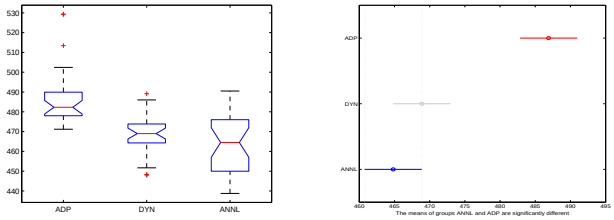
Table 3. Results obtained by ADP, DYN, and ANNL on instances I_4, I_5 and I_6.

Ins	ADP			DYN			ANNL		
–	Median	Avg	Evals	Median	Avg	Evals	Median	Avg	Evals
I_4	482.29	486.89	9131	469.01	468.60	10165	**464.42**	**464.88**	**6127**
I_5	498.30	502.64	9139	**478.62**	**480.47**	9809	479.08	481.01	**7077**
I_6	514.36	514.51	9167	490.69	494.60	10057	**490.23**	**492.61**	**7008**

kilometers traveled obtained by the three penalty functions. From the analysis we observed that not significative statical differences were found between DYN and ADP, so both algorithms were used in the third experiment.

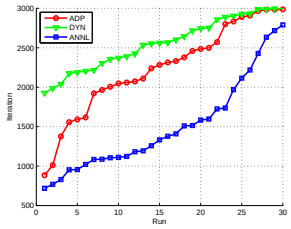
Table 3 shows the results obtained from the third experiment. It can be observed that for two instances (I_4 and I_6) for all variables shown ANNL obtained the minimum values. While DYN obtained minimum values for instance I_5 for median and average kilometers traveled. To make a deeper analysis of the results obtained we applied the ANOVA test. In Figure 1(a) for I_4 it can be observed that ANNL obtained the minimum value for the median kilometers traveled. Nevertheless the median obtained by DYN is very similar and this can also be observed in Figure 1(b) where the differences between ANNL and ADP are statistically significatives. Finally, for the evals variable it can be observed in Figure 1 (c) less computational effort (evaluations) required by ANNL that the effort required by the other two approaches.

In Figure 2(a) for I_5 it can be observed that there is little difference between the medians of DYN and ANNL, with the median obtained by DYN slightly smaller than the obtained by ANNL. This is confirmed in Figure 2(b) where statistical differences are detected between ADP and the other two techniques, ANNL and DYN. Finally, for the evals variable it can be observed in Figure 2 (c) that also for this instance less computational effort (evaluations) is required by ANNL with respect other techniques. In Figure 3 it can be seen a similar behavior of the algorithms to that described for Figure 1.



(a)

(b)



(c)

Fig. 1. (a) Box-plot , (b) Statistical differences and (c) Computational effort for I_4 with ADP, DYN and ANNL approaches, respectively.

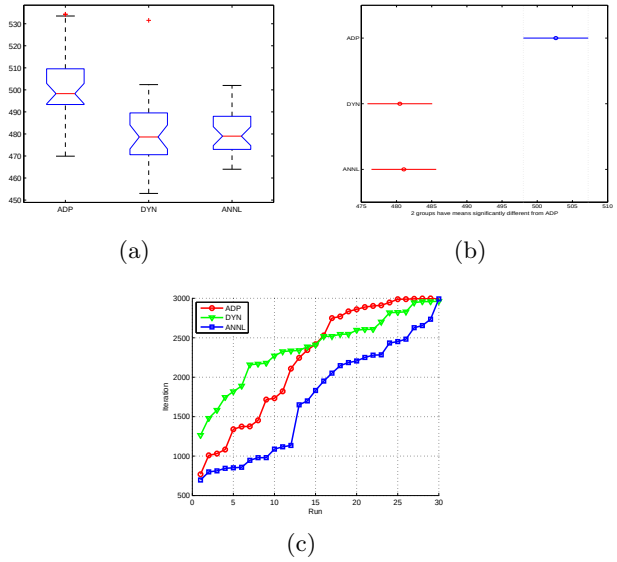
6 Conclusions

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

- In general, the Dynamic and Annealing approaches obtained similar values for the analyzed quality variables (median and average kilometers traveled).
- In regards of the computational effort (evaluations) we can said with a 95% of confidence that the Annealing approach requires less number of evaluations with the same quality of solutions that the others approaches.

Future works will include the design, implementation and study of a more advanced constraint handling technique taking into account the results observed in this work. Also, the formulation of different types of constraints and schedules based on multiple maintenance teams will be considered.

Acknowledgments We acknowledge the cooperation of the project group of LabTEm and the Universidad Nacional de la Patagonia Austral from which



(a) (b)

(c)

Fig. 2. (a) Box-plot , (b) Statistical differences and (c) Computational effort for I_5 with ADP, DYN and ANNL approaches, respectively.

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. Handling constraints in global optimization using artifitial immune system. *ICARIS*, 3627:234–247, 2005.
4. D. Goldberg and R. Lingle. Alleles, loci and the traveling salesman problem. In *International Conference on Genetic Algorithms*, pages 154–159, 1987.
5. S. Kirkpatrick, J. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671–6801, 1983.
6. 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.
7. Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionaty Computation*, 4(1):1–32, 1996.

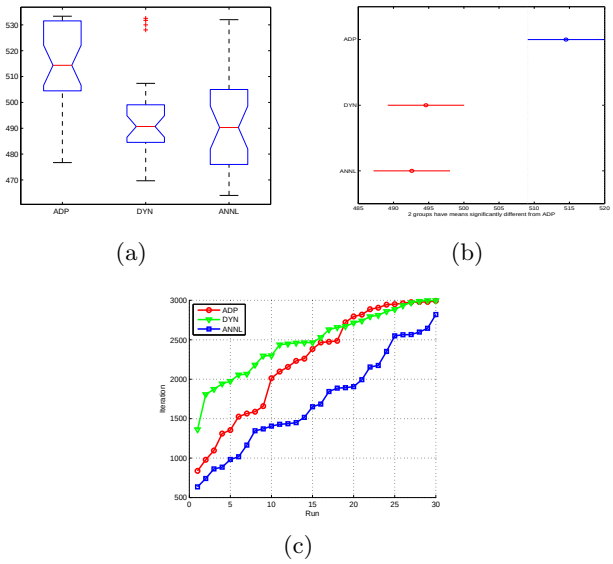


Fig. 3. (a) Box-plot , (b) Statistical differences and (c) Computational effort for I_6 with ADP, DYN and ANNL approaches, respectively.

8. D. Pandolfi, A. Villagra., E. De San Pedro, M. Lasso, and G. Leguizamón. An experimental study of an evolutionary tool for scheduling in oil wells. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA-AIE)*, pages 576–585. Springer-Verlang, 2010.
9. M. Pinedo. *Scheduling: Theory, Algorithms and System.* First edition Prentice Hall, 1995.
10. 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.
11. E. J. Steele, R. A. Lindley, and R. V. Blanden. *Lamarckś Signature. How Retrogenes Are Changing Darwinś Natural Selection Paradigm.* Perseus Books, 1998.
12. D. Whitley, S. Gordon, and K. Mathias. Lamarckian evolution, the baldwin effect and function optimization. In *Parallel Problem Solving from Nature - PPSN III*, 1994.