Differential Evolution with a Repair Method to Solve Dynamic Constrained Optimization Problems

María-Yaneli Ameca-Alducin, Efrén Mezura-Montes and Nicandro Cruz-Ramírez
Artificial Intelligence Research Center, University of Veracruz
Xalapa, Veracruz, México
yaneliameca@gmail.com, {emezura,ncruz}@uv.mx

ABSTRACT

An algorithm inspired in two differential evolution variants is proposed to solve Dynamic Constrained Optimization Problems (DCOPs). It is also added a repair method based on the differential mutation, which does not require feasible solutions as reference. This approach is compared against state-of-the-art algorithms to solve DCOPs. Different performance measures are employed in the tests to show the competitiveness of our proposal at different change frequencies.

CCS Concepts

•**Computing methodologies** → **Genetic algorithms;**
•**Mathematics of computing** → *Continuous functions;*

Keywords

Differential Evolution; Constraint-handling; Dynamic optimization

1. INTRODUCTION

A DCOP can be seen as a search problem where its fitness landscape and feasible region change through time. Evolutionary algorithms (EAs) were not designed to deal with dynamic search spaces because they lack mechanisms to detect search space changes [11, 12]. Without loss of generality, a DCOP can be defined as to:

Find $\vec{x}$, at each time t, which:

$$\min_{\vec{x} \in F_t \subseteq [L,U]} f(\vec{x}, t)$$

where $t \in N^+$ is the current time,

$$[L, U] = \{\vec{x} = (x_1, x_2, ..., x_D) | L_i \leq x_i \leq U_i, i = 1 \dots D\}$$

is the search space,

GECCO '15, July 11 - 15, 2015, Madrid, Spain
 ISBN 978-1-4503-3488-4/15/07…$15.00
DOI: http://dx.doi.org/10.1145/2739482.2768471

subject to:

$$F_t = \{\vec{x} | \vec{x} \in [L, U], g_i(\vec{x}, t) \leq 0, i = 1 \dots m, \\ h_j(\vec{x}, t) = 0, j = 1 \dots p\}$$

which is called the feasible region at time t. $\forall \vec{x} \in F_t$ if there exists a solution $\vec{x}^* \in F_t$ such that $f(\vec{x}^*, t) \leq f(\vec{x}, t)$, then $\vec{x}^*$ is called a feasible optimal solution and $f(\vec{x}^*, t)$ is called the feasible optimal value at time t.

Four types of DCOPs are defined: i) a static objective function and static constraints (i.e. a static constrained optimization problem), ii) a dynamic objective function and static constraints, iii) a static objective function and dynamic constraints, and iv) a dynamic objective function and dynamic constraints.

The Genetic Algorithm (GA) is the most popular EA to solve DCOPS, where different mechanisms to deal with fitness landscape and feasible region changes have been considered, such as diversity maintenance and repair methods [3, 4, 11, 12]. Other meta-heuristics like the Dynamic Constrained T-Cell (DCTC) [2], and the Gravitational Search Algorithm (GSA) [14] were proposed to solve DCOPs as well. Differential Evolution (DE) has been also recently adapted to deal with DCOPs [13]. As it was mentioned above, a popular mechanism added to an EA when solving DCOPs is the repair method, which requires feasible solutions as reference to convert infeasible solutions into feasible ones. The main motivation of this work lies on proposing an algorithm based on a scarcely EA used for DCOPs, DE in our case, but with the novelty of combining two of its variants (DE/rand/1/bin and DE/best/1/bin) and also a novel but simple repair method which does not require feasible solutions to work. A set of experiments focused in the frequency of the change in the search space is carried out, because the repair method has an important role in the algorithm's recovery after a change. The results obtained are compared against different state-of-the-art approaches to solve DCOPs.

The rest of the paper is divided as follows: Section 2 introduces the proposed algorithm. Section 3 presents the experiments and results obtained by the algorithm . Finally, Section 4 includes the conclusions and future research.

2. PROPOSED ALGORITHM

The proposed algorithm is based on the so-called Differential Evolution with Combined Variants (DECV), originally proposed to solve static constrained optimization problems [8]. A preliminary version adapted to solve DCOPs, called dynamic differential evolution with combined variants (DDECV) was presented in [1]. The elements of the algo-

rithm are the following:

i) *The change detection mechanism* consists on solution re-evaluation [15]. At each generation, two solutions are evaluated and their objective function values and constraints values are compared against their previous values. If any value is different, this indicates that a change has been detected.

ii) *The exploration promotion mechanism* is activated after the change detection mechanism, and here is where the two DE variants are switched because DDECV starts by using DE/rand/1/bin. However, after the change detection mechanism, DE/best/1/bin is adopted for a number of generations, while the F value is increased during the same period of time to promote exploration [8].

iii) *Constraint-handling*. Three feasibility rules [5] are used to deal with the constraints: (1) Between two feasible solutions, the one with the best objective function value is chosen, (2) between a feasible solution and an infeasible solution, the feasible one is chosen, and (3) between two infeasible solutions, the one with the lowest sum of constraint violation is chosen.

iv) *Random-immigrants*. A number of randomly generated solutions called immigrants [16] are inserted into the current population at the end of each generation. The number of immigrants is increased after a detected change and returns to its original value after that.

v) *Convergence promotion*. A hill-climber-like local search operator [7] is applied to a randomly chosen solution from the current population by a determined number of iterations. The final obtained solution replaces the worst one in the current population.

Recalling from the DCOPs literature, the algorithms that present a competitive performance are those with a repair method as GA+ Repair [11], DE + Repair [13] and GSA + Repair [14]. The use of repair methods has been employed as constraint-handlers. The repair method consists in the use of a set of feasible solutions that serve as reference to convert infeasible solutions of the current population in feasible solutions [9, 11, 13, 14].

Unlike the above mentioned way to work, our repair method does not use feasible solutions, it is a resampling approach based on the differential mutation operator. For each infeasible solution, three new and temporal solutions are generated at random with the only aim to apply the differential mutation operator (see line 4 in Algorithm 1) as if a mutant vector is created in DE. At each generation, before the selection between the parent and offspring solutions, if the offspring is infeasible, the repair method is applied until it is repaired or Repair_Limit attempts are computed. The details of the repair method can be seen in Algorithm 1.

It is important to remark that DDECV + Repair does not consider the convergence promotion mechanism (i.e. it does not use the local search operator). The details of DDECV + Repair are shown in Algorithm 2, where the repair method is remarked in boldface.

3. EXPERIMENTS AND RESULTS

3.1 Experimental setup

DDECV + Repair was tested on the 18 functions of the G24 benchmark, whose details can be found in [11, 12]. The parameter settings for the benchmark problems were the following: number of runs = 50, number of changes = 12, change frequency at 500, 1000 and 2000 evaluations, the objective function severity was medium ($k = 0.5$) and the constraint severity was medium ($S = 20$).

The results obtained by DDECV + Repair were compared against algorithms from the state-of-the-art to solve DCOPs: i) a Genetic Algorithm (GA) with elitism (GAElit) [11], ii) a GA with random immigrants (RIGAElit) [4], iii) a GA with hypermutation (HyperMElit) [3], iv) a GA with a repair method (GA + Repair) [11], v) a differential evolution with a repair method (DE + Repair) [13], vi) the gravitational search algorithm with a repair method (GSA + Repair) [14], and vii) the original DDECV [1]. DE + Repair and GSA + Repair were compared only with 1000 evaluations as change frequency because no results were found for the remaining frequencies.

The parameter values used by DDECV and DDECV + Repair were taken from [1]: NP = 25 CR= 0.8399, F= 0.9644 F (after change)= 1.0820, Immigrants before change= 5, Immigrants after change = 3, iterations for local search =8 (just for DDECV) and Repair_Limit=100 (just for DDECV + Repair).

The performance measures adopted in this work were the following:

offline error [11, 14] is defined as the average of the sum of errors in each cycle divided by the sum of the number of cycles. The offline error is always greater than or equal to zero.

The recovery rate (RR) is used to analyze how quickly an algorithm recovers after a change and starts converging to the new best solution before the next change occurs. Such new solution is not necessarily the global optimum. The RR value would be 1 in the best case where the algorithm is able to recover and converge to the best solution immediately after a change, and closer to zero where the algorithm is unable to recover.

The absolute recovery rate (ARR) is similar to the RR, but is used to analyze how fast is an algorithm to start converging to the global optimum before the next change occurs. The ARR value would be 1 in the best case when the algorithm is able to recover and converge to the global optimum immediately after a change, and would be zero in case the algorithm is unable to recover.

For more details about these measures the reader is referred to [11, 12].

3.2 Results

The experiments were divided as follows:

i) An indirect comparison, where the results of other algorithms were taken from [1, 10, 13, 14] and compared with our proposal by using offline error. The statistical validation was made with the non-parametric 95%-confidence Kruskal-Wallis (KW) test and a post-hoc test (Bonferroni Dunn) [6]. Table 1 shows the results of the KW test, where the results obtained by DDECV + Repair are compared against other algorithms by using different change frequencies (500, 1000, and 2000 evaluations) and a medium severity of change (i.e., k=0.5 and S=20).

At 500, 1000 and 2000 evaluations for a change, DDECV + Repair outperformed the first four algorithms, including GA + Repair, with the exception of DDECV. Particularly for a change frequency of 1000 evaluations, DDECV + Repair performed in a similar way as DE + Repair, GSA + Repair and DDECV. Figure 1 shows the results of the Bonferroni Dunn post-hoc test, which confirm the findings in

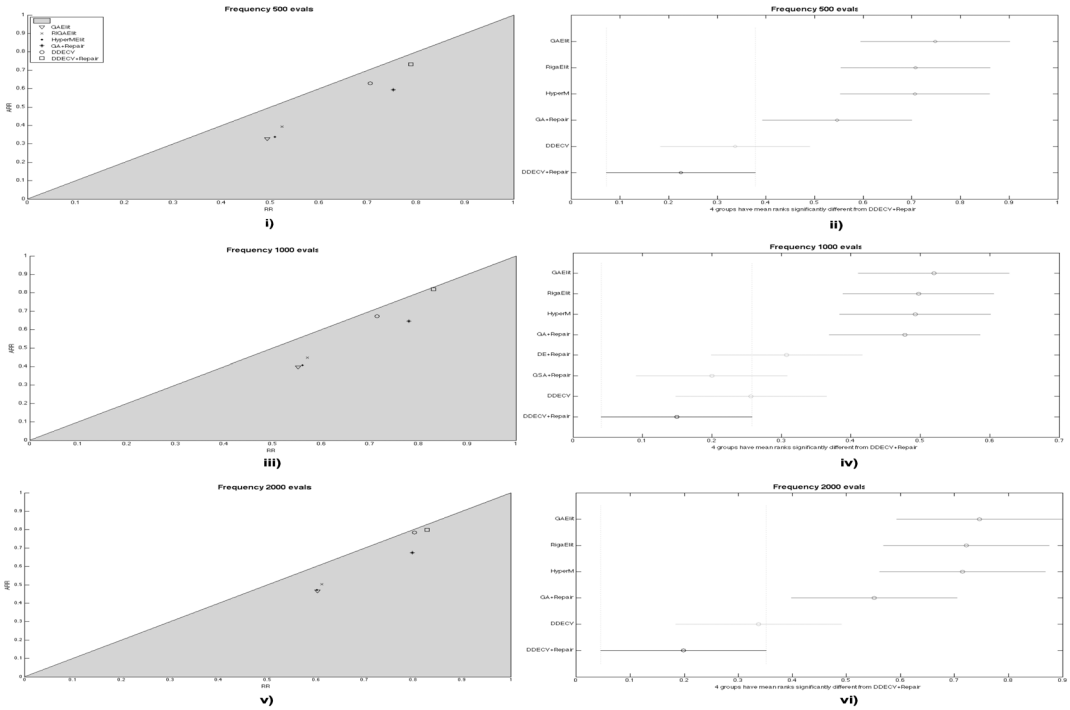

Figure 1: Mapping of the RR/ARR scores of GAElit, RIGAElit, HyperMElit, GA + Repair, DDECV and DDECV + Repair to the RR-ARR diagram for three change frequencies: i) 500 evaluations, iii) 1000 evaluations, and v) 2000 evaluations. If a point is closer to the right hand-side area of the graph, it indicates a faster recovery. Moreover, if the point lies on the diagonal line, the algorithm has been able to recover from the change and converge to the new global optimum. Post-hoc Bonferroni test results based on the offline error values with different change frequencies are showed in: ii) 500 evaluations, iv) 1000 evaluations and vi) 2000 evaluations.

Table 1.

ii) A direct comparison, where the algorithms from the specialized literature to solve DCOPs were implemented and their corresponding performances were evaluated by using RR and ARR. To improve the results in the GAElit and GA + Repair algorithms, a change detection mechanism employed in DDECV [1] was added. Figure 1 depicts the RR and ARR diagrams of the following algorithms: GAElit, RIGAElit, HyperMElit, GA + Repair, DDECV and DDECV + Repair in three change frequencies (500, 1000 and 2000 evaluations) and a medium change severity (i.e., k=0. 5 and S=20). Such results suggest that DDECV + Repair recovers faster than the compared algorithms, including DDECV, particularly when the change is more frequent (i.e. every 500 evaluations). Furthermore, DDECV + Repair provided the best recovery to the global optimum after a change, regardless of its frequency.

4. CONCLUSIONS

An EA based on two variants of differential evolution (DE/rand/1/bin and DE/best/1/bin), coupled with a resampling-based feasible-solutions-free repair method inspired in the differential mutation operator, was proposed to solve DCOPs. Two experiments based on three performance measures (offline error, recovery rate and absolute recovery rate) with three change frequencies were carried out. Seven algorithms were used for comparison purposes. The overall results showed that DDECV + Repair outperformed most of them based on the offline error measure, regardless of the change frequency. Particularly with 1000 evaluations for a change, it was just comparable with respect to three algorithms (the original DDECV, DE + Repair and GSA + Repair). However, the main advantage of the proposed algorithm was its capability to recover faster than others after a change, mostly in frequent changes (i.e. at every 500 evaluations). It is important to remark that, unlike other repair methods, the repair method proposed in this work does not require feasible solutions to operate.

The future work considers using other measures in order to verify the performance of DDECV + Repair. Also, different values for the severity of the change will be included in the tests.

Algorithm 1 Repair_Method

Require: $\vec{u}_{i,G}$ {trial vector}
1: $counter = 0$
2: **while** $\vec{u}_{i,G}$ is infeasible and $counter \leq$ Repair_Limit **do**
3: Generate three random vectors ($\vec{u}_{r0,G}$, $\vec{u}_{r1,G}$ and $\vec{u}_{r2,G}$)
4: $\vec{u}_{i,G} = \vec{u}_{r0,G} + F(\vec{u}_{r1,G} - \vec{u}_{r2,G})$
5: $counter = counter + 1$
6: **end while**
7: Return $\vec{u}_{i,G}$

5. ACKNOWLEDGMENTS

The first author acknowledges support from the Mexican Council for Science and Technology (CONACyT) to pursue graduate studies at the University of Veracruz. The second author acknowledges support from CONACyT through project No. 220522.

Table 1: Offline error comparison of DDECV+Repair against other algorithms with three change frequencies, k=0.5 and S=20. "$X^{(+)}$" means that the algorithm in the corresponding column outperformed algorithm X based on the 95%-confidence KW test. "$X^{(-)}$" means that the algorithm in the corresponding column was outperformed by algorithm X based on the 95%-confidence KW test. "NA" means not available.

Freq.	GAElit (1)	RIGAElit (2)	HyperMElit (3)	GA + Repair (4)	DE + Repair (5)	GSA + Repair (6)	DDECV (7)	DDECV + Repair(8)
500	$7^{(-)}$, $8^{(-)}$	$7^{(-)}$, $8^{(-)}$	$7^{(-)}$, $8^{(-)}$	$8^{(-)}$	NA	NA	$1^{(+)}$, $2^{(+)}$, $3^{(+)}$	$1^{(+)}$, $2^{(+)}$, $3^{(+)}$, $4^{(+)}$
1000	$6^{(-)}$, $7^{(-)}$, $8^{(-)}$	$6^{(-)}$, $7^{(-)}$, $8^{(-)}$	$6^{(-)}$, $7^{(-)}$, $8^{(-)}$	$6^{(-)}$, $7^{(-)}$, $8^{(-)}$		$1^{(+)}$, $2^{(+)}$, $3^{(+)}$, $4^{(+)}$	$1^{(+)}$, $2^{(+)}$, $3^{(+)}$, $4^{(+)}$	$1^{(+)}$, $2^{(+)}$, $3^{(+)}$, $4^{(+)}$
2000	$7^{(-)}$, $8^{(-)}$	$7^{(-)}$, $8^{(-)}$	$7^{(-)}$, $8^{(-)}$	$8^{(-)}$	NA	NA	$1^{(+)}$, $2^{(+)}$, $3^{(+)}$	$1^{(+)}$, $2^{(+)}$, $3^{(+)}$, $4^{(+)}$

Algorithm 2 DDECV+Repair algorithm

```
1: G=0
2: Create a randomly-generated initial population x⃗_{i,G} ∀i, i = 1,...,NP
3: Evaluate each x⃗_{i,G} ∀i, i = 1,...,NP
4: eval = eval + NP
5: while eval ≤ Max_eval do
6:    for i ← 1 to NP do
7:       if i = 1 or i = NP/2 then
8:          Change_detection_Mechanism (x⃗_{i,G})
9:          eval = eval + 1
10:      end if
11:      Exploration_promotion_mechanism
12:      if u⃗_{i,G} is infeasible then
13:         Repair_Method(u⃗_{i,G}) {Algorithm 1}
14:      end if
15:      eval = eval + 1
16:      if f(u⃗_{i,G}) is better than f(x⃗_{i,G}) based on the feasibility
         rules then
17:         x⃗_{i,G+1} = u⃗_{i,G}
18:      else
19:         x⃗_{i,G+1} = x⃗_{i,G}
20:      end if
21:   end for
22:   Add NI immigrants to the current population and evaluate
      them
23:   eval = eval + NI
24:   G = G + 1
25: end while
```

6. REFERENCES

[1] M.-Y. Ameca-Alducin, E. Mezura-Montes, and N. Cruz-Ramirez. Differential evolution with combined variants for dynamic constrained optimization. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 975–982, July 2014.

[2] V. Aragón, S. Esquivel, and C. Coello. Artificial immune system for solving dynamic constrained optimization problems. In E. Alba, A. Nakib, and P. Siarry, editors, *Metaheuristics for Dynamic Optimization*, volume 433 of *Studies in Computational Intelligence*, pages 225–263. Springer Berlin Heidelberg, 2013.

[3] H. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical report, Naval Research Lab Washington DC, 1990.

[4] H. Cobb and J. Grefenstette. Genetic algorithms for tracking changing environments. In S. Forrest, editor, *ICGA*, pages 523–530. Morgan Kaufmann, 1993.

[5] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(24):311–338, 2000.

[6] J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[7] S. Hernandez, G. Leguizamon, and E. Mezura-Montes. A hybrid version of differential evolution with two differential mutation operators applied by stages. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2895–2901, 2013.

[8] E. Mezura-Montes, M. E. Miranda-Varela, and R. del Carmen Gómez-Ramón. Differential evolution in constrained numerical optimization. an empirical study. *Information Sciences*, 180(22):4223–4262, 2010.

[9] Z. Michalewicz and G. Nazhiyath. Genocop iii: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In *Evolutionary Computation, 1995., IEEE International Conference on*, volume 2, pages 647–651 vol.2, Nov 1995.

[10] T. Nguyen and X. Yao. Detailed experimental results of ga, riga, hyperm and ga+repair on the g24 set of benchmark problems. Technical report, School Comput. Sci., Univ. Birmingham, Birmingham, U.K., 2010. available at: http://www.staff.livjm.ac.uk /enrtngu1/Papers/DCOP fulldata.pdf.

[11] T. Nguyen and X. Yao. Continuous dynamic constrained optimization: The challenges. *IEEE Transactions on Evolutionary Computation*, 16(6):769–786, 2012.

[12] T. Nguyen and X. Yao. Evolutionary optimization on continuous dynamic constrained problems - an analysis. In S. Yang and X. Yao, editors, *Evolutionary Computation for Dynamic Optimization Problems*, volume 490 of *Studies in Computational Intelligence*, pages 193–217. Springer Berlin Heidelberg, 2013.

[13] K. Pal, C. Saha, and S. Das. Differential evolution and offspring repair method based dynamic constrained optimization. In B. Panigrahi, P. Suganthan, S. Das, and S. Dash, editors, *Swarm, Evolutionary, and Memetic Computing*, volume 8297 of *Lecture Notes in Computer Science*, pages 298–309. Springer International Publishing, 2013.

[14] K. Pal, C. Saha, S. Das, and C. Coello-Coello. Dynamic constrained optimization with offspring repair based gravitational search algorithm. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2414–2421, 2013.

[15] H. Richter. Detecting change in dynamic fitness landscapes. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pages 1613–1620, 2009.

[16] Y. Shengxiang. Memory-based immigrants for genetic algorithms in dynamic environments. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 1115–1122, New York, NY, USA, 2005. ACM.