A Novel Coral Reefs Optimization Algorithm for Multi-objective Problems

S. Salcedo-Sanz, A. Pastor-Sánchez,
D. Gallo-Marazuela, and A. Portilla-Figueras

Universidad de Alcalá, Alcalá de Henares, Madrid, Spain

Abstract. In this paper we detail a new algorithm for multi-objective optimization, the Multi-Objective Coral Reefs Optimization (MO-CRO) algorithm. The algorithm is based on the simulation of the coral reefs processes, including corals' reproduction and fight for the space in the reef. The adaptation to multi-objective problems is an easy process based on domination or non-domination during the process of fight for the space in the reef. The final MO-CRO is an easily implementing and fast algorithm, quite simple, but able to keep diversity in the population of corals (solutions) in a natural way. Experiments in different multi-objective benchmark problems have shown the good performance of the proposed approach in cases with limited computational resources, where we have compared it with the well known NSGA-II algorithm as reference.

1 Introduction

The Coral Reefs Optimization Algorithm (CRO) is an evolutionary bio-inspired approach based on the simulation of the processes in a coral reefs, recently proposed in [1]. The CRO can be classified into the family of bio-inspired algorithms which try to artificially simulate the behavior of a specific natural ecosystem to tackle optimization problems, similarly to ant colony optimization [2], particle swarm optimization algorithm [3], artificial bee colony approach (ABC) [4] or the weed colonization algorithm [5]. The CRO has been proven to be effective in several single-objective optimization problems, obtaining better solutions than alternative optimization algorithms in the literature.

Basically, the CRO algorithm starts from a population of individuals encoding different solutions to a given optimization problem. These solutions are located in an square grid (*reef*), where there are also empty spaces at the beginning of the algorithm. The algorithm is though to simulate the process of coral reproduction (sexual and asexual reproduction operators are applied), and the process of coral reef formation, where a fight for space occurs. Thus, in each step of the CRO algorithm a coral larvae formation is carried out, and each larva tries to occupy a place in the reef. It depends on how strong the larva is (how good the solution to the optimization problem is), or if it is lucky enough to find an empty place in the reef. Note that empty places in the reef are scarce after some generations of larvae, though a process of corals depredation ensures the possibility of empty places in the reef even at the final stages of the algorithm.

H. Yin et al. (Eds.): IDEAL 2013, LNCS 8206, pp. 326–333, 2013.

In this paper we describe a multi-objective version of the CRO algorithm (MO-CRO). The MO-CRO can be directly obtained from the basic CRO algorithm with very few adaptations, resulting in an effective algorithm for multi-objective optimization problems. The main characteristic of the MO-CRO is its simplicity and the easy implementation of the algorithm, and its good performance in cases where computational resources are limited. In the paper we describe in detail the algorithm and its characteristics and provide some experimental proofs of its performance in different benchmark optimization problems, where we compare with the well-known NSGA-II algorithm.

The rest of the paper has been structured as follows: next section states the main concepts and definitions of multi-objective optimization problems. Section 3 details the basic CRO algorithm, and the adaptations needed to make it appropriate to solve multi-objective optimization problems. Section 4 describes the experimental part of the paper, in which the MO-CRO has been applied to different benchmark problems and compared to the NSGA-II approach. Section 5 closes the paper by giving some final remarks and conclusions.

2 Multi-objective Problems: Basic Concepts and Definitions

Following [6], we define next several important concepts in multi-objective optimization such as domination, equivalency, non-domination with respect to sets and pareto optimal vectors and fronts, that will be helpful later on, in the definition of the MO-CRO.

Let us consider a multi-objective optimization algorithm, defined in a search space $\mathcal{S}$, formed by vectors with n-components $\mathbf{x} = x_1, \ldots, x_n$. Let us consider a vector of m objective functions $(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n))$, that maps each vector $\mathbf{x}$ with a fitness space. The aim of a multi-objective optimization problem is to find the set of optimal tradeoff solutions in $\mathcal{S}$ called *Pareto optimal set*. Note that usually not all the pareto set is calculated, but a representative subset is enough in the majority of cases. The following definitions are useful to clearer define the problem:

Given two parameter vectors $\mathbf{x}$ and $\mathbf{y}$, we say that $\mathbf{x}$ *dominates* $\mathbf{y}$ ($y \prec x$) iff $\mathbf{x}$ is at least as good as $\mathbf{y}$ in all objectives, and better in at least one. Similarly, $\mathbf{x}$ is equivalent to $\mathbf{y}$ ($\mathbf{x} \equiv \mathbf{y}$) iff they are identical in all objectives considered. Two parameter vectors are incomparable iff they are not equivalent, and neither dominates the other. A parameter vector $\mathbf{x}$ is non-dominated with respect to a set Ψ of vectors iff there is no vector in Ψ that dominates $\mathbf{x}$. Ψ is called *nondominated set* iff all vectors in Ψ are mutually nondominating. The set of corresponding objective vectors for a nondominated set is a nondominated front. A parameter vector is defined as *Pareto optimal* iff is nondominated with respect to the set of all parameter vectors. Note that an improvement in any one objective of a pareto optimal vector means worsening at least one other objective. The Pareto optimal set is the set of all Pareto optimal parameter vectors, and the corresponding set of objective vectors is the Pareto optimal front. Note that the Pareto optimal

set is a subset of the search space, whereas the Pareto optimal front is a subset of the fitness space.

3 The Multi-objective Coral Reefs Optimization Algorithm

3.1 Basic CRO Algorithm

The CRO is a novel meta-heuristic approach based on corals' reproduction and coral reefs formation, proposed in [1]. Basically, the CRO is based on the artificial modeling of a coral reef, Λ, consisting of a $N \times M$ square grid. We assume that each square (i, j) of Λ is able to allocate a coral (or colony of corals) $\Xi_{i,j}$, representing a solution to a given optimization problem, which is encoded as a string of numbers in a given alphabet $\mathcal{I}$. The CRO algorithm is first initialized at random by assigning some squares in Λ to be occupied by corals (i.e. solutions to the problem) and some other squares in the grid to be empty, i.e. holes in the reef where new corals can freely settle and grow in the future. The rate between free/occupied squares in Λ at the beginning of the algorithm is an important parameter of the CRO algorithm, which is denoted as ρ, and note that $0 < \rho_0 < 1$. Each coral is labeled with an associated *health* function $f(\Xi_{ij}) : \mathcal{I} \rightarrow \mathbb{R}$, that represents the problem's objective function. The CRO is based on the fact that reef will progress, as long as healthier (stronger) corals (which represent better solutions to the problem at hand) survive, while less healthy corals perish.

After the reef initialization described above, a second phase of reef formation is artificially simulated in the CRO algorithm: a simulation of the corals' reproduction in the reef is done by sequentially applying different operators. This sequential set of operators is then applied until a given stop criteria is met. Several operators to imitate corals' reproduction are defined, among them: a modeling of corals' sexual reproduction (broadcast spawning and brooding), a model of asexual reproduction (budding), and also some catastrophic events in the reef, i.e. polyps depredation. After the sexual and asexual reproduction, the set of larvae formed (new solutions to the problem), try to locate a place to grow in the reef. It could be in a free space, or in an occupied once, by fighting against the coral actually located in that place. If larvae are not successful in locate a place to grow in a given number of attempts, they are depredated in this phase.

1. *Broadcast Spawning (external sexual reproduction)*: the modeling of coral reproduction by *broadcast spawning* consists of the following steps:

   1.a. In a given step k of the reef formation phase, select uniformly at random a fraction of the existing corals ρ_k in the reef to be broadcast spawners. The fraction of broadcast spawners with respect to the overall amount of existing corals in the reef will be denoted as F_b. Corals that are not selected to be broadcast spawners (i.e. $1-F_b$) will reproduce by brooding later on, in the algorithm.

1.b. Select couples out of the pool of broadcast spawner corals in step k. Each of such couples will form a coral larva by sexual crossover, which is then released out to the water. Note that, once two corals have been selected to be the parents of a larva, they are not chosen anymore in step k (i.e. two corals are parents only once in a given step).

2. *Brooding (internal sexual reproduction)*: as previously mentioned, at each step k of the reef formation phase in the CRO algorithm, the fraction of corals that will reproduce by brooding is $1 - F_b$. The brooding modeling consists of the formation of a coral larva by means of a random mutation of the brooding-reproductive coral (self-fertilization considering hermaphrodite corals). The produced larva is then released out to the water in a similar fashion than that of the larvae generated in step 1.b.

3. *Larvae setting*: once all the larvae are formed at step k either through broadcast spawning (1.) or by brooding (2.), they will try to set and grow in the reef. First, the health function of each coral larva is computed. Second, each larva will randomly try to set in a square (i, j) of the reef. If the square is empty (free space in the reef), the coral grows therein no matter the value of its health function. By contrast, if a coral is already occupying the square at hand, the new larva will set only if its health function is better than that of the existing coral. We define a number κ of attempts for a larva to set in the reef: after κ unsuccessful tries, it will be depredated by animals in the reef.

4. *Asexual reproduction*: in the modeling of asexual reproduction (budding or fragmentation), the overall set of existing corals in the reef are sorted as a function of their level of healthiness (given by $f(\Xi_{ij})$), from which a fraction F_a duplicates itself and tries to settle in a different part of the reef by following the setting process described in Step 3. Note that a maximum number of identical corals (μ) will be allowed in the reef.

5. *Depredation in polyp phase*: corals may die during the reef formation phase of the CRO algorithm. At the end of each reproduction step k, a small number of corals in the reef can be depredated, thus liberating space in the reef for next coral generation. The depredation operator is applied with a very small probability P_d at each step k, and exclusively to a fraction F_d of the worse health corals in Λ.

3.2 Multi-objective CRO

The adaptation of the CRO to multi-objective problems is an easy task by starting from the basic CRO approach. It is established in the *Larvae Setting* process of the algorithm and in a generalization of the depredation operator: once all the larvae from broadcast spawning and brooding have been produced, they start the setting process one by one, trying to established themselves into the reef. When an existing coral occupies a given position in the reef that is tried by a

larva, a fight for the space occurs. In the MO-CRO, this fight for the space is based on domination of solutions. Let us call Ξ_A to the coral currently occupying a given location on the reef, and Ξ_B the larva challenging for the space. In the MO-CRO, Ξ_B wins the fight (and occupies the place of Ξ_A) iff $\Xi_A \prec \Xi_B$. In any other case, Ξ_A wins, and the challenging larva either tries another place in the reef, or die, depending on its current κ value. Note that in case of equivalency between solutions ($\Xi_A \equiv \Xi_B$), the current solution in the reef is maintained. A second adaptation is needed in order to provide diversity to the reef: In the MO-CRO, a fix number μ of corals with the same value in all objective functions is allowed in the population. After the larvae setting process, we obtain the number of corals in the reef with the same value in all objectives, let us call it β, and if $\mu < \beta$, $(\beta - \mu)$ randomly chosen corals are depredated. We call this adaptation as Extreme Depredation Operator (EDO).
Several points must be notated:

1. The algorithm will never destroy vectors belonging to the pareto optimal front, since they will dominate or will be incomparable to any other possible vector in the search space.
2. Dominated solutions may coexist with better individuals if empty places exist in the coral reef.
3. The generation of diversity in the first stages of the algorithm is comparable to that of evolutionary approaches.
4. Due to 1., the genetic drift causing the loss of diversity in the population of evolutionary algorithms does not occur within the CRO algorithm.
5. Thanks to the EDO, the The CRO does not, therefore, need to establish a ranking on the individuals, nor needs for modification in fitness values in order to preserve diversity in the population.
6. The MO-CRO is nearly as fast as the basic CRO, and it is straightforward to implement from the basic CRO version.

4 Experimental Part

In order to test the performance of the proposed MO-CRO, we have tackled a number of benchmark functions, usually employed in the literature for the evaluation of multi-objective approaches [7], that are defined in Table 1. In order to contrast the results obtained with the MO-CRO, we will carry out a comparison with a successful algorithm in the literature, the NSGA-II algorithm [7]. The methodology of the comparison will be the following: a fix (limited) number of function evaluations is set for both the NSGA-II and MO-CRO. The idea is to analyze the performance of the algorithms in cases of reduced computational resources available. Specifically, the NSGA-II has been run with 100 individuals, during 200 generations, and the MO-CRO has been adjusted to approximately evaluate the same number of functions. A matlab implementation of the NSGA-II algorithm has been used [8]. In this version of the algorithm, a Simulated Binary Crossover [9] and a polynomial mutation [10] are implemented. We have also used this operators in the MO-CRO algorithm.

Table 1. Benchmark function to test the MO-CRO proposed in this paper

Function	n	variable bounds	Expression
FON	3	$[-4, 4]$	$f_1(\mathbf{x}) = 1 - \exp\left(-\sum_1^3 \left(x_i - \frac{1}{\sqrt{3}}\right)^2\right)$
			$f_2(\mathbf{x}) = 1 - \exp\left(-\sum_1^3 \left(x_i + \frac{1}{\sqrt{3}}\right)^2\right)$
SCH	1	$[-1000, 1000]$	$f_1(x) = x^2$
			$f_2(x) = (x-2)^2$
ZDT1	30	$[0, 1]$	$f_1(\mathbf{x}) = x_1$
			$f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{\frac{x_1}{g(\mathbf{x})}}\right]$
			$g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^n x_i\right)/(n-1)$
ZDT2	30	$[0, 1]$	$f_1(\mathbf{x}) = x_1$
			$f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - (\frac{x_1}{g(\mathbf{x})})^2\right]$
			$g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^n x_i\right)/(n-1)$
ZDT3	30	$[0, 1]$	$f_1(\mathbf{x}) = x_1$
			$f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{\frac{x_1}{g(\mathbf{x})}} - \frac{x_1}{g(\mathbf{x})}\sin(10\pi x_1)\right]$
			$g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^n x_i\right)/(n-1)$
ZDT6	10	$[0, 1]$	$f_1(\mathbf{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$
			$f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - (\frac{f_1(\mathbf{x})}{g(\mathbf{x})})^2\right]$
			$g(\mathbf{x}) = 1 + 9\left[\left(\sum_{i=2}^n x_i\right)/(n-1)\right]^{0.25}$

The comparative results obtained with the MO-CRO and NSGA-II algorithms are shown in Figure 1. As can be seen, in the FON function, the MO-CRO is able to get the optimal pareto front, though in this case the NSGA-II algorithm covers better the extremes of the front. In SCH benchmark the MO-CRO covers the optimal front in a similar way that the NSGA-II. In the most difficult functions (ZDT), the performance of the MO-CRO with the resources considered (number of function evaluation) is better than the NSGA-II. In ZDT1 function, the NSGA-II does not obtain the optimal pareto front with the number of function evaluations fixed. The MO-CRO is able to get this optimal front, covering it in a reasonable way. The behavior of the algorithms is quite similar in function ZDT2, the NSGA-II is not able to obtain the optimal pareto front with the limited number of objective function evaluation, and the MO-CRO obtains it, reasonably covering the complete front. Function ZDT3 considers a discontinuous optimal front. The MO-CRO is able to obtain it, but it does not cover completely part of the front. The NSGA-II obtains a suboptimal front, but it covers it almost completely. Finally, in Function ZDT6, both algorithms as able to obtain the optimal pareto front. The NSGA-II covers it better than the MO-CRO, but it also shows suboptimal solutions at the left-most part of the front.

Summarizing, the proposed MO-CRO obtains good results in terms of optimal front location even with limited computational resources (limited number of possible function evaluation), improving the behaviour of the NSGA-II algorithm. It seems that the NSGA-II outperforms the MO-CRO in terms of diversity of solutions in the optimal front. Due to its simplicity, the proposed MO-CRO algorithm could be a good option in real-world multi-objective problems and applications.

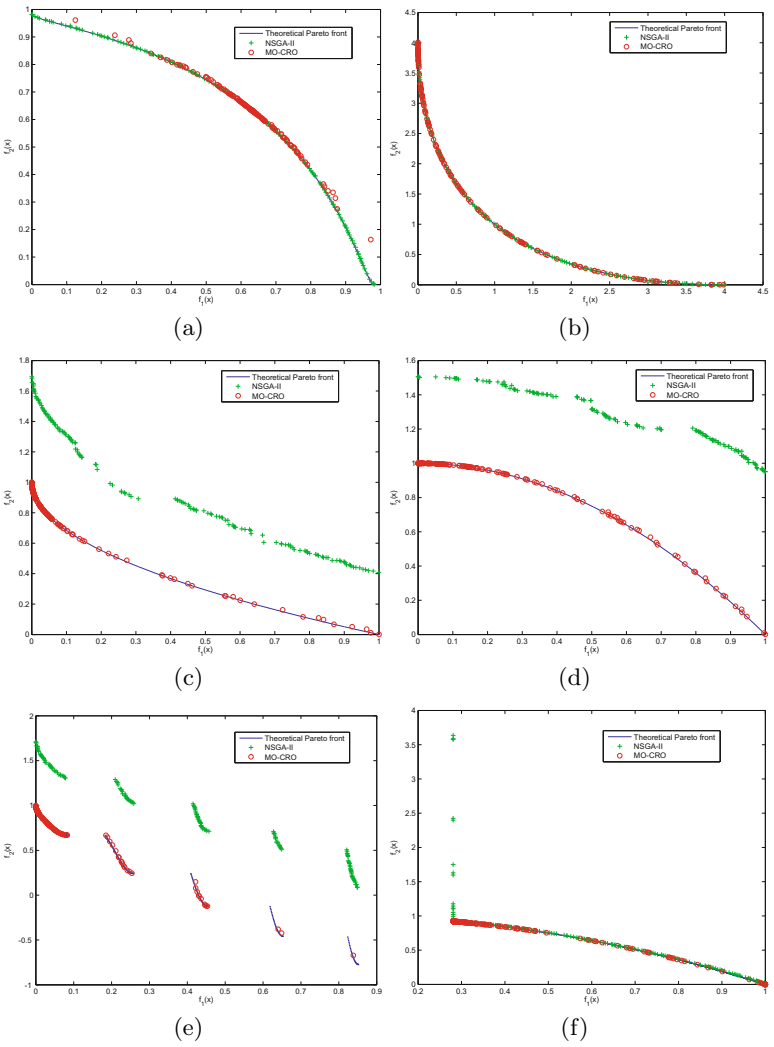

Fig. 1. Comparison MO-CRO vs. NSGA-II in the different benchmark functions considered; (a) FON; (b) SCH; (c) ZDT1; (d) ZDT2; (e) ZDT3 and (f) ZDT6.

5 Conclusions

In this paper we have presented a novel algorithm for multi-objective optimization problems: the Multi-Objective Coral Reefs Optimization algorithm (MO-CRO). We have detailed the algorithm's structure and the adaptations necessary in the CRO to convert it into a multi-objective algorithm. We have verified the goodness of the proposed approach in several multi-objective optimization benchmark problems with limited computational resources available, where we have compared its performance with that of a popular evolutionary algorithm for multi-objective optimization (the NSGA-II algorithm). The proposed MO-CRO main characteristics are its simplicity and easy implementation, its low-computational cost and the good results obtained in several test functions.

Acknowledgements. This work has been partially supported by Spanish Ministry of Science and Innovation, under project numbers ECO2010-22065-C03-02.

References

1. Salcedo-Sanz, S., Del Ser, J., Gil-López, S., Landa-Torres, I., Portilla-Figueras, J.A.: The Coral Reefs Optimization Algorithm: A new metaheuristic algorithm for hard optimization problems. In: Proc. of the 15th International Conference on Applied Stochastic Models and Data Analysis (ASMDA), Mataró, Barcelona (2013)
2. Dorigo, M., Maziezzo, V., Colorni, A.: The ant system: optimization by a colony of cooperating ants. IEEE Transactions on Systems, Man and Cybernetics B 26(1), 29–41 (1996)
3. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. of the 4th IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
4. Karaboga, D., Basturk, B.: On the performance of the artificial bee colony (ABC) algorithm. Applied Soft Computing 8, 687–697 (2008)
5. Mehrabian, A.R., Lucas, C.: A novel numerical optimization algorithm inspired from weed colonization. Ecological Informatics 1, 355–366 (2006)
6. Huban, S., Hingston, P., Barone, L., While, L.: A Review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation 10(5), 477–506 (2006)
7. Deb, K., Pratab, A., Agrawal, S., Merayivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
8. `http://www.mathworks.es/matlabcentral/fileexchange/10429-nsga-ii-a-multi-objective-optimization-algorithm`
9. Deb, K., Agarwal, R.B.: Simulated Binary Crossover for continuous search space. Complex Systems 9, 115–148 (1995)
10. Raghuwanshi, M.M., Kakde, O.G.: Survey on multiobjective evolutionary and real coded genetic algorithms. In: Proc. of the 8th Asia Paciffc Symposium on Intelligent and Evolutionary Systems, pp. 150–161 (2004)