Proceedings, 15th Applied Stochastic Models and Data Analysis (ASMDA2013)
International Conference, Mataró (Barcelona), Spain 25 - 28 June 2013

The Coral Reefs Optimization Algorithm: An Efficient Meta-heuristic for Solving Hard Optimization Problems

S. Salcedo-Sanz[1], J. Del Ser[2], I. Landa-Torres[2], S. Gil-López[2], and A. Portilla-Figueras[1]

[1] Department of Signal Processing and Communications, Universidad de Alcalá, 28805 Alcalá de Henares, Madrid, Spain.
(E-mail: sancho.salcedo@uah.es)
[2] Tecnalia Research & Innovation. Parque Tecnológico de Bizkaia, 48170 Zamudio, Bizkaia, Spain.

Abstract. This paper presents a novel bio-inspired algorithm to tackle complex optimization problems: the Coral Reefs Optimization (CRO) algorithm. The CRO algorithm artificially simulates a coral reef, where different corals (namely, solutions to the optimization problem considered) grow and reproduce in coral colonies, fighting by choking out other corals for space in the reef. This fight for space, along with the specific characteristics of the corals' reproduction, produces a robust meta-heuristic algorithm, shown to be powerful for solving hard optimization problems. In this research the CRO algorithm is detailed and tested in several continuous and discrete optimization problems, obtaining advantages over other existing meta-heuristic techniques. The obtained results confirm the excellent performance of the proposed algorithm.
Keywords: Coral Reefs Optimization algorithm, optimization problems, modern meta-heuristics, bio-inspired algorithms.

1 Introduction

In the last years, huge research efforts have been conducted towards solving hard optimization problems, by well balancing the trade-off between the complexity incurred by the utilized method and the optimality of the produced solutions. These problems, often characterized by search spaces of high dimensionality (either discrete or continuous), non-linear objective functions and/or stringent constraints, arise frequently in Science and Engineering applications. In such fields, classical optimization approaches do not provide in general good solutions to these problems, or are just not applicable, due to the unmanageable search space structure or its huge size.

In this context, modern optimization heuristics and meta-heuristics have been lately the core of research, aimed at solving the aforementioned lack of efficient methods. A good number of such algorithms are *bio-inspired* techniques such as evolutionary algorithms (EA), which includes a whole family of techniques such as Genetic Algorithms [1], Evolutionary Strategies [2], Evolutionary Programming [3], Differential Evolution [4], among others. These schemes are based on concepts borrow from natural evolution and survival of the fittest individuals in Nature. Likewise, Ant Colonies Optimization (ACO) [5] are based

on the social behavior of ants, Particle Swarm Optimization (PSO) approaches [6] are in essence elegant algorithms specially well-suited for continuous optimization problems. They imitate the behavior of birds flocks or fish schools. There have been more research activity on bio-inspired meta-heuristics, with approaches such as Artificial Bee Colony [7], the Invasive Weed Optimization Algorithm (IWO), [8], based on weed growth and their invasive properties, or the so-called Cuckoo search approach [9], built upon the reproduction and breeding of the cuckoo bird, among others.

In this paper we present a novel bio-inspired meta-heuristic for optimization problems, which will be hereafter coined as the Coral Reefs Optimization (CRO) algorithm. The CRO algorithm is based on an artificial simulation of the process of coral reefs' formation and reproduction. During this process, the CRO algorithm emulates different phases of coral reproduction and fight for space in the reef, which ultimately renders an efficient algorithm for solving difficult optimization problems. The proposed CRO approach can be regarded as a cellular-type evolutionary scheme, with superior exploration-exploitation properties thanks to the particularities of the emulated reef structure and coral reproduction. The performance of the proposed approach has been tested in different benchmark problems obtaining very good results in comparison with alternative approaches in the literature.

The rest of this article is structured as follows: for the sake of self-completeness of the manuscript, the next section provides an introduction to coral reefs and corals' structure and reproduction. Next, Section 2 presents the CRO algorithm in detail, including an analysis of similarities and differences with other existing meta-heuristics. Section 3 shows the performance of the CRO algorithm in different optimization problems. Finally, Section 4 ends the paper by giving some concluding remarks.

2 The Coral Reefs Optimization algorithm

The CRO is a novel meta-heuristic approach based on corals' reproduction and coral reefs formation. 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 rho, and note that $0 < \rho_0 < 1$. Each coral is labeled with an associated *health* function $f(\Xi_{ij}) : \mathcal{I} \to \mathbb{R}$, 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. This second phase of the CRO can be detailed as follows:

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). These couple selection can be done uniformly at random or by resorting to any fitness proportionate selection approach (e.g. roulette wheel).

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) are 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. For the sake of simplicity in the parameter setting of the CRO algorithm, the value of this fraction may be set to $F_d = F_a$. Any other assignment may also apply provided that $F_d + F_a \leq 1$ (i.e. no overlap between the asexually reproduced and the depredated coral sets).

3 Experiments and Numerical Results

In this paper we carry out a first performance assessment of the proposed CRO algorithm in different test problems. Specifically, different well-known continuous and discrete benchmark problems are under consideration: continuous analytical functions and several instances of the *Max-Ones* and *3-bit Deceptive*.

We have selected other meta-heuristic algorithms for comparison: Evolutionary Algorithms, Genetic Algorithms (EA and GA, [1]) and Harmony Search (HS, [10]), which have obtained excellent results in a wide range of optimization problems during the last years. Regarding the continuous benchmark problems, we have compared the results obtained by the CRO in the same problems tackled in [11].

Following this rationale, the encoding strategy used to represent the produced solutions for the aforementioned problems is set identical for all the algorithms under comparison. Specifically, real encoding has been adopted for the continuous benchmark problems, whereas the *Max Ones* and *3-bit Deceptive* problems resort to standard binary encoding. On the other hand, values of all parameters controlling the CRO approach have been set to be comparable to that of its counterparts tested in every benchmark function. Therefore we have kept the number of function evaluations constant for all the compared algorithms in *Maxones* (15000), whereas for the *3-bit Deceptive* problem the total number of function evaluations is set to 50000 for GA and HS, and 30000 for the proposed CRO. In the continuous benchmark functions, we have set the number of function evaluations to be comparable with the results in [11]. For every simulation instance, 30 executions of each algorithm have been launched so as to obtain well-sampled performance statistics (best, average and standard

deviation of the metric after all iterations are done). Note that the size of the population – $N \times M$ for CRO, population length L for the GA, and harmony memory size HM for HS – have been set equal for all the experiments for the sake of fairness in the comparison of the algorithms: in the *Max Ones* problem $N \times M = 5 \times 10$, $L = 50$ and $HM = 50$ and in the *3-bit Deceptive* problem $N \times M = 10 \times 10$, $L = 100$ and $HM = 100$. The CRO parameters F_b and ρ has been set to $F_b = 0.9$ and $\rho = 0.7$, unless otherwise stated in the discussion on the specific simulated application.

3.1 CRO Evaluation in Continuous Benchmark Problems

This first round of experiments includes four well-known benchmark functions, on which the proposed CRO is comparatively assessed with respect to different hybrid evolutionary algorithms described in [11]. In these experiments we have incorporated Gaussian and Cauchy mutations [3] in the internal reproduction (brooding) of the corals in order to accommodate the corresponding operator to the real encoding of the solutions. In the Gaussian mutation we have established a fixed standard deviation $\sigma = (max - min)/100$, where max and min are the maximum and minimum values that each component of the solution can take, whereas in the Cauchy mutation the value of the τ parameter has been fixed to 1 following the guidelines in [3]. The rest of operators in the CRO are the ones shown in Section 2.

Table 1 lists the results obtained by three different versions of the CRO (with Gaussian, Cauchy and Gaussian-Cauchy internal reproduction) in the benchmark functions tackled in this first round of experiments. Also included are the results for different versions of the hybrid evolutionary algorithm proposed in [11], labelled as Hybrid Adaptive Evolutionary Algorithm (HAEA) in what follows. It is straightforward to note that the CRO approach is able to obtain better results than the different versions of HAEA consistently – and with statistical significance positively checked through Kruskal-Wallis tests – in all the functions under consideration. The inclusion of both Gaussian and Cauchy mutations in the brooding coral reproduction (always maintaining the number of functions evaluations) appears to improve the performance of the CRO solver.

3.2 CRO Evaluation in Discrete Benchmark Problems

The first discrete benchmark problem considered is the well-known *Max Ones* problem, often used in a number of previous works aimed at evaluating different approaches of genetic algorithms (e.g. see [**?**,11] and references therein). This optimization problem is defined in a binary search space $\mathcal{S} = \{0, 1\}^n$, where n stands for the dimension of the space. The *One Max* problem is then defined as

$$\max_{\mathbf{x} \in \mathcal{S}} f(\mathbf{x}) = \frac{100}{n} \sum_{i=1}^{n} x_i \qquad [\%]. \tag{1}$$

Table 1. Results (mean/standard deviation) obtained in the different continuous benchmark functions tested.

Algorithm	Rosenbrock	Schwefel	Rastrigin	Griewank
CRO (G)	$7.0 \cdot 10^{-5}/5.0 \cdot 10^{-6}$	$1.3 \cdot 10^{-4}/2.0 \cdot 10^{-6}$	$7.1 \cdot 10^{-3}/3.0 \cdot 10^{-3}$	0.22/0.05
CRO (C)	$7.2 \cdot 10^{-5}/5.6 \cdot 10^{-6}$	$1.3 \cdot 10^{-4}/1.6 \cdot 10^{-6}$	$6.7 \cdot 10^{-3}/2.3 \cdot 10^{-3}$	0.03/0.02
CRO (G+C)	$2.3 \cdot 10^{-5}/1.0 \cdot 10^{-6}$	$1.3 \cdot 10^{-4}/1.4 \cdot 10^{-6}$	$4.3 \cdot 10^{-3}/1.5 \cdot 10^{-3}$	0.05/0.02
HAEA (XUG)	$7.0 \cdot 10^{-4}/1.0 \cdot 10^{-5}$	$5.6 \cdot 10^{-3}/0.01$	0.05/0.02	0.055/0.03
HAEA (XU)	$4.1 \cdot 10^{-3}/4.0 \cdot 10^{-3}$	1.3/0.93	0.24/0.15	0.5/0.2
HAEA (XG)	$1.3 \cdot 10^{-3}/3.6 \cdot 10^{-3}$	140.5/123.7	7.7/3.2	0.05/0.02
HAEA (GU)	$1.4 \cdot 10^{-4}/2.5 \cdot 10^{-3}$	201.9/81.2	6.3/1.4	1.6/0.38

Despite the evident simplicity of its definition, this problem is challenging for optimization algorithms when dealing with large values of the space dimensionality n.

Table 2 summarizes the results (maximum, average and standard deviation) obtained by CRO, GA and HS in *Max Ones* instances of varying size from $n = 50$ to $n = 500$. As one may expect, in the scenarios of smallest dimension all the utilized heuristic approaches are able to obtain the optimum solution (100%) in every run of the algorithm. However, when the dimensions of the simulated problem increase, the differences between the CRO and the other tested algorithms become more significant. Specially remarkable is the fact that the CRO obtains the best value in all the instances with a very high probability (over 99% of the times in which the algorithm was run). HS also obtains good solutions, but notably worse than the GA even in the smallest instances. By contrast, the CRO clearly dominates GA and HS, specially in the largest *Max Ones* instances.

Table 2. Results obtained by CRO, GA and HS in *Max Ones* problems of increasing size. The results are shown in best/average/standard deviation over 30 runs of the algorithms.

n	CRO	GA	HS
50	100/100/0	100/100/0	100/100/0
100	100/100/0	100/100/0	98/95.67/0.92
150	100/100/0	100/100/0	94.67/90.84/1.13
200	$100/99.98/9.12 \cdot 10^{-4}$	100/99.93/0.17	90/87.32/0.88
250	$100/99.97/7.3 \cdot 10^{-4}$	100/99.81/0.25	86.80/84.64/1.04
300	$100/99.96\ /8.45 \cdot 10^{-4}$	100/99.61/0.39	83.67/82.0700/0.62
350	$100/99.96/9.8 \cdot 10^{-4}$	100/99.21/0.46	81.4300/80.03/0.71
400	$100/99.95/7.3 \cdot 10^{-4}$	99.50/98.67/0.58	79.50/78.45/1.04
450	100/99.93/0.13	99.55/98.11/0.67	78.67/76.97/0.99
500	100/99.92/0.1	98.60/97.04/0.75	78/75.99/0.69

The second discrete benchmark problem addressed is the maximization of the aforementioned *3-bit Deceptive* function, which has been previously utilized to evaluate different improvements in genetic and evolutionary heuristics [11]. The 3-bit deceptive function is a binary optimization problem defined in blocks of 3 bits. Each 3-bit block is assigned a value according to Table 3. The optimization of the function is known to be computationally hard for heuristic algorithms, since the `111` block (optimum since it is assigned the highest value) is "surrounded" by low-valued blocks of two 1s (i.e. with small Hamming distance with respect to `111`). Different size functions (integer multiple of 3) are considered in this study, i.e. $n = \{15, 30, 45, 60, 75, 90, 105, 120\}$.

Table 4 shows the results obtained by the CRO in the considered 3-bit Deceptive functions, and its comparison to those of HS and GA. In this problem the CRO clearly obtains the best results among all the compared algorithms. Indeed, it is able to obtain the optimum (maximum) value in all the instances and in almost every executed run. The performance of the alternative algorithms degrades significantly in the largest instances, though in the smallest ones the GA is able to obtain the optimum value.

Table 3. Value assignment in the considered *3-bits Deceptive* function.

Groups of 3 bits	Value
1 1 1	80
0 0 0	70
0 0 1	50
0 1 0	49
1 0 0	30
1 1 0	3
1 0 1	2
0 1 1	1

Table 4. Results obtained by CRO, HS and GA in the considered *3-bit Deceptive* instances. The results are shown in best/average/standard deviation over 30 runs of the algorithms.

n	CRO	HS	GA	Upper Bound
15	400/400/0	400/399.66/1.82	400/400/0	400
30	800/800/0	800/792/8.05	800/795/6.82	800
45	1200/1200/0	1190/1159/14.93	1200/1179.3/13.37	1200
60	1600/1600/0	1560/1517.3/21.96	1590/1562.70/18.74	1600
75	2000/2000/0	1910/1882/20.97	1990/1940.30/22.04	2000
90	2400/2400/0	2280/2243/22.63	2340/2297.30/21.16	2400
105	2800/2799.70/1.82	2660/2598.80/34.20	2730/2687.70/27.75	2800
120	3200/3200/0	2990/2924.8/37.90	3090/3049/21.22	3200

4 Conclusions

In this paper we have presented a novel algorithm to solve optimization problems, inspired by the process of coral reefs formation, and guided by coral reproduction, reef expansion and fight for the space in the reef. The algorithm, named as the Coral Reef Optimization (CRO) algorithm, is a kind of cellular evolutionary algorithm rendering very good properties of convergence to global optima. In this paper we have studied the main characteristics of the proposed CRO and analyzed its comparison to other existing meta-heuristic approaches in different benchmark problems. The promising obtained results encourage the application of the proposed CRO approach to other practical optimization paradigms of high complexity.

Acknowledgements

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

References

1.A. E. Eiben and J. E. Smith, "Introduction to evolutionary computing," Springer-Verlag, Natural Computing Series 1st edition, 2003.
2.H. G. Beyer and H. P. Schwefel, "Evolution strategies - A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3-52, 2002.
3.X. Yao, Y. Liu and G. Lin, "Evolutionary Programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82-102, 1999.
4.R. Storn and K. Price, "Differential Evolution - A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341-359, 1997.
5.M. Dorigo, V. Maziezzo and A. Colorni, "The ant system: optimization by a colony of cooperating ants," *IEEE Transactions on Systems, Man and Cybernetics B*, vol. 26, no. 1, pp. 29-41, 1996.
6.J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of the 4th IEEE International Conference on Neural Networks*, pp. 1942-1948, 1995.
7.D. Karaboga and B. Basturk, "On the performance of the artificial bee colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, pp. 687-697, 2008.
8.A. R. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecological Informatics*, vol. 1, pp. 355-366, 2006.
9.X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. of the World Conference on Nature & Biologically Inspired Computing*, pp. 210-214, 2009.
10.Z. W. Geem, J. Hoon Kim and G. V. Loganathan, A New Heuristic Optimization Algorithm: Harmony Search, *Simulation*, vol. 76, no. 2, 2001. 60-68.
11.J. Gómez, "Self Adaptation of operator rates in evolutionary algorithms," GECCO 2004, *Lecture Notes in Computer Science*, vol. 3102, pp. 1162-1173, 2004.