Performance Comparison of Cuckoo Search and Differential Evolution Algorithm for Constrained Optimization

Mahmud Iwan Solihin, Mohd Fauzi Zanil
Faculty of Engineering, Technology & Built Environment, UCSI University, Malaysia

Abstract. Cuckoo Search (CS) and Differential Evolution (DE) algorithms are considerably robust meta-heuristic algorithms to solve constrained optimization problems. In this study, the performance of CS and DE are compared in solving the constrained optimization problem from selected benchmark functions. Selection of the benchmark functions are based on active or inactive constraints and dimensionality of variables (i.e. number of solution variable). In addition, a specific constraint handling and stopping criterion technique are adopted in the optimization algorithm. The results show, CS approach outperforms DE in term of repeatability and the quality of the optimum solutions.

1. Introduction
Cuckoo Search (CS) algorithm is a novel meta-heuristic optimization algorithm that has been developed in 2009 [1]. On the other hand, the Differential Evolution (DE) algorithm is introduced in 1995 [2]. DE is considerably robust optimization algorithm as it has been proven in many applications when the performance is compared with other algorithms such as particle swarm optimization [3] and ant colony algorithm [4]. These optimization algorithms have gained popularity recently due to their simplicity [5] and efficiency to solve difficult many problems [6, 7].

In this paper, the performance of CS and DE algorithm in solving constrained optimization problem is compared. CS and DE are considered as a robust optimization algorithm which the DE optimization algorithm was proposed earlier [3]. The constrained optimization problem used in this paper is taken from the established benchmark functions and it has been combined with a specific constrain handling.

2. The constrained optimization problem
The aim of a constrained optimization is to get the optimal values of solution variable (which the solution is within the boundary), so that the objective function is optimized and all constraints are satisfied. To achieve this goal, the optimization algorithm evaluate the objective function by searching a value of solution variable in the bounded space, $S$. Thus, an optimal solution is found in the feasible space, $F$, once the objective (minimum or maximum) is met, where $F \subseteq S$ [8].

Assuming a minimization problem, the general constrained problem is defined as:
Minimize:
\[ f(X), \ X = (x_1, x_2, ..., x_D) \]
Subject to:
\[ g_p(X) \leq 0, \ p = 1, 2, ..., n_p \]
\[ h_q = 0, \ q = 1, 2, ..., n_q \]
Boundary constraints: \( X \in [l_b, u_b] \)

where \( X \) is solution variables, \( D \) is dimension of the problem, \( n_p \) and \( n_q \) are the number of inequality and equality constraints respectively, \( l_b \) and \( u_b \) are specified lower and upper bounds for variable solutions respectively.

2.1. Constraint handling

An efficient and adequate constraint-handling technique is a key element in the design of optimization algorithm. Although the use of penalty functions is the most common technique for constraint-handling, there are a lot of different approaches for dealing with constraints [9]. A comprehensive discussion about constraint handling is presented by [10].

In constraint handling using penalty approach, a penalty is added to the objective function to penalize an individual for constraint violation so that the constrained optimization seems to be converted to unconstraint optimization. The optimization might be inefficient with this technique due to the conversion and boundary space.

In this study, a dynamic constraint handling approach is adopted in order to improve the efficiency, i.e reducing the computation time. The dynamic constraint handling called dynamic-objective constraint-handling method (DOCHM) is adopted from Lu and Chen work [11].

Through defining auxiliary function \( F(X) \), the dynamic constraint handling converts the original problem into bi-objective optimization problem \( \min(F(X), f(X)) \), where \( F(X) \) is treated as the first objective function and \( f(X) \) is the second (the main) objective.

The auxiliary function \( F(X) \) will be merely used to determine whether or not an individual (candidate solution) is within the feasible region and how close a particle is to the feasible region. If an individual lies outside the feasible region, the individual will take \( F(X) \) as its optimization objective. Otherwise, the individual will instead optimize the real objective function \( f(X) \). During the optimization process if an individual leaves the feasible region, it will once again optimize \( F(X) \). Therefore, the optimizer has the ability to dynamically drive the individuals into the feasible region. The dynamic constraint handling can be illustrated in the following pseudo-code:

\[
\begin{align*}
\text{If} & \quad F(X) = 0 \quad ;(\text{constrains are satisfied}) \\
& \quad f(x) = f(X) \quad ;(\text{the main objective func.}) \\
\text{else} & \quad f(x) = F(X) \quad ;(\text{the auxiliary objective func.})
\end{align*}
\]

The auxiliary objective function is defined as:
\[ F(X) = \sum_{i=1}^{D} \max(0, d_i) \]

where \( d_i \) represents the distance of an individual (candidate solution), represented by ‘*’ marks, to the constraint violation boundary as illustrated in Figure 1.
2.2. Stopping Criterion
The objective of min-max optimization is usually clear, i.e. the global optimum should be found. However, it is not easy to decide when the execution of an optimization algorithm should be terminated. For practical applications, the stopping criteria selection can significantly influence the computational time of an optimization process. Due to bad selection, an optimization process might be terminated before the population has converged, or computational resources might be wasted because the optimization process took longer time to complete [3].

In this study, the number of function evaluation (FE) in feasible region is used as stopping criterion, i.e. to terminate the optimization run. This FE counts for how many times the objective function has been evaluated in the optimization loop and the constraint condition are satisfied, i.e. evaluation in feasible region.

2.3. Benchmark functions
Three benchmark functions are selected for the constrained optimization experiment. These benchmark functions are taken from [12]. These function varies from the simplest representation to a complicate form in terms of constraint and dimensionality (number of variable).

**Benchmark function 1:** Minimize the function
\[ f_1(X) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \]
Subject to nonlinear constraints,
\[ x_1 + x_2^2 \geq 0 \]
\[ x_1^2 + x_2 \geq 0 \]
with \( x_1 \in [-0.5, 0.5] \) and \( x_2 \leq 1.0 \). The global optimum is \( X^* = (0.5, 0.25) \) for \( f(X^*) = 0.25 \). Both constraints are inactive.

**Benchmark function 2:** Minimize the function
\[ f_2(X) = (x_1 - 2)^2 - (x_2 - 1)^2 \]
Subject to nonlinear constraint,
\[ -x_1^2 + x_2 \geq 0 \]
and linear constraint,
\[ x_1 + x_2 \leq 2 \]
The global optimum is \( X^* = (1, 1) \) for \( f(X^*) = 1 \). Both constraints are active.

**Benchmark function 3:** Minimize the function
\[ f_3(X) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^2 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \]
Subject to nonlinear constraints,
\[-127 + 2x_1^2 + 3x_2^3 + x_3 + 4x_4^2 + 5x_5 \leq 0\]
\[-282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0\]
\[-196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0\]
\[4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0\]
with \(-10 \leq x_i \leq 10\) for \(i = 1, 2, \ldots, 7\).

The global optimum is \(X^* = (2.3305, 1.9514, -0.4775, 4.3657, -0.6245, 1.0381, 1.5942)\) for \(f_3(X^*) = 680.6301\). The first and the last constraints are active.

3. Results

Series of experiments on the selected benchmark functions have been conducted. The experiment is to evaluate the performance of the proposed approach for the constrained optimization problem by using CS and DE. The program has been written and executed in MATLAB 2008 using a Laptop with i3 processor and 4GB RAM. The results are presented in the following section.

3.1. Optimization parameters

Prior to the optimization run, a number of parameters (solution variables) must be specified. In meta-heuristics optimization algorithms, the common parameter that all algorithms share is the number of population (population size). The rest number of parameters differs from each other depending on the algorithm complexity.

For DE algorithm, the main parameters are \(F_m\) (mutation factor) and \(C_r\) (crossover constant). In addition, population size \((N_p)\) has been included. A preliminary study has been carried out such that \(F_m = 0.5\) and \(C_r = 0.9\) gives the optimum solution, for \(N_p = 50\) [3]. Thus, these DE parameters are also used in this optimization as comparison with CS algorithm.

For CS algorithm, in addition to \(N_p\) (number of nest in this case), only one parameter needs to be set, i.e. discovery rate \((p_a)\). It is suggested that \(N_p = 15\) to \(25\) and \(p_a = 0.15\) to \(0.30\) are sufficient for most optimization problems [13]. In this study \(p_a = 0.25\) will be used and population size \(N_p = 50\) is set in order to have un-bias comparison with that of DE solution.

3.2. Optimization for the benchmark functions

Experiment on the constrained optimization for the selected benchmark functions has been carried out. The number of objective function evaluation (FE) in feasible region is used as stopping criterion of the optimization run as mentioned earlier. Performance comparison between CS and DE is evaluated based on the optimal results of each objective function (the benchmark functions). This means, with the same number of FE, the optimum value of \(f(X)\) for a number of independent optimization runs are recorded.

The statistical measure of the optimum solution for 20 independent runs is presented (mean, best, worst and standard deviation) as showed in Table 1. It can be seen from the quality of results, CS outperforms the DE solution for all cases of dimensionality and constraint. CS can produce better solutions, i.e. faster convergence. In addition, for the case of \(f_1(X)\) where constraints are inactive, CS can give the exact result with 100 times evaluation of the function (FE=100), while DE still could not get the exact solution.
In the next experiment, the number of FE is increased to 1000 in order to further investigate the capability of the DE to reach the optimal solution.

The results as in Table 2 is consistent with the previous experiment (Table 1) with the addition that DE can reach the exact solution for \( f_1(X) \) and reach nearly exact for \( f_2(X) \). For CS, solution of \( f_2(X) \) and \( f_3(X) \) are improved. Furthermore, CS produced a slight robust (slightly better repeatability) solution of \( f_3(X) \) as compared to DE approach.

In addition, this result confirms the advantage of CS over DE as also mentioned in some previous studies such as in [14] and [15]. In the study done in [15], it is mentioned that both CS and DE are more robust optimization algorithms as compared to PSO (particle swarm optimization) and ABC (artificial bee colony) algorithm.

However, it is shown that CS performs slightly better than DE. Another advantage of CS over DE, and perhaps over many other algorithms such as PSO and ABC, is that only one parameter needs to be adjusted, i.e. discovery rate \( (p_d) \).

4. Conclusions

The performance of CS and DE for constrained optimization problems have been evaluated in several investigations. The analysis cases are adopted from the established benchmark functions. The study demonstrate the effectiveness of CS algorithm combined with the specific constraint handling for constrained optimization problem. From the results, it can be concluded that the CS algorithm outperforms DE algorithm in terms of convergence speed to reach the optimum solution with the same number of the function evaluation (FE) in the feasible region, i.e. how many times the objective function is evaluated in the feasible region.

In Addition, another advantage of CS over DE, and perhaps over many other algorithms, is that only one parameter needs to be adjusted, i.e. discovery rate \( (p_d) \). Meanwhile for DE, it needs at least two parameters to set, i.e. \( F_m \) (mutation factor) and \( C_r \) (crossover constant).

| Function | Statistic | Using CS | Using DE |
|----------|-----------|----------|----------|
| \( f_1(X^*) = 0.25 \) | Mean | 0.25 | 0.34 |
| | Best | 0.25 | 0.27 |
| | Worst | 0.25 | 0.42 |
| | Std dev. | 0 | 0.04 |
| \( f_2(X^*) = 1 \) | Mean | 1.124 | 1.347 |
| | Best | 1.008 | 1.012 |
| | Worst | 1.360 | 1.923 |
| | Std dev. | 0.088 | 0.259 |
| \( f_3(X^*) = 680.63 \) | Mean | 744.21 | 932.04 |
| | Best | 700.8 | 852.92 |
| | Worst | 836.47 | 1081.60 |
| | Std dev. | 32.51 | 72.51 |
Table 2. Optimization results for 20 runs (stopping at FE=1000)

| Function  | Statistic | Using CS | Using DE |
|-----------|-----------|----------|----------|
|           | f(X) value | f(X) value |          |
| $f_1(X^*) = 0.25$ | Mean 0.25 | 0.25 |          |
|           | Best 0.25  | 0.25 |          |
|           | Worst 0.25 | 0.25 |          |
|           | Std dev. 0 | 0 |          |
| $f_2(X^*) = 1$ | Mean 1 | 1.007 |          |
|           | Best 1 | 1.001 |          |
|           | Worst 1 | 1.018 |          |
|           | Std dev. 0 | 0.005 |          |
| $f_3(X^*) = 680.63$ | Mean 680.6365 | 697.12 |          |
|           | Best 680.63 | 687.79 |          |
|           | Worst 680.65 | 719.03 |          |
|           | Std dev. 0.0058 | 9.06 |          |

References

[1] Xin-She, Y. and S. Deb. Cuckoo Search via Levy flights. in World Congress on Nature & Biologically Inspired Computing, 2009. NaBIC 2009. . 2009.
[2] Storn, R. and K. Price, Differential Evolution- A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, in Technical Report TR-95-012. 1995.
[3] Iwan, M., et al., Performance Comparison of Differential Evolution and Particle Swarm Optimization in Constrained Optimization. Procedia Engineering, 2012. 41: p. 1323-1328.
[4] Civicioglu, P. and E. Besdok, A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. Artificial Intelligence Review, 2013. 39(4): p. 315-346.
[5] Musigawan, P., S. Chiewchanwattana, and K. Sunat, Improved Differential Evolution via Cuckoo Search Operator, in Neural Information Processing, T. Huang, et al., Editors. 2012, Springer Berlin Heidelberg. p. 465-472.
[6] Cuevas, E. and A. Reyna-Orta, A Cuckoo Search Algorithm for Multimodal Optimization. The Scientific World Journal, 2014. 2014: p. 20.
[7] Jamil, M., Hans-J, and J. Zepernick, Multimodal function optimisation with cuckoo search algorithm. Int. J. Bio-Inspired Comput., 2013. 5(2): p. 73-83.
[8] Engelbrecht, A.P., Introduction to Computational Intelligence, in Computational Intelligence. 2007, John Wiley & Sons, Ltd. p. 1-13.
[9] Coello Coello, C.A., 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, 2002. 191(11–12): p. 1245-1287.
[10] Mezura-Montes, E., Constraint-Handling in Evolutionary Optimization. 2009: Springer Publishing Company, Incorporated. 264.
[11] Lu, H. and W. Chen, Dynamic-objective particle swarm optimization for constrained optimization problems. Journal of Combinatorial Optimization, 2006. 12(4): p. 409-419.
[12] Engelbrecht, A.P., Fundamentals of Computational Swarm Intelligence. 2006: John Wiley \& Sons.
[13] Yang, X.-S. and S. Deb, *Engineering optimisation by cuckoo search*. International Journal of Mathematical Modelling and Numerical Optimisation, 2010. 1(4): p. 330-343.

[14] Adnan, Md Akhtaruzzaman, and M. A. Razzaque. *A comparative study of particle swarm optimization and Cuckoo search techniques through problem-specific distance function*. International Conference of IEEE Information and Communication Technology (ICoICT), 2013.

[15] Civicioglu, Pinar and Besdok, Erkan. *A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms*. Artificial Intelligence Review, 2013. 39(4): p. 315-346.