Efficiency Competition on N-Queen Problem: DE vs. CMA-ES

Shahryar Rahnamayan[1] and Paul Dieras[2]
[1] Department of Mechatronic System Engineering, Simon Fraser University, Vancouver, Canada
[2]Department of Electrical Engineering, National Institute of Applied Sciences (INSA) of Lyon, France
shahryar@pami.uwaterloo.ca, paul.dieras@insa-lyon.fr

***Abstract*— In this paper, two well-known evolutionary algorithms, namely, Differential Evolution (DE) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES), are compared on solving N-queen problem (a classical combinatorial optimization problem). Convergence velocity and robustness are our main measures in the current comparative study. Experiments are conducted on twelve chessboard dimensions. Results confirm that CMA-ES performs better than DE with respect to success rate and also success performance. Details about the N-queen problem, comparison strategies, metrics, and results are provided.**

I. Introduction

Originately, the 8-queen problem has been enounced in the mid 1800s by a chess player, Max Bezzel. Then, famous mathematicians such as Gauss and Cantor worked on this subject. Franz Nauck was the first person who gave a solution and also extended the problem to N queens.

Basically, the problem can be expressed as follow: Placing N queens on N by N cheassboard such that no queen is attacked by another one. A sample solution for 8-queen problem is shown in Figure 1.

The question of solutions' existence and retrieval has been analytically solved for the general case by Ahrens [1]. However, as explained by Sosic in [18], the major drawback of the proposed analytical solution is its limitation to a certain class of solutions. This drawback was the first motivation to try evolutionary algorithms. Although, many algorithms have been designed to solve N-queen problem, such as backtracking algorithms [6], probabilistic search algorithms [18], genetic algorithms [4] and neural networks [20]. Currently the task is not anymore to solve this problem, since it has already been done successfully for very large dimensions. By this way, the N-queen problem can be utilized as a non-trivial benchmarking test-bed to compare optimization algorithms. The exponentially growing cardinality of the search space allows evolutionary algorithm to show their capabilities to explore and escape from many local optimums.

CCECE/CCGEI May 5-7 2008 Niagara Falls. Canada
978-1-4244-1643-1/08/$25.00

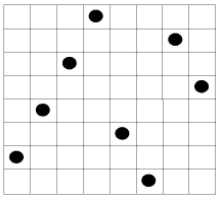

Fig. 1. A sample solution for 8-queen problem, the 8-tuple $(4, 7, 3, 8, 2, 5, 1, 6)$.

II. Differential Evolution (DE)

Differential Evolution is a population-based directed search method [11]. Like other evolutionary algorithms, it starts with an initial population vector, which is randomly generated when no preliminary knowledge about the solution space is available. Each vector of the initial population can be generated as follows [12]:

$$X_{i,j} = a_j + rand_j(0,1) \times (a_j - b_j); j = 1, 2, ..., D, \quad (1)$$

where D is the problem dimension; a_j and b_j are the lower and the upper boundaries of the variable j, respectively. $rand(0,1)$ is the uniformly generated random number in $[0,1]$.

Let us assume that $X_{i,G}(i = 1, 2, ..., N_p)$ are candidate solution vectors in generation G (N_p : population size). Successive populations are generated by adding the weighted difference of two randomly selected vectors to a third randomly selected vector. For classical DE ($DE/rand/1/bin$), the mutation, crossover, and selection operators are straightforwardly defined as follows:

Mutation - For each vector $X_{i,G}$ in generation G a mutant vector $V_{i,G}$ is defined by

$$V_{i,G} = X_{a,G} + F(X_{c,G} - X_{b,G}), \quad (2)$$

where $i = \{1, 2, ..., N_p\}$ and a, b, and c are mutually different random integer indices selected from $\{1, 2, ..., N_p\}$. Further, i, a, b, and c are different so that $N_p \geq 4$ is required. $F \in [0, 2]$ is a real constant which determines the amplification of the added differential variation of $(X_{c,G} - X_{b,G})$. Larger values for F result in

higher diversity in the generated population and lower values cause faster convergence.

Crossover - DE utilizes the crossover operation to generate new solutions by shuffling competing vectors and also to increase the diversity of the population. For the classical DE ($DE/rand/1/bin$), the binary crossover (shown by 'bin' in the notation) is utilized. It defines the following trial vector:

$$U_{i,G} = (U_{1i,G}, U_{2i,G}, ..., U_{Di,G}), \tag{3}$$

$$U_{ji,G} = \begin{cases} V_{ji,G} & \text{if } rand_j(0,1) \leq C_r \vee j = k, \\ X_{ji,G} & \text{otherwise.} \end{cases} \tag{4}$$

$C_r \in (0,1)$ is the predefined crossover rate, and $rand_j(0,1)$ is the j^{th} evaluation of a uniform random number generator. $k \in \{1, 2, ..., D\}$ is a random parameter index, chosen once for each i to make sure that at least one parameter is always selected from the mutated vector, $V_{ji,G}$. Most popular values for C_r are in the range of $(0.4, 1)$ [5].

Selection - This is an approach which must decide which vector ($U_{i,G}$ or $X_{i,G}$) should be a member of next (new) generation, $G + 1$. For a minimization problem, the vector with the lower value of objective function is chosen (greedy selection).

This evolutionary cycle (i.e., mutation, crossover, and selection) is repeated N_p (population size) times to generate a new population. These successive generations are produced until meeting the predefined termination criteria. Algorithm 1 presents the details of the classical DE algorithm. Starting point for the mutation, crossover and selection is indicated by the comments in the algorithm.

Algorithm 1 Differential Evolution (DE). P_0: Initial population, N_p: Population size, V: Noise vector, U: Trial vector, D: Problem dimension, BFV: Best fitness value so far, VTR: Value-to-reach, NFC: Number of function calls, MAX_{NFC}: Maximum number of function calls, F: Mutation constant, $rand(0,1)$: Uniformly generated random number, C_r: Crossover rate, $f(\cdot)$: Objective function, P': Population of the next generation.

```
1: Generate uniformly distributed random population P_0
2: while ( BFV > VTR and NFC < MAX_NFC ) do
3:     {Generate-and-Test-Loop}
4:     for i = 0 to N_p do
5:         Select three parents X_a, X_b, and X_c randomly from current
           population where i ≠ a ≠ b ≠ c
           {Mutation}
6:         V_i ← X_a + F × (X_c − X_b)
           {Crossover}
7:         for j = 0 to D do
8:             if rand(0,1) < C_r then
9:                 U_{i,j} ← V_{i,j}
10:            else
11:                U_{i,j} ← X_{i,j}
12:            end if
13:        end for
           {Selection}
14:        Evaluate U_i
15:        if (f(U_i) ≤ f(X_i)) then
16:            X'_i ← U_i
17:        else
18:            X'_i ← X_i
19:        end if
20:    end for
21:    X ← X'
22: end while
```

III. Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

CMA-ES, introduced in 1996 [8], has demonstrated to be a reliable and robust optimizer for both local and global optimization. It is an Evolutionary Algorithm (EA) with selection and recombination processes which supports self-adaptation. However, its evolutionary mechanism is completely derandomized and is based on a probability density which tries to be as close as possible to the inverse Hessien matrix of the objective function.

As explained in [9], the general steps of CMA-ES are as follows:

1) Parents are chosen from the initial population.
2) The probability distribution is estimated according to the parents.
3) The probability distribution is sampled leading to the evaluation of offsprings.
4) The new population is chosen according to the selection process.

In step 2, the employed probability distribution has a normal distribution. The mean is equal to a weighted sum of the parent vectors in which the best members, in term of fitness value, are assigned higher weights. In other words, the choice of the mean should be interpreted as the recombination step. The standard deviation is defined as the squared step size by the covariance matrix. Formally, the offsprings are defined as follows:

$$x_k^{(g+1)} \sim N\left(m^{(g)}, \left(\sigma^{(g)}\right)^2 C^{(g)}\right), k = 1, 2, ..., \lambda \tag{5}$$

$$m^{(g)} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{(g)} \tag{6}$$

where g: generation index, k: offspring index, x: n-dimensional offspring, $N(\cdot)$: normal distribution, m: the mean vector computed in Eq. 6, σ: the standard deviation, step size, of generation g, C: covariance matrix, $\mu \leq \lambda$: number of selected parents, w_i: i^{th} weight of the i^{th} best parent.

More details about how the covariance matrix and the step size are adapted have been given in [7].

CMA-ES has been widely applied in different domains such as filter design [2], computer vision [3] and

turbomachinery [17].

IV. Comparative Experiments

A. Fitness Function

Simply, the pairs of collisions are counted as a fitness value. Reaching to the zero value for that means finding the solution (mutually non-attack queens). By participating some constraints, we can simplify the N-Queen problem, but the aim is keeping it more challenging for both optimizers.

B. Handling Integer Variables

DE and CMA-ES work directly with floating-point variables, but by applying some minor modifications it is able to handle integer variables as well. One of the common ways is simply using the integer variables during evaluation of the objective function and leaving other parts unchanged [10]. Following rules present this approach:

$$f(y_j);\ j = 1, ..., D$$

$$y_j = \begin{cases} x_j & x_j \text{ is a continuous variable,} \\ \text{INT}(x_j) & x_j \text{ is an integer variable.} \end{cases} \quad (7)$$

or

$$y_j = \begin{cases} x_j & x_j \text{ is a continuous variable,} \\ \text{ROUND}(x_j) & x_j \text{ is an integer variable.} \end{cases} \quad (8)$$

Converting a continuous number to the integer can be performed by INT$(\cdot)$ or ROUND$(\cdot)$ functions. INT$(\cdot)$ truncates the real part of the continuous number and ROUND$(\cdot)$ returns the closest integer number. If the INT$(\cdot)$ is the conversion case, Eq. 1 should be replaced with the following equation because of truncation effect of INT$(\cdot)$ function:

$$X_{i,j} = a_j + rand_j(0,1) \times (a_j - b_j + 1), j = 1, 2, ..., D. \quad (9)$$

In this study, ROUND$(\cdot)$ function has been utilized. Through this method, DE and CMA-ES internally search a superset of the solution space instead, because the integer numbers are the subset of the continuous numbers. This feature increases the population diversity and the robustness of the algorithm as well.

C. Comparison Strategies and Metrics

In this study, three metrics, namely, *number of function calls* (NFC), *success rate* (SR), and *success performance* (SP) have been utilized to compare the algorithms. We compare the convergence speed by measuring the number of function calls which is the most commonly used metric in literature [13]–[16]. A smaller NFC means higher convergence speed. The termination criterion is to find a value smaller than the value-to-reach (VTR) before reaching the maximum number of function calls MAX_{NFC}. In order to minimize the effect of the stochastic nature of the algorithms on the metric, the reported number of function calls for each function is the average over 20 trials.

The number of times, for which the algorithm succeeds to reach the VTR (zero in the current study) for each test function is measured as the success rate SR:

$$\text{SR} = \frac{\text{number of times reached VTR}}{\text{total number of trials}}. \quad (10)$$

The average success rate (SR_{ave}) over n test functions are calculated as follows:

$$\text{SR}_{ave} = \frac{1}{n} \sum_{i=1}^{n} \text{SR}_i. \quad (11)$$

Both of NFC and SR are important measures in an optimization process. So, two individual objectives should be considered simultaneously to compare competitors. In order to combine these two metrics, a new measure, called success performance (SP), has been introduced as follows:

$$\text{SP} = \frac{\text{mean (NFC for successful runs)}}{\text{SR}}. \quad (12)$$

SP is our main measure to judge which algorithm performs better than others.

D. Setting Control Parameters

Parameter settings for all conducted experiments are as follows:

- Common settings for DE and CMA-ES
  - Population size, $N_p = 5 \times N$
  - Maximum number of function calls, $\text{MAX}_{\text{NFC}} = 2 \times 10^6$
  - Value to reach, VTR$= 0$ (mutually non-attack queens)
- DE settings [19]
  - Differential amplification factor, $F = 0.5$
  - Crossover probability constant, $C_r = 0.9$
  - Mutation strategy: $DE/rand/1/bin$ (classic version of DE)
- CMA-ES settings [7], [8]
  - Initial vector, x_0 is chosen randomly
  - Step size, $\sigma_0 = (N-1)/3$

E. Results

The results of applying CMA-ES and DE to solve N-Queen problems (N=4 to N=15) are given in Table I. According to the reported success performances (SP), CMA-ES outperforms DE on 8 problems (six of them

TABLE I

COMPARISON OF DE AND CMA-ES. N×N: CHESS BOARD DIMENSIONS, NFC: NUMBER OF FUNCTION CALLS (AVERAGE OVER 20 TRIALS), SR: SUCCESS RATE, SP: SUCCESS PERFORMANCE.

	CMA-ES			DE		
N	*NFC*	*SR*	*SP*	*NFC*	*SR*	*SP*
4	456	1.00	456	**134**	1.00	**134**
5	656	1.00	656	**254**	1.00	**254**
6	**22013**	**1.00**	**22013**	111136	0.65	170979
7	**9964**	**1.00**	**9964**	24338	0.95	25619
8	84962	**1.00**	84962	**7576**	0.75	**10101**
9	133628	**1.00**	133628	**19296**	0.50	**38592**
10	**263572**	**0.95**	**277444**	286208	0.30	954028
11	284382	**0.95**	**299349**	**68255**	0.10	682550
12	295740	**0.75**	**394330**	**99120**	0.25	396480
13	376631	**0.85**	**443100**	**95485**	0.15	636570
14	450654	**0.85**	**530181**	**160475**	0.10	1604750
15	627391	**0.50**	**1254782**	**223425**	0.10	2234250
Ave.		**0.90**			0.49	
$\sum$			**3, 450, 864**			6, 754, 306

are on $N \geq 10$). DE performs better than CMA-ES on 4 problems (for N=$4, 5, 8, 9$). DE solves 9 of problems faster than CMA-ES (smaller NFC), but as seen, it suffers from a lower success rate. The average success rate to solve 12 problems for CMA-ES is 0.90, while this value is 0.49 for DE. So, DE shows lower robustness than CMA-ES. In fact, capability of escaping from the local minima for CMA-ES is higher than DE. For higher dimensions ($N \geq 10$), DE's success rate is reduced sharply. The last row of the table presents the total success performance ($\sum_{i=1}^{12}$ SP) for both algorithms. According to these numbers, CMA-ES performs 1.96 times better than DE ($3,450,864$ vs. $6,754,306$).

V. CONCLUSION

In this comparative study, DE and CMA-ES have been employed to solve 12 non-separable combinatorial problems with many local minima. The results confirm that CMA-ES performs better than DE with respect to success performance. CMA-ES shows higher success rate than DE in particular on high dimension problems. It is interesting to be mentioned, DE and CMA-ES have already been compared in [7]. They concluded that CMA-ES can be outperformed by DE on separable functions, whereas better performance is shown in non-separable problem by CMA-ES. Interestingly, the same conclusion is confirmed by the conducted experiments in this paper.

DE and CMA-ES are compared over finding a first solution, however, competing on obtaining more unique or distinct solutions can be considered as new direction for future study.

REFERENCES

[1] W. Ahrens. *Mathematische Unterhaltungen und Spiele*. B.G. Teubner, 1910.

[2] S. Baskar, P. Suganthan, N. Ngo, A. Alphones, and R. Zheng. Design of triangular fbg filter for sensor applications using covariance matrix adapted evolution algorithm. *Optics Communications*, 260(2):716–722, 2006.

[3] T. Bergener, C. Bruckhoff, and C. Igel. Parameter optimization for visual obstacle detection using a derandomized evolution strategy. *Imaging and Vision Systems: Theory, Assessment and Applications*, 9:265–279, 2001.

[4] K. Crawford. Solving the n-queens problem using genetic algorithms. In *Proceedings ACM/SIGAPP Syposium on Applied Computing, Kansas City*, pages 1039–1047, 1992.

[5] S. Das, A. Konar, and U. Chakraborty. Improved differential evolution algorithms for handling noisy optimization problems. In *Proceedings of IEEE Congress on Evolutionary Computation Conference*, pages 1691–1698, Napier University, Edinburgh, UK, September 2005.

[6] S. W. Golomb and L. Baumert. Backtrack programming. *Journal of the ACM*, 12:516–524, 1965.

[7] N. Hansen and S. Kern. Evaluating the cma evolution strategy on multimodal test functions. In *Eighth International Conference on Parallel Problem Solving from Nature PPSN VIII*, pages 282–291, 2004.

[8] N. Hansen and Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *IEEE International Conference on Evolutionary Computation*, pages 312–317, 1996.

[9] S. Kern, S. Müller, N. Hansen, D. Büche, J. Ocenasek, and P. Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms - a comparative review. *Natural Computing*, 3(1):77–112, 2004.

[10] G. C. Onwubolu and B. Babu. *New Optimization Techniques in Engineering*. Springer, Berlin, New York, 2004.

[11] K. Price. *An Introduction to Differential Evolution*. McGraw-Hill, London (UK), 1999. ISBN: 007-709506-5.

[12] K. Price, R. Storn, and J. Lampinen. *Differential Evolution : A Practical Approach to Global Optimization*. Springer-Verlag, Berlin/Heidelberg/ Germany, 1st edition edition, 2005. ISBN: 3540209506.

[13] S. Rahnamayan, H. Tizhoosh, and M. Salama. Opposition-based differential evolution algorithms. In *Proceedings of the 2006 IEEE World Congress on Computational Intelligence (CEC-2006)*, pages 2010–2017, Vancouver, BC, Canada, July 2006.

[14] S. Rahnamayan, H. Tizhoosh, and M. Salama. Opposition-based differential evolution with variable jumping rate. In *IEEE Symposium on Foundations of Computational Intelligence*, pages 81–88, Honolulu, Hawaii, USA, April 2007.

[15] S. Rahnamayan, H. Tizhoosh, and M. Salama. Opposition-based differential evolution. *Journal of IEEE Transactions on Evolutionary Computation*, 12(1):64–79, Feb. 2008.

[16] S. Rahnamayan, H. Tizhoosh, and M. Salama. Opposition versus randomness in soft computing techniques. *Elsevier Journal on Applied Soft Computing*, 8:906–918, March 2008.

[17] T. Sonoda, Y. Yamaguchi, T. Arima, M. Olhofer, B. Sendhoff, and H. Schreiber. Advanced high turning compressor airfoils for low reynolds number condition. *Part I: Design and Optimization. Journal of Turbomachinery*, 126:350–359, 2004.

[18] R. Sosič and J. Gu. Fast search algorithms for the n-queens problem. *IEEE TRansactions on Systems, Man and Cybernetics*, 21:1572–1576, 1991.

[19] R. Storn and K. Price. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[20] T. Tambouratzis. A simulated annealing artificial neural network implementation of the N-queens problem. *International Journal of Intelligent Systems*, 12:739–752, 1997.