2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Can Deterministic Chaos Improve Differential Evolution for the Linear Ordering Problem?

Pavel Krömer, Ivan Zelinka, and Václav Snášel

***Abstract*— Linear ordering problem is a popular NP-hard combinatorial optimization problem attractive for its complexity, rich library of test data, and variety of real world applications. It has been solved by a number of heuristic as well as metaheuristic methods in the past. The implementation of nature-inspired metaheuristic optimization and search methods usually depends on streams of integer and floating point numbers generated in course of their execution. The pseudo-random numbers are utilized for an in-silico emulation of probability-driven natural processes such as arbitrary modification of genetic information (mutation, crossover), partner selection, and survival of the fittest (selection, migration) and environmental effects (small random changes in particle motion direction and velocity). Deterministic chaos is a well known mathematical concept that can be used to generate sequences of seemingly random real numbers within selected interval in a predictable and well controllable way. In the past, it has been used as a basis for various pseudo-random number generators with interesting properties. Recently, it has been shown that it can be successfully used as a source of stochasticity for nature-inspired algorithms solving a continuous optimization problem. In this work we compare effectiveness of the differential evolution with different pseudo-random number generators and chaotic systems as sources of stochasticity when solving the linear ordering problem.**

I. Introduction

Nature-inspired optimization and search methods including the differential evolution [1] (DE) operate in a stochastic (probability-driven) manner. It corresponds to the way real natural systems that have originally inspired evolutionary and swarm-like methods work. An example of highly complex stochastic biological system represent e.g. ant colonies and the stochastic way tasks are distributed among the ants within a colony [2].

Computers, however, are deterministic machines and non-determinism has to be emulated in order to mimic the probability-driven decision making observed in the nature. That is usually achieved by the means of pseudo-random number generators, less frequently by deterministic chaos, and only seldom by the means of true (hardware based) random number generators.

Recently, it has been shown that deterministic chaos can be successfully used as a source of stochasticity for nature-inspired algorithms solving a battery of continuous test problem [3]. This study compares the effect the use of different sources of stochasticity has on the DE solving a well-known real-world combinatorial optimization problem – the linear ordering problem (LOP). The sources of non-determinism investigated in this work include different pseudo-random number generators (PRNGs) and generators based on deterministic chaos DCh) [4].

The LOP is a well-known NP-hard combinatorial optimization problem. It has been intensively studied and there are plenty of exact, heuristic and metaheuristic algorithms for the LOP. With its large collection of well described testing data sets, the LOP represents an interesting testbed for metaheuristic and soft computing algorithms for combinatorial optimization [5], [6].

The LOP can be formulated as a graph problem and as a matrix triangulation problem [5]. It can be seen as a search for simultaneous permutation of rows and columns of a matrix C such that the sum of the elements in the upper diagonal of the modified matrix is as large as possible. In this work we use the matrix formulation of the LOP.

II. LOP as a Matrix Triangulation Problem

The LOP can be defined as a search for optimal column and row reordering of a weight matrix C [7], [8], [9], [5]. Consider a matrix $C^{n\times n}$, permutation Π and a cost function f:

$$f(\Pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\Pi(i)\Pi(j)} \tag{1}$$

Solving the LOP involves a search for permutation Π so that $f(\Pi)$ is maximized, i.e. the permutation restructures the matrix C so that the sum of its elements above main diagonal is maximized. The LOP is an NP-hard problem with a number of applications in scheduling (scheduling with constraints), graph theory, economy, sociology (paired comparison ranking), tournaments and archaeology among others.

In economics, LOP algorithms are deployed to triangularize input-output matrices. The resulting permutation provides an useful information about the stability of the

Pavel Krömer is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton AB T6G 2V4, Canada (email: pavel.kromer@ualberta.ca) and with the VŠB-Technical University of Ostrava, Ostrava, Czech Republic (email: pavel.kromer@vsb.cz). Ivan Zelinka, and Václav Snášel are with the Department of Computer Science and IT4Innovations, VŠB-Technical University of Ostrava, 17. listopadu 2172/15, 708 00 Ostrava-Poruba, Czech Republic (email: {ivan.zelinka,vaclav.snasel}@vsb.cz)

This work was supported by the Natural Science and Engineering Research Council (NSERC) of Canada, the Helmholtz-Alberta Initiative, and by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), by the Development of human resources in research and development of latest soft computing methods and their application in practice project, reg. no. CZ.1.07/2.3.00/20.0072 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic, by the Grant Agency of the Czech Republic - GACR P103/13/08195S, and by SGS, VSB-Technical University of Ostrava, under the grants SP2014/110 and SP2014/159.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

investigated economy. In archaeology, LOP algorithms are used to process the Harris Matrix, a matrix describing most probable chronological ordering of samples found in different archaeological sites [8]. Other applications of the LOP include the equivalent graph problem, related graph problem, aggregation of individual preferences, ranking in sport tournaments and e.g. the minimization of crossing [5].

A. LOP Data sets

There are several test libraries used for benchmarking LOP algorithms. They are well preprocessed, thoroughly described and the optimal (or so far best) solutions are available. Majority of investigated algorithms were tested against the LOLIB library. The original LOLIB library contains 49 instances of input-output matrices describing European economies in the 70s. Although LOLIB features real world data, it is considered rather simple and easy to solve [10].

Mitchell and Bochers [11] published an artificial LOP data library and LOP instance generator to evaluate their algorithm. The data (from now on addressed as MBLB) and code are available at Rensselaer Polytechnic Institute[1].

Schiavinotto and Stützle [7], [8] showed that the LOLIB and MBLB instances are significantly different, having diverse high-level characteristics of the matrix entries such as sparsity or skewness. The search space analysis revealed that MBLB instances typically have higher correlation length and also a generally larger fitness-distance correlation than LOLIB instances. It suggests that MBLB instances should be easier to solve than LOLIB instances of the same dimension. Moreover, a new set of large artificial LOP instances (based on LOLIB) called XLOLIB was created and published. Another set of LOP instances is known as the Stanford Graph Base (SGB). The SGB is composed of larger input-output matrices describing US economies. In this study we use the well-known LOLIB and SGB LOP instances.

B. LOP Algorithms

There are several exact and heuristic algorithms for the linear ordering problem. The exact algorithms are strongly limited by the fact that LOP is an NP-hard problem (i.e. there are no exact algorithms that could solve LOP in polynomial time). Among the exact algorithms, branch & bound approach based on LP-relaxation of the LOP for the lower bound, a branch & cut algorithm, and interior point/cutting plane algorithm attracted attention [8]. Exact algorithms are able to solve rather small general instances of the LOP and bigger instances (with the dimension of few hundred rows and columns) of certain classes of LOP [8].

A number of heuristics and soft computing algorithms was used for solving LOP instances: greedy algorithms, local search algorithms, elite tabu search, scattered search and iterated local search [8], [12], [5].The metaheuristic algorithms used to solve LOP in the past include genetic algorithms [13], [14], differential evolution [15], and ant colony based algorithms [16]. The investigation of metaheuristics and soft computing algorithms is motivated by previous success of such methods in real world and industrial applications [17], [18], [19], [20]. This study investigates whether the use of deterministic chaos as a source of stochasticity can contribute to more efficient (faster, more accurate, more energy efficient) solving of LOP instances. Differential evolution was selected as a metaheuristic LOP solver based on its previous good results on LOP [21] as well as on experiments with deterministic chaos [3].

[1] http://www.rpi.edu/~mitchj/generators/linord/

III. Differential Evolution

The DE is a versatile and easy to use stochastic evolutionary optimization algorithm [1]. It is a population-based optimizer that evolves a population of real-encoded vectors representing the solutions to given problem. The DE was introduced by Storn and Price in 1995 [22], [23] and it quickly became a popular alternative to the more traditional types of evolutionary algorithms. It evolves a population of candidate solutions by the iterative modification of the candidate solutions via the application of differential mutation and crossover [1].

The DE starts with an initial population of N real-valued vectors. The vectors are initialized with real values either randomly or so, that they are evenly spread over the problem space. The latter initialization leads to better results of the optimization [1].

During the optimization, the DE generates new vectors that are scaled perturbations of existing population vectors. The algorithm perturbs selected base vectors with the scaled difference of two (or more) other population vectors in order to produce the trial vectors. The trial vectors compete with members of the current population with the same index called the target vectors. If a trial vector represents better solution than the corresponding target vector, it takes its place in the population [1].

There are two most significant parameters of the DE [1]. The scaling factor $F \in [0, \infty]$ controls the rate at which the population evolves and the crossover probability $C \in [0, 1]$ determines the ratio of bits that are transferred to the trial vector from its opponent. The size of the population and the choice of operators are another important parameters of the optimization process.

The basic operations of the classic DE can be summarized using the following formulas [1]: the random initialization of the ith vector with N parameters is defined by

$$x_i[j] = rand(b_j^L, b_j^U), \quad j \in \{0, \ldots, N-1\} \tag{2}$$

where b_j^L is the lower bound of jth parameter, b_j^U is the upper bound of jth parameter and $rand(a, b)$ is a function generating a random number from the range $[a, b]$. A simple form of the differential mutation is given by

$$v_i^t = v_{r1} + F(v_{r2} - v_{r3}) \tag{3}$$

where F is the scaling factor and v_{r1}, v_{r2} and v_{r3} are three random vectors from the population. The vector v_{r1} is the

base vector, v_{r2} and v_{r3} are the difference vectors, and the ith vector in the population is the target vector. It is required that $i \neq r1 \neq r2 \neq r3$. The uniform crossover that combines the target vector with the trial vector is given by

$$l = rand(0, N-1) \tag{4}$$

$$v_i^t[m] = \begin{cases} v_i^t[m] & \text{if } (rand(0,1) < C) \text{ or } m = l \\ x_i[m] \end{cases} \tag{5}$$

for each $m \in \{1, \ldots, N\}$. The uniform crossover replaces with probability $1-C$ the parameters in v_i^t by the parameters from the target vector x_i.

There are also many other modifications to the classic DE. Mostly, they differ in the implementation of particular DE steps such as the initialization strategy, the vector selection, the type of differential mutation, the recombination operator, and control parameter selection and usage [1].

IV. Related Work

Stochasticity is an important property of non-deterministic natural systems that have inspired evolutionary computation and swarm-intelligent algorithms including the DE. The effects of various software and hardware based sources of non-determinism on the performance and results of nature-inspired algorithms were in the past investigated in several studies.

A. Random Number Generators and DE

Cárdenas-Montes et al. [24] studied the sensitiveness of four evolutionary algorithms including the GA and DE to the change of PRNG (Mersenne Twister and GCC rand). The study featured a massive experimental evaluation on a set of test functions and statistical analysis of the results and the authors concluded that both GA and DE *are* sensitive to the choice of RNG. No guidelines for the choice of PRNG for particular algorithm and problem were found though.

Another study of the influence of PRNG quality on the DE is due to Tirronen et al. [25]. The authors have used a battery of test functions and concluded that the quality of PRNG does not correlate with the performance of the algorithm (i.e. a lower-quality PRNG can yield a DE with good performance).

B. Deterministic Chaos and DE

The use of deterministic chaos for selected tasks of GA, DE, and PSO has been investigated by a number of research works.

A study by Yang et al. [26] proposed a self-adapting DE with chaos. Sequence of chaotic numbers generated by logistic map was used as a part of adaptive mechanism for self-tuning of DE parameters F and C. Computational experiments performed with selected test functions have shown that the adaptive algorithm finds better solutions than non-adaptive DE.

C. Deterministic Chaos and PRNG

Deterministic chaos has been exploited for general pseudo-random number generation in several studies. Wagner [27] proposed in 1993 the use of coupled map lattice for pseudo-random number generation. The Logistic Lattice algorithm featured good statistical properties and a long period of generated numbers. Moreover, the algorithm addressed the problems caused by finite precision of computer arithmetics by re-mapping the logistic equation from interval [0, 1] to interval [-1, 1]. A design of PRNG based on logistic equation is due to Andrecut [28]. The generator was shown to outperform linear congruential generator in terms of quality of generated numbers.

The work of Szczepański and Kotulski [29] proposed physical model of a chaotic PRNG with good statistical properties. Shastry et al. [30] proposed a generalization of Logistic Equation and a new PRNG. The algorithm sampled its parameters from several chaotic trajectories.

Chen et al. [31] used a nonlinear digitalized modified logistic map as a basis of novel PRNG. The algorithm was shown to pass selected randomness tests. A recent analysis of PRNGs based on Logistic Equation is due to Persohn and Povinelli [32]. The authors have illustrated some of the problems caused by finite precision of computer arithmetics and proposed a new algorithm to test the properties of chaos based generators.

The studies outlined in this section often used sophisticated approaches to design a chaos-based PRNGs that would satisfy the requirements usually imposed on traditional PRNGs that are necessary e.g. for cryptographical applications. In this work, however, we investigate the ability of PRNGs and DCh-based generators to drive nature-inspired stochastic methods. Because of that, also simple DCh-based methods for generation of sequences of real numbers are considered.

V. Experiments

In order to evaluate the effect different sources of stochasticity have on the DE when solving LOP, a modified DE with adaptive mutation rate was used to find solutions of LOLIB and SGB instances. The LOLIB data set contained 49 LOP matrices with dimensions ranging from 50 to 56 and one instance of the dimension 79 (describing the US economy). The SGB data set contained 25 LOP matrices of the dimension 75.

The DE as well as a battery of RNGs and DCh-based generators was implemented in C++. The overview of implemented PRNGs and DCh-based generators is shown in table I. All generators were implemented from scratch with the exception of GCC rand, which was provided by the GNU C Library (libc). The DE was executed without any type of local search as a pure metaheuristic search. The algorithm was executed for 5000 generations with population size 100, i.e. the search involved 50000 fitness function evaluations. LOP solutions were represented as real number vectors using the convenient random keys encoding [33], [34]. The DE was

TABLE I
IMPLEMENTED SOURCES OF STOCHASTICITY.

Generator	Type	Note
GCC rand	PRNG	rand() function from GCC 4.7.2; linear congruential generator.
Mersenne Twister	PRNG	Mersenne Twister PRNG according to [35].
Ranlux D1	PRNG	Luxury random numbers generator by Luscher [36]. Adapted from GNU Scientific library, luxury level 1.
Ranlux D2	PRNG	Luxury random numbers, luxury level 2.
Logistic Equation	DCh	Logistic Equation (Map) $x_{n+1} = rx_n(1-x_n)$, $r = 4$, $x_0 = 0.02$ [4].
Re-mapped Logistic Equation	DCh	Logistic Equation re-mapped to [-1, 1] according to [27].
Logistic Lattice	DCh	Full implementation of Logistic Lattice according to [27].
B-exponential Map	DCh	Generalized Logistic Equation according to [30].

for each generator and each LOP instance executed 30 times independently and this work presents average results in order to evaluate the effect of different sources of stochasticity. The average accuracy and standard deviation of LOLIB solutions found with the help of different generators is summarized in table II and the average accuracy and standard deviation of SGB solutions found using different generators is shown in table III. Averages over all LOP instances in both data set are shown at the bottom of each table.

It can be observed that the DE with all generators achieved good average accuracy (for a pure metaheuristic solver): 95.22% - 99.93% for LOLIB instances and 97.65% - 99.27% for SGB instances. The average precision of solutions found with the help of PRNGs is better than those of LOP solutions found with DCh. Moreover, the standard deviation of PRNG-based solutinons is a little lower. Student's paired t-test [37] at significance level $\alpha = 0.05$ showed that the difference between PRNG-based and DCh-based solutions is statistically significant. The only exception was the generator based on logistic lattice which performed on par with the PRNGs.

VI. CONCLUSIONS

This study investigated the effects that the use of different sources of stochasticity might have on the performance of a nature-inspired algorithm (namely differential evolution) when solving a real-world combinatorial optimization problem (namely the linear ordering problem). Two sets of well-known LOP instances from the LOLIB and SGB data sets were solved by the DE utilizing several pseudo-random number generators and generators based on deterministic chaos. The results suggest that the use of deterministic chaos (or at least the generators used in this work) brings no improvement to LOLIB and SGB solutions. On the contrary, the DE with generators based on logistic equation, re-mapped logistic, and b-exponential map yielded significantly worse solutions than the DE utilizing PRNGs. Only the DCh-based generator with logistic lattice performed similarly as the PRNGs.

Our future work on this topic will include a large-scale evaluation of other nature-inspired algorithms utilizing different sources of stochasticity for different well-known LOP instances, other combinatorial optimization problems and other real-world problems.

REFERENCES

[1] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution A Practical Approach to Global Optimization*, ser. Natural Computing Series, G. Rozenberg, T. Bäck, A. E. Eiben, J. N. Kok, and H. P. Spaink, Eds. Berlin, Germany: Springer-Verlag, 2005. [Online]. Available: http://www.springer.com/west/home/computer/foundations?SGWID=4-156-22-32104365-0\&\#38;teaserId=68063\&\#38;CENTER_ID=69103

[2] D. M. Gordon, *Ant Encounters: Interaction Networks and Colony Behavior*, ser. Primers in complex systems. Princeton University Press, 2010. [Online]. Available: http://books.google.com/books?id=MabwdXLZ9YMC

[3] P. Krömer, V. Snášael, and I. Zelinka, "Randomness and chaos in genetic algorithms and differential evolution," in *Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on*, 2013, pp. 196–201.

[4] H. Schuster and W. Just, *Deterministic Chaos*. Wiley, 2006. [Online]. Available: http://books.google.cz/books?id=9y2qSGpQR7QC

[5] R. Martí and G. Reinelt, *The Linear Ordering Problem - Exact and Heuristic Methods in Combinatorial Optimization*, ser. Applied Mathematical Sciences. Springer Heidelberg Dordrecht London New York, 2011, vol. 175.

[6] R. Martí, G. Reinelt, and A. Duarte, "A benchmark library and a comparison of heuristic methods for the linear ordering problem," *Computational Optimization and Applications*, pp. 1–21, 2011, 10.1007/s10589-010-9384-9. [Online]. Available: http://dx.doi.org/10.1007/s10589-010-9384-9

[7] T. Schiavinotto and T. Stützle, "Search space analysis for the linear ordering problem," in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science, G. R. Raidl, J.-A. Meyer, M. Middendorf, S. Cagnoni, J. J. R. Cardalda, D. Corne, J. Gottlieb, A. Guillot, E. Hart, C. G. Johnson, and E. Marchiori, Eds., vol. 2611. Berlin, Germany: Springer-Verlag, 2003, pp. 322–333.

[8] ——, "The linear ordering problem: Instances, search space analysis and algorithms," *Journal of Mathematical Modelling and Algorithms*, vol. 3, no. 4, pp. 367–402, 2004.

[9] V. Campos, F. Glover, M. Laguna, and R. Martí, "An experimental evaluation of a scatter search for the linear ordering problem," *J. of Global Optimization*, vol. 21, no. 4, pp. 397–414, 2001.

[10] G. Reinelt, *The Linear Ordering Problem : Algorithms and Applications*, ser. Research and Exposition in Mathematics. Heldermann Verlag Berlin, 1985, vol. 8.

[11] J. E. Mitchell and B. Borchers, "Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm," Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180–3590, Tech. Rep., September 1997, accepted for publication in *Proceedings of HPOPT97*, Rotterdam, The Netherlands. [Online]. Available: http://www.math.rpi.edu/~mitchj/papers/combined.ps

[12] G. Huang and A. Lim, "Designing a hybrid genetic algorithm for the linear ordering problem," in *GECCO*, 2003, pp. 1053–1064.

[13] P. Krömer, V. Snášel, and J. Platoš, "Evolving feasible linear ordering problem solutions," in *CSTST '08: Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology*. New York, NY, USA: ACM, 2008, pp. 337–342.

[14] P. Krömer, V. Snášel, J. Platoš, and D. Husek, "Genetic Algorithms for the Linear Ordering Problem," *Neural Network World*, vol. 19, no. 1, pp. 65–80, 2009, ISSN 1210-0552, impact factor 0.395.

[15] V. Snášel, P. Krömer, and J. Platoš, "Differential evolution and genetic algorithms for the linear ordering problem." in *KES (1)*, ser. Lecture Notes in Computer Science, J. D. Velásquez, S. A. Ríos, R. J. Howlett, and L. C. Jain, Eds., vol. 5711. Springer, 2009, pp. 139–146.

[16] C. Chira, C. M. Pintea, G. C. Crisan, and D. Dumitrescu, "Solving the linear ordering problem using ant models," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ser. GECCO '09. New York, NY, USA: ACM, 2009, pp. 1803–1804. [Online]. Available: http://doi.acm.org/10.1145/1569901.1570170

TABLE II

AVERAGE PRECISION AND STD. DEVIATION OF LOLIB SOLUTIONS FOUND BY DE WITH DIFFERENT SOURCES OF STOCHASTICITY (IN PERCENT).

LOP Instance	GCC rand	Mersenne Twister	Ranlux D1	Ranlux D2	Logistic Equation	Re-mapped Log. Eq.	Logistic Lattice	B-exponential Map
N-be75eec	98.61 (1.03)	98.69 (0.84)	98.71 (0.64)	98.67 (0.32)	98.10 (0.97)	97.97 (1.08)	98.57 (0.71)	98.31 (0.84)
N-stabu74	98.76 (0.56)	98.68 (0.70)	98.65 (0.60)	98.73 (0.38)	97.80 (0.84)	97.70 (0.84)	98.50 (0.74)	97.68 (0.95)
N-t59i11xx	99.25 (0.61)	99.18 (0.64)	99.49 (0.39)	99.29 (0.59)	97.43 (1.43)	97.50 (1.24)	99.19 (0.88)	97.30 (1.44)
N-t65i11xx	99.02 (1.01)	99.00 (0.83)	99.51 (0.25)	99.00 (0.87)	96.91 (1.72)	97.15 (1.61)	99.19 (0.56)	96.43 (1.74)
N-t70b11xx	98.64 (1.03)	98.72 (0.89)	99.20 (0.73)	98.70 (1.28)	96.57 (1.74)	96.44 (1.78)	98.86 (0.89)	96.91 (1.85)
N-t70k11xx	98.89 (0.79)	98.96 (0.51)	99.01 (0.61)	98.84 (0.82)	96.90 (2.02)	96.72 (1.85)	98.86 (0.63)	97.19 (2.13)
N-t70x11xx	98.49 (1.06)	98.73 (1.12)	98.31 (1.33)	98.50 (0.96)	96.47 (1.70)	96.32 (1.43)	98.59 (0.85)	96.34 (2.20)
N-t75k11xx	99.31 (0.47)	99.11 (0.68)	99.16 (0.65)	98.99 (0.72)	96.83 (2.40)	96.82 (2.13)	99.29 (0.59)	97.18 (2.29)
N-tiw56n62	98.13 (1.34)	98.09 (1.41)	98.86 (0.67)	97.88 (1.30)	96.89 (1.67)	96.84 (1.57)	98.16 (1.52)	96.94 (1.52)
N-tiw56r58	98.27 (1.32)	98.50 (1.00)	98.44 (0.78)	98.90 (0.57)	96.98 (1.41)	97.13 (1.43)	98.28 (0.92)	96.95 (1.40)
N-be75np	98.60 (1.09)	98.49 (1.10)	98.18 (1.37)	98.27 (1.23)	97.32 (1.79)	97.50 (1.69)	98.47 (1.15)	97.12 (2.29)
N-stabu75	98.63 (0.53)	98.67 (0.42)	98.73 (0.51)	98.87 (0.58)	97.67 (0.75)	97.79 (0.73)	98.61 (0.54)	97.46 (0.92)
N-t59n11xx	98.51 (0.75)	98.69 (0.64)	98.39 (0.94)	98.36 (0.74)	97.10 (1.67)	97.20 (1.60)	98.60 (0.86)	97.26 (1.67)
N-t65l11xx	99.82 (0.21)	99.86 (0.11)	99.93 (0.11)	99.78 (0.24)	99.40 (0.84)	99.42 (0.85)	99.87 (0.15)	99.34 (0.89)
N-t70d11xx	98.35 (0.92)	97.94 (1.49)	98.00 (1.32)	97.65 (1.38)	96.25 (1.79)	96.07 (1.64)	97.79 (1.30)	96.12 (1.88)
N-t70l11xx	99.59 (0.29)	99.73 (0.24)	99.77 (0.18)	99.63 (0.32)	99.08 (0.59)	99.12 (0.57)	99.39 (0.47)	99.10 (0.55)
N-t74d11xx	98.40 (1.06)	98.36 (1.07)	97.98 (2.10)	98.34 (1.48)	96.81 (1.69)	96.62 (1.68)	98.15 (0.99)	97.03 (1.84)
N-t75n11xx	98.82 (0.54)	98.77 (0.72)	99.14 (0.53)	99.34 (0.53)	97.35 (1.45)	97.47 (1.31)	98.53 (0.68)	96.77 (1.69)
N-tiw56n66	98.27 (1.37)	98.25 (1.04)	98.26 (0.65)	98.63 (0.66)	96.78 (1.46)	96.64 (1.59)	98.20 (1.12)	96.77 (1.46)
N-tiw56r66	98.41 (1.00)	98.67 (0.88)	98.29 (1.02)	98.55 (0.89)	96.89 (1.29)	96.93 (1.31)	98.15 (1.04)	96.95 (1.47)
N-be75oi	99.41 (0.33)	99.38 (0.34)	99.43 (0.40)	99.26 (0.41)	98.85 (0.84)	98.77 (0.87)	99.31 (0.47)	98.89 (0.74)
N-t59b11xx	98.70 (1.27)	98.59 (1.54)	99.25 (0.38)	99.31 (0.45)	96.45 (2.28)	96.88 (2.03)	99.22 (0.49)	96.90 (2.20)
N-t65b11xx	98.46 (0.80)	98.47 (0.90)	98.20 (0.64)	98.26 (0.98)	96.40 (1.48)	96.33 (1.52)	98.09 (0.71)	96.50 (1.57)
N-t65n11xx	98.78 (0.54)	99.00 (0.53)	99.25 (0.20)	98.91 (0.46)	97.27 (1.44)	97.65 (1.22)	99.12 (0.40)	97.63 (1.49)
N-t70d11xxb	97.40 (1.60)	97.58 (1.62)	97.04 (2.36)	98.00 (1.47)	95.56 (1.87)	95.49 (1.76)	97.50 (1.39)	96.06 (2.17)
N-t70n11xx	99.08 (0.37)	98.77 (0.72)	98.98 (0.32)	98.98 (0.35)	97.44 (1.44)	97.34 (1.29)	98.82 (0.48)	97.00 (1.45)
N-t75d11xx	98.15 (0.95)	98.32 (0.66)	98.49 (0.77)	98.19 (0.89)	96.39 (1.50)	96.38 (1.40)	98.00 (0.92)	96.19 (1.71)
N-t75u11xx	97.77 (1.74)	98.50 (1.18)	98.35 (1.43)	97.88 (1.15)	96.98 (1.62)	96.75 (1.75)	98.12 (1.58)	96.83 (2.04)
N-tiw56n67	98.48 (0.82)	98.20 (1.44)	98.41 (0.52)	97.89 (2.17)	96.81 (1.36)	96.72 (1.47)	98.20 (0.90)	96.93 (1.39)
N-tiw56r67	98.44 (0.80)	98.77 (0.67)	98.32 (1.19)	98.84 (0.23)	97.43 (1.27)	97.45 (1.27)	98.84 (1.02)	97.28 (1.45)
N-be75tot	98.59 (0.80)	98.05 (1.24)	98.17 (1.37)	98.68 (0.97)	96.33 (1.66)	96.55 (1.55)	98.11 (1.31)	96.70 (1.90)
N-t59d11xx	99.27 (0.48)	99.19 (0.33)	99.00 (0.83)	99.24 (0.35)	98.10 (1.20)	98.22 (1.06)	99.26 (0.46)	98.10 (1.20)
N-t65d11xx	97.90 (1.69)	97.94 (1.17)	97.28 (1.43)	98.00 (1.07)	96.11 (1.76)	96.09 (1.70)	97.90 (1.26)	95.92 (1.92)
N-t65w11xx	97.73 (1.12)	98.19 (1.16)	98.27 (0.85)	97.65 (1.02)	96.03 (1.98)	95.90 (1.93)	97.91 (1.14)	95.22 (3.45)
N-t70f11xx	98.37 (1.18)	98.59 (0.80)	98.47 (0.95)	98.69 (1.46)	96.10 (1.78)	96.20 (2.03)	98.50 (1.00)	96.57 (1.63)
N-t70u11xx	98.73 (1.38)	98.52 (1.63)	98.05 (1.80)	99.13 (0.41)	97.11 (2.03)	96.93 (2.03)	98.43 (1.69)	97.49 (2.07)
N-t75e11xx	98.84 (0.76)	98.56 (1.16)	98.87 (0.92)	98.56 (0.82)	96.07 (2.15)	95.95 (2.21)	98.85 (0.92)	95.72 (2.56)
N-tiw56n54	98.86 (0.59)	98.82 (0.83)	99.19 (0.44)	99.31 (0.25)	97.37 (1.27)	97.48 (0.89)	98.80 (0.99)	97.48 (1.38)
N-tiw56n72	98.17 (1.12)	98.19 (1.12)	98.44 (0.76)	98.36 (1.07)	96.72 (1.24)	96.88 (1.17)	97.95 (1.21)	96.53 (1.41)
N-tiw56r72	98.27 (1.13)	98.57 (0.96)	98.62 (0.91)	98.55 (0.85)	97.15 (1.37)	97.21 (1.38)	98.43 (1.06)	96.46 (2.84)
N-stabu70	98.92 (0.49)	98.66 (0.67)	98.87 (0.59)	99.13 (0.20)	97.89 (0.96)	97.80 (1.05)	98.82 (0.59)	97.59 (0.95)
N-t59f11xx	98.64 (1.31)	98.91 (0.91)	99.40 (0.28)	98.80 (0.79)	96.17 (2.24)	96.39 (1.97)	98.30 (1.44)	95.88 (2.18)
N-t65f11xx	98.57 (0.80)	98.61 (0.73)	98.76 (0.78)	98.76 (0.66)	96.31 (1.55)	96.41 (1.45)	98.58 (0.74)	96.38 (1.69)
N-t69r11xx	98.77 (0.56)	98.33 (0.95)	98.65 (0.72)	98.43 (1.13)	97.09 (2.36)	97.23 (2.28)	98.85 (0.70)	96.71 (2.39)
N-t70i11xx	98.93 (1.48)	99.07 (0.80)	99.53 (0.23)	98.81 (1.24)	97.47 (1.38)	97.48 (1.35)	99.01 (0.81)	96.98 (1.55)
N-t70w11xx	98.18 (1.46)	98.60 (1.32)	98.18 (1.29)	98.18 (1.29)	96.37 (1.77)	96.25 (1.72)	98.47 (1.18)	96.29 (1.61)
N-t75i11xx	99.13 (0.59)	98.85 (0.70)	99.06 (0.48)	99.17 (0.35)	97.14 (1.61)	97.00 (1.72)	98.93 (0.64)	97.10 (1.71)
N-tiw56n58	98.79 (0.81)	98.71 (0.77)	98.84 (0.88)	98.41 (1.13)	97.27 (1.24)	97.40 (1.09)	98.74 (0.89)	97.09 (1.35)
N-tiw56r54	98.80 (0.66)	98.87 (0.85)	98.44 (0.96)	98.90 (0.41)	97.34 (1.27)	97.62 (1.08)	98.60 (0.64)	97.12 (1.39)
N-usa79	97.20 (1.90)	97.09 (1.70)	97.51 (1.53)	98.18 (0.94)	95.65 (1.63)	95.86 (1.49)	97.12 (1.42)	95.60 (4.14)
Average	98.60 (0.93)	98.62 (0.91)	98.67 (0.83)	98.67 (0.82)	97.02 (1.54)	97.04 (1.47)	98.56 (0.90)	96.97 (1.73)

[17] A. Abraham, "Editorial - hybrid soft computing and applications," *International Journal of Computational Intelligence and Applications*, vol. 8, no. 1, 2009.

[18] E. Corchado, A. Arroyo, and V. Tricio, "Soft computing models to identify typical meteorological days," *Logic Journal of the IGPL*, vol. 19, no. 2, pp. 373–383, 2011.

[19] S.-Z. Zhao, M. W. Iruthayarajan, S. Baskar, and P. Suganthan, "Multi-objective robust pid controller tuning using two lbests multi-objective particle swarm optimization," *Information Sciences*, vol. 181, no. 16, pp. 3323 – 3335, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002002551100171X

[20] J. Sedano, L. Curiel, E. Corchado, E. de la Cal, and J. R. Villar, "A soft computing method for detecting lifetime building thermal insulation failures," *Integr. Comput.-Aided Eng.*, vol. 17, no. 2, pp. 103–115, Apr. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1804647.1804653

[21] P. Krömer, J. Platoš, and V. Snášel, "Differential evolution for the linear ordering problem implemented on cuda," in *Proceedings of the 2011 IEEE Congress on Evolutionary Computation*, A. E. Smith, Ed., IEEE Computational Intelligence Society. New Orleans, USA: IEEE Press, 5-8 June 2011, pp. 790–796.

[22] R. Storn and K. Price, "Differential Evolution- A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces," Tech. Rep., 1995. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.9696

[23] R. Storn, "Differential evolution design of an IIR-filter," in *Proceeding of the IEEE Conference on Evolutionary Computation ICEC*. IEEE Press, 1996, pp. 268–273.

[24] M. Cárdenas-Montes, M. A. Vega-Rodríguez, and A. Gómez-Iglesias, "Sensitiveness of evolutionary algorithms to the random number generator," in *Proceedings of the 10th international conference on Adaptive and natural computing algorithms - Volume Part I*, ser. ICANNGA'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 371–380. [Online]. Available: http://dl.acm.org/citation.cfm?id=1997052.1997093

[25] V. Tirronen, S. yrm, and M. Weber, "Study on the effects of pseudorandom generation quality on the performance of differential evolution," in *Adaptive and Natural Computing Algorithms*, ser. Lecture Notes in Computer Science, A. Dobnikar, U. Lotri, and B. ter, Eds. Springer Berlin Heidelberg, 2011, vol. 6593, pp. 361–370. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20282-7_37

TABLE III

AVERAGE PRECISION AND STD. DEVIATION OF SGB SOLUTIONS FOUND BY DE WITH DIFFERENT SOURCES OF STOCHASTICITY (IN PERCENT).

LOP Instance	GCC rand	Mersenne Twister	Ranlux D1	Ranlux D2	Logistic Equation	Re-mapped Log. Eq.	Logistic Lattice	B-exponential Map
N-sgb75.01	98.54 (0.90)	98.79 (0.73)	98.59 (0.52)	98.90 (0.55)	98.07 (1.19)	98.20 (1.14)	98.87 (0.65)	97.94 (1.02)
N-sgb75.04	98.77 (0.90)	98.95 (0.53)	98.67 (0.59)	98.73 (1.04)	98.07 (1.01)	98.09 (0.95)	99.04 (0.55)	97.99 (1.05)
N-sgb75.07	98.87 (0.74)	98.43 (0.93)	98.40 (0.63)	97.86 (0.99)	98.05 (0.97)	98.19 (0.93)	98.55 (0.82)	97.94 (1.17)
N-sgb75.10	98.75 (0.90)	99.00 (0.50)	99.15 (0.49)	98.89 (0.96)	98.17 (1.07)	97.98 (1.06)	98.58 (0.94)	98.24 (1.13)
N-sgb75.13	98.30 (1.09)	98.93 (0.79)	99.21 (0.13)	98.95 (0.53)	98.07 (0.99)	98.26 (0.90)	98.71 (0.72)	97.86 (1.31)
N-sgb75.16	98.75 (0.87)	99.00 (0.57)	98.92 (0.44)	98.63 (0.86)	97.92 (1.13)	97.85 (1.15)	98.47 (0.97)	98.17 (1.07)
N-sgb75.19	98.75 (0.82)	98.68 (1.08)	98.47 (0.58)	98.80 (0.58)	98.19 (1.00)	98.21 (0.85)	98.66 (1.07)	97.86 (1.10)
N-sgb75.22	98.98 (0.68)	98.98 (0.77)	98.84 (1.16)	98.37 (1.02)	98.05 (1.21)	98.21 (1.06)	98.54 (1.03)	98.11 (1.09)
N-sgb75.25	98.54 (0.99)	98.94 (0.67)	98.70 (0.84)	98.26 (0.98)	97.88 (1.22)	97.67 (1.31)	98.70 (0.98)	97.93 (1.31)
N-sgb75.02	99.13 (0.70)	98.70 (0.97)	97.65 (0.67)	98.55 (0.74)	97.97 (1.18)	98.19 (1.02)	98.88 (0.93)	98.05 (1.14)
N-sgb75.05	98.72 (0.81)	98.78 (0.66)	98.66 (0.91)	98.89 (0.77)	97.94 (1.10)	97.92 (1.14)	98.78 (0.63)	97.75 (1.07)
N-sgb75.08	98.72 (0.82)	98.70 (0.82)	97.93 (0.86)	98.62 (0.81)	97.93 (0.97)	97.89 (0.89)	98.62 (0.72)	97.93 (0.93)
N-sgb75.11	98.60 (0.78)	98.72 (0.74)	98.33 (0.87)	98.69 (0.70)	98.04 (1.04)	98.06 (1.02)	98.61 (0.81)	98.00 (1.00)
N-sgb75.14	98.49 (0.92)	99.00 (0.56)	98.72 (0.73)	98.92 (0.44)	97.98 (1.13)	97.86 (1.10)	98.64 (0.92)	97.82 (1.40)
N-sgb75.17	98.83 (0.58)	98.63 (0.91)	98.67 (0.84)	98.51 (0.46)	98.09 (1.03)	98.25 (1.04)	98.92 (0.67)	98.16 (0.96)
N-sgb75.20	98.53 (0.84)	98.65 (0.77)	98.79 (0.63)	98.77 (0.62)	98.01 (1.11)	97.99 (1.09)	98.75 (0.52)	98.05 (1.01)
N-sgb75.23	99.04 (0.49)	98.86 (0.53)	98.87 (0.21)	98.45 (0.79)	98.20 (1.09)	98.02 (1.21)	98.93 (0.83)	98.16 (1.11)
N-sgb75.03	98.54 (0.83)	98.74 (0.85)	98.58 (0.47)	98.20 (1.20)	97.77 (1.17)	97.57 (1.21)	98.57 (0.97)	97.81 (1.58)
N-sgb75.06	98.83 (0.91)	98.82 (0.71)	98.95 (0.59)	99.12 (0.44)	98.05 (1.13)	97.96 (1.21)	98.83 (0.86)	98.18 (1.04)
N-sgb75.09	98.72 (0.87)	99.01 (0.60)	98.56 (0.76)	98.58 (0.85)	98.14 (0.89)	98.18 (0.90)	98.89 (0.74)	97.99 (1.06)
N-sgb75.12	98.82 (0.69)	98.43 (0.81)	99.27 (0.31)	98.53 (0.81)	98.12 (1.01)	98.18 (1.03)	98.48 (0.99)	97.88 (1.21)
N-sgb75.15	98.53 (0.91)	98.90 (0.69)	98.95 (0.85)	98.66 (0.70)	97.93 (1.16)	97.75 (1.23)	98.66 (0.65)	98.06 (1.21)
N-sgb75.18	98.97 (0.75)	98.67 (1.03)	99.05 (0.57)	98.94 (0.83)	98.44 (1.01)	98.44 (0.94)	98.71 (0.97)	98.37 (1.15)
N-sgb75.21	98.56 (0.68)	98.83 (0.61)	98.67 (0.87)	98.62 (0.99)	97.83 (1.22)	97.91 (1.32)	98.80 (0.75)	97.95 (1.40)
N-sgb75.24	98.32 (0.97)	98.30 (1.00)	98.58 (0.96)	98.75 (0.52)	97.86 (1.10)	97.83 (1.16)	98.55 (0.88)	97.87 (1.12)
Average	98.71 (0.81)	98.78 (0.75)	98.69 (0.67)	98.64 (0.78)	98.03 (1.08)	98.02 (1.07)	98.70 (0.83)	98.01 (1.15)

[26] M. Yang, J. Guan, Z. Cai, and L. Wang, "Self-adapting differential evolution algorithm with chaos random for global numerical optimization," in *Proceedings of the 5th international conference on Advances in computation and intelligence*, ser. ISICA'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 112–122. [Online]. Available: http://dl.acm.org/citation.cfm?id=1926680.1926694

[27] N. R. Wagner, "The logistic lattice in random number generation," in *Proceedings of the Thirtieth Annual Allerton Conference on Communications, Control, and Computing*. Coordinated Science Lab and Department of Electrical and Computer Engineering, University of Illinois at Urbabn-Champaign, 1993, pp. 922–931.

[28] M. Andrecut, "Logistic map as a random number generator," *International Journal of Modern Physics B*, vol. 12, no. 09, pp. 921–930, 1998. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S021797929800051X

[29] J. Szczepański and Z. Kotulski, "Pseudorandom number generators based on chaotic dynamical systems," *Open Systems & Information Dynamics*, vol. 8, no. 2, pp. 137–146, Jun. 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1011950531970

[30] M. C. Shastry, N. Nagaraj, and P. G. Vaidya, "The b-exponential map: A generalization of the logistic map, and its applications in generating pseudo-random numbers," *CoRR*, vol. abs/cs/0607069, 2006.

[31] S.-L. Chen, T. Hwang, and W.-W. Lin, "Randomness enhancement using digitalized modified logistic map," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 57, no. 12, pp. 996–1000, 2010.

[32] K. Persohn and R. Povinelli, "Analyzing logistic map pseudorandom number generators for periodicity induced by finite precision floating-point representation," *Chaos, Solitons & Fractals*, vol. 45, no. 3, pp. 238 – 245, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960077911002384

[33] L. V. Snyder and M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 174, no. 1, pp. 38–53, 2006.

[34] P. Krömer, J. Platoš, and V. Snášel, "Modeling permutations for genetic algorithms," in *Proceedings of the International Conference of Soft Computing and Pattern Recognition (SoCPaR 2009)*. IEEE Computer Society, 2009, pp. 100 – 105.

[35] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, Jan. 1998. [Online]. Available: http://doi.acm.org/10.1145/272991.272995

[36] M. Luscher, "A portable high-quality random number generator for lattice field theory simulations," *Computer Physics Communications*, vol. 79, no. 1, pp. 100 – 110, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0010465594902321

[37] G. Box, J. Hunter, and W. Hunter, *Statistics for experimenters: design, innovation, and discovery*, ser. Wiley series in probability and statistics. Wiley-Interscience, 2005.