Double-Stranded Differential Evolution and Particle Swarm Optimization with LibreOffice Nonlinear Programming Solver

Gergana Mateeva[1], Delyan Keremedchiev[2], Kalin Kopanov[1], Velizar Varbanov[1], and Todor Balabanov[1](✉)

[1] Bulgarian Academy of Sciences, Institute of Information and Communication Technologies, acad. Georgi Bonchev Street, block 2, 1113 Sofia, Bulgaria
{gergana.mateeva,kalin.kopanov,velizar.varbanov, todor.balabanov}@iict.bas.bg

[2] Department of Informatics, New Bulgarian University, 21 Montevideo Street, block 2, 1618 Sofia, Bulgaria
delyank@nbu.bg
http://iict.bas.bg/, http://www.nbu.bg/

Abstract. Differential Evolution and Particle Swarm Optimization are heuristic global optimization methods inspired by natural evolution and swarm behavior. They are often used to solve complex optimization and simulation problems that are time-consuming or impossible to solve using exact numerical methods. Traditionally, RNA ideas are closer to Differential Evolution population formation. This paper proposes a double-stranded (more DNA-like) implementation of population in LibreOffice Calc NLP Solver. The proposed implementation is validated with well-known optimization benchmark functions.

Keywords: Double-stranded genetic algorithms · Nonlinear optimization · LibreOffice

1 Introduction

The fundamental concept behind Differential Evolution is to emulate the process of natural selection and reproduction to generate a population of potential solutions to a problem. The calculation begins with a random population of candidate solutions and subsequently employs operators, such as selection, crossover, and mutation [1], to generate new generations of candidate solutions. Similarly, Particle Swarm Optimization draws inspiration from the social behavior of birds

Supported by Velbazhd Software LLC

T. Zlateva and G. Tuparov (Eds.): CSECS 2024, LNICST 609, pp. 146–153, 2025.
https://doi.org/10.1007/978-3-031-84312-9_10

and fish, initially proposed by James Kennedy and Russell Eberhart in 1995 [2]. Both algorithms are effectively employed together in a hybrid implementation for global optimization.

1.1 Selection in Population-Based Algorithms

In the selection stage, solutions demonstrating superior performance on the given problem are more likely to be reproduced. This stage constitutes a fundamental component of population-based algorithms, wherein the fittest individuals from a population are selected to serve as parents for the subsequent generation [3]. The selection operator is pivotal in deciding which individuals will contribute genetic material to the next generation, making it a critical factor influencing the performance and efficiency of population-based algorithms. Population-based algorithms employ various selection methods, each presenting distinct advantages and disadvantages.

The selection operator is crucial because it enables population-based algorithms to identify and preserve the best solutions, gradually enhancing the population's overall fitness over time. By choosing the fittest individuals as parents for the next generation, the population-based algorithm can converge toward an optimal solution more rapidly and efficiently than if it were to select parents randomly. However, it is essential to strike a balance between selection pressure and genetic diversity to ensure the algorithm smoothly converges to an optimal solution, avoiding sub-optimal outcomes.

1.2 Crossover in Population-Based Algorithms

In the crossover stage, two or more candidate solutions are merged to form a new candidate solution. This stage holds significance in population-based algorithms, as it emulates the process of sexual reproduction in biological organisms. This process entails the exchange of genetic material between parent individuals to generate offspring possessing a combination of their traits. The crossover operator is employed to enhance the population's genetic diversity and provide novel solutions that may better suit the problem at hand.

The crossover operator is typically applied with a probability known as the crossover rate, which determines the likelihood of performing crossover on a given pair of parents. Various types of crossover operators can be utilized in population-based algorithms [4].

The choice of which type of crossover operator to use depends on the problem being solved and the characteristics of the population being evolved. Crossover generally combines the positive traits of parent individuals while minimizing the negative traits; however, it can also introduce new genetic material that may need to be more fit than the original.

One challenge with the crossover operator is the potential for premature convergence, where the algorithm converges too quickly to a sub-optimal solution.

To mitigate this risk, it is essential to balance crossover with other genetic operators, such as mutation, and carefully control the crossover rate to ensure that the algorithm maintains sufficient genetic diversity throughout evolution.

1.3 Mutation in Population-Based Algorithms

In the mutation stage, a random change is made to a candidate solution to introduce diversity in the population. It is a fundamental component of population-based algorithms that introduces small, random changes to an individual's genetic material in a population. The mutation operator is used to maintain genetic diversity in the population of individuals and to prevent the genetic algorithm from becoming stuck in local optima, which can be sub-optimal solutions to the problem being solved.

The mutation operator is typically applied with probability (mutation rate) [5]. This rate determines the likelihood that a given genes in an individual will undergo mutation. Generally, the mutation rate is set to a low value, 1% or 5%, to ensure that only a tiny fraction of the population is mutated at any given time. Various types of mutation operators can be employed in population-based algorithms.

The choice of which type of mutation operator to use depends on the problem being solved and the characteristics of the evolving population. In general, mutation introduces new genetic material that may benefit the solution to the problem. However, excessive mutation can lead to a loss of fitness in the population. Therefore, it is crucial to balance mutation with other genetic operators, such as crossover, to ensure that the population-based algorithm converges efficiently toward an optimal solution.

Over time, the population evolves towards better solutions as the algorithm repeatedly applies these operators to generate new generations of candidate solutions. Population-based algorithms have been employed for a wide range of problems, from optimization problems in engineering to game-playing and machine learning.

1.4 Natural Genetics

Ribonucleic acid is a molecule that plays a crucial role in transferring genetic information in living organisms. Similar to how RNA serves as a genetic code conveying information from DNA to ribosomes, specific population-based algorithms utilize a population of candidate solutions as a form of genetic code, conveying information about potential solutions to a problem. Like RNA, some population-based algorithms employ selective pressure and random mutation to generate diverse populations of candidate solutions. These solutions are then assessed based on their fitness or ability to address the problem. The evolution of these populations over time mirrors the process of genetic information evolution in living organisms, where the fittest solutions survive and propagate while less fit solutions die out. Heuristics in population-based algorithms provide a distinctive simulation of the processes representing evolution in the real world.

The Proposition

In this study, single chromosomes are paired to mimic the structure of DNA. Each candidate solution has its complementary counterpart, and together, they form a complementary pair.

The remainder of the paper is organized as follows: The second section presents the conceptual idea; the third section describes the practical implementation as a LibreOffice Calc model; the fourth section discusses the experimental part and the results achieved; the last section concludes and proposes further research.

2 Double-Stranded Candidate Solutions

As is commonly known, the population of widely employed heuristics comprises chromosomes, each with its distinct existence within the population. These chromosomes emulate the RNA structure found in the biological realm.

While genetic algorithms commonly employ binary encoding, it may only sometimes be optimal for some optimization or simulation problems. Floating-point number encoding is often more suitable for problems involving real numbers, such as benchmark functions like Rastrigin, Sphere, Rosenbrock, and Styblinski-Tang [6].

Using floating-point number encoding allows a more direct representation of the problem domain. In this encoding, genes in the chromosome represent floating-point values instead of binary values, potentially leading to quicker convergence and superior solutions, particularly in problems with continuous variables.

Various methods for floating-point number encoding exist, including Gray encoding. In floating-point encoding, the chromosome's genes are represented as floating-point numbers. In contrast, the genes are depicted as binary digits converted to real numbers using a specific formula in Gray encoding.

Extending the analogy of genetic algorithms to paired DNA strands is feasible. In this approach, each chromosome in the population would have a complementary pair, akin to the base pairs in DNA strands, termed double-stranded genetic algorithms (DSGAs) [7]. In DSGAs, chromosomes are paired, forming complementary pairs treated as single entities during selection, crossover, and mutation operations.

Using paired chromosomes in DSGAs can provide advantages over traditional population-based algorithms, facilitating more efficient search space exploration and enhanced handling of constraints in optimization problems.

It is important to note that DSGAs may entail higher computational costs than traditional algorithms, given the need to maintain complementary pairs and perform operations on pairs rather than individual chromosomes.

The proposal suggests utilizing chromosomes organized as floating-point twins with sign-inverted values in the population, known as floating-point twin optimization. In this method, each chromosome is represented by two twins

[8], where values in one twin have the opposite sign to form the other twin. This approach leverages the universality of floating-point encoding.

One advantage of twin chromosomes is their shared fitness value, calculated separately for each twin, mitigating premature convergence to sub-optimal solutions. Additionally, twin chromosomes can contribute complementary information about the search space, aiding in maintaining diversity in the population.

However, it is crucial to acknowledge that using twin chromosomes may incur additional computational costs due to the inversion operation applied to each value. Additionally, this approach may only be suitable for problems that align well with floating-point encoding.

3 Pragmatic Realization

LibreOffice Calc has a useful module called NLP Solver for global non-linear problem optimization [9]. Several algorithms are available, but a hybrid combination of Differential Evolution and Particle Swarm Optimization is the most applicable.

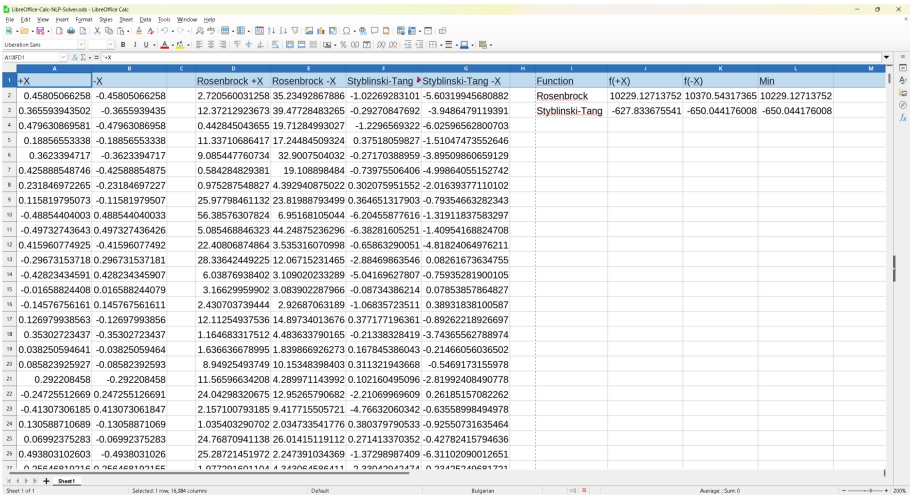
There are many software implementations of evolutionary algorithms. Mathematical - oriented tools like Matlab [10], R [11], Python [12], and many others can offer great possibilities for experimenting with ideas in evolutionary algorithms. Each tool has advantages and disadvantages. The selection of a particular software instrument in the current study is related to its openness. The NLP Solver in LibreOffice Calc is far from being state-of-the-art in evolutionary computing. Anyway, the NLP Solver is an open-source module. It has an open implementation of the most used global optimization metaheuristics.

The NLP Solver has the advantage of visually presenting the proposed double-stranded chromosomes. The encoding of genes is illustrative, and running the optimization for comparative analysis between single-stranded and double-stranded is numerically expressive.

The numerical problem for which an optimal solution will be sought is entered into the spreadsheet sheet. Initial values, randomly selected, are set in consecutive cells, preferably in a column, but not necessarily required. In an adjacent column, calculations are performed for each value of the function being optimized. For functions that involve sums, a specific cell is designated to receive the target value for optimization. The proposed spreadsheet model can be accessed at the following information source [13].

As illustrated in Fig. 1, the initial data column comprises the X vector supplied to the target function. The second column replicates the same vector but with negative values. These two vectors constitute a complementary pair. The Rosenbrock and Styblinski-Tang function values are computed for each vector. From the resulting values, the minimum is populated into the target cell. Given that the objective is minimization, the smaller of the two values is selected as the fitness value of the complementary pair.

As depicted in Fig. 2, the target cell comprises the computed value of the function, and only the first twin of the complement pair is permitted to change



+X	-X	Rosenbrock +X	Rosenbrock -X	Styblinski-Tang +X	Styblinski-Tang -X	Function	f(+X)	f(-X)	Min
0.45805066258	-0.45805066258	2.720560031258	35.23492867886	-1.02269283101	-5.60319945680882	Rosenbrock	10229.12713752	10370.54317365	10229.12713752
0.365593943502	-0.3655939435	12.37212923673	39.47728483265	-0.29270847692	-3.9486479119391	Styblinski-Tang	-627.833675541	-650.044176008	-650.044176008
0.47963086981	-0.47963086958	0.442845043655	19.71284993027	-1.2296569322	-6.02596562800703				
0.18856553338	-0.18856553338	11.33710686417	17.24484509324	0.37518059827	-1.51047473552646				
0.3623394717	-0.3623394717	9.085447760734	32.9007504032	-0.27170388959	-3.89509860659129				
0.425888548746	-0.42588854875	0.584284829381	19.108898484	-0.73975506406	-4.99864055152742				
0.231846972265	-0.23184697227	0.975287548827	4.392940875022	0.302075951552	-2.01639377110102				
0.115819795073	-0.11581979507	25.97798461132	23.81988793499	0.364651317903	-0.79354663282343				
-0.48854404003	0.488544040033	56.38576307824	6.95168105044	-6.20455877616	-1.31911837583297				
-0.49732743643	0.497327436426	5.085468846323	44.24875236296	-6.38281605251	-1.40954168824708				
0.415960774925	-0.41596077492	22.40806874864	3.535316070998	-0.65863290051	-4.81824064976211				
-0.29673153718	0.296731537181	28.33642449225	12.06715231465	-2.88469863546	0.08261673634755				
-0.42823434591	0.428234345907	6.03876938402	3.109020233289	-5.04169627807	-0.75935281900105				
-0.01658824408	0.016588244079	3.16629959902	3.083902287966	-0.08734386214	0.078538578648 27				
-0.14576756161	0.145767561611	2.430703739444	2.92687063189	-1.06835723511	0.38931838100587				
0.126979938563	-0.12697993856	12.11254937536	14.89734013676	0.377177196361	-0.89262218926697				
0.35302723437	-0.35302723437	1.164683317512	4.483633790165	-0.21338328419	-3.74365562788974				
0.038250594641	-0.03825059464	1.636636678995	1.839866926273	0.167845386043	-0.21466056036502				
0.085823925927	-0.08582392593	8.94925493749	10.15348398403	0.311321943668	-0.5469173155978				
0.292208458	-0.292208458	11.56596634208	4.289971143992	0.102160495096	-2.81992408490778				
-0.24725512669	0.247255126691	24.04298320675	12.95265790682	-2.21069969609	0.26185157082262				
-0.41307306185	0.413073061847	2.157100793185	9.417715505721	-4.76632060342	-0.63558998494978				
0.130588710689	-0.13058871069	1.035403290702	2.034733541776	0.380379790533	-0.92550731635464				
0.06992375283	-0.06992375283	24.76870941138	26.01415119112	0.271413370352	-0.42782415794636				
0.493803102603	-0.4938031026	25.28721451972	2.247391034369	-1.37298987409	-6.31102090012651				

Fig. 1. LibreCalc optimization problem model

through Differential Evolution and Particle Swarm Optimization processes. The second twin is automatically generated due to the modifications made to the first twin.

4 Experiments and Results

All experiments were conducted on a computer with the following specifications: Asus-2023, featuring a 12th Gen Intel(R) Core(TM) i7-12700 processor running at 2.10 GHz, 32.0 GB of RAM, and a 64-bit operating system (x64-based processor). The specific Windows version used was Windows 11 Pro 23H2 22631.3007, and the software employed for the experiments was LibreOffice Community 7.5.9.2 (X86_64) with a total of 20 CPU threads.

Rosenbrock and Styblinski-Tang's functions were employed as benchmark functions for experimental simulations. They were selected due to the asymmetry present in the negative twin. Calculations were conducted in a 1000-dimensional space over 100 optimization cycles.

This study does not focus on the implementation of Differential Evolution and Particle Swarm Optimization, done almost two decades ago [14,15]. This study focuses on a comparison between single-strained chromosomes and double-strained chromosomes in all other equal conditions.

The performance ratio of the two optimization algorithms, Differential Evolution and Particle Swarm Optimization, is 50 to 50%. The crossover probability is set at 90% in Differential Evolution, and the scaling factor is 0.5. For the Particle Swarm Optimization, the cognitive constant is 1.494, and the constric-

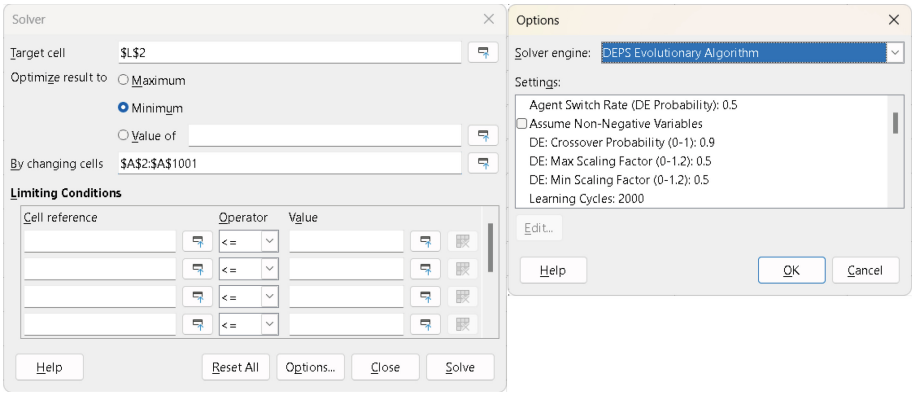


Fig. 2. NLP Solver configuration

tion coefficient is 0.729. No mutation is applied in the particle swarm, and the social constant is maintained at 1.494. The swarm size is configured to have 70 individuals.

Table 1. Experimental results

	Rosenbrock - Single		Rosenbrock - Twins		Styblinski-Tang - Single		Styblinski-Tang - Twins	
	Value	Time [s]	Value	Time [s]	Value	Time [s]	Value	Time [s]
Mean	5265.11	17.79	5131.65	17.64	-293.30	17.32	-280.76	17.30
SD	749.38	0.70	613.06	0.66	61.40	0.74	45.50	0.77

The optimization process has been initiated 30 times. The average values and their standard deviation are presented in Table 1. It is evident that double-stranded optimization yields better results within the same time frame compared to regular optimization.

5 Conclusions

In conclusion, enhancing population-based algorithms inspired by DNA in LibreOffice Calc can potentially enhance problem-solving efficiency through evolutionary optimization. This adaptation can prove especially advantageous for resource-limited instances of LibreOffice Calc, empowering it to execute intricate tasks with increased speed and precision. For future investigations, exploring additional benchmark functions and evaluating performance on various mobile hardware platforms could be pursued.

Acknowledgements. This research is financed by the Central Strategic Development Fund of New Bulgarian University.

References

1. Lambora, A., Gupta, K., Chopra, K.: Genetic algorithm- a literature review. In: International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, pp. 380–384 (2019)
2. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN 1995 - International Conference on Neural Networks, Perth, WA, Australia, vol. 4, pp. 1942–1948 (1995)
3. Miller, B.L., Goldberg, D.E.: Genetic algorithms, selection schemes, and the varying effects of noise. Evol. Comput. **4**(2), 113–131 (1996)
4. Umbarkar, A.J., Sheth, P.D.: Crossover operators in genetic algorithms: a review. ICTACT J. Soft Comput. **6**(1), 1083–1092 (2015)
5. Greenwell, R.N., Angus, J.E., Finck, M.: Optimal mutation probability for genetic algorithms. Math. Comput. Model. **21**(8), 1–11 (1995)
6. Jamil, M., Yang, X., Zepernick, H.J.: 8 - Test Functions for Global Optimization: A Comprehensive Survey, pp. 193–222. Elsevier, Swarm Intelligence and Bio-Inspired Computation (2013)
7. Zang, W., Zhang, W., Wang, Z., Jiang, D., Liu, X., Sun, M.: A novel double-strand DNA Genetic algorithm for multi-objective optimization. IEEE Access **7**, 18821–18839 (2019)
8. Yang, S.: PDGA: The Primal-dual Genetic Algorithm, pp. 1–10. IOS Press (2003)
9. Tomov, P.: Multilayer perceptron fast prototyping with differential evolution and particle swarm optimization in LibreOffice Calc. Probl. Eng. Cybern. Robot. **75**, 5–14 (2021)
10. Fiala, J., Kocvara, M., Stingl, M.: PENLAB: A MATLAB solver for nonlinear semidefinite optimization (2013). arXiv preprint arXiv:1311.5240
11. Nash, J.C.: Nonlinear Parameter Optimization Using R Tools. John Wiley and Sons, Hoboken (2014)
12. Beck, A.: Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with Python and MATLAB. Society for Industrial and Applied Mathematics (2023)
13. Balabanov, T.: Rosenbrock and Styblinski-Tang benchmark function reproduced as double-stranded chromosomes for LibreOffice Calc NLP Solver, ResearchGate GmbH (2024). https://doi.org/10.13140/RG.2.2.14156.59528
14. Zhang, W.J., Xie, X.F.: DEPSO: hybrid particle swarm with differential evolution operator. In: IEEE International Conference on Systems, Man and Cybernetics, Washington D C, USA, pp. 3816-3821 (2003)
15. Xie, X.F., Zhang, W.J.: SWAF: swarm algorithm framework for numerical optimization. In: Genetic and Evolutionary Computation Conference (GECCO), Seattle, WA, USA, pp. 238–250 (2004)