MATHEMATICAL MODELING AND COMPUTING, Vol. 10, No. 3, pp. 764–771 (2023)

A new improved simulated annealing for traveling salesman problem

Adil N., Lakhbab H.

Hassan II University, Fundamental and applied Mathematics Laboratory, Casablanca, Morocco

(Received 15 February 2023; Revised 18 July 2023; Accepted 19 July 2023)

Simulated annealing algorithm is one of the most popular metaheuristics that has been successfully applied to many optimization problems. The main advantage of SA is its ability to escape from local optima by allowing hill-climbing moves and exploring new solutions at the beginning of the search process. One of its drawbacks is its slow convergence, requiring high computational time with a good set of parameter values to find a reasonable solution. In this work, a new improved SA is proposed to solve the well-known travelling salesman problem. In order to improve SA performance, a population-based improvement procedure is incorporated after the acceptance phase of SA, allowing the algorithm to take advantage of the social behavior of some solutions from the search space. Numerical results were carried out using known TSP instances from TSPLIB and preliminary results show that the proposed algorithm outperforms in terms of solution quality, the other comparison algorithms.

Keywords: *simulated annealing; travelling salesman problem; metaheuristics.*

2010 MSC: 90B06, 90C59, 90-08, 90C27, 90C90 **DOI:** 10.23939/mmc2023.03.764

1. Introduction

Metaheuristics are approximate methods designed to solve complex optimization problems. Some of these methods imitate certain strategies or metaphors from nature. Their goal is to provide a good solution in a polynomial time for problems arising in science and engineering.

The advantage of metaheuristic methods is that they don't depend on the characteristics of the problem being optimized. The disadvantage is the number of parameters that should be fine-tuned to obtain good approximate solutions. We can classify those methods as single-individual methods and population-based methods. Single-based methods start with and improve a single solution along iterations, like Tabu Search [1], and Simulated Annealing (SA) [2]. While population-based methods require the use of multiple candidate solutions rather than one. Particles Swarm Optimization (PSO) [3] is a famous example of this type of method.

In this paper, we introduce an improved simulated annealing algorithm we call NISA, where, we incorporate a population-based mechanism, to increase the diversification and enhance intensification in the algorithm. The robustness and efficiency of NISA method are tested on 12 instances from TSPLIB95. Therefore, the paper is organized as follows. Related work is presented in Section 2. Traveling salesman problem is presented in Section 3. The standard algorithm of SA is described in Section 4. Our New Improved Simulated Annealing (NISA) is introduced in Section 5, followed by the numerical results in Section 6, and finally the conclusion in Section 7.

2. Related work

The simulated annealing algorithm SA is a stochastic search algorithm, proposed by Kirkpatrick et al. [2]. Its development originates from an analogy with metallurgical annealing process and its power remains in its ability to escape from local optimums by occasionally allowing uphill moves.

Although SA is a classical method, it still grabs researchers' attention by its simplicity and its effectiveness in avoiding local optima traps. It was first designed to solve discrete optimization problems, then it was adapted to be used in the context of continuous optimization also. Discrete optimization problems define optimization problems where the decision variable is discrete. Examples of this type of optimization would be the Travelling salesman Problem (TSP) [7], Vehicle routing problem [19], assignment problem [18], and many more.

TSP was first presented by Dantzig in 1959 [4]. TSP belongs to a class of problems known as being NP-complete [5], and represent a good benchmark to analyze algorithms and validate newly proposed ones, thus it still grabs the researchers' attention and is still an interesting research topic. The state of the art around this problem is so wide, that we will focus on recent advances using new variants or hybridization of SA to solve the TSP.

Zhong et al. [6] is among the works that used metropolis criterion inspired by SA, to improve and obtain a better balance between intensification and diversification in their proposed discrete pigeon-inspired optimization algorithm to solve large-scale TSP Metropolis. Euzugwu et al. [7] proposed a hybrid metaheuristic for solving TSP based on SA and symbiotic Organisms Search method known as an effective new metaheuristic search algorithm. SA was also used to enhance the diversification of the population of a Gene Expression Programming method in the work of Zhou et al. [8]. In the work of Zhong et al., [9], a discrete comprehensive particle swarm optimization with metropolis criterion was proposed aiming to enhance its ability to escape from premature convergence. Geng et al. [10] combined the application of simulated annealing with a greedy search algorithm to speed up its convergence rate. Results from all these works accessed the efficiency of SA mechanisms in increasing other metaheuristics performance.

Another category of research has focused on studying SA and proposed new strategies to improve the method. In the paper [11] an evolutionary SA is developed for the TSP and compared with the Tabu Search method. Another interesting work represented in [12], the authors introduced a list-based simulated annealing (LBSA) algorithm to solve TSP, the objective is to simplify cooling schedule parameters setting, using an adaptive list-based schedule to control the decrease in temperature. The results showed that LBSA is a robust and highly competitive method. Johannes J. Schneider et al. [13] propose a new simplified approach to derive adaptive cooling schedule based on retained data in memory. Another interesting work is the paper of Roberto da Silva et al. [14] that studies the effects of the statistics on the coordinates of the points when they apply a standard simulated annealing algorithm to the traveling salesman problem.

3. Traveling salesman problem

Traveling salesman problem can simply be described as the problem of finding the shortest path possible in order to visit N cities only once and returning to the starting point at the end. The objective is to find a route that minimizes the total travelled distance.

Let $G = (V, E)$ a complete graph, where C is the set of vertices representing the cities to be visited and $E = (c_i, c_j), c_i, c_j \in C, i \neq j$ is the set of edges that links the cities. Let d_{ij} be the cost associated to each edge. In this paper, the symmetric TSP is considered, i.e. that the cost $d_{ij} = d_{ji}$ for all instances of the problem. Thus the problem can be mathematically formulated as follows:

$$\underset{\mathbf{X}}{\text{minimize}} \quad f(\mathbf{X}) = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ i\neq j}}^{N} d_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{\substack{j=1 \\ i\neq j}}^{N} x_{ij} = 1, \quad \forall j \in \{1, \dots, N\}, \tag{2}$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{N} x_{ij} = 1, \quad \forall i \in \{1, \dots, N\}, \tag{3}$$

$$\sum_{i,j\in S} x_{ij} \leqslant |S|-1, \quad \forall S\subset V. \tag{4}$$

Where $x_{ij}\in\{0,1\}$ is the decision variable and takes 1 if the edge (i,j) is used in the solution or 0 otherwise. In addition, the objective function is represented in Expression (1). Expressions (2) and (3) ensure that each vertex must be visited once and only once. Lastly, (4) guaranties the absence of sub-tours.

4. Simulated annealing for TSP

As seen in the previous section, simulated annealing algorithm SA is a simple and effective method widely used to enhance other metaheuristics performance. SA begins with a single solution from the search space and an initial temperature, then it explores both good and bad neighboring solutions until a predefined stopping criterion is met. The steps of SA for travelling salesman problem can be described as follows.

Initialization step. The initialization step of SA consists of getting an initial set to start the algorithm with. The set includes the starting solution x_0 that can be generated randomly or using one of the known constructive methods for the TSP, the initial temperature $T_{\max}$, the minimum permissible temperature $T_{\min}$ and the stopping criterion. The fitness function f must also be defined. In this work, the initial solution was generated randomly and the fitness function in our case is the total travelled distance.

Generate new neighbouring solution. In this step, the process of SA explores new neighbours of the current solution x_{current}. The definition of a neighbouring solution or a neighbourhood of solution is always a difficult task and problem specific. In the case of TSP, many operators can be used. As the improvement we propose later is independent of the choice of such operators, we decided to adopt the 2-opt operator in our work, a simple yet effective operator to generate a new tour from the current one. The general idea of 2-opt is that two random edges in the current tour are replaced by two new edges as long as the result is a tour better than the previous one. In the case of SA, recall that we need some uphill moves, and thus our 2-opt should be able to produce both improving and non-improving solutions for the process to work correctly. For that, using the current Temperature T (updated during the process of SA), we introduced metropolis criterion (explained in step 3) in 2-opt, so that the replacement of two random edges can be done occasionally even if the length of the resulting tour is higher than the previous tour. That can be called 2-opt-metropolis to avoid confusion.

Acceptance criteria. After the step 2, the next step is to decide whether the new solution is to replace the current one in the search process or not. This step is known as the acceptance step, and it is based on the current temperature, and the fitness of the solutions. We can distinguish two cases:

— If the fitness of the new solution is better $f(x_{\text{new}}) < f(x_{\text{current}})$, then this one becomes the new current one.

— Otherwise, the solution can be accepted with a probability MC. This probability is known as metropolis criterion and is computed with the formula:

$$MC=\exp\left(-\frac{f(x_{\text{new}})-f(x_{\text{current}})}{T}\right).$$

Metropolis criterion is what allows the uphill moves, to avoid local optima and increase the exploration of the search space.

Temperature update. The temperature is an important parameter in the SA process. The performance of the algorithm is highly influenced by the cooling schedule. The cooling schedule is the strategy used to decrease the current temperature T along iterations. If the cooling process is too fast, SA would not be given enough time to explore the solution space while if it is too high, it can be computationally expensive. Therefore, a good temperature update equation is crucial to have satisfying results. We used a monotonically decreasing formula that can be defined as follows:

$$T=T_{\max}-[(T_{\max}-T_{\min})*k]/N_{\max}. \tag{5}$$

With k is the current iteration and $N_{\max}$ is the maximum number of iterations for SA process to stop.

Stopping criterion. The stopping criterion in algorithms can be defined in many ways and often depends on the decision-makers choice. The algorithm can stop if it reaches a maximum number of function evaluations $FE_{\max}$ or a predefined maximum number of iterations $N_{\max}$ or if the temperature got to $T_{\min}$. Finally, SA can be summarized in Algorithm 1.

Algorithm 1 SA pseudo-code

```
Require: initial temperature $T_{\max}$, minimum temperature $T_{\min}$, maximum number of iterations $N_{\max}$;
Ensure: Randomly generate initial solution $x_0$; Evaluate fitness for initial solution $f(x_0)$;
 1: set $x_{\text{best}} \leftarrow x_0; T \leftarrow T_0$;
 2: repeat
 3:     Generate new solutions $x_{\text{new}}$ using 2-opt-metropolis;
 4:     compute $\Delta f = f(x_{\text{new}}) - f(x_0)$;
 5:     if $\Delta f < 0$ then
 6:         Accept new solution;
 7:     if random$[0,1] < \exp(-\Delta f/T)$ then
 8:         Accept new solution;
 9:     Update temperature $T$ using Eq. (5);
10:     Update $x_{\text{best}}$;
11: until termination criterion reached.
```

5. Proposed improved method NISA

The improvement procedure is inspired by population-based algorithms, it can be seen as a local search component developed to enhance SA.

After the acceptance step in SA, and with a probability p, a population of solutions $RPop$ is randomly generated from the search space. Then the new current solution of SA x_{SA} is appended as a member into $RPop$. Next, the members of $RPop$ evolve toward the best element in the population (it can be different from the best-found solution of SA). So each member of the population represents a solution. The member i at iteration t is provided with a velocity v_i^t and a position x_i^t. The velocity and solution's update formulas are defined as in the Improved Bat Algorithm from [15] and are given as follows. The velocity was defined using the number of differences of positions between the best solution found so far x_* and the current solution x_i^t, also known as hamming distance:

$$v_i^t = \text{random}(1, \text{hammingdistance}(x_*, x_i^t)).$$

Then to update the position of the member x_i^t of the population, the 2-opt or 3-opt is used v_i^t times, and a mechanism was introduced based on the number of cities to choose between two operators, so the formula is:

$$\begin{cases} x_{inew}^t = 2 - \text{opt}(x_{iold}^t, v_i^t) & \text{if} \quad v_i^t \leqslant N/2, \\ x_{inew}^t = 3 - \text{opt}(x_{iold}^t, v_i^t) & \text{otherwise.} \end{cases}$$

The probability p will be updated with the formula:

$$r_i^t = r_i^0(1 - e^{-\gamma t}).$$

In other words, the new current solution of SA is occasionally improved by exploiting a set of random solutions, taking advantage of social behaviour we find in the population-based methods. This will speed up the convergence of the algorithm, as it not only improves the single point of SA randomly, but it takes into account information about better solutions from different neighbours in each iteration, thus attaining larger search regions from the solution space.

The flowshart of the method is represented in Figure 1.

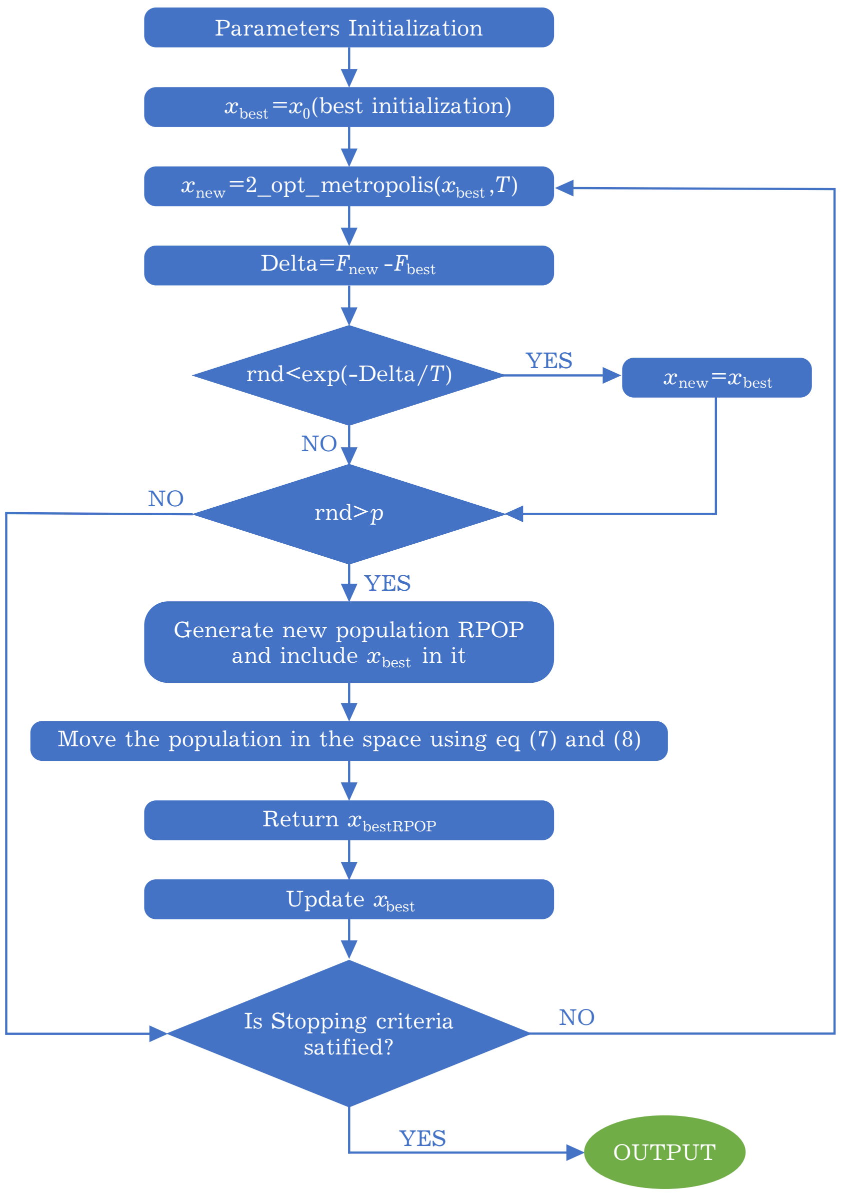
Parameters Initialization
x_{best}=x_0(best initialization)
x_{new}=2_opt_metropolis(x_{best},T)
Delta=F_{new}-F_{best}
rnd<exp(-Delta/T)
YES
x_{new}=x_{best}
NO
rnd>p
NO
YES
Generate new population RPOP and include x_{best} in it
Move the population in the space using eq (7) and (8)
Return x_{bestRPOP}
Update x_{best}
Is Stopping criteria satified?
NO
YES
OUTPUT

Fig. 1. Flowshart of NISA.

6. Numerical results

Comparison tests were done between the proposed NISA, classical SA as described in Section 2, Improved Bat Algorithm from [15] and particle swarm optimization PSO that was adapted to use the same position update formulas as IBA and the velocity update equation being dependent on both the best particle and best past of the incumbent particle like follows:

$$v_i^t = \text{random}\left(1, \max\left[\text{hammingdistance}(x_*, x_i^t), \text{hammingdistance}(x_p, x_i^t)\right]\right).$$

It is worth mentioning that the algorithms were implemented using Python 3.9 and all the tests were performed on an Intel Core i7 laptop, with 2.80GHz and RAM of 16GB.

Note that the instances used in the experimental results are from the famous TSPLIB95 [16], for which optimal values or best-known solutions are provided. The parameters were set as follows:

— Common parameters: $FE_{\max} = 3000$, $T_{\max} = 1000$, $T_{\min} = 0.001$, $N_{\max} = 200$.

— In NISA, the population size of the improvement was set to 4 and maximum iterations set to 5 and $r_p = 0.4$.

— For BA and PSO, the population size was set to 15 and maximum iterations set to 200.

Comparison and discussion. Table 1 represents the best solution (Min), the mean value (Mean), the standard deviation (Std) and the worst solution (Max) values that were calculated after running the algorithm 10 times.

Table 1. Results of NISA vs BA, PSO and SA.

Instances		PSO	BA	SA	NISA	optimum
burma14	**Mean**	3330.2	3332.8	3545.8	**3323**	3323
	STD	15.178933	12.345039	204.229609	**0**	
	Best	**3323**	**3323**	**3323**	**3323**	
	Worst	3359	3359	3793	**3323**	
ulysses16	**Mean**	6862.6	6864.8	7125.8	**6859**	6859
	STD	11.3842	11.792653	199.971554	**0**	
	Best	**6859**	**6859**	**6859**	**6859**	
	Worst	6895	6895	7487	**6859**	
ulysses22	**Mean**	7050.9	7069.5	7482.9	**7013**	7013
	STD	35.132922	47.209345	320.488672	**0**	
	Best	**7013**	**7013**	7047	**7013**	
	Worst	7100	7131	7938	**7013**	
bayg29	**Mean**	1642	1649.6	1718.4	**1624.6**	1610
	STD	14.306176	18.833776	87.510253	**11.296017**	
	Best	1630	1625	1625	**1610**	
	Worst	1677	1677	1856	**1646**	
bays29	**Mean**	2060.6	2079	2188.7	**2034**	2020
	STD	23.538149	30.940804	92.514323	**8.944272**	
	Best	2031	2033	2030	**2020**	
	Worst	2099	2132	2316	**2052**	
gr17	**Mean**	2096.1	2098.4	2160.1	**2085**	2085
	STD	12.887979	18.148768	68.549171	**0**	
	Best	**2088**	**2085**	2085	**2085**	
	Worst	2129	2142	2294	**2085**	
gr21	**Mean**	2774.7	2764.1	2959.8	**2707**	2707
	STD	48.737506	64.205313	181.129052	**0**	
	Best	**2707**	**2707**	**2707**	**2707**	
	Worst	2833	2853	3202	**2707**	
gr24	**Mean**	1283.3	1283.8	1363.3	**1272**	1272
	STD	17.372072	14.589189	59.522265	**0**	
	Best	**1272**	**1272**	1286	**1272**	
	Worst	1326	1317	1461	**1272**	
fri26	**Mean**	947.8	961.7	1040.5	**939.8**	937
	STD	8.337332	10.46741	24.509635	**5.266245**	
	Best	**937**	**937**	1000	**937**	
	Worst	957	975	1085	**953**	
dantzig42	**Mean**	715.1	718.8	861.9	**710.5**	699
	STD	13.543756	14.972568	54.744761	**7.905694**	
	Best	**699**	701	807	**699**	
	Worst	742	753	948	**727**	
eil51	**Mean**	439.5	442.8	531.9	**432.7**	426
	STD	5.948856	5.202563	54.000926	**4.667857**	
	Best	431	437	464	**427**	
	Worst	451	451	648	**442**	
berlin52	**Mean**	**7856.3**	7908.1	8284	7870.2	7542
	STD	135.329269	207.245131	402.855088	**133.010275**	
	Best	7666	**7542**	**7542**	7596	
	Worst	8073	8158	8845	**8050**	

From those results our NISA clearly outperforms PSO, BA and SA in 11 out of 12 TSP instances considering four criteria. Besides, it has reached the optimum value in all instances but two: eil51 and berlin52. NISA algorithm combines in a way the advantages of SA and its uphill moves, and the population-based methods' social behaviour enabling the algorithm to get more information about the new solution's neighbourhood. Besides, as the population is always created from anew, we think that this actually gives a certain balance between diversification and intensification in the method, enhancing its convergence.

To effectively evaluate the performance of NISA, we use both Wilcoxon and sign tests for pairwise comparison. The tests' p-values were computed using the tests from the SciPy library in Python, and are presented in Table 2.

Table 2. Results of wilcoxon Test.

NISA vs	**PSO**	**BA**	**SA**
Win/loss	11/1	12/0	12/0
sign test pvalue	**6.34E-02**	**4.88E-04**	**4.88E-04**
wilcoxon pvalue	**1.22E-02**	**1.22E-02**	**4.88E-04**

The first row presents the number of wins and losses of NISA against PSO, BA and SA. The algorithm is considered a winner on a given instance if the mean of the best of 10 runs is better than the other algorithms.

Now as we have 12 instances, and with a level of significance $\alpha = 0.05$, our algorithm outperforms its standard version (an algorithm is considered better if it has at least 9 wins, see Derrac et al. [17]). The remaining rows present the p-value of the sign test and Wilcoxon's respectively. Both show that NISA has the upper hand with a level of significance of 0.05 over the methods we compared with, thus we can say that our proposed algorithm is very competitive.

7. Conclusion

In this work, an improvement was integrated into the classical SA algorithm. The improvement was inspired by population-based methods. Tests were performed on 12 instances from the TSPLIB. The results of these tests assess that the mechanism is efficient and can enhance the performance of the algorithm. Although the results of the tests are encouraging and the algorithm is competitive with other algorithms, an extensive analysis of other instances is required and the application of some statistical tests should be performed. A parametric study will also be considered.

[1] Glover F. Tabu Search — Part I. ORSA Journal on Computing. **1** (3), 190–206 (1989).

[2] Kirkpatrick S., Gelatt C. D., Vecchi M. P. Optimization by Simulated Annealing. Science. **220** (4598), 671–680 (1983).

[3] Kennedy J., Eberhart R. Particle swarm optimization. Proceedings of ICNN'95 – International Conference on Neural Networks. **4**, 1942–1948 (1995).

[4] Dantzig G. B., Fulkerson D. R., Johnson S. M. On a Linear-Programming, Combinatorial Approach to the Traveling-Salesman Problem. Operations Research. **7** (1), 58–66 (1959).

[5] Brucker P. NP-Complete operations research problems and approximation algorithms. Zeitschrift für Operations – Research. **23**, 73–94 (1979).

[6] Zhong Y., Wang L., Lin M., Zhang H. Discrete pigeon-inspired optimization algorithm with Metropolis acceptance criterion for large-scale traveling salesman problem. Swarm and Evolutionary Computation. **48**, 134–144 (2019).

[7] Ezugwu A. E., Adewumi A. O., Froncu M. E. Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem. Expert Systems with Applications. **77**, 189–210 (2017).

[8] Zhou A.-H., Zhu L.-P., Hu B., Deng S., Song Y., Qiu H., Pan S. Traveling-Salesman-Problem Algorithm Based on Simulated Annealing and Gene-Expression Programming. Information. **10** (1), 7 (2019).

[9] Zhong Y., Lin J., Wang L., Zhang H. Discrete comprehensive learning particle swarm optimization algorithm with Metropolis acceptance criterion for traveling salesman problem. Swarm and Evolutionary Computation. **42**, 77–88 (2018).

[10] Geng X., Chen Z., Yang W., Shi D., Zhao K. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. Applied Soft Computing. **11** (4), 3680–3689 (2011).

[11] Osaba E., Carballedo R., Lopez-Garcia P., Diaz F. Comparison between Golden Ball Meta-Heuristic, Evolutionary Simulated Annealing and Tabu Search for the Traveling Salesman Problem. Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion. 1469–1470 (2016).

[12] Zhan S.-h., Lin J., Zhang Z.-j., Zhong Y.-w. List-Based Simulated Annealing Algorithm for Traveling Salesman Problem. Computational Intelligence and Neuroscience. **2016**, 1712630 (2016).

[13] Schneider J. J., Puchta M. Investigation of acceptance simulated annealing — A simplified approach to adaptive cooling schedules. Physica A. **389** (24), 5822–5831 (2010).

[14] Da Silva R., Filho E. V., Alves A. A thorough study of the performance of simulated annealing in the traveling salesman problem under correlated and long tailed spatial scenarios. Physica A. **577**, 126067 (2021).

[15] Osaba E., Yang X.-S., Diaz F., Lopez-Garcia P., Carballedo R. An improved discrete bat algorithm for symmetric and asymmetric Traveling Salesman Problems. Engineering Applications of Artificial Intelligence. **48** (24), 59–71 (2016).

[16] Reinelt G. Tsplib — A Traveling Salesman Problem Library. ORSA Journal on Computing. **3** (4), 376–384 (1991).

[17] Derrac J., García S., Molina D., Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation. **1** (1), 3–18 (2011).

[18] Sundar A., Rahmin N. A. A., Chen C. Y., Nazihah M. A. Simulated annealing approach for outpatient scheduling in a haemodialysis unit. Mathematical Modeling and Computing. **9** (4), 860–870 (2022).

[19] Yu V. F., Redi A. A. N. P., Hidayat Y. A., Wibowo O. J. A simulated annealing heuristic for the hybrid vehicle routing problem. Applied Soft Computing. **53** (C), 119–132 (2017).

Нова покращена симуляція відпалу для задачі комівояжера

Аділь Н., Лахбаб Х.

Університет Хасана II, лабораторія фундаментальної та прикладної математики, Касабланка, Марокко

Алгоритм симулювання відпалу (SA) є однією з найпопулярніших метаевристик, яка успішно застосовувалася до багатьох задач оптимізації. Головною перевагою SA є його здатність відходити від локальних оптимумів, дозволяючи рухи вверх та досліджувати нові розв'язки на початку процесу пошуку. Одним із його недоліків є його повільна збіжність, що вимагає великого часу обчислення для хорошого набору значень параметрів для пошуку розумного розв'язку. У цій роботі пропонується новий покращений SA для розв'язання відомої задачі комівояжера. Щоб покращити продуктивність SA, після фази прийняття SA включено процедуру покращення на основі популяції, що дозволяє алгоритму використовувати переваги соціальної поведінки деяких розв'язків із простору пошуку. Чисельні результати були проведені з використанням відомих екземплярів TSP від TSPLIB, і попередні результати показують, що запропонований алгоритм перевершує інші алгоритми порівняння з точки зору якості розв'язків.

Ключові слова: *імітований відпал; задача комівояжера; метаевристики.*