UDC 338.33
DOI https://doi.org/10.24144/2616-7700.2020.1(36).105-111

Y. Selyutin[1], I. Kozin[2]

[1] Zaporizhzhya National University, Zaporizhzhya,
Post-graduate student
wizz92@gmail.com
ORCID: https://orcid.org/0000-0002-5458-9880

[2] Zaporizhzhya National University, Zaporizhzhya,
The professor of the Economic Cybernetics Department,
Doctor of Physics and Math Science
ainc00@gmail.com
ORCID: https://orcid.org/0000-0003-1278-8520

COMPARATIVE EFFECTIVENESS OF METAHEURISTIC METHODS

The essence of metaheuristic methods and conditions of their application are considered, in particular, limited amount of knowledge and availability of some candidate for optimality. The formal statement of the traveling salesman problem and its solution are presented with 4 algorithms: genetic, annealing, Lin-Kernigan and the method of jumping frogs.

The advantages and disadvantages of the annealing algorithm are analyzed. A parallel is drawn between the annealing algorithm and the gradient methods. The set tasks of the salesman in the parameters of the genetic algorithm. The principles of operation of the jumping frog method and the Lin-Kernigan algorithm are given.

To perform the experiment, a database of random data was generated that formed a 1000×1000 dimension problem with a known exact solution. For the conclusions on the effectiveness of the methods, the rate of convergence of the task was estimated with the maximum approximation to the global extremum and the standard deviation from the exact solution. The genetic algorithm was found to perform best under the given conditions. The further application of the jumping frog algorithm for optimization problems, implemented with a large number of iterations, is promising. One of the ways of using the jumping frog algorithm is the task of placing production.

Keywords: discrete optimization, metaheuristics, genetic algorithm, jumping frog algorithm, leap frog method, annealing method, Lin-Kernighan algorithm.

1. Introduction. The essence of metaheuristic methods is often difficult to understand precisely because of the specifics of the term itself. Metaheurism is not a heuristic for heuristics as it might seem due to the meta part, it is in some way a black box optimization. Black box optimization is also close to the term "stochastic optimization". Algorithms use some degree of randomness when searching for the most optimal solutions to complex problems.

In order for the metaheuristics method to be suitable for solving the problem, several conditions must be met:

– knowledge about the task itself is very limited – it is not known in advance what the optimal solution will look like, there is no specific solution method, there is too little information to use heuristics;

– there is a certain "candidate for optimality" – a suitable option for solving the problem.

In this case, the simple enumeration method does not work due to obvious shortcomings: cumbersomeness, significant time and resources. Gradient methods are not

suitable due to similar reasons, in addition, they do not have the right to be applied if it is a discrete problem.

2. The results of the study. There are many different types of metaheuristics: combinatorial optimization algorithms (ant colony), parallel (island model), multi-criteria and some others. However, all of them have the above-described properties, according to which they are referred to as metaheuristics.

Let us compare the effectiveness of 4 methods: the classical genetic algorithm [1], the jumping (leap) frog method [1], the annealing method [1] and Lin-Kernighan algorithm [2].

One of the one-state metaheuristic methods is also simulated **annealing**, which also belongs to the group of Monte Carlo methods. The algorithm is based on a simulation of a physical process that occurs during crystallization of a substance, including during annealing of metals. The transition of an atom from one cell to another occurs with some probability, and the probability decreases with decreasing temperature.

The current state of the optimized system is modified according to the normal law. The algorithm may remain at the current point, or may go into a new state. The probability of a transition to a new state depends on the temperature and the energy difference, that is, the value of the optimized function, in the current and possible new state, and is calculated in accordance with some distribution, for example, with the Gibbs distribution.

The Gibbs distribution is used classically, while the application of the Boltzmann distribution gives the method a new name – "Boltzmann annealing".

The simulated annealing algorithm is similar to gradient descent, but due to the randomness of the choice of an intermediate point, it will have to fall into local minima less often than gradient descent. Using the logarithmic law of lowering the temperature, a global minimum can be guaranteed, with a probability tending to unity, but in practice this requires too many iterations, so this approach is not used directly.

Imagine a formal statement of the problem and its solution.

1. Choose the initial solution $i_0 \hat{\mathrm{I}} I$ and put $f^* := f(i_0)$, $k := 0$.

2. Until the stopping criterion is met, do the following:

2.1. Randomly select $j < N$ (i_k).

2.2. If $f(j) - f(i_k) < t_k$ then $i_{k+1} := j$.

2.3. If $f^* > f(i_k)$, then $f^* := f(i_k)$.

2.4. Put $k := k+1$.

The following quantity is also introduced t_k, k = 0, 1, 2,... – random variable with mathematical expectation $E(t_k) = c_k > 0$ – local search option when arbitrary deterioration in the objective function is allowed, but the probability such a transition is inversely proportional to the magnitude of the deterioration, more precisely, for any $j \in N(i)$

$$P_{ij} = \begin{cases} 1, \ if \ f(j) \le f(i), \\ \exp\left(\frac{f(i)-f(j)}{c_k}\right), \ if \ f(j) \ge f(i). \end{cases}$$

The sequence $\{c_k\}$ plays an important role in the analysis. Sometimes the parameter c_k is called the temperature.

There are many heuristic ways to select the final sequence $\{c_k\}$ in order to increase the probability of detecting a global optimum.

1. The initial value: $c_0 = D\, f_{max}$ – the maximum difference between two adjacent solutions.

2. Lowering the threshold: $c_{k+1} = a\ c_k$, $k = 0, 1, \ldots, K-1$, where a is a positive constant, quite close to 1, for example, $a \in [0.8; 0.99]$.

3. Final value: $c_K > 0$ is determined either by the number of changes made, or as the maximum c_k, at which the algorithm does not change the current solution within a given number of steps. For each value of c_k, the algorithm performs the order $N(i)$ steps without changing the threshold value.

For example, the statement of the traveling salesman problem for a simulation algorithm looks like as follows.

Each city is represented as a pair of coordinates with a corresponding index $(x_i;\ y_i)$.

By condition, the solution is the route between all cities, which means that the set of states S is all possible routes passing through each city. In other words, the set of all ordered sequences of elements of C, in which every city occurs exactly once. Obviously, the length of each such sequence $|C|$.

As we recall, in order to use the simulated annealing method, we must define two functions that depend on each specific task. This is the energy function E (or the "objective function" in conventional terminology) and the function F, generating a new state.

Since we strive to minimize the distance, it will be "energy". Therefore, our objective function will look like this:

$$E(s_i) = \ E_i = \ \left[\sum_{1}^{|C|-1} \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}\right] + \sqrt{(x_{|C|} - x_1)^2 + (y_{|C|} - y_1)^2}$$

Consider a **genetic algorithm** – this is a heuristic search algorithm used to solve optimization and modeling problems by sequentially selecting, combining and varying the desired parameters using mechanisms reminiscent of biological evolution. It is a type of evolutionary computation. A distinctive feature of the genetic algorithm is the emphasis on the use of the "crossing" operator, which performs the operation of recombination of candidate solutions, whose role is similar to the role of crossing in wildlife.

A simple and pure genetic algorithm can be defined in the following steps.

Step 1. Create an initial population of P chromosomes.

Step 2. Evaluate the fitness of each chromosome.

Step 3. Choose P/2 parents from the current population via proportional selection.

Step 4. Randomly select two parents to create offspring using crossover operator.

Step 5. Apply mutation operators for minor changes in the results.

Step 6. Repeat Steps 4 and 5 until all parents are selected and mated.

Step 7. Replace old population of chromosomes with new one.

Step 8. Evaluate the fitness of each chromosome in the new population.

Step 9. Terminate if the number of generations meets some upper bound; otherwise go to Step 3.

Let's condifer also **the Lin-Kernighan algorithm.**

The Lin-Kernigan as 2-opt algorithm is a special case of the λ-opt algorithm, where in each step λ links of the current tour are replaced by λ links in such a way that a shorter tour is achieved. In other words, in each step a shorter tour is obtained by deleting λ links and putting the resulting paths together in a new way, possibly reversing one or more of them.

The λ-opt algorithm is based on the concept λ-optimality. A tour is said to be λ-optimal (or simply λ-opt) if it is impossible to obtain a shorter tour by replacing any λ of its links by any other set of λ links.

From this definition it is obvious that any λ-optimal tour is also λ-optimal for $1 \leq \lambda' \leq \lambda$. It is also easy to see that a tour containing n cities is optimal if and only if it is n-optimal. In general, the larger the value of λ, the more likely it is that the final tour is optimal. For fairly large λ it appears, at least intuitively, that a λ-optimal tour should be optimal. Unfortunately, the number of operations to test all 1-exchanges increases rapidly as the number of cities increases. In a naive implementation the testing of a λ-exchange has a time complexity of $O(n^{\lambda})$. Furthermore, there is no nontrivial upper bound of the number of λ-exchanges. As a result, the values $\lambda = 2$ and $\lambda = 3$ are the most commonly used. In one study the values $\lambda = 4$ and $\lambda = 5$ were used. However, it is a drawback that λ must be specified in advance. It is difficult to know what λ to use to achieve the best compromise between running time and quality of solution. Lin and Kernighan removed this drawback by introducing a powerful variable λ-opt algorithm. The algorithm changes the value of λ during its execution, deciding at each iteration what the value of λ should be. At each iteration step the algorithm examines, for ascending values of λ, whether an interchange of λ links may result in a shorter tour. Given that the exchange of r links is being considered, a series of tests is performed to determine whether r+1 link exchanges should be considered. This continues until some stopping conditions are satisfied. At each step the algorithm considers a growing set of potential exchanges (starting with r = 2). These exchanges are chosen in such a way that a feasible tour may be formed at any stage of the process. If the exploration succeeds in finding a new shorter tour, then the actual tour is replaced with the new tour.

Consider the **jumping (leap) frog method**.

Its name is due to the social behavior of a group of frogs. In this process, unlike evolutionary algorithms, genetic operators are not used, and the interaction of the solutions found is considered taking into account the success of their neighbors in the search space. The statement of the traveling salesman problem for solving it by the jumping frog method, as well as a comparison of the effectiveness of this algorithm with other methods, is described in [1] (Skobtsov, 2008).

The detailed steps of jumping frog method are as follows.

Step 1 (swarm generation): Let x_i denotes i-th frog's position and f_i is its fitness. A frogs population $X = \{x_i, f_i, i = 1, \ldots, F\}$ is initialized with position within the searching space and sorted in descending order of fitness values.

Step 2 (memeplexes partition): Partition the population into m memeplexes $\{Y_1, Y_2, \ldots, Y_m\}$, each contains n frogs and

$$Y_i = \left[(x_j, f_j)|x_j = x_{i+m(j-1)}, f_j = f_{i+m(j-1)}, j = 1, \ldots, n\right].$$

Step 3 (submemeplexs generation): The selection strategy of a submemeplex

(including q frogs) in each memeplex is that the larger coefficients are distributed to the frogs with better positions. A selection method is that the frogs with better positions have bigger weights to be assigned to the submemeplex. The weights are assigned with a triangular probability distribution as follows

$$p_i = 2(n+1-i)/(n+1), \quad i = 1, \ldots, n.$$

Step 4 (submemeplexs evolution): Let x_B denotes the best position and x_W denotes the worst position in submemeplex. Then, local search is started from the worst frog to leap to the best frog in any memeplex. The worst frog in submemeplex leaps towards to the best frog in the memeplex, and thus the new position is obtained by a leaping step. A random position is generated to replace it, that is

$$x_q^{k+1} = a + Int[r^k(b-a)].$$

where [a,b] is the boundary of frogs' feasible location. Afterwards, the frogs are sorted in a descending order according to their fitness. Repeat above steps and evolve the submemeplexes with G_1 generations.

Step 5 (memeplexs shuffle): After the local search of each memeplex is finished, all memeplexs are shuffled in which the frogs are reorganized in descending order of fitness. Repeatedly divide the population into memeplexs and carry out local search process, until memetic evolution generation G_2 is reached.

The following subsections show the general information about different metaheuristic methods, and their steps. For numerical experiments we used a single randomly generated task with well-known exact solution and the dimension of the 1000×1000 problem. For each of the 4 algorithms 50 trials were performed with different initial solutions. Several tests of the algorithm were carried out with 10, 15, 30 iterations. The result table (Table 1) included averages. Let's compare their effectiveness.

Table 1. The effectiveness of metaheuristic methods for travel salesman problem

Metaheuristic Method	The number of tests in which the algorithm found a global extremum	The number of iterations when the global extremum was found	Average algorithm running time	Standard deviation (%)
Annealing algorithm	93%	15, 30	5 minutes	2,5
Genetic algorithm	95%	10, 15, 30	4,5 minutes	1,3
Lin-Kernighan algorithm	90%	15,30	6,3 minutes	1,7
Jumping (leap) frog method	80%	15, 30	6,2 minutes	1,8

Also compare several tests of the jumping frog algorithm that were carried out with different numbers of iterations: 10, 15, 30 (Fig. 1).

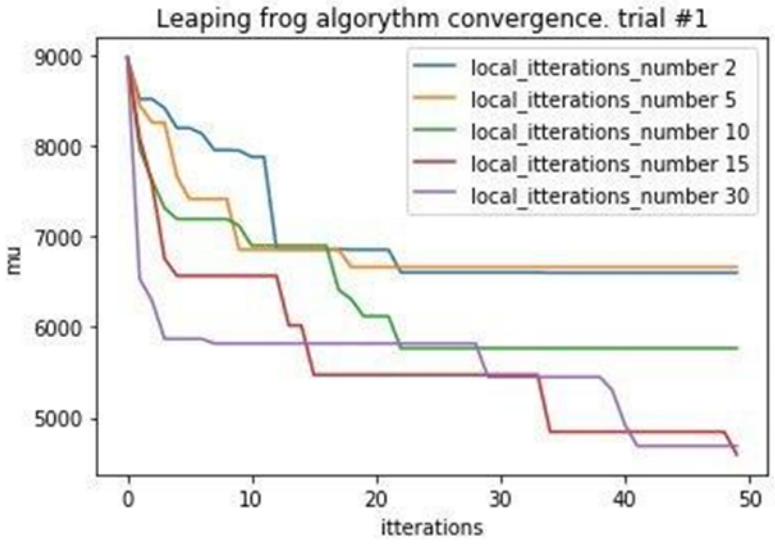

Fig. 1. Comparison of the efficiency of the jumping frog algorithm with a different number of iterations according to the convergence criterion

The best algorithm is considered, implemented with 15 iterations of local search. With an increase in the number of iterations, in most cases the algorithm wins with their maximum number (Fig. 2).

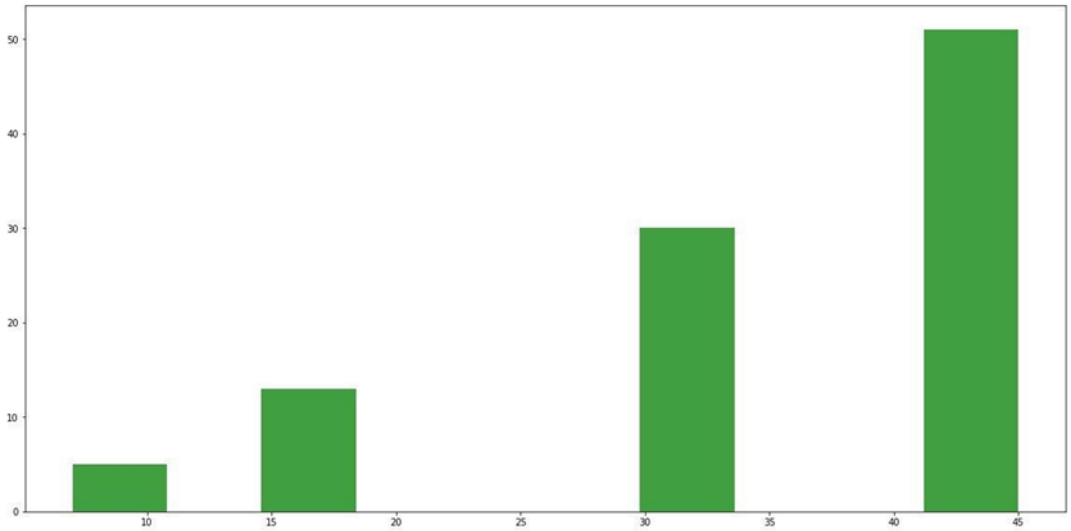

Fig. 2. Comparison of the efficiency of the jumping frog algorithm with a different number of iterations according to the search criterion for the global maximum

Analysis Fig. 2 makes it possible to conclude that its principle of operation of the jumping frog method is outwardly similar to the random search algorithm, which emphasizes its belonging to the category of metaheuristics methods.

3. Conclusions. Promising is the further application of a simple hybrid algorithm based on a combination of the jumping frog method and a fragmented algorithm. It is also necessary to investigate methods for evaluating the effectiveness of various metaheuristic algorithms.

References

1. Skobtsov, Y. (2008). Fundamentals of evolutionary computing: textbook. benefits Donetsk: DonNTU. [in Russian].
2. Lin, S., & Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research. Vol. 21, No. 2.* DOI: https://doi.org/10.1287/opre.21.2.498
3. Eusuff, M.M. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. *J. Water Resour. Planning Mgmt. Vol. 129.*, 210 – 225. DOI: https://doi.org/10.1061/(ASCE)0733-9496(2003)129:3(210)
4. Narimani, M.R. (2011). A New Modified Shuffle Frog Leaping Algorithm for NonSmooth Economic Dispath. *World Applied Sciences Journal*, 803–814.
5. Khumawala, B.M. (1972). An Efficient Branch-Bound Algorithm for the Warehouse Location Problem. *Management Science. v18.*, 718-731. DOI: https://doi.org/10.1287/mnsc.18.12.B718
6. Krarup, J., & Pruzan, P.M. (1983). The simple plant location problem: Survey and synthesis. *European Journal of Operational Research. v12*, 36-81. DOI: https://doi.org/10.1016/0377-2217(83)90181-9

Сєлютін, Є., Козін, І. Порівняльна ефективність використання метаевристичних методів.

Розглянуто суть метаевристичних методів та умови їх застосування, зокрема, обмежена кількість знань і наявність деякого кандидата на оптимальність. Наведена формальна постановка задачі комівояжера і її рішення 4 алгоритмами: генетичним, відпалу, Лін-Кернігана і методом стрибаючих жаб.

Проаналізовано переваги та недоліки алгоритму відпалу. Проведена паралель між алгоритмом відпалу і градієнтними методами. Задані змінні завдання комівояжера в параметрах генетичного алгоритму. Наведено принципи дії методу стрибаючих жаб та алгоритму Лін-Кернігана.

Для проведення експерименту була згенерована база випадкових даних, які утворили задачу розмірності 1000×1000 з наперед відомим точним розв'язком. Для висновків по результативності методів були оцінені швидкість збіжності завдання за умови максимального наближення до глобального екстремуму і середньоквадратичне відхилення від точного розв'язку. Виявлено, що генетичний алгоритм за заданих умов демонструє найкращі результати. Перспективним є подальше застосування алгоритму стрибаючих жаб для задач оптимізації, реалізоване з великим числом ітерацій. Одним з напрямків використання алгоритму стрибаючих жаб є завдання розміщення виробництва.

Ключові слова: дискретна оптимізація, метаевристика, генетичний алгоритм, алгоритм стрибаючих жаб, метод відпалу, алгоритм Лін-Кернігана.

Список використаної літератури

1. Скобцов Ю.А., Федоров Е.Е. Основы эволюционных вычислений. Донецк: Изд-во «Ноулинджж». 2013. 426 с.
2. Lin S., Kernighan B. W. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*. 1973. Vol. 21, No. 2. DOI: https://doi.org/10.1287/opre.21.2.498
3. Eusuff M.M. Optimization of water distribution network design using the shuffled frog leaping algorithm. *J. Water Resour. Planning Mgmt.* 2003. Vol. 129. P. 210 – 225. DOI: https://doi.org/10.1061/(ASCE)0733-9496(2003)129:3(210)
4. Narimani M.R. A New Modified Shuffle Frog Leaping Algorithm for NonSmooth Economic Dispath. *World Applied Sciences Journal*. 2011. P. 803–814
5. Khumawala B.M. An Efficient Branch-Bound Algorithm for the Warehouse Location Problem. *Management Science*. 1972. v18, P. 718-731. DOI: https://doi.org/10.1287/mnsc.18.12.B718
6. Krarup J., Pruzan P.M. The simple plant location problem: Survey and synthesis. *European Journal of Operational Research*. 1983. v12. P. 36-81. DOI: https://doi.org/10.1016/0377-2217(83)90181-9.

Recived 03.02.2020