УДК 004.023; 004.94

O. Muliarevych, V. Golembo
National university "Lviv Polytechnic"
Department of Computer Engineering

THE IMPLEMENTATION OF LOCAL OPTIMIZATION METHODS IN COMPUTER SYSTEM FOR SOLVING DYNAMIC TRAVELLING SALESMAN PROBLEM USING SWARM INTELLIGENCE BEHAVIOR MODEL OF AGENTS

This paper is devoted to the solving one of the combinatorial optimization task – the Dynamic Travelling Salesman Problem (DTSP) by using computer system based on swarm behavior model of collective agents and benefits of local optimization methods usage.

Keywords - agents, DTSP, ant colony method, local optimization methods, swarm behavior.

Abstract

Nowadays, very popular become investigations of collectives of autonomous agents for possible practical applications expanding. Experiments are especially focused on combinatory optimization tasks, including well-known Travelling Salesman Problem (TSP) [1]. This task is formulated next way: given a list of cities and the distances between each pair of cities (instead of distance can be cost, time), find out the shortest possible route that visits each city exactly once and returns to the origin (start point) city. It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

Created computer system [2, 3] using swarm intelligence behavior model of agents is able to solve both type TSP for acceptable on practice time of calculation: static (without changes of input data during calculation process) and dynamic (in conditions of dynamic changes of input data). But received result routes of TSP frequent till 10% from value of optimal route. With aim to improve results without losing speed and ability to solve in conditions of dynamic changes of input data was decided to overview possibility of local methods usage as additional program module of system, which take as input data received formed TSP route for further improvement.

Approaches to solving combinatorial optimization tasks

Using developed classification of methods for solving TSP [4] during process of analyze of existing methods was chosen the most perspective for solving dynamic TSP methods with usage of agents collective. Dynamic TSP is the closest task to reality, especially during usage for solving task in logistic for transport and informational networks.

Methods with usage of agents collective include next swarm intelligence methods: social insects algorithms (Artificial Bee Colony algorithms, Ant Colony Optimization algorithms [5,6]), Particle Swarm Optimization algorithm, Bacterial Foraging Optimization algorithm, Artificial fish swarm algorithm, Migrating Birds Optimization algorithm [7] and others [5,8]. Ant Colony Optimization algorithm demonstrate one of the best results of solving dynamic TSP. It is important to mention that this method is able to solve TSP without restarting of calculation cause of constructive type. Ant Colony Optimization algorithm also can solve task in conditions of partly unknown input data.

Separate positions have methods of local search [9], or also known as – local optimization methods. These methods make able to improve result – decrease length of received TSP route. Local optimization methods differ in complexity coefficient k, where k – count of edges, which take part in permutations. The most popular are methods with low complexity (2-opt, 2.5-opt, 3-opt algorithms) and local optimization method with dynamically changeable complexity coefficient k [10] (Fig. 1).

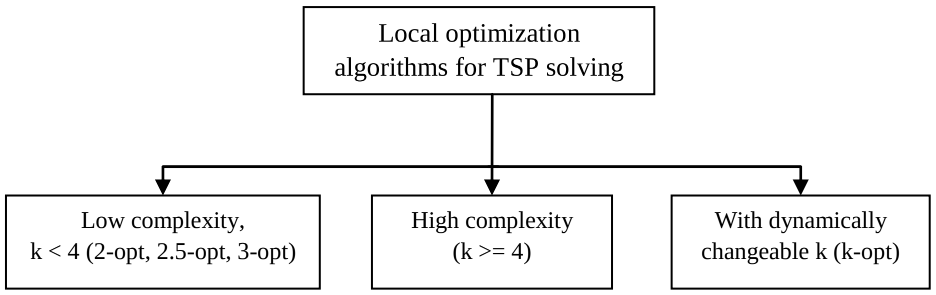

Local optimization algorithms for TSP solving
Low complexity, k < 4 (2-opt, 2.5-opt, 3-opt)
High complexity (k >= 4)
With dynamically changeable k (k-opt)

Fig.1. Classification of local optimization algorithms for TSP solving

In classical realization of 2-opt algorithm 2 edges are excluded from route and added two new in such way that new route become shorter. There is only one variant of exchange of two edges on new ones, during which route will left instead of two cycle creation.

Same way are used others local optimization algorithms with difference only in count of edges which take part in combinatorial permutations, 3 edges in 3-opt algorithm, k edges in mostly efficient and universal k-opt algorithm with dynamically changeable coefficient k, which was developed as modification of local optimization algorithms and well-known as Lin-Kernighan method [9]. In this method value k is dynamic and changes during calculation process of result route. Helsgaun K. has developed LKH application [9] for TSP solving using Lin-Kernighan and decomposition methods.

Features of the application of local optimization methods in developed system

Local optimization methods [9] belongs to metaheuristic methods used for solving complex in calculations optimization tasks. Local optimization methods can be used for tasks, which are formulated as result search that is maximal by some criteria in array of possible results. Local optimization algorithms check all possible solutions by method of local changes while optimal result will not be found or time limit or try count limit will not be reached. The most popular for solving TSP is k-opt algorithm [9, 11].

Most tasks can be formulated in terms of “space” and “aim” several ways. For example, for TSP solution can be Hamilton loop, and criteria of maximization will be combination of some points count and loop length. But solution from other side can be also route, and create loop will be one more part of task for local search. This approach is used in program application LKH during “union” of separate routes. Local optimizations algorithms start to analyze from some input solution or its part and sequentially go to neighbor one. It is possible only when neighborhood relationships between solutions or its parts are defined in space of search. As example, if in created route – TSP solution exchange placement of just one point, we will get new route. Start route and retrieved after point replacement route will be neighbors. For TSP solution count of results is defined by count of points. Count of permutations increase according complexity of local optimization method – k, which define edge count that will be replaced and also count of neighbor solutions which will take part in analyze.

Condition of algorithm execution shut down usually set as time limit or count of local optimization attempts. Local optimization algorithms can be stopped in any time of execution, but are able to work only without presence of dynamic changes during their calculation process. Moreover, when calculation is finished local optimization algorithms don’t guarantee the most optimal result in case of impossibility of local improvements. These cases have place when complexity of local optimization method isn’t enough for founding better neighbor solution.

Lin-Kernighan algorithm on practice is very efficient, but in some case reaches exponential complexity and time on calculation of result. Feature of Lin-Kernighan algorithm is that it works with changeable k, which value is redefined during iteration of local search cycle.

During setting of neighbor list for point local optimization method decrease complexity from $O(n^2)$ to $O(m*n)$, but if m value will be not enough big local optimization algorithm efficient will decrease. Most often are used algorithm with low complexity: 2-opt, 2.5-opt, 3-opt algorithms. According Lin S. [9], 3-opt algorithm demonstrated the highest results in most optimal calculation time.

So, local optimization methods with k < 4 can highly improve accuracy of results, which are retrieved from developed computer system for dynamic TSP solving, based on ant colony optimization method including developed modifications, without growth of calculation time. Analyze of optimization possibility to improve route segment don't cancel possibility TSP solving with dynamic changes of input data if time for analyze process is less then period of dynamic changes.

Program realization of local optimization methods

Local optimization methods usage increase accuracy of received result with help of edges replacements in found route. For implementation of local optimization methods is necessary to develop program realization cause of not high complexity and time of calculation for next 3 algorithms: 2-opt, 2.5-opt, 3-opt. The way and details of implementation for TSP solving is absent in open informational sources. Cause of that was decided to describe all used permutation which are used during selected local optimization algorithms. This methods and described schemas were implemented in developed computer system as additional program module.

Let's describe on example of randomly chosen part of graph with already set edges between points according retrieved route. Points which take part in checks for permutations are defined as s1 and s2 (station). According route, previous to s1 point marked as p_s1, next after s1 – s_s1 (sequent after s1).

Same as for point s1 are defined for point s2: p_s2 and s_s2. If current point s1 yet was not optimized, start process of search s2 point in such way, that distance between s1 and s2 was smaller than distance between s1 and s_s1, which is named on schema R – radius on fig.2.

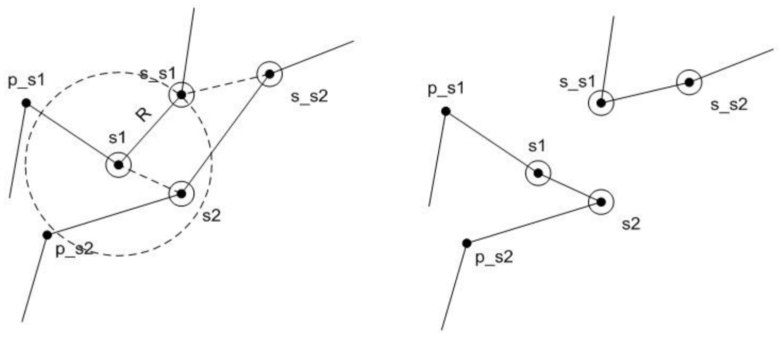


Fig.2. Execution of 1-st type permutation of 2-opt algorithm on the randomly chosen graph area.

After which is running check for expediency of permutation: excluding of s1-s_s1 and s2-s_s2 edges and adding s1-s2 and s_s1-s_s2 edges. Next time we will call this permutation as 2-opt permutation of 1-st type: permutation of next going points by route according current s1 and found s2 points. Execution of this permutation is demonstrated on figure 2. Here and further at left side is shown initial state, at right side – state of route after permutation. Defined points with circles around take part in permutations.

If current permutation (fig. 2) will decrease cost of instant route, than will be activated switch to the permutation execution – correction of input route, current point will be marked as optimized and algorithm will go to the next one. If current permutation isn't suitable, than start search of point s2 (fig. 3) in such way that distance between p_s1 and p_s2 will be smaller than distance between p_s1 and s1, which is marked on schema as R – radius on fig.3. After that will be checked permutation with excluding of edges p_s1-s1 and p_s2-s2, adding edges p_s1-p_s2 and s1-s2. This permutation will call as 2-nd type permutation of 2-opt algorithm, in which take part previous by route points according current s1 and some found s2 point. Such type of permutation for randomly chosen graph area was demonstrated in Fig.3.

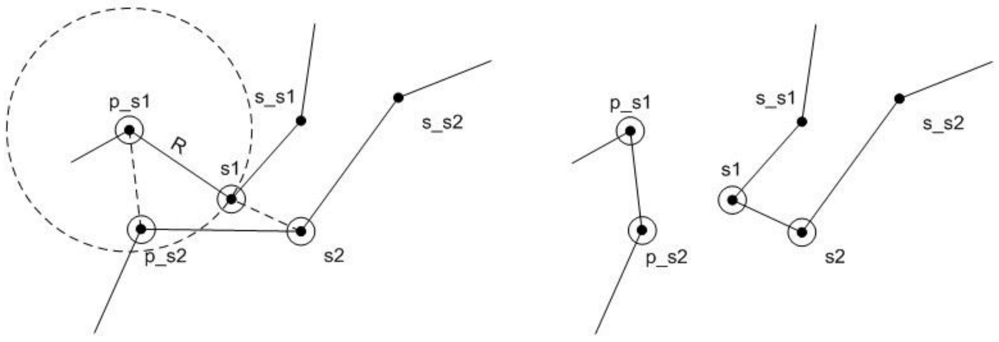


Fig.3. Execution of 2-nd type permutation of 2-opt algorithm on the randomly chosen graph area.

If 2-nd type permutation of 2-opt algorithm will decrease cost of instant route similarly as with 1-st type permutation will be activated switch to the permutation execution – route correction and continue local optimization cycle.

In fact, this algorithm if necessary can be stopped in any time. Similar to observed 2-opt algorithm implemented 2.5-opt algorithm, which include some stages of 2-opt algorithm and some elements of 3-opt algorithm as three edges take part in process of excluding point from one place and inserting into other place of route.

Firstly attempts of 2-opt optimization permutations of 1-st and 2-nd type are checked same as during 2-opt algorithm execution. After that starting analyze for expediency of insertion s2 point, from pool of neighbor point according to instant, between s1 and s_s1 points –will be called such insertion as 1-st type insertion of 2.5-opt algorithm. Such type of insertion for randomly chosen area of TSP result route is demonstrated in Fig.4.

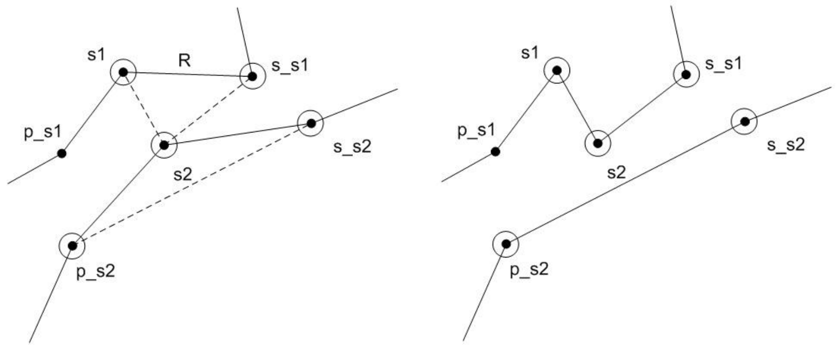


Fig.4. Execution of 1-st type insertion of 2.5-opt algorithm on the randomly chosen graph area.

If such type of insertion will not decrease cost of route, then will be checked expediency of insertion in route some s2 point between p_s1 and s1 points. Such type of insertion will be called as 2-nd type insertion of 2.5-opt algorithm, which is shown in Fig.5. Local optimization method with k=3 is similar to two observed methods. Stage of optimization is more complex than in 2-opt and 2.5-opt algorithms, where only 2 and 4 checks have place.

At 3-opt algorithm firstly two 2-opt checks are executed (Fig.6), if they improve result route appears possibility to end optimization or save result and continue analyze for next possible permutations (Fig.7).

First type permutation of 2-opt algorithm according to s1 point shown in Fig.6.a and 2-nd type permutation according s_s1 point shown in Fig.6.b. As in previous pictures, points, which take part in edge permutations, are marked around with circles. With dotted line are represented edges, which are created after permutation.

In one case is excluded s2-s_s2 edge, in another – p_s2-s2 edge, s1-s_s1 edge is excluded in both cases.

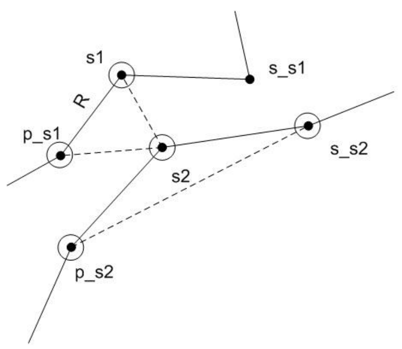
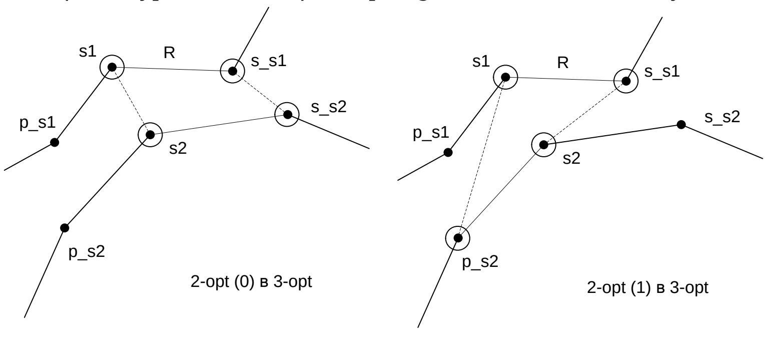
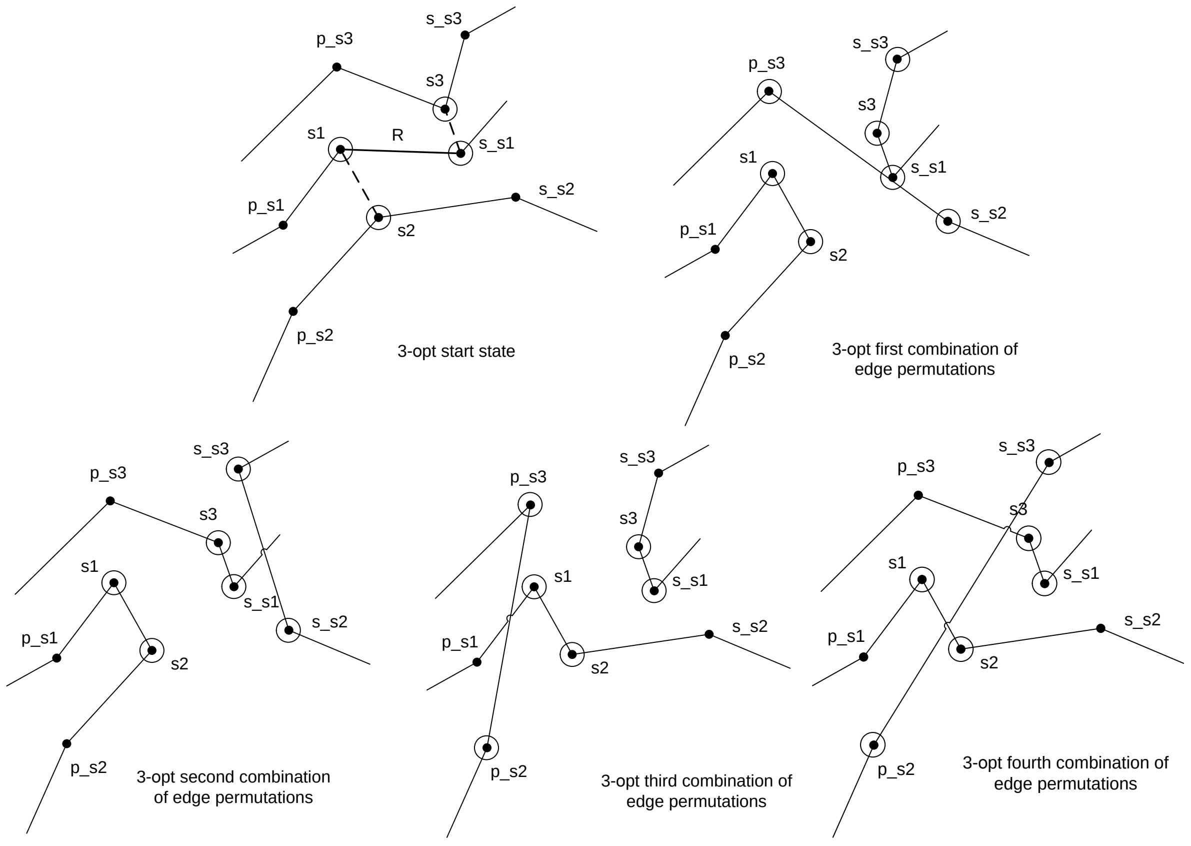


Fig.5. Execution of 2-nd type insertion of 2.5-opt algorithm on the randomly chosen graph area.



(a) (b)

Fig.6. Execution of 2-opt algorithm permutations during 3-opt algorithm on the randomly chosen graph area.



3-opt start state
3-opt first combination of edge permutations
3-opt second combination of edge permutations
3-opt third combination of edge permutations
3-opt fourth combination of edge permutations
s1
s_s1
s2
s_s2
s3
s_s3
p_s1
p_s2
p_s3
R

Fig.7. Permutation combinations in 3-opt algorithm on the randomly chosen graph area.

After which start further check for expediency of permutations of two edges with found during 2-opt check s2 point and currently found s3 point, which is neighbor according s_s1 point. Should be noted, that s2 and s3 points are chosen in such way that s3-s_s1 and s1-s2 edges will be smaller than s1-s_s1 edge, which is marked as R – radius.

If a permutation with excluding of s1-s_s1 edge, adding s1-s2 and s_s1-s3 edges is considered to be expediency, then start the search process for best permutation combination, which will reach the highest decrease of route length. In developed implementation of 3-opt algorithm in analyze take part four variants of edge permutation combinations, shown in Fig.7. Edge s1-s_s1 is excluded in each of these combinations.

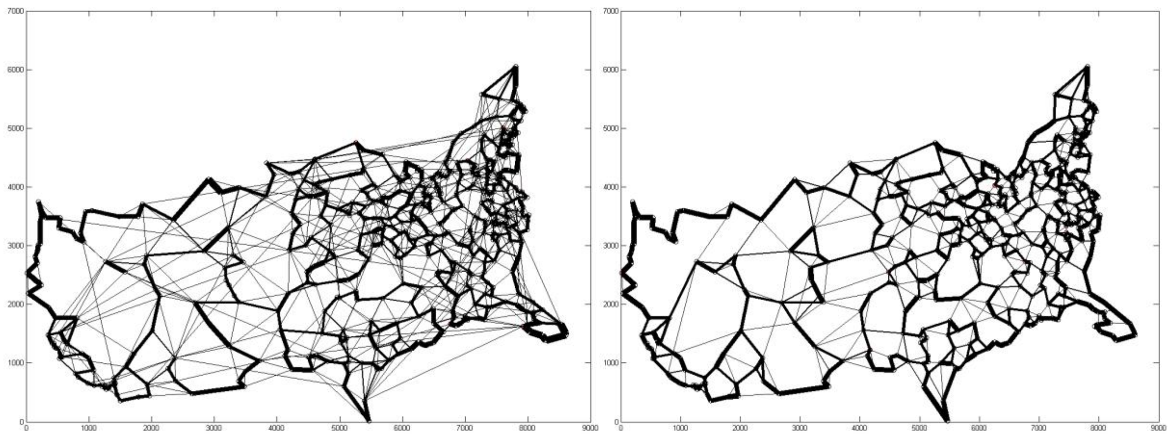
After check of each combination, length of possible improved route is stored for further summary decision and choosing most optimal combination according this criteria and its execution.

If after all check and tries of corrections improvement of result route was not reached, then current s1 point is added to list of optimized and algorithm go to next by random sequence not optimized yet point of TSP route. According carried out investigations, was decided to use local optimization methods for improvement of all found by agents routes.

Research and evaluation of implementation results of additional program module based on local optimization methods in developed computer system

For research and evaluation of implementation results of additional program module based on local optimization methods in developed computer system was used library file att532.tsp [12], which include all input data for TSP on 532 points (in current case coordinates of USA cities). During process of TSP resolving with usage of ant colony optimization method appears phenomenon of so called “ant colony memory” – network of pheromones, left marks, on which ant-agents are oriented in process of route search. Just because of existence of such phenomenon agents can easier adapt to dynamic changes of beginning input data and conditions, during process of TSP solving. Overview of such “ant colony memory” for TSP on 532 points is shown in Fig.8. Width of edge is as big as bigger value of digital mark (pheromone at biological ants) placed on it.

In Fig.8 is shown state of marks after execution of 10 iteration of route search cycle. In Fif.8.a is demonstrated state of marks without usage of local optimization methods, in Fig.8.b – with usage of 3-opt local optimization algorithm. As we can see, usage of local optimization methods boost process of disappearance of additional edges, which excluded from usage of agents as not suitable, cause of periodical decreasing of mark values on it and no update process – passing through it by agents.



(a) (b)

Fig.8. Overview of collective agent’s memory (“ant colony memory”) – matrix of marks in process of solving TSP (532 points): a) without usage of local optimization methods;
b) with usage of 3-opt local optimization algorithm.

In Table is represented calculation process of TSP result during first 3 iterations of search cycle by agent collective: 1) without usage of local optimization methods; 2) after each iteration of search routes cycle by agents with usage of additional one of the implemented local optimization methods: 2-opt, 2.5-opt, 3-opt. In table also is shown average calculation time spent for TSP solving in observed functional modes of developed computer system.

Table

Calculation process of TSP 3K (532 points)

	Length of route				Count of improvements \ permutations			Average calculation time, s			
I	without usage of local optimization methods	with usage of local optimization methods:			2-opt	2,5-opt	3-opt	without usage of local optimization methods	with usage of local optimization methods:		
		2-opt	2.5-opt	3opt					2-opt	2.5-opt	3opt
1	49787	30105	29387	28780	428	1239	614	0.9	4.4	4.6	4.95
2	39067	29806	29327	28675	202	1012	625				
3	38939	29665	29202	28375	224	797	625				

Optimal route for current TSP have length 27686 (according data from library file). Route of such length was found by developed modeling system in mode with usage of 3-opt local optimization method. As we can see, usage of each local optimization method according its complexity increase calculation time of result route, but highly decrease iteration count of search routes cycle by agents and increase accuracy of retrieved result. In Fig. 9 graphically are represented TSP solutions for 532 points retrieved in four functional modes of developed system after 30 iterations of "ant run": without usage of local optimization methods (Fig.9.a); with usage of 2-opt local optimization method (Fig.9.b); with usage of 2.5-opt local optimization method (Fig.9.c); with usage of 3-opt local optimization method (Fig.9.d).

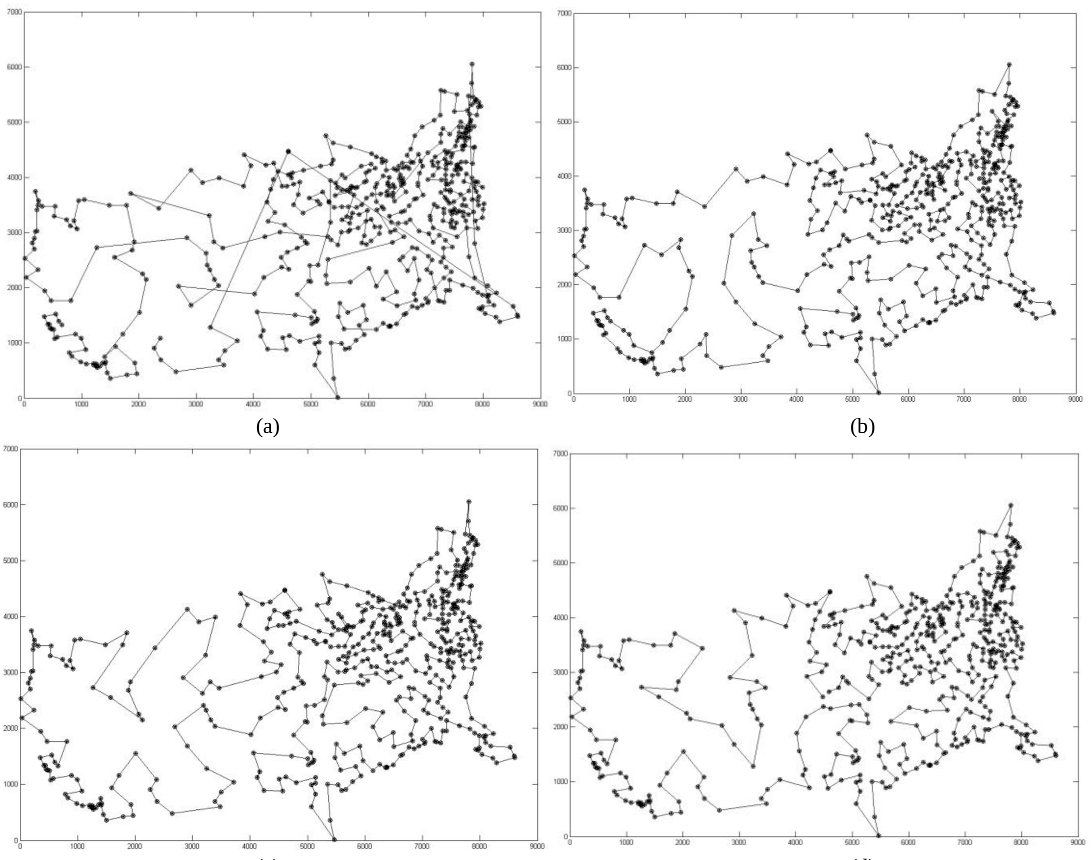


Fig.9. Result TSP (532 points) routes.

Length of result route, according table and pictures, decrease in proportion to increase of complexity of used local optimization method. Developed computer system based on swarm intelligence behavior model of agents, even with usage of local optimization methods, is able to solve TSP of 532 points in time less than program application LKH, which need in average nearly 7 seconds for TSP solving.

Developed system is able to output quasi-optimal result route with difference from optimal less than 6%, spent nearly 1 second for calculation of it. Constructive character of used ant colony optimization method make possible retrieving of route even after first iteration of search route cycle, that means after several milliseconds from start, after which this result is checked in additional program module based on local optimization methods with aim to decrease length of retrieved route.

Conclusions

Implementation of local optimization methods into developed computer system for solving TSP with usage of swarm behavior model of agents make possible to decrease difference between quasi-optimal and optimal routes and decrease iterations count of search cycle of route by agents. In such way, to solve used in experiments TSP of 532 points modeling system in conditions of dynamic changes of input data after 1 second is able to output result route with difference from optimal less than 6%. And without existed dynamic changes of input data, already after 5 seconds can output optimal result using 3-opt local optimization method.

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ, 2-ое издание. - Вильямс, 2005. 2. Голембо В.А., Муляревич О.В. Модифікація методу мурашиної колонії для розв'язання задачі комівояжера колективом автономних агентів // Вісник Національного університету "Львівська політехніка". – 2011. – №717: Комп'ютерні системи та мережі. – 24 - 30 с. 3. Голембо В.А., Бочкарьов О.Ю., Муляревич О.В. Нові підходи до розв'язку задач комбінаторної оптимізації колективом автономних агентів // Матеріали 5-ої Міжнародної науково-технічної конференції ACSN-2011 "Сучасні комп'ютерні системи та мережі: розробка та використання". — Львів. — 2011. — 227 - 230 с. 4. Муляревич О. Переваги застосування колективної поведінки агентів для розв'язку задачі комівояжера динамічного характеру // Тези доповідей студентської секції «Кібер фізичні системи в метрології» IX Міжнародної науково-технічної конференції «Методи і засоби вимірювань фізичних величин» - «Температура-2012», 25-28 вересня 2012р. – Львів: ПП Сорока Т.Б., 2012. – 165 - 168 с. 5. Bonabeau E., Dorigo M., Theraulaz G. Swarm intelligence: From Natural to Artificial Systems. – Oxford University Press, 1999. 6. Stützle T., López-Ibáñez M., Pellegrini P., Maur M., Oca M., Birattari M., Michael Maur, Dorigo M. "Parameter Adaptation in Ant Colony Optimization" // Technical Report, IRIDIA, Université Libre de Bruxelles, 2010. 7. Vahit Tongur and Erkan Ulker Migrating Birds Optimization for Traveling Salesman Problem // Матеріали 6-ої Міжнародної науково-технічної конференції ACSN-2013 "Сучасні комп'ютерні системи та мережі: розробка та використання". — Львів. — 2013. — 219 - 223 с. 8. Субботін С.О., Олійник А.О., Олійник О.О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: Монографія / Під заг. ред. С.О. Субботіна. – Запоріжжя: ЗНТУ, 2009. 9. Helsgaun K. General k-opt submoves for the Lin-Kernighan TSP heuristic // Mathematical Programming Computation , 2009. – 119 - 163 p. 10. Базилевич Р., Кутельмах Р. Дослідження ефективності існуючих алгоритмів для розв'язання задачі комівояжера. – НУ "Львівська політехніка", Вісник № 638, 2009. – 235 – 244 с. 11. Hoos H.H., Stützle T. Stochastic Local Search: Foundations and Applications. – San Francisco: Morgan Kaufmann Publ., 2005. 12. TSPLIB official homepage. Exist from 1995, © Copyright Universität Heidelberg. URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/