Solve traveling salesman problem using EMF-CE algorithm

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

24-10-2020 / 28-10-2020

CITATION

Luo, Meng; Gu, Shiliang (2020): Solve traveling salesman problem using EMF-CE algorithm. TechRxiv. Preprint. <https://doi.org/10.36227/techrxiv.13139042.v2>

DOI

[10.36227/techrxiv.13139042.v2](https://doi.org/10.36227/techrxiv.13139042.v2)

Solve traveling salesman problem using EMF-CE algorithm

Meng Luo^{a*}, Shiliang Gu^{a*}

^a Jiangsu Provincial Key Laboratory of Crop Genetics and Physiology/Co-Innovation Center for Modern Production Technology of Grain Crops; Key Laboratory of Plant Functional Genomics of Ministry of Education; Wheat Research Center, Yangzhou University, Yangzhou, Jiangsu 225009, China

* Corresponding author.

E-mail addresses: slgu@yzu.edu.cn (Shiliang Gu); czheluo@gmail.com (Meng Luo);

Abstract—In this paper, a novel search algorithm that based on the Contraction-Expansion algorithm and integrated three operators Exchange, Move, and Flip (EMF-CE) is proposed for the traveling salesman problem (TSP). EMF-CE uses a negative exponent function to generate critical value as the feedback regulation of algorithm implementation. Also, combined Exchange Step, Move step with Flip step and constitute of more than twenty combinatorial optimizations of program elements. It has been shown that the integration of local search operators can significantly improve the performance of EMF-CE for TSPs. We test small and medium scale (51-1000 cities) TSPs were taken from the TSPLIB online library. The experimental results show the efficiency of the proposed EMF-CE for addressing TSPs compared to other state-of-the-art algorithms.

Index Terms— Contraction-Expansion algorithm (CE), Exchange-Move-Flip (EMF), combinatorial optimization, traveling salesman problem (TSP)

I .INTRODUCTION

The traveling salesman problem (TSP) is well known for the classical and fundamental NP-hard combinatorial optimization problems. The classical TSP that can be described as following: find a path through a weighted graph that starts and ends in the same city, includes every other city exactly once and minimizes the total distance tour of n cities. The above-described path is always a Hamiltonian cycle or tour. In the mathematical field of graph theory, the Hamiltonian path problem and the Hamiltonian cycle problem are problems of determining whether a Hamiltonian path or a Hamiltonian cycle exists in a given graph (whether directed or undirected). Both problems are NP-complete [1-3]. Special cases of the TSP that included metric TSP, Euclidean TSP, asymmetric TSP, analyst's traveling salesman problem, and TSP path length for random sets of points in a square. TSPs raise important issues because various problems in

33 science, engineering, and bioinformatics fields such as very large scale integration(VLSI) design[4],
34 vehicle routing[5], scheduling problems[6], integrated circuits designs[7], genome sequencing[8, 9],
35 physical mapping problems[10], and constructing phylogenetic trees[11] can be formulated as TSPs. An
36 important characteristic of these problems is that tests performed on challenging TSP instances provide a
37 basis for analyzing the performance characteristics of global search meta-heuristic techniques[2]. On the
38 other hand, it falls into a class of NP-hard or NP-complete problems. Therefore, the research on the TSP is
39 theoretically important. This paper focuses on a Contraction-expansion algorithm based on Move, Change,
40 and Fliplr technique that generates near to optimal TSP solutions.

41 After all these years, a huge number of methods have been developed for solving TSPs. Exact
42 methods[12], heuristics[3, 13-15], and metaheuristics[16-19] have been proposed to solve the static TSPs.
43 Due to the fierce increase in the computation time when the problem size increases, heuristics and
44 metaheuristics are more preferable than exact methods since they trade optimality for efficiency. And also
45 the TSP has attracted extensive research and has been repeatedly used as the basis of comparison for
46 different optimization algorithms such as genetic algorithms (GAs)[19, 20], tabu search[21-24], simulated
47 annealing(SA)[25, 26], local search[27, 28], ant colony system(ACS)[16, 29], evolutionary algorithms
48 (EAs)[18], neural network[30, 31], and so on. These diverse approaches have demonstrated various degrees
49 of strength and success.

50 Most exact methods, e.g., Concorde, and heuristic methods, e.g., Lin–Kernighan, 2-Opt or 3-Opt, are not
51 (directly) applicable for asymmetric TSP cases since they are based on the triangle inequality of Euclidean
52 distance associated with symmetric TSPs. Methods for asymmetric TSPs are less studied with only a few
53 exceptions[32, 33]. Of course, symmetric methods can be applied with some modifications to the problem
54 or the method itself. But, this may significantly increase the computation time or degrade their
55 effectiveness [29], [52]. Evolutionary algorithms(EAs) and Ant colony optimization (ACO) algorithms
56 have proved to be powerful methods to tackle such problems due to their adaptation capabilities. Such
57 methods are capable of finding the global optimum (or close to the global optimum) solution for symmetric
58 TSP cases in seconds.

59 Recently, there were many new approaches derived from evolutionary algorithms(EAs), ant colony
60 optimization (ACO), simulated annealing(SA), and neural network, which have been presented effectually
61 to solve the TSPs[18, 25, 26, 34-36]. Such as Genetic-based Particle Swarm Optimization (GPSO)
62 algorithm for TSPs, and this evolutionary algorithm includes two phases. Where the first two was fuzzy C-
63 Means and Genetic-based PSO procedure applied to reduce the complexity and better efficiency of solving
64 TSPs with large number of cities, respectively[37]. Another two methods that hybrid method (based on
65 Particle Swarm Optimization, Ant Colony Optimization, and 3-Opt algorithms) and adaptive simulated

annealing algorithm with greedy search to solve TSP, which was better than the former of the performance[25, 38]. In hybrid methods that PSO is used for determining parameters, and which affected the performance of the ACO, and the 3-Opt is used for getting rid of the local solution found in the ACO algorithm. The metaheuristics approach looks for more effective mutations, a more rational greedy search strategy, and more universal adaptive parameter control of the proposed algorithm to achieve better results and faster convergence in solving TSP with large-scale cities. The solution accuracy of the Move, Change and Flip Search Algorithm(MCFSA) are still not comparable to some other state-of-the-art heuristics and metaheuristics for the TSP, including Ant Colony Optimization, Simulated Annealing[25], and Evolutionary algorithms(EAs)[18] for tackling the large-scale cities.

The rest of this paper is organized as follows. Section II introduces the TSP problem definition and describes how the TSP cases are generated. Section III describes the approaches from the literature that have been applied to solve the TSPs. Section IV gives details of the proposed EMF-CE algorithm. Section V presents the experimental studies that include comparisons with other popular heuristics and state-of-the-art algorithms. Finally, concluding remarks and future work directions are presented in Section VI.

80

81

82 II. DEFINITION OF TRAVELING SALESMAN PROBLEMS

83 A. Problem Formulation

Quintessentially, a TSP instance is modeled by a fully connected weighted graph $G = (N, A)$, where $N = \{x_1, \dots, x_n\}$ is a set of n nodes and $A = \{(x_i, x_j) \mid x_i, x_j \in N, i \neq j\}$ is a set of arcs. For the classic TSP, nodes and arcs represent the cities and the links between them. Each arc $(x_i, x_j) \in A$ is associated with a nonnegative value $d_{ij} \in \mathbb{R}^+$, which for the classic TSP represents the distance between cities x_i and x_j . While TSP instance is considered to be symmetric if $d_{ij} = d_{ji}$, $(x_i, x_j) \in A$, or asymmetric if $d_{ij} \neq d_{ji}$ for at least one $(x_i, x_j) \in A$, and in this paper we only considered the symmetric TSP. The most studied and well known special case of symmetric TSP is the version in which the distances among cities always satisfy the triangle inequality (e.g., $d_{ij} \leq d_{jk} + d_{ik}$), known as Euclidean TSP. But the definition of d_{ij} that the distance among cities may have multiple choices, while we also used the Euclidean distance here. Currently, there are many existing methods, both exact and heuristic, such that are based on the triangle inequality[12]. However, the corresponding asymmetric special case may not always satisfy the triangle inequality.

95

96

97 Evaluation function: It measures the cost of the route and is calculated using the sum of the Euclidean
 98 distances between each pair of cities with coordinates $\{x_i, x_j\}$. Assumed that the route has a length of n
 99 cities, then the fitness function[39] is defined in this paper as follows:

$$100 \quad \text{tsp}(x) = \sum_{i=1}^{n-1} d(x_i, x_{j+1}) \quad (1)$$

101 Where $d(x_i, x_j)$ is the distance between cities x_i and x_j . If both cities have y and z coordinates, then the
 102 distance is calculated as:

$$103 \quad d(x_i, x_j) = \sqrt{(y_i - y_j)^2 + (z_i - z_j)^2} \quad (2)$$

104

105 III.METHODOLOGIES FOR SOLVING TSPs

106 The challenges of algorithms in solving NP-complete problems and in static environments are well-
 107 known in terms of computational complexity. In the following sections, we briefly review existing and
 108 comparing algorithms designed to address the TSPs.

109 A. Evolutionary Algorithms

110 Evolutionary algorithms (EAs) forebode a very promising direction. However, only general problem-
 111 independent EAs are frequently inefficient in solving TSPs, especially large TSPs. Tsai et al.[18] proposes
 112 an evolutionary algorithm, called the heterogeneous selection evolutionary algorithm (HeSEA), which
 113 integrates edge assembly crossover (EAX) and Lin–Kernighan (LK) local search, through family
 114 competition and heterogeneous pairing selection. Where the HeSEA is especially efficient to cope with the
 115 large TSPs.

116 Jin et al.[19] proposed the integrated self-organizing map (ISOM) that combination the canonical genetic
 117 algorithm(GA) in the neural-evolutionary system, in which every individual (chromosome) represents an
 118 effective learning scheme(rule). The evolved ISOM (eISOM) has made at least a 1% improvement over the
 119 SOM developed by Budinich[30], the ESOM[40], the convex elastic net[41], and the FLEXMAP[42]
 120 algorithm that examined on a wide spectrum of TSPs. Feng et al.[43] developed the hybrid evolutionary
 121 fuzzy learning scheme combines the adaptive fuzzy C-means (FCM), simple MAX–MIN merging concept,
 122 simulated annealing (SA) learning algorithm, and an efficient table transform-based particle swarm
 123 optimization (TPSO) that automatically determines the near-optimal traveling path in LSTSPs. It denotes
 124 the clustering-based FCM algorithm as a classifier to reduce the complexity of the full traveling cities by
 125 dividing samples into the proper number of subgroups. After the TPSO and SA local optimal learning
 126 scheme identify the optimal traveling paths in each subgroup, and the novel and effective MAX–MIN
 127 algorithm rebuilds the complete traveling path.

Baraglia et al.[44] showed that combines the compact genetic algorithm(Cga) for the TSP with an efficient implementation of LK local search heuristics designed by Lin and Kernighan[14](Cga-LK). Which exploited the low memory requirements and the high computational efficiency. Nguyen et al.[45] further proposed hybrid genetic algorithm (HGA) was based on a multipopulation steady-state GA and combined with the local search heuristics. The offspring replacing the worst individual in the population that using a steady-state update strategy and linear ranking for selecting parents to maintain a more diverse population for a GA. They also use LK local search heuristics for tour improvement. And these approaches aimed at solving the large-scale TSPs.

136
137

138 *B. Simulated Annealing*

139 Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given
140 function. Specifically, it is a metaheuristic to approximate global optimization in a large search space. It is
141 often used when the search space is discrete (e.g., all tours that visit a given set of cities)[7]. But the major
142 shortage of SA is that it can be extremely slow and require significantly more processing time than other
143 meta-heuristics. Several studies have tried to improve SAs performance by changing the generation and
144 acceptance mechanisms.

145 Pepper et al.[46] applied a conventional simulated annealing(SA) algorithm to solve Euclidean TSPs,
146 which SA showed as the best of the non-demon algorithms with average accuracy and running time. Geng
147 et al.[25] proposed an adaptive simulated annealing algorithm with greedy search (ASA-GS) to deal with
148 the TSPs, and based on the standard simulated annealing that it can find a better solution at the cost of vast
149 time. Chen et al.[26] presented a method that called the genetic simulated annealing ant colony system with
150 particle swarm optimization techniques for solving the TSPs. The system simulates the annealing process
151 of metal atoms. Metal atoms at a high temperature will become unstable from their initial states and search
152 for the newest states. Li et al.[47] proposed a new hybrid scheme of parallel tempering and simulated
153 annealing (hybrid PT/SA) to copy with the TSPs. Which the SA process was to allow the temperature at
154 each temperature level to decrease gradually toward a target temperature.

155 Wang et al.[48] proposed a multi-agent SA algorithm with instance-based sampling (MSA-IBS) aims at
156 both improving the efficiency of SA and adding parallelism of SA more naturally for traveling salesman
157 problem, which employs a population of agents running SA algorithm and evolving individuals of instance-
158 based meta-heuristics collaboratively. Motivated by the success of using instance-based learning to
159 improve the efficiency of SA algorithm[49, 50] and ACO algorithm. Wang et al[51] presented swarm SA

160 (SSA) algorithm which exploited swarm of agents running SA algorithm and used the solution component
161 selection scheme of ACO algorithm to generate candidate solutions that can be efficient solve the TSPs.

162 Inspired by a metaheuristic algorithm called the list-based threshold-accepting (LBTA) algorithm which
163 has shown significant performance for combinatorial optimization problems that are NP-hard[52, 53]. Zhan
164 et al.[54] proposed the list-based simulated annealing (LBSA) algorithm to solve traveling salesman
165 problem algorithm, and used the novel list-based cooling scheme to control the decrease of temperature,
166 also generated a list of maximum temperature which was used by Metropolis acceptance criterion to decide
167 whether to accept the candidate solution.

168

169

170 *C. Ant Colony Optimization Algorithms*

171 Ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational
172 problems which can be reduced to discovery good paths through graphs. However, for multifaceted large-
173 scale optimization problems, the drawback of the ACO that was leisurely premature convergence.

174 The AS[55] is the first ACO algorithm applied for the TSP. Dorigo et al.[16] proposed ant colony system
175 (ACS) algorithm that built on the previous ant system in the direction of enlightening efficiency when
176 applied to symmetric and asymmetric TSPs. The main idea that was sets of cooperating agents called ants,
177 which search in parallel for good solutions to the TSP and also cooperated through pheromone-mediated
178 indirect and global communication. They also combined the local optimization heuristic with the restricted
179 3-opt procedure, which inserting/removing three edges on the path, and considers only 3-opt moves that did
180 not return the order in which the cities were visited. Furthermore, also Chu et al.[56] presented the parallel
181 ant colony system that combined seven communication strategies, which the communication was to update
182 the pheromone level for each route according to the best route found by neighboring groups. However,
183 Chen et al.[57] further built on the PACS and cooperating with GA, while the PGACS outperformed the
184 PACS in terms of accuracy solution quality.

185 Mavrovouniotis and Yang [58] further improved the MMAS algorithm[59] based on with or without
186 direct communication(DC) scheme, where the DC scheme reduced computational runtime and improved
187 the solution quality of conventional ACO algorithms to solve the large-scale TSPs. Which DC scheme
188 allowing to exchange cities and utilize the different solutions constructed by ants on each iteration.

189 *D. Heuristics And Neural Network*

190 Local search with k-change neighborhoods, k-opt, are the most widely used heuristic method for the
191 traveling salesman problem. Helsgaun [60, 61] proposed a new LKH called LKH-2, and based on the LKH-
192 1. which eliminates many of the limitations and shortcomings LKH-1. Combined use of general K-opt

193 moves with six partitioning schemes, for obtaining high-quality solutions and reducing the complexity of
194 solving large-scale problem instances, respectively. As we all know, LKH-2 is the best search algorithm to
195 achieve the best optimal tour so far. Rego et al.[15] surveyed leading heuristics for the TSPs, which
196 gathered with Lin–Kernighan (LK) and stem-and-cycle (S&C) methods, as well known the most effective
197 and efficient local search ejection chain (EC) methods. Proposing six and two variants of the Lin–
198 Kernighan (LK) and stem-and-cycle (S&C) ejection chain method, respectively.

199 Mahdi Saadatmand-Tarzjan et al.[62] proposed a novel constructive-optimizer NN (CONN), combined
200 uses a feedback structure and competitive training algorithm, which similar to Hopfield-type NNs
201 (HNN)[63] and Kohonen-type self-organizing maps(K-SOM)[19] were presented to provide the best
202 conciliation between the convergence speed and solution quality, respectively. Consequently, CONN has
203 consisted of a constructive part, which grew the tour and an optimizer part to optimize it. In the training
204 algorithm, an initial tour was created first and introduced to CONN. Thiago A.S. Masutti et al.[36] further
205 modified the Real-Valued Antibody Network designed to solve the Traveling Salesman Problem
206 (RABNET-TSP) [64], which cooperated with the immune-inspired self-organizing neural network based on
207 the immune system, and aim at improving its efficacy (quality of the solutions found) and reducing the
208 computational time.

209

210

211 IV. PROPOSED EXCHANGE-MOVE-FLIP BASED ON CE ALGORITHM

212 In this paper, a novel search algorithm that based on we proposed Contraction-Expansion algorithm[65]
213 is introduced to provide the best compromise between the convergence speed and solution quality. The
214 main idea of the proposed EMF-CE is taking advantage of integrated three operators Move, Exchange, and
215 Flip techniques, and cooperated the CE to search for the optimal solution of TSPs. We will use ten nodes as
216 an example to illustrate the process of the EMF-CE algorithm, as **Fig.1-a** show.

Fig 1 the original path of ten nodes (a) and exchange x_2 and x_4 (b)

A.Exchange Step

Assume that have $x_1, \dots, x_n$ nodes, if exchange x_i and x_j ($i \neq j = 1 \dots n$) can make the total distance tsp (1) value less than before, then achieved this exchange step, furthermore, anew generated another better tour. Those exchanges should include that the position of x_i and x_j in a two-dimensional array-based data structure with an n -vector to control two-dimensional matrix ($n \times 2$) original data. Each row corresponds to each current city and each column corresponds to the order of traveling sequence of this current tour. Also, the position wherein the Euclidean distance matrix the row and column of x_i and x_j will be complete exchange. To illustrate how this Exchange step is formulated, the following example is used for explanation. Which there are ten cities(nodes), as Fig.1b, after exchanged the x_2 and x_4 , the original path sequence 1-2-3-4-5... became 1-4-3-2-5.... The difference of tour length between the old path and the new path as follows:

$$X(1) = d(1,2) + d(2,3) + d(3,4) + d(4,5) - d(1,4) - d(4,3) - d(3,2) - d(2,5) = 65.9! \quad (3)$$

Where $X < 0$, it is indicated that the new path superior old path, then exchanged x_2 and x_4 two nodes. Here, if $X \leq 2$, then X that the difference of tour length only is between the red and blue dashed lines **Fig1-b**. And the other line segments (arcs) without change. Identically, it isn't necessary to calculate the distance in order to that decrease the computation burden. If $X \geq 2$, which would be involved eight line segments (arcs) (**Fig.2a-b** and **Fig3a-b**, the four red and four blue dashed lines). The difference in tour length X should be the following:

238

$$X = 1 - \frac{d(i-1, i) + d(i, i+1) + d(j-1, j) + d(j, j+1) + d(i-1, j) + d(j, i-1) + d(j, i+1) + d(i, j+1)}{4} \quad (4)$$

$i = 1, 2, 3, \dots, n-1, j = i+1, i+2, \dots, n$

239

240

241

242

243

244

245

246

247

Fig.2 exchange x_4 , x_7 and x_4 , x_8 nodes

It is worth noting that should be immediately exchanged only if $X < 0$. Meanwhile, exchange the row position of x_i and x_j in two-dimensional matrix of original data, also corresponding to exchange row and column in the distance matrix. The exchanged sequence can from the smallest to largest (such an exchange was called sequential[66], i was small to large, $j > i$) executed or inverse (i was large to small, $j < i$) executed. Identically, for the next Move step and Flip step both should be executed and following this principle. Given a feasible TSP tour, the algorithm repeatedly performs exchanges that reduce the length of the current tour, until a tour is reached for which no exchange yields an improvement.

Fig.3 exchange x_5 , x_7 (a) and x_7 , x_8 (b) nodes

All possible tours are made of ten nodes that more than one hundred and eighty thousand. But in the original tour of ten nodes (the sequence was random permutation), where only exchanged 5 times and instantly achieved optimal result **Fig.1a-b**. In the completely random (or random permutation nodes) condition, where the exchange times will be an increase, while will not considerably increase, and also can achieve preferably (maybe optimal) result. If found the better tour or smallest tsp (1) value should be marker X_m and $tspm$ variables archived. **Algorithm1-7 (supplement1)**.

B.Move Step

k-Opt algorithm (2-, 3-, and 4-opt move, etc.) was a simple local search algorithm for solving the TSP in optimization[66]. In this algorithm, k-Opt analysis involves deleting k connections (or edges) on a tour, reconnecting the tour in all other possible ways, and then evaluating each reconnection method to find the optimum one. Here, we proposed the move (1-move, 2-move, 3-move, etc.) algorithm of concept was different from k-opt, which only moves the nodes forward or backward, and also can move one, two, and three nodes etc together. For instance, tried to move the position of x_i ($i=1 \dots n$) forward or backward, and only change it that in the sequence of the TSP (Traveling salesman problems) tour. If made the $tsp(1)$ value less than before, then executed this step. The number of move nodes could be one node or simultaneously moved to adjoin nodes from two to n nodes were chose. Buts large number of experiments show that move simultaneously chose from two to five nodes together was the best choice. Such as, based on the original of

ten nodes path **Fig.4a**, and moved x_3 to the front of x_2 **Fig.4b** (this step is equivalent to exchange x_3 and x_2 , both moves adjoin nodes are equivalent to exchange step), and the difference of tour length that the old path and generated new path only included four segments **Fig.4b** (two red dashed line and two blue dashed line) (5). Cause $X < 0$, this move step made the TSP path shorter.

$$X = 1 \cdot d_{1,3} + 1 \cdot d_{2,4} - d_{1,2} - d_{3,4} = 2.1 \quad (5)$$

Fig 4 the original path of ten nodes (a) and moved x_3 to the front of x_2 (b)

if $X \neq 2$ then involved six segments line (or edges) **Fig.5a**, three red dashed lines and three blue dashed lines, respectively. And the difference of tour length between the segments line should be as following:

$$X = 1 \cdot d_{i,1,j} + 1 \cdot d_{j,i} + 1 \cdot d_{j,1,i} - d_{i,1,i} - d_{j,1,j} - d_{j,j,i} = 2, 3, 1, \dots, n-1, j-i-1, i-2, \dots, n \quad (6)$$

As a show, if $X < 0$, only exchange the adjoin nodes x_i and x_j or move x_j to the front of x_i . When moving multiple nodes (from two to five) that should make a judge whether may insert forward or insert backward, and only picking the proper insert direction that can make X a decrease. Depending on the orientations of the paths within the current structure, the path between i and j city may have to be reversed. As illustrated above that move could be forward or backward. For example, the original path (**Fig.4a**) that only by four times single node moving and instantly can achieve the optimum result as shown in **Fig.5-Fig.6** show. But after another random permutation of the current tour that the move times will increase. Also, the move step always stops after may n iterations if its rerouting strategy fails to improve the current best solution and will immediately execute another step. A similar local search algorithm starts at some location in the search

space and subsequently moves from the present location to a neighboring location or interval two and three nodes. Move algorithm is used for the purpose of improving tour arrangement operations.

Fig 5 moved x_4 to the front of x_2 (a) and x_5 to the front of x_3 (b)

Fig 6 moved x_4 to the front of x_3 (a) and the optimal path of ten nodes problem (b)

C.Flip Step

299 The concept of Flip in this paper that means Fliplr. If flip a segment line (or an edge) can make the $tsp(1)$
 300 value decrease, then executed flip step operation. As **Fig.7a**, there is a crossover link between x_4 , x_5 , and x_6 ,
 301 and this kink between edge(x_3 , x_4) and edge(x_6 , x_7) make tours more complex and lengthy. Therefore,
 302 removing these kinks is wise action. Eliminating kinks from the current tour that means the path 3-4-5-6-7
 303 should become 3-6-5-4-7, and achieve that only to flip x_4 , x_5 , x_6 to x_6 , x_5 , x_4 . The difference of tour length
 304 between the new path and old path as following (**Fig.7b**, two blue and two red dashed lines):

305
$$\chi 1 d_{3,6} \tilde{d}_{4,7} d_{3,4} d_{6,7} \quad 15.9! (7)$$

306 As shown not only the $tsp(1)$ value decrease by 15.9, and also eliminated the crossover link. For another
 307 flip edge (x_4 , x_7) and edge (x_5 , x_7), and the red and blue dashed lines that were the change segments line
 308 **Fig.8a-b**.

310 Fig 7 the original path ten nodes (a) and flip edge (x_4 , x_6) path (b)

enforceability). It can help to drop out the pitfall of the local optimal, and greatly increase the possibility of realizing global optimal.

With the feasibility criterion that the loop is stopped as soon as the chain represents a gainful exchange, and move, or when the chain cannot be produced better optimum tsp (1) value (no longer less than tsp value) anymore. In theory, there will be up to n iterations in the loop, where n is the number of nodes. In practice, however, the number of iterations is much smaller (due to the confirm of critical value). So we set rules for the critical value. if crt is too small, it cannot drop out the pitfall of local optimum. And if crt is too large that should not have exchanged and moved, it's would be forced to implement, and after one loop or more loops that the sequence of the tour would be disarray and have a strong impact on the effectiveness of implementation. Meanwhile, as the change of crt with the search loops alters, it appeared the form of the wave for changing. Therefore, we spent plenty of time researching and seeking the best function that can reasonably adjust and control the crt changing. Here, we found the function (9) that was a negative exponent function, which was used to the method of feedback regulation. This stop criterion (a product the feasibility critical value) plays a major role in the efficiency of the algorithm.

$$crt \leftarrow crt * e^{(-0.001)} \quad (9)$$

E. Combinatorial Optimization of Program Elements

Pseudocode of those programs that included exchange, move and flip (involved one node and from two to five nodes choose together), besides, combined the contraction-expansion algorithm and process of redeploying of enforceability which compiled more than twenty program elements. And the implement of the sequence of those programs that is the core of high effectiveness achieve global optimal. The implement of rules that are after exchange forward following on exchange backward with the different types of program elements, also, such that avoid an ineffective exchange of the nodes which should be legitimately configuration exchange between the one node and multiple nodes. A large computational experiment show that reasonably arrange the sequence of program elements is the point to increase the power of achieving the global optimum for traveling salesman problems. As far, that combinatorial optimization of program elements still not the best and yet have the space to reach the optimal.

Practically, after certainly loops that random to implement those programs of elements, it can prevent happened of invalid remove and vibration from the process of exchange and move nodes, also, yielding a better tour solution and more efficient to nearness optimal solution. Included all process of these program of elements is called loop times (round). It is impossible to achieve the global optimal for complicated traveling salesman problems just one loop times, and even hundreds of cycle rounds for large scare TSPs. But for small or middle scare TSPs that above procedure can produce a better tour solution.

370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401

F.Rotating TSP Loop

Above that procedure of exchange, move and flip the position of the start point, and endpoint won't implement the exchange and move steps. Also, it was hard to generate efficient exchange for the closely linked nodes. Sometimes, it will be not only left the dead corner to achieving the optimal tour, and also obstruct from yield the global optimal tour. Therefore, after certainly loop times that need to redefine the start node and keep the search for the best optimal tour. Usually, it is based on before generating the best tour that random forward or backward move to certainly nodes, and rotate TSP cycle that the lots of position as the new start(end) point. By doing this made the procedure of exchange, move and flip more unobstructed (no dead corner) for the complicate TSPs and increase the speed of the process of search the best tour solution.

V. EXPERIMENTAL STUDY

In this section, we describe the test problems and implementation details of the algorithms that we use in our computational study. We present and discuss the results generated by other different algorithms that have reported from the literature on the same problems in TSPLIB. Also, in our experiments, which we presented a new evaluation index described as following, and it was used to evaluate the performance of EMF-CE algorithm. And used the following index that was described as the efficiency index of EMF-CE algorithm.

A.Evaluation Index

To test developed operator performance, which test problems for small and medium scale (51-1000 cities) TSPs were taken from the TSPLIB online library[67] and the website at (<http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>). As we all know, if an algorithm can reach the nearby global optima for all kinds of traveling salesman problems, then it should be the best program or software. For all the test problems, we both random permutation for the initial path (disorganize the original data sequence) as the restart path after yield optimal solution. Which also records the optimal rates and time of achieved the global optima twenty times. And gave the optimal solution of the result was from the Condore 1.1(current best software) running in windows as the benchmark.

We used the rates of optimal solution ($R_o=N_o/N_r$, N_o is the number of optimal, N_r is the number of runs), the average tour length, and median tour length as the power of evaluation index. And the efficiency of the index included the shortest time, average time, and median time. By reason of complicated problems in a set of loops can't all achieve global optima and early quit, which also can't calculate the meantime to reach to global optima. In some way, it was fake that if we take early quit time into consideration. While median time (must have more than half to achieve global optima) isn't affected by extremes value; therefore, it is more stable and more valuable as an efficiency index.

The first two sets of experiments were *eil51.tsp* and *ch150.tsp* two small scale problems were used to test the executable of CPU time and rates of optimal. For random permutation the initial path as start path, and instantly record the CPU time as long as obtained the optimal solution. The CPU time-frequency distribution resulted from multiple replications executes. In early, obtained a few times for executing, while the CPU time was unimodal skewed normal distribution which was incompatible with the theory. But with the continuous running 1000 times, its CPU time of frequency distribution was similar to the formerly cardinal principle and showed the extremely skewed distribution as follows **Fig.9**.

Fig 9 the frequency distribution of CPU time with 1000 replicates for *eil51.tsp* problem

Executed one thousand times for this issue and the rates of optimal was 100% ($R_o=1000/1000$), also all of the lengths of the tour were 426, but between all of them, the CPU time has a more significant

422 difference **Fig.9**. As the show that the last group was 53 times for CPU time more than 30 seconds. The
 423 minimum and maximum CPU time was 0.05s and 65.01s, respectively. Because exist of the extreme value,
 424 the mean of the CPU time was 5.09s (the red line show) which wasn't reasonable as the efficiency of the
 425 index for indicating the effectiveness of the algorithm. While the median was 1.61s (the green line show)
 426 and has run for 775 times that the CPU time was less than 3s to achieve global optimal, also, it is indicating
 427 the median better than average as the efficiency of index.

428 As we all know, the tour length of the global optimal of *eil51.tsp* problem was 426. While we dig
 429 deeper and found that situation was different. We tried to retain 3 decimals, the tour length was 429.118,
 430 but this is not the real global optimal, and the tour length of global optimal should be 428.871 (the rounded
 431 number was 427) **Fig.10-b**. Otherwise, if round the tour length to nearest the integer was 426 Fig.9, keep
 432 integer distance of between $d_{25,26} \tilde{\sim} d_{26,27} \tilde{\sim} d_{27,28}$ and $d_{25,27} \tilde{\sim} d_{27,26} \tilde{\sim} d_{26,28}$, there are not the only one
 433 optimal tours. There are groups of test problems from TSPLIB that existed retain integer for the tour length
 434 will be influenced to produce optimal results. While for a large data TSP problem, the integer has little
 435 effect on

436
 437 Fig 10 the global optimal solution of integer (a) and non-integer (b) for *eil51.tsp* problem

438 the optimal results and relatively easy to obtain the optimal results, also convenient to compare each other
439 with different algorithms, which maybe was the reason for the widespread use. But there are some test TSP
440 problems, if retain 4~6 decimals, the optimal results will be the global optimal.

441 Where also executed one thousand times for *ch150.tsp* issue and the rates of optimal was
442 90.5% ($Ro=905/1000$), all of the lengths of the tours were 426, but between all of them, the CPU time has a
443 more significant difference **Fig.10**. And have 156 times for the CPU time more than 60s in the last group as
444 show **Fig.10**. The results indicated that minimum, average, maximum, and median tour length was 6528
445 (integer), 6593, 6531.06, and 6528, respectively. Also, the minimum, average, maximum, and median time
446 were 1.04s, 58.12, 760.66, and 1.61s, respectively. Indicating that already have $Ro=726/1000$ for the CPU
447 time less than 3s. It was also indicating that median time or tour length will be better than average time or
448 tour length as an index for proving the effectiveness of the algorithm. But the set of iteration times less than
449 300 that can't achieve global optimal.

450 There were three weak peaks that respectively nearby the 20s, 40s, and 50s both achieved the global
451 optimal. After researched the program and found that appeared these phenomena, which has certain
452 relations with the periodic variation of *crt*. While these peaks relatively the *crt* value both in the bottom and
453 easy to achieve global optimal after early big *crt* value broke the pitfall of local optimal. For the other test
454 problems that the distribution of CPU time both corresponding to the first two problems. This indicates

455
456 Fig 10 the frequency distribution of CPU time with 1000 replicates for *ch150.tsp*

that the new algorithm to achieve global optimal with a shorter median time, and another small part of it takes a long time to run. With the complexity of the problem, this phenomenon will become more obvious. This also reflects algorithmic programs to improve the possibility of need.

B. Implementation Details

All the experimental results were obtained by an Intel(R) Core(TM)2 Duo 2.93GHz with 2-GB main memory using MATLAB environment and running the Windows 7 operating system. For comparing EMF-CE with other algorithms in terms of CPU time, we scaled the CPU time of each algorithm by an appropriate scaling coefficient related to its processing system. We utilized the results reported to obtain the scaling coefficients as shown in Table I. Note that the codes made by 3-GL programming languages such as C/C++, PASCAL, and FORTRAN are more efficient than M-codes in the MATLAB interpreter. Nevertheless, we did not consider any scaling coefficient for comparing M-codes with 3-GL codes. In other words, it is expected to obtain better performance by implementing our algorithm using an efficient programming language like C or C++.

Table I
SCALING COEFFICIENTS FOR COUNTERPART ALGORITHMS

Counterpart Algorithm	Processing System	Program Language	Scaling Coefficients	
			3GL Prog.Lan	MATLAB
ASA-GS[26]	2.8GHz	C++	1.00	>1.00
CONN[62]	1.4GHz	C++	0.49	>0.49
eISOM, ESOM, Budinich[19]	1.4GHz	C++	0.48	>0.48
RABNET-TSP[36]	3.0GHz	MATLAB	1.06	>1.06
GCGA[68]	2.8GHz	C++	0.99	>0.99
HGA[69]	2.3GHz	C++	NA	NA
GSTM[70]	2.8GHz	Delphi	NA	NA
ACS[16]	200MHz	C++	NA	NA
SA[46]	SPARCULTRA5	C++	NA	NA
PSO-ACO-3Opt[38]	2.8GHz	C++	NA	NA

The percent differences (relative error) have been computed using:

$$RE(\%) = 1 - \frac{l}{l_{opt}} \quad 100\% \quad (48)$$

Where l were best tour lengths (BL, the result from EMF-CE), average tour lengths(AL), or median tour lengths(ML), however, l_{opt} is the optimal tour lengths(OPT). Note that for each TSP EMF-CE gives the same solution in all runs.

481 We select 43 problems from the TSPLIB online library. Each problem is randomly generated. We use Euclidean
482 distances (all of the Euclidean distances are real distances). The problems range in size from small to middle-scale 51
483 to 1000. Following Johnson and McGeoch[71], and only one running pr2392 was performed for this instance, which
484 were used in previous computational studies (see Jin et al. [19, 25, 26, 62, 70]. Each entry gives the number of nodes
485 in the problem (e.g., eil51 has 51 nodes). We use the approximate Euclidean distance metric specified in TSPLIB.
486 Moreover, we did not apply EMF-CE to the instances with 3000 or more cities since it would have taken too much
487 time (future work for large TSP). Therefore, we considered only a qualitative comparison between the different
488 algorithms. The results are listed in Table II. For each benchmark problem, the first ten columns of this table are 1)
489 number; 2) the TSP name; 3) optimal tour length (OPT) as reported in TSPLIB, and some instances renew calculated
490 that used the Concorde (<http://www.math.uwaterloo.ca/tsp/concorde/gui/gui.htm>), and 4)described in V.A 5) were
491 Shortest tour lengths (SL, the result from EMF-CE) and RE(%); 6) average tour lengths(AL) and RE(%); 7) median
492 tour lengths(ML) and RE(%); 8) the shortest of running time in second; 9) the average of running time in second; 9)
493 the median of running time in second. And the CPU time of average and median distribution as the Fig.11 scatter
494 diagram demo. Also, in order to facilitate comparison with different algorithms. According to Pepper et al.[46], we
495 split the set of selection problems into the first three sets that 51 to 299 nodes(31 instances), 318 to 574 nodes(7
496 instances), and 666 to 100 and one 2392 nodes(5instances).

497 TABLE II
498 EMF-CE SOLUTIONS TO 43 BENCHMARK TSP FROM TSPLIB

No.	Instances	OPT	Ro	SL(%)	AL(%)	ML(%)	ST(s)	AT(s)	MT(s)
1	eil51	426	20/20	Optimal	426(0)	426(0)	0.21	20.71	12.95
2	st70	675	20/20	Optimal	675(0)	675(0)	0.52	15.98	13.37
3	eil76	538	20/20	Optimal	538(0)	538(0)	1.09	75.96	66.14
4	pr76	108159	20/20	Optimal	108159(0)	108159(0)	12.85	32.62	29.37
5	rat99	1211	20/20	Optimal	1211(0)	1211(0)	2.33	77.97	73.00
6	kroA100	21282	20/20	Optimal	21282(0)	21282(0)	0.67	39.54	21.37
7	kroB100	22141	20/20	Optimal	22141(0)	22141(0)	7.88	150.29	156.76
8	kroC100	20749	20/20	Optimal	20749(0)	20749(0)	2.20	43.71	27.62
9	kroD100	21294	20/20	Optimal	21294(0)	21294(0)	1.00	37.05	24.24
10	kroE100	22068	7/20	Optimal	22097.3(0.13)	22100(0.15)	32.11	1287.12	182.31
11	rd100	7910	20/20	Optimal	7910(0)	7910(0)	16.26	149.86	159.14
12	eil101	629	14/20	Optimal	630.35(0.21)	629(0)	24.83	173.40	173.51
13	lin105	14379	20/20	Optimal	14379(0)	14379(0)	16.92	37.33	26.66
14	pr107	44303	20/20	Optimal	44303(0)	44303(0)	0.36	8.86	3.75
15	pr124	59030	20/20	Optimal	59030(0)	59030(0)	6.09	70.23	50.38
16	bier127	118282	3/20	Optimal	118483.4(0.17)	118476.7(0.16)	101.83	396.94	388.89
17	ch130	6110	7/20	Optimal	6130.45(0.33)	6128(0.29)	149.25	337.28	335.73
18	gr137	697	20/20	Optimal	697(0)	697(0)	2.02	166.41	172.51
19	pr144	58537	20/20	Optimal	58537(0)	58537(0)	4.68	118.55	96.50

20	ch150	6528	18/20	Optimal	6529.71(0.03)	6528(0)	1.80	57.99	1.90
21	kroA150	26524	8/20	Optimal	26578.8(0.21)	26566(0.16)	204.79	429.27	346.90
22	kroB150	26130	5/20	Optimal	26166.6(0.14)	26150(0.08)	306.43	1291.90	1350.45
23	pr152	73682	20/20	Optimal	73709.2(0.04)	73682(0)	0.89	114.19	84.65
24	u159	42080	18/20	Optimal	42104.4(0.06)	42080(0)	1.90	106.94	41.01
25	kroA200	29368	1/20	Optimal	29543.4345(0.6)	29545.5(0.6)	532.67	2088.46	861.85
26	kroB200	29437	0/20	29440(0.01)	29597.7(0.55)	29530.5(0.32)	628.99	2622.08	2181.19
27	ts225	126643	20/20	Optimal	126643(0)	126643(0)	119.99	399.69	377.58
28	pr226	80369	2/20	Optimal	80378.5(0.01)	80373(0)	1592.06	2737.44	2434.60
29	pr264	49135	12/20	Optimal	49155.95(0.04)	49135(0)	260.69	864.18	857.53
30	a280	2579	8/20	Optimal	2585.2(0.24)	2590(0.43)	988.79	1746.69	1601.42
31	pr299	48191	0/20	48226(0.07)	48418.9(0.47)	48435.5(0.51)	188.22	4633.67	2082.20
32	lin318	42029	0/10	52152.99(0.30)	42152.99(0.3)	42152.99(0.3)	7.30	225.98	9.36
33	rd400	15281	0/10	15338.60(0.40)	15480.174(1.3)	15475.61(1.27)	626.35	28878.17	31025.00
34	fl417	11861	0/10	11873(0.10)	11882.3(0.18)	11882(0.18)	6761.00	14498.14	14801.00
35	pr439	107217	0/10	107267(0.05)	107985.5(0.72)	107941(0.68)	8343.34	24983.92	30897.71
36	pcb442	50778	0/10	51009.68(0.46)	51009.68(0.46)	51009.68(0.46)	17.58	1293.36	152.25
37	att532	86729	0/10	86779(0.06)	88232.17(1.73)	88145(1.63)	30961.84	42216.81	39462.64
38	u574	36905	0/10	36955(0.14)	37662.67(2.05)	37819(2.48)	43611.95	77116.72	74097.63
39	gr666	3062	0/5	3067.09(0.17)	3098.836(1.2)	3112(1.63)	68419.00	136507.80	145240.00
40	u724	41910	0/5	41960(0.12)	42744.8(1.99)	42840(2.22)	136850.00	169554.00	145740.00
41	rat783	8806	0/5	8846(0.45)	9007.115(2.28)	9033.23(2.58)	194695.35	227887.42	225529.61
42	dsj1000 (CEIL2D)	18660188	0/5	18660238 (0.00027)	18942217.2 (1.51)	19020000 (1.93)	313540.00	344652.20	347430.00
43	pr2392	378032	0/1	378062.83 (0.0082)	378062.83 (0.0082)	378062.83 (0.0082)	434484.00	434484.00	434484.00

499

500

501

Fig.11 EMF-CE average CPU time versus the number of cities for 43 benchmark TSPs from TSPLIB

Fig. 11 shows the EMF-CE CPU time versus the number of cities for 43 benchmark TSPs. Fig. 7 shows the CONN CPU time versus the number of cities for all benchmark TSPs. In this figure, the CPU time versus n has been fitted to a polynomial.

Fig.12 EMF-CE percent difference versus the number of cities for 43 benchmark TSPs from TSPLIB

As illustrated in **Fig. 12**, the three evaluation indexes showed that the solution qualities of EMF-CE are range 0.00% ~ 0.46%, 0.00%~2.28%, and 0.00%~2.58% for all test benchmark TSPs TABLE II. Furthermore, EMF-CE gave the optimal solutions for 29 benchmark TSPs, in which the number of cities less than 300.

D. Comparing with Computationally Comparable Algorithms

In this section, we compared EMF-CE with several well-known evolutionary algorithms, simulated annealing, evolutionary algorithms, and neural network including ASA-GS, GCGA, HGA, CONN, and LBSA, and so on. Which were reported in[26], [68], [69],[62] and [54], respectively. Also, these algorithm methodologies that we already briefly described in section II. EMF-CE has a competitive computational complexity with respect to these algorithms ($O(n^3)$). The third set of experiments was performed on 38 benchmark TSPs from TSPLIB with 51(small-scale) to 783 cities (medium-scale). The results are only given the percent difference of shortest tour lengths in Table III.

For average shortest tour length that corresponding to the same TSP instances and EMF-CE provided 0.35%, 1.49%, 0.82, 3.11%, and 0.02% improvements over ASA-GS, GCGA, HGA, CONN, and LBSA, respectively. When we also split the set of problems into three sets, as described in section V.C. Over the corresponding to first set problems **Fig.13**, we see that EMF-CE and LBSA was the most accurate algorithm with the difference of 0.003%. However, EMF-CE provided 0.22%, 0.92%, 0.47%, and 2.75% improvements over ASA-GS, GCGA, HGA and CONN to the average shortest tour length, respectively; also, corresponding to second set problems, EMF-CE

provided 0.16%, 0.80%, 3.09%, 4.62% and 4.51% improvements over LBSA, ASA-GS, HGA, GCGA and CONN to the average shortest tour length, respectively. We can see that for the medium-scale TSPs, and the EMF-CE gave the best tour length.

TABLE IV
COMPARING EMF-CE AND ITS FIVE COUNTERPARTS INCLUDING ASA-GS, GCGA, HGA, CONN AND LBSA FOR THE SAME BENCHMARK TSPS FROM TSPLIB IN TERMS OF THE SOLUTION QUALITY AND COMPUTATIONAL COMPLEXITY.

Instances	OPT	EMF-CE	ASA-GS	GCGA	HGA	CONN	LBSA
		SL(%)	SL(%)	SL(%)	SL(%)	SL(%)	SL(%)
eil51	426	0	0.67	0.23	0.67	2.58	0
st70	675	0	0.31	0	0.31	2.96	0
eil76	538	0	1.18	2.23	1.18	5.02	0
pr76	108159	0	0	0.14	0	4.34	0
rat99	1211	0	0.68	0.99	0.68	0.33	0
kroA100	21282	0	0.02	0.05	0.02	2.57	0
kroB100	22141	0	0	0.24	na	2.6	0
kroC100	20749	0	0.01	0.32	0.01	1.53	0
kroD100	21294	0	0	1.22	0	1.42	0
kroE100	22068	0	0.17	0.24	na	1.97	0
rd100	7910	0	0.01	0.96	0	3.59	0
eil101	629	0	1.78	1.59	1.78	5.09	0
lin105	14379	0	0.03	0.56	0.03	0.38	0
pr107	44303	0	0	1.15	0	2.77	0
pr124	59030	0	0	0	0	1.74	0
bier127	118282	0	0.01	0.91	na	2.45	0
ch130	6110	0	0.01	na	0.01	4.88	0
pr144	58537	0	0	0	0	2.34	0
ch150	6528	0	0.04	na	0.04	5.5	0
kroA150	26524	0	0	1.4	0	2.27	0
kroB150	26130	0	0.04	1.63	—	3.09	0
pr152	73682	0	0	0.19	0	0.79	0
u159	42080	0	0.74	1.74	na	na	0
kroA200	29368	0	0.15	1.33	0	0.79	0
kroB200	29437	0.01	0.23	2.09	0.05	1.07	0
ts225	126643	0	0	0.64	1.18	na	na
pr226	80369	0	0.22	0.35	0.08	2.35	na
pr264	49135	0	0	0.62	0.03	3.58	na
a280	2579	0	na	na	3.13	4.24	na
pr299	48191	0.07	0.16	4.11	2.64	4.85	0
lin318	42029	0.3	0.66	3.74	1.42	na	0.1
rd400	15281	0.38	0.46	6.11	5.03	1.93	0.2
fl417	11861	0.1	0.67	1.65	na	3.58	na

pr439	107217	0.05	2.61	5.19	2.76	4.07	na
pcb442	50778	0.46	0.56	7.66	4.13	4.8	0.11
u574	36905	0.14	0.89	na	na	5.77	na
u724	41910	0.12	0.87	na	na	6.48	0.7
rat783	8806	0.45	1.68	na	na	6.72	0.92
Average		0.05	0.4	1.54	0.87	3.16	0.07

Note: 'na' represented no data.

Fig.13 Average shortest tour lengths percent above optimality for EMF-CE and five comparable algorithms

VI. CONCLUSION

In this paper, we presented a new method, which Contraction-expansion algorithm based on Move, Change, and Flip three operators called EMF-CE for TSPs. The main purpose of designing EMF-CE is to achieve near to optimal solution quality and the most accurate algorithm while the method would give comparatively slower of convergence speed. Its computational intensity is $O(n^3)$. But it is now can achieve global optimal for the small scale TSPs. For the generalized traveling salesman problems that round number and round integer may affect the end optimal result. Usually can take 4 to 6 significant digit to calculate the distance and TSP path length, in order to prevent the loss of precision and affect the realization of the optimal solution.

The EMF-CE performance was compared with those of several computationally comparable including ASA-GS, GCGA, HGA, CONN, and LBSA, etc.. in terms of the qualitative comparison between those algorithms. However,

we only show that five algorithms compares with the EMF-CE(have compared and reported in[25, 26, 38, 54], etc.). Cause the others state-of-the-art (GATM, GSM, SA, ITS, GSA-ACS-PSO, RABNET-TSP, and MSA-IBS, etc. have described and overviewed in section II) algorithm that can't give the best tour lengths (reach to global optimal) for the small scale TSPs and let alone the medium-scale TSPs. Where EMF-CE has outperformed these state-of-the-art NNs and evolution algorithms in terms of accuracy and in a reasonable time for the best tour length and average tour length. But these algorithms provided a better and promising CPU time with poverty solution quality. Also, the CPU time of EMF-CE with M code and only using single-core for computing. Moreover, the scaling coefficients large than any 3G program language **TABLE I**. In other words, it is expected to obtain better performance by implementing our algorithm using an efficient programming language like C/C++.

Future work should investigate and improve convergence speed. While the encouraging results presented here led us to adapt the algorithm to solve more complex routing problems, such as both symmetric and asymmetric DTSPs, and the capacitated vehicle routing problem (CVRP). Also, further investigations must be performed with larger TSP instances and hybrid use of the proposed algorithm with route improvement heuristics, such as k-opt and simulated annealing.

- 582 [1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.
583 H. Freeman & Co., 1979, p. 338.
- 584 [2] D. Gamboa, C. Rego, and F. Glover, "Implementation analysis of efficient heuristic algorithms for the
585 traveling salesman problem," *Computers & Operations Research*, vol. 33, no. 4, pp. 1154-1172, 4// 2006.
- 586 [3] D. Karapetyan and G. Gutin, "Lin–Kernighan heuristic adaptations for the generalized traveling salesman
587 problem," *European Journal of Operational Research*, vol. 208, no. 3, pp. 221-232, 2/1/ 2011.
- 588 [4] W. Cook, "VLSI Data Sets," [Online], 2003.
- 589 [5] M. Fisher, "Chapter 1 Vehicle routing," in *Handbooks in Operations Research and Management Science*, vol.
590 Volume 8: Elsevier, 1995, pp. 1-33.
- 591 [6] L. D. Whitley, T. Starkweather, and D. A. Fuquay, "Scheduling Problems and Traveling Salesmen: The
592 Genetic Edge Recombination Operator," presented at the Proceedings of the 3rd International Conference
593 on Genetic Algorithms, 1989.
- 594 [7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no.
595 4598, pp. 671-680, 1983.
- 596 [8] R. Agarwala, D. L. Applegate, D. Maglott, G. D. Schuler, and A. A. Schäffer, "A Fast and Scalable Radiation
597 Hybrid Map Construction and Integration Strategy," *Genome Research*, vol. 10, no. 3, pp. 350-364, March 1,
598 2000 2000.
- 599 [9] P. Avner *et al.*, "A radiation hybrid transcript map of the mouse genome," *Nat Genet*, 10.1038/ng1001-194
600 vol. 29, no. 2, pp. 194-200, 10//print 2001.
- 601 [10] F. Alizadeh, R. M. Karp, L. A. Newberg, and D. K. Weissner, "Physical mapping of chromosomes: A
602 combinatorial problem in molecular biology," *Algorithmica*, vol. 13, no. 1, pp. 52-76, 1995// 1995.
- 603 [11] C. Korostensky and G. H. Gonnet, "Using traveling salesman problem algorithms for evolutionary tree
604 construction," *Bioinformatics*, vol. 16, no. 7, pp. 619-627, July 1, 2000 2000.
- 605 [12] W. J. Cook, "In pursuit of the traveling salesman. Mathematics at the limits of computation," *Princeton*
606 *University Press*, 2012.
- 607 [13] G. Tao and Z. Michalewicz, "Inver-over operator for the TSP," in *Parallel Problem Solving from Nature —*
608 *PPSN V: 5th International Conference Amsterdam, The Netherlands September 27–30, 1998 Proceedings*, A.
609 E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg,
610 1998, pp. 803-812.
- 611 [14] S. Lin and B. W. Kernighan, "An Effective Heuristic Algorithm for the Traveling-Salesman Problem,"
612 *Operations Research*, vol. 21, no. 2, pp. 498-516, 1973/04/01 1973.
- 613 [15] C. Rego, D. Gamboa, F. Glover, and C. Osterman, "Traveling salesman problem heuristics: Leading methods,
614 implementations and latest advances," *European Journal of Operational Research*, vol. 211, no. 3, pp. 427-
615 441, 6/16/ 2011.
- 616 [16] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling
617 salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53-66, 1997.
- 618 [17] N. L. J. Ulder, E. H. L. Aarts, H.-J. Bandelt, P. J. M. van Laarhoven, and E. Pesch, "Genetic local search
619 algorithms for the traveling salesman problem," in *Parallel Problem Solving from Nature: 1st Workshop,*
620 *PPSN I Dortmund, FRG, October 1–3, 1990 Proceedings*, H.-P. Schwefel and R. Männer, Eds. Berlin,
621 Heidelberg: Springer Berlin Heidelberg, 1991, pp. 109-116.
- 622 [18] T. Huai-Kuang, Y. Jinn-Moon, T. Yuan-Fang, and K. Cheng-Yan, "An evolutionary algorithm for large traveling
623 salesman problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no.
624 4, pp. 1718-1729, 2004.
- 625 [19] J. Hui-Dong, L. Kwong-Sak, W. Man-Leung, and Z. B. Xu, "An efficient self-organizing map designed by
626 genetic algorithms for the traveling salesman problem," *IEEE Transactions on Systems, Man, and*
627 *Cybernetics, Part B (Cybernetics)*, vol. 33, no. 6, pp. 877-888, 2003.
- 628 [20] C. Contreras-Bolton and V. Parada, "Automatic Combination of Operators in a Genetic Algorithm to Solve
629 the Traveling Salesman Problem," *PLoS ONE*, vol. 10, no. 9, p. e0137724, 2015.

- 630 [21] J. Knox, "Heuristic, Genetic and Tabu SearchTabu search performance on the symmetric traveling salesman
631 problem," *Computers & Operations Research*, vol. 21, no. 8, pp. 867-876, 1994/10/01 1994.
- 632 [22] C. N. Fiechter, "A parallel tabu search algorithm for large traveling salesman problems," *Discrete Applied
633 Mathematics*, vol. 51, no. 3, pp. 243-267, 1994/07/06 1994.
- 634 [23] E. Osaba, R. Carballedo, P. Lopez-Garcia, and F. Diaz, "Comparison between Golden Ball Meta-heuristic,
635 Evolutionary Simulated Annealing and Tabu Search for the Traveling Salesman Problem," presented at the
636 Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, Denver,
637 Colorado, USA, 2016.
- 638 [24] J. M. Bostamam and Z. Othman, "Hybrid water flow-like algorithm with Tabu search for traveling salesman
639 problem," *AIP Conference Proceedings*, vol. 1761, no. 1, p. 020058, 2016.
- 640 [25] X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao, "Solving the traveling salesman problem based on an
641 adaptive simulated annealing algorithm with greedy search," *Applied Soft Computing*, vol. 11, no. 4, pp.
642 3680-3689, 6// 2011.
- 643 [26] S.-M. Chen and C.-Y. Chien, "Solving the traveling salesman problem based on the genetic simulated
644 annealing ant colony system with particle swarm optimization techniques," *Expert Systems with
645 Applications*, vol. 38, no. 12, pp. 14439-14450, 11// 2011.
- 646 [27] Y. Wu, T. Weise, and R. Chiong, "Local search for the Traveling Salesman Problem: A comparative study," in
647 *Cognitive Informatics & Cognitive Computing (ICCI*CC), 2015 IEEE 14th International Conference on*, 2015,
648 pp. 213-220.
- 649 [28] D. S. Johnson, *The Traveling Salesman Problem: A Case Study in Local Optimization*. 2010, pp. 215--310.
- 650 [29] M. Mavrovouniotis, M. F. M. x00Fc, Iler, and S. Yang, "Ant Colony Optimization With Local Search for
651 Dynamic Traveling Salesman Problems," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1-14, 2016.
- 652 [30] M. Al-Mulhem and T. Al-Maghrabi, "Efficient convex-elastic net algorithm to solve the Euclidean traveling
653 salesman problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no.
654 4, pp. 618-620, 1998.
- 655 [31] B. Avşar and D. E. Aliabadi, "Parallelized neural network system for solving Euclidean traveling salesman
656 problem," *Applied Soft Computing*, vol. 34, pp. 862-873, 9// 2015.
- 657 [32] L. Buriol, P. M. França, and P. Moscato, "A New Memetic Algorithm for the Asymmetric Traveling Salesman
658 Problem," *Journal of Heuristics*, vol. 10, no. 5, pp. 483-506, 2004// 2004.
- 659 [33] B. Freisleben and P. Merz, "A genetic local search algorithm for solving symmetric and asymmetric traveling
660 salesman problems," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*,
661 1996, pp. 616-621.
- 662 [34] X. Yang and J. s. Wang, "Application of improved ant colony optimization algorithm on traveling salesman
663 problem," in *2016 Chinese Control and Decision Conference (CCDC)*, 2016, pp. 2156-2160.
- 664 [35] T. Saenphon, S. Phimoltares, and C. Lursinsap, "Combining new Fast Opposite Gradient Search with Ant
665 Colony Optimization for solving travelling salesman problem," *Engineering Applications of Artificial
666 Intelligence*, vol. 35, pp. 324-334, 10// 2014.
- 667 [36] T. A. S. Masutti and L. N. de Castro, "A self-organizing neural network using ideas from the immune system
668 to solve the traveling salesman problem," *Information Sciences*, vol. 179, no. 10, pp. 1454-1468, 4/29/
669 2009.
- 670 [37] Y.-F. Liao, D.-H. Yau, and C.-L. Chen, "Evolutionary algorithm to traveling salesman problems," *Computers &
671 Mathematics with Applications*, vol. 64, no. 5, pp. 788-797, 9// 2012.
- 672 [38] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on Particle Swarm Optimization, Ant
673 Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem," *Applied Soft Computing*, vol.
674 30, pp. 484-490, 5// 2015.
- 675 [39] G. Vahdati, M. Yaghoubi, M. Poostchi, and S. M. B. N, "A New Approach to Solve Traveling Salesman
676 Problem Using Genetic Algorithm Based on Heuristic Crossover and Mutation Operator," in *Soft Computing
677 and Pattern Recognition, 2009. SOCPAR '09. International Conference of*, 2009, pp. 112-116.
- 678 [40] K. S. Leung, H. D. Jin, and Z. B. Xu, "An expanding self-organizing neural network for the traveling salesman
679 problem," *Neurocomputing*, vol. 62, no. 1, pp. 267-292, 2004.

680 [41] M. Budinich, "A Self-Organizing Neural Network for the Traveling Salesman Problem That Is Competitive
681 with Simulated Annealing," *Neural Computation*, vol. 8, no. 2, pp. 416-424, 1996/02/15 1996.

682 [42] B. Fritzke and P. Wilke, "FLEXMAP-a neural network for the traveling salesman problem with linear time
683 and space complexity," in *Neural Networks, 1991. 1991 IEEE International Joint Conference on*, 1991, pp.
684 929-934 vol.2.

685 [43] H.-M. Feng and K.-L. Liao, "Hybrid evolutionary fuzzy learning scheme in the applications of traveling
686 salesman problems," *Information Sciences*, vol. 270, pp. 204-225, 6/20/ 2014.

687 [44] R. Baraglia, J. I. Hidalgo, and R. Perego, "A hybrid heuristic for the traveling salesman problem," *IEEE
688 Transactions on Evolutionary Computation*, vol. 5, no. 6, pp. 613-622, 2001.

689 [45] H. D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga, "Implementation of an Effective Hybrid GA for
690 Large-Scale Traveling Salesman Problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B
691 (Cybernetics)*, vol. 37, no. 1, pp. 92-99, 2007.

692 [46] J. W. Pepper, B. L. Golden, and E. A. Wasil, "Solving the traveling salesman problem with annealing-based
693 heuristics: a computational study," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems
694 and Humans*, vol. 32, no. 1, pp. 72-77, 2002.

695 [47] Y. Li, V. A. Protopopescu, N. Arnold, X. Zhang, and A. Gorin, "Hybrid parallel tempering and simulated
696 annealing method," *Applied Mathematics and Computation*, vol. 212, no. 1, pp. 216-228, 6/1/ 2009.

697 [48] C. Y. Wang, M. Lin, Y. W. Zhong, and H. Zhang, "Solving travelling salesman problem using multiagent
698 simulated annealing algorithm with instance-based sampling," *International Journal of Computing Science &
699 Mathematics*, vol. 6, no. 4, pp. 336-353, 2015.

700 [49] Y. Zhong, L. Wang, C. Wang, and H. Zhang, "Multi-agent simulated annealing algorithm based on differential
701 evolution algorithm," *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, pp. 217-228, 2012.

702 [50] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo, "Model-Based Search for Combinatorial Optimization:
703 A Critical Survey," *Annals of Operations Research*, vol. 131, no. 1, pp. 373--395, 2004.

704 [51] C. Wang, M. Lin, Y. Zhong, and H. Zhang, "Swarm simulated annealing algorithm with knowledge-based
705 sampling for travelling salesman problem," *International Journal of Intelligent Systems Technologies &
706 Applications*, vol. 15, no. 1, pp. 74-94, 2016.

707 [52] C. D. Tarantilis and C. T. Kiranoudis, "A list-based threshold accepting method for job shop scheduling
708 problems," *International Journal of Production Economics*, vol. 77, no. 2, pp. 159-171, 2002.

709 [53] C. D. Tarantilis, C. T. Kiranoudis, and V. S. Vassiliadis, "A list based threshold accepting metaheuristic for the
710 heterogeneous fixed fleet vehicle routing problem," *Journal of the Operational Research Society*, vol. 54,
711 no. 54, pp. 65-71(7), 2003.

712 [54] S. H. Zhan, J. Lin, Z. J. Zhang, and Y. W. Zhong, "List-Based Simulated Annealing Algorithm for Traveling
713 Salesman Problem," *Computational Intelligence & Neuroscience*, vol. 2016, no. 5, pp. 1-12, 2016.

714 [55] A. Colomi, M. Dorigo, and V. Maniezzo, "Distributed Optimization by Ant Colonies," in *European Conference
715 on Artificial Life*, 1991.

716 [56] S. C. Chu, J. F. Roddick, and J. S. Pan, "Ant colony system with communication strategies," (in English),
717 *Information Sciences*, Article vol. 167, no. 1-4, pp. 63-76, Dec 2004.

718 [57] S.-M. Chen and C.-Y. Chien, "Parallelized genetic ant colony systems for solving the traveling salesman
719 problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3873-3883, 4// 2011.

720 [58] M. Mavrovouniotis and S. Yang, "Ant colony optimization with direct communication for the traveling
721 salesman problem," in *2010 UK Workshop on Computational Intelligence (UKCI)*, 2010, pp. 1-6.

722 [59] T. Stützle and H. H. Hoos, "The MAX-MIN ant system and local search for the traveling salesman problem,"
723 1997.

724 [60] K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," *European
725 Journal of Operational Research*, vol. 126, no. 1, pp. 106-130, 2000.

726 [61] K. Helsgaun, "General k-opt submoves for the Lin-Kernighan TSP heuristic," *Mathematical Programming
727 Computation*, vol. 1, no. 2, pp. 119-163, 2009// 2009.

728 [62] M. Saadatmand-Tarzjan, M. Khademi, M. R. Akbarzadeh-T, and H. A. Moghaddam, "A Novel Constructive-
729 Optimizer Neural Network for the Traveling Salesman Problem," *IEEE Transactions on Systems, Man, and
730 Cybernetics, Part B (Cybernetics)*, vol. 37, no. 4, pp. 754-770, 2007.

731 [63] Y. Takahashi, "Mathematical improvement of the Hopfield model for feasible solutions to the traveling
732 salesman problem by a synapse dynamical system," *IEEE Transactions on Cybernetics*, vol. 28, no. 6, pp.
733 906-19, 1998.

734 [64] R. Pasti and L. N. De Castro, "A Neuro-Immune Network for Solving the Traveling Salesman Problem," in
735 *International Joint Conference on Neural Networks*, 2006, pp. 3760-3766.

736 [65] GU Shi-Liang, WAN Lin-Sheng, HUANG Li-Juan, and WANG Wen-Ping, "Improved Contraction-Expansion
737 Algorithm for Curve and Surface Fitting," *Acta Agronomica Sinica*, no. 04, pp. 583-589, 2007.

738 [66] K. Helsgaun, "An Effective Implementation of K-opt Moves for the Lin-Kernighan TSP Heuristic,"
739 *Transactions of the Japan Society of Mechanical Engineers A*, vol. 69, no. 12, pp. 1615-1628, 2010.

740 [67] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library," *ORSA Journal on Computing*, vol. 3, no. 4, pp.
741 376-384, 1991.

742 [68] J. Yang, C. Wu, H. P. Lee, and Y. Liang, "Solving traveling salesman problems using generalized chromosome
743 genetic algorithm," *Progress in Natural Science*, vol. 18, no. 7, pp. 887-892, 7/10/ 2008.

744 [69] Y. Wang, "The hybrid genetic algorithm with two local optimization strategies for traveling salesman
745 problem," *Computers & Industrial Engineering*, vol. 70, pp. 124-133, 4// 2014.

746 [70] M. Albayrak and N. Allahverdi, "Development a new mutation operator to solve the Traveling Salesman
747 Problem by aid of Genetic Algorithms," *Expert Systems with Applications*, vol. 38, no. 3, pp. 1313-1320, 3//
748 2011.

749 [71] D. S. Johnson and L. A. McGeoch, "Experimental Analysis of Heuristics for the STSP," in *The Traveling*
750 *Salesman Problem and Its Variations*, G. Gutin and A. P. Punnen, Eds. Boston, MA: Springer US, 2007, pp.
751 369-443.