DISCRETE HEAT TRANSFER SEARCH FOR SOLVING TRAVELLING SALESMAN PROBLEM

POONAM SAVSANI
Department of Industrial Engineering, Pandit Deenadayal Petroleum University
Gandhinagar, Gujarat, India
and
Postdoctoral Fellow, Department of Mathematics and Statistics, Faculty of Science
Thompson Rivers University, Kamloops, BC, Canada V2C 0C8

MOHAMED A. TAWHID*
Department of Mathematics and Statistics, Faculty of Science
Thompson Rivers University, Kamloops, BC, Canada V2C 0C8

(Communicated by Zhipeng Cai)

Abstract. Within the academic circle the Traveling Salesman Problem (TSP), this is one of the most major NP-hard problems that have been a primary topic of discussion for years. Developing efficient algorithms to solve TSP have been the goal of many individuals, and so this has been addressed efficiently in this article. Here, a discrete heat transfer search (DHTS) is proposed to solve TSP. DHTS uses three distinct phases to update the city tours namely, conduction, convection, and radiation. Each phase performs a certain function as the conduction phase is a replica of the 2-Opt local search technique, the convection phase exchanges the random city with the finest city tour, and the radiation phase exchanges the random city among two separate city tours without compromising the basics of HTS algorithm. Bench test problems taken from TSPLIB successfully test the algorithm and demonstrate the fact that the proposed algorithm can attain results near the optimal values, and do so within an acceptable duration.

1. Introduction. The renowned NP-hard problem, the traveling salesman problem (TSP) compromises in basic terms of a set of cities and the distance between each pair of the cities. For scholars, the overall challenge is to determine which of the routes is not only the shortest in length, but also returns to the origin city after visiting each city exactly once. The entire course including the different cities is labelled as the Hamiltonian circuit (HC) while the shortest Hamiltonian circuit in length is known as the optimal Hamiltonian circuit (OHC) for Euclidean TSP. In the 20th century, [49] proved the TSP to be NP-complete. Intriguingly, due to its
theoretical and practical values [37], TSP has been analyzed extensively in scientific fields of diverse types including computer science, combinatorial mathematics, and graph theory. Despite the extensive research done, there has to be developed polynomial algorithms that operate on all cases of the problem, and not only when NP = P [2]. Though the TSP has complications and is challenging, scholars utilize it as a platform to test an algorithms competence. Looking through the various developed methods to solve TSP, one can observe that these methods can be split into two disjoint categories, exact methods (which assure a final, optimal result) and approximation algorithms. Several researchers would rather use exact methods such as the branch and bound method [23], the dynamic programming method [1] (DM), and the integer programming method [5]. Unfortunately, there exists a limiting factor for these methods, for example, and they only have the ability to solve TSPs < 1000 cities [17]. In fact, there exists a pattern in which the scale becomes larger, the performance decreases and the time required for solving the problems increases at an exponential rate (though if utilizing potent Turing machines, one can still find the OHC within a satisfactory duration). Hence for larger scale problems, approximate algorithms provide finer results. What is even more interesting is that approximation algorithms themselves are also divided into two categories; local search algorithms and heuristic optimization methods. The first type makes use of a problem’s individual attributes to determine its local optimal solution, while the second type strives to find solutions near the optimum. In the literature, some of the mentioned local search algorithms include 2-Opt [6], the LKH [14], Inver over [12], LK [25], and 3-Opt [24]. The artificial immune algorithm (AIS) [16], the ant colony algorithm [20] (ACO), the simulated annealing algorithm (SA) [22], the genetic algorithm (GA) [40], particle swarm optimization (PSO) [21], and the artificial neural networks (ANN) [26], [15], [13] are common examples of developed heuristic methods. The primary reason as to why methods such as ones above-mentioned are utilized is that they do not rely on the problem itself, which enables the methods to exhibit potent global search capabilities. As can be seen, each type has separate qualities that aid in solving TSP. As such, various scholars have synthesized the two types into one, thus, developing a new, hybrid algorithm which contains both types of qualities making it more effective in solving TSP. In the literature, systematic procedures produced by mixing the genetic operators and LK [30], combining the genetic algorithm with a 2-Opt [39], combining the ant colony algorithm and mutation strategy [47], proposing the discrete glow worm swarm algorithm (DGSO) [50], proposing an improved and complete 2-Opt (Complete2-Opt.C2OPT) [10], proposing the honey bee mating algorithm [28], proposing the discrete cuckoo search algorithm (DCS) [33], and the hybrid genetic algorithm which contains two optimization strategies (O(n) and O(n^3)) [46] are examples of what scholars have developed to optimize Euclidean TSP. Results attained by utilizing hybrid methods such as the improved genetic algorithm show that one can reach a satisfactory solution in fewer runs. This is also proven as the author reports the fact that by using the hybrid GA, they attained finer results than when using GA with 2-opt or the classical GA.

One of the new metaheuristic algorithms inspired by the natural law of thermodynamics and heat transfer, called Heat Transfer Search (HTS) algorithm [35] was developed to solve continuous constrained optimization problems. HTS is a population-based algorithm and attracts many researchers to solve various optimization and engineering design problems such as structural optimization [43],
sizing optimization of truss structures [7], and multi-objective engineering design problems [42].

As seen from the above researches it can be noted that still TSP problems are not addressed completely and efficient method is required to solve TSP effectively. Many real-life applications can be modelled as discrete complex problems like TSP, and this has motivated us to develop a discrete version of HTS in order to deal with such problems. To add to the collection of efficient methods that can solve TSP in a comparable, if not better fashion, a discrete heat transfer search (DHTS) is proposed within this paper. This method contains three separate phases through which the HC can be updated. The proposed method is developed in such as way that it takes care of the local and global search for the optimum solution. To validate its ability, a benchmark problem (from TSPLIB) is used to judge the algorithm, and the results acquired infer that the proposed algorithm can attain the nearest OHC.

The rest of the paper is structured as follows. Section 2 explains the TSP, Section 3 introduces the basic HTS, Section 4 proposes/explains the DHTS, while results are presented and discussed in Section 5, and Section 6 gives the conclusions and future work.

2. The traveling salesman problem. The traveling salesman problem (TSP) is a paradox in the way that it seems effortless to understand, but challenging to solve. For that reason, it is one of the most cited NP-hard combinatorial optimization problems. Graph theory describes this problem as a weighted graph (WG) where variables are defined as follows, \( G = (V, E, d) \) in which \( V = (v_1, v_2, \ldots, v_m) \) (vertex set), \( v_i (1 \leq i \leq m) \) is the vertex, \( E = \{e_{ij} \}_{m \times m} \) (edges set) where \( e_{ij} (1 \leq i, j \leq n) \) is the edge which links the two distinct vertices \( v_i \) and \( v_j \), and \( d : E \rightarrow R^+ \) (weight function) where \( d \) is usually a factor such as cost or distance. \( d \) is vital in determining whether TSP is symmetrical (which according to a report [14] has been proven to be more challenging) or not. If \( d_{ij} = d_{ji} \), then it is symmetric, if not it is asymmetric. The reason for the report concluded the fact regarding the challenging nature is that when testing, it was found that within a specific time frame only 7397 cities of symmetrical TSP were resolved vs. 500,000 cities.

It is found that given a weighted graph of a symmetrical TSP, including the vertices, there are a total of \((m-1)!/2\) HC's. The OHC is selected based on which HC is smallest in length. As stated in a previous section, if one is to discover the shortest Hamilton circuit (OHC) of a TSP in which the vertices are visited once, then they achieved the overall goal. \( HC = (v_1, v_2, \ldots, v_m, v_1) \) if the circuit includes \( m \) vertices. From there, one can calculate the minimal \( f(HC) \) by adding the sum of the entire Euclidean distances and the \( d \) between each city from same path Hamiltonian circuit. The computational model of symmetrical TSP is illustrated below in equation 1.

\[
f(HC) = \sum_{i=1}^{m-1} d(v_i, v_{i+1}) + d(v_m, v_1) \tag{1}
\]

\( d \) is the Euclidean distance, between any two cities with coordinates \((x_1, y_1)\) and \((x_2, y_2)\), and can be calculated using the following equation.

\[
d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2}
\]
Later in the work, one can see the ability of the proposed method when solving a symmetrical benchmark retrieved from TSPLIB.

3. **Basic heat transfer search algorithm.** The HTS algorithm, proposed in [35], operates by following and understanding the thermal equilibrium of systems. This equilibrium is achieved as a result of applying the natural law of thermos dynamics to systems. The law states “A system will always try to be in equilibrium with its surroundings”, meaning that for thermos dynamically imbalanced systems, they will attempt to achieve thermal equilibrium by initiating heat transfers between themselves and their environment. The three kinds of heat transfer, namely, conduction, convection, and radiation, play a key role in achieving thermal equilibrium. The HTS algorithm utilizes the conduction phase, the convection phase, and the radiation phase to attain an equilibrium state. In the HTS algorithm, all three forms of heat transfer have equal probability to transfer the heat. As a result, during the course of optimization, the form of heat transfer is determined randomly for each generation.

Like other methods which aim to solve TSP, the HTS algorithm is a population-based technique, which begins solving by first initiating a randomly generated population. In the system, variables are defined as $n$ is the number of molecules/photons (i.e., population size) and $m$ is equal to the temperature level (i.e., design variables). In the following phase, the population is updated for each of the generations ($g$) by utilizing one of the randomly chosen forms of heat transfer. It is important to note that the updated solution obtained by the HTS algorithm is accepted only if it is a finer functional value. If the solution attained is identical to an existing value, they are substituted by a random existing solution. This method of replacing solutions ensures that the fittest solution can be attained by performing the difference between the current solution and either of the best solutions, another random solution, or the mean value of solutions from the population. A detailed description of the three phases used in the HTS algorithm is explained below. In the conduction phase, heat transfer occurs as a result of the conduction between the molecules of the system. The more energetic molecules transfer their heat to the less energetic molecules until a state of thermal equilibrium is reached. Not only has that, but conduction, which follows Fourier’s law of heat conduction, also occurred between a system and its surroundings when they are in physical contact. This phase, which follows Newton’s law of cooling, occurs between a system and an adjacent fluid. Heat transfer occurs as a result of the convection between the system and the adjacent fluid in motion, which means that the system’s temperature (the mean temperature) interacts with the adjacent fluids temperature (the surroundings) to achieve a state of thermal equilibrium. Here, the best solution is the surrounding. The radiation phase is unique in that the heat transfer occurs as a result of the radiation emitted in the form of electromagnetic waves/photons due to temperature level. This means that the system interacts with either the surrounding temperature (surrounding= best solution) or within the system itself (other solution) to establish thermal equilibrium. One must know that anybody above the temperature of absolute zero emits some form of radiation. Stefan-Boltzmann states that the maximum rate of the radiation heat transfer depends on the level of the absolute temperature level. The HTS is explained in detail below in Algorithm-1.
Algorithm-1: Basic HTS algorithm

1. First, initialize population size (n), Number of design variables (m), limits on design variables (L, U), and a stopping criteria \((FE_{\text{max}}, g_{\text{max}})\) CDF, COF, RDF /* initialization */

   \[ T_{j,i} = L_{j,i} + \text{rand} \ast (U_{j,i} - L_{j,i}), \text{for}\forall j \in [1, n], \text{for}\forall i \in [1, m] /* initialize population */ \]

2. \(FE=0;\)
3. Evaluate the population then arrange the population in ascending order \(\rightarrow FE = n\)
4. While \((g < g_{\text{max}}\text{and} FE < FE_{\text{max}}) /* begin the optimization loop */\)
5. \(R=\text{rand} \in [0, 1] /* R decides the probability for the selection of phases */\)
6. \(j = 1 : n /* update the population in the conduction phase */\)
7. \(k \in [1, n] \neq j /* select any random solution */\)
8. \(i \in [1, m] /* Select any random design variable */\)
9. if \(F(T_{j,i}) > F(T_k)\)
10. if \(g \leq g_{\text{max}}/\text{CDF} \rightarrow T_{j,i}' = T_{k,i} + (-R_{k,i}^2)\)
11. else \(T_{j,i}' = T_{k,i} + (-r_{k,i})\)
12. Evaluate \(f(T_{j,i}') \rightarrow FE = FE + 1\)
13. \(f(T_{j,i}') < f(T_j) \leftrightarrow T_j = T_{j,i}' /* greedy selection */\)
14. Else
15. if \(g \leq g_{\text{max}}/\text{CDF} \rightarrow T_{k,i}' = T_{j,i} + (-R_{j,i}^2)\)
16. else \(T_{k,i}' = T_{j,i} + (-r_{j,i})\)
17. Evaluate \(f(T_{k,i}') \rightarrow FE = FE + 1\)
18. \(f(T_{k,i}') < f(T_k) \leftrightarrow T_k = T_{k,i}' /* greedy selection */\)
19. end if
20. else if \(0.3333 \leq R \& \& R \leq 0.6666 \)
21. \(T_{j,i} \rightarrow T_{j,i} + R \ast (T_{k,i} - T_{j,i}) /* update the population in the radiation phase */\)
22. if \(F(T_{j,i}) > F(T_k)\)
23. if \(g \leq g_{\text{max}}/\text{RDF} \rightarrow \forall i : T_{j,i}' = T_{j,i} + R \ast (T_{k,i} - T_{j,i})\)
24. else \(\forall i : T_{j,i}' = T_{j,i} + r_i \ast (T_{k,i} - T_{j,i})\)
25. else
26. if \(g \leq g_{\text{max}}/\text{RDF} \rightarrow \forall i : T_{j,i}' = T_{j,i} + R \ast (T_{j,i} - T_{k,i})\)
27. else \(\forall i : T_{j,i}' = T_{j,i} + r_i \ast (T_{j,i} - T_{k,i})\)
28. endif
29. Evaluate \(f(T_{j,i}') \rightarrow FE = FE + 1\)
30. \(f(T_{j,i}') < f(T_j) \leftrightarrow T_j = T_{j,i}' /* greedy selection */\)
31. else /* update the population in the convection phase */
32. if \(g \leq g_{\text{max}}/\text{CDF} \rightarrow \forall i : T_{j,i}' = T_{j,i} + R \ast (T_{j,i} - T_{k,i})\)
33. else \( \forall i : T'_{j,i} = T_{j,i} + R \ast (T_{j,1} - T_{j,mean} \ast (1 + r_i)) \)
34. Evaluate \( f \left( T'_j \right) \rightarrow FE = FE + 1 \)
35. \( f \left( T'_j \right) < f (T_j) \leftrightarrow T_j = T'_j / * \text{greedy selection} / * \)
36. end if
37. end for
38. for \( j=1:2:n / * \text{remove duplicate solution} / * \)
39. If \( T_j = T_{j+1} \rightarrow T_{j+1,i} = L_i + \text{rand} \left( U_i - L_i \right) \), not\( \forall i \rightarrow FE = FE + 1 \)
40. endfor
41. endwhile

It can be observed that the basic HTS is designed to solve continuous problems. So, effective modifications are required in the basic version of HTS so that it can solve discrete optimization problems. The next section highlights the development of discrete version of HTS.

4. **Discrete heat transfer search (DHTS) algorithm.** Since the DHTS algorithm is built upon the same concept as the HTS it also has three phases of heat transfer which are also used to improve the HC of the TSP. This section briefly explains the three various phases of DHTS.

4.1. **Conduction phase:** One of the differing points between the HTS and DHTS is that in the proposed method, each Hamilton circuit is taken as the number of molecules of an object (population). The current molecule is denoted as \( HC_j \) where \( j = 1,2,\ldots,n \); and \( HC_k \) is a molecule chosen randomly where \( j \neq k; k \in (1,2,\ldots,n) \). In the case that \( HC_j < HC_k \) then \( HC_k \) will receive energy from \( HC_j \) and update itself to attain equilibrium. If it is the opposite, then \( HC_j \) is the one receiving some energy and update itself. This phase modifies the temperature of the molecules depending on the temperature of neighborhood molecules. Usually, the neighborhood search is utilized for continuous improvement of the solution. Within the DHTS, the conduction factor (CDF) represents the number of edges which are exchanged until no further improvement occurs in the HC. In the proposed systematic procedure, to improve the temperature of the molecules, the CDF-opt local search mechanism is utilized. Equations 3 and 4 showcase the mathematical formulation of the updated molecule.

\[
HC_j = \text{updated}HC_k, \text{if} F(HC_j) > F(HC_k) \\
HC_j = \text{updated}HC_j, \text{if} F(HC_j) < F(HC_k)
\]

DHTS conduction phase is explained as given in Algorithm-2.

---

**Algorithm-2: Conduction phase of DHTS**

1. Assume conduction factor(CDF)
2. CDFFoptConduction(HC, i, j)
3. If \( d(v_i,v_j) + d(v_{i+1},v_{j+1}) < d(v_i,v_{i+1}) + d(v_j,v_{j+1}) \)
4. \( (v_i, v_{i+1}) \) and \( (v_j, v_{j+1}) \) will be deleted
5. \( (v_i, v_j) \) and \( (v_{i+1},v_{j+1}) \) will be added
6. Reverse the part between cities \( v_{i+1}, v_j \)
7. New_HC= HC[1:i-1]+reverse of HC[i:j]+HC[j:m];
8. end if
9. repeat until no improvement is made
10. start again:
11. best_distance = calculateTotalDistance(existing_route)
12. for i = 0:m-CDF
13. for j = i + 1:m
14. if \( F(HC_j) > F(HC_k) \)
15. new_HC = CDFoptConduction( HC_k , i , j )
16. new_distance = calculateTotalDistance(new_route)
17. else
18. new_HC = CDFoptConduction( HC_j , i , j )
19. new_distance = calculateTotalDistance(new_route)
20. end if
21. if new_distance < best_distance
22. HC_j = new_HC
23. goto start again
24. end if
25. end for
26. end for
27. end while

4.2. The convection phase: Like the basic HTS, the surroundings in the convection phase is considered to be the best solution. As such, the temperature of the molecules also interacts with the adjacent fluid temperature (the surrounding) to attempt to attain thermal equilibrium. In the proposed procedure, the convection factor (COF) signifies the number of cities saved from the current molecules \( HC_k \) from the sub-tour, from when this last one crosses with the surrounding \( HC_j \). Since the surrounding = best solution, when \( HC_k > HC_j \), the current molecule receives energy from the surroundings and heats up by adopting a number of cities from the surrounding \( HC_j \). The Convection phase is explained in Algorithm-3.

Algorithm-3: Convection phase of DHTS

1. \( m=\)number of molecules/cities
2. Assume convection factor (COF)
3. for k=1:m
4. Choose molecules \( HC_k , HC_j \ k \neq j , j = 1 \)
5. for i = COF:m-1
6. \( u_k = HC_k (i) \)
7. \( u_{k+1} = HC_k (i + 1) \)
8. \( u_j = HC_j (p) , p = position (u_k in HC_j) \)
9. \( u_{j+1} = HC_j (p + 1) , \ if p = m \rightarrow (p + 1) = 1 \)
10. if \( d(u_k , u_{k+1}) > d(u_j , u_{j+1}) \)
11. \( HC_k (i + 1) = HC_j (p + 1) \)
12. end if
13. end for
14. end for

4.3. The radiation phase: The third phase within the DHTS operates in the same fashion as that of the standard HTS. Since the occurring heat transfer is
a result of the radiation emitted (form of electromagnetic waves or photons) due to temperature, the system colludes with either the surrounding temperature (best solution) or within the system itself (another solution) to reach thermal equilibrium. Within this phase, the radiation factor (ROF) operates in a manner similar to COF. The current molecule ($HC_k$) colludes with $HC_j$, which may either be the surroundings or another molecule within the system. In the case that $HC_k > HC_j$, the current molecule will receive energy and adopt city of $HC_j$, if not, then $HC_k$ will not adopt a city from $HC_j$. The radiation phase within the proposed within is explained in Algorithm-4.

Algorithm-4: Radiation phase of DHTS

1. $m=$number of molecules/cities
2. Assume radiation factor (RDF)
3. for $k=1:m$
4. select molecules $HC_k, HC_j, k \neq j, j=1, 2, \ldots , m$
5. for $i =$ RDF: m-1
6. $u_k = HC_k(i)$
7. $u_{k+1} = HC_k(i+1)$
8. $u_j = HC_j(p), p =$ position ($u_k$in$HC_j$)
9. $u_{j+1} = HC_j(p+1), if p = m \rightarrow (p + 1) = 1$
10. if $d(u_k, u_{k+1}) > d(u_j, u_{j+1})$
11. $HC_k(i+1) = HC_j(p+1)$
12. else
13. $HC_k(i+1) = HC_k(i+1)$
14. end if
15. end for
16. end for

DHTS for the travelling salesman problem is summarized in Algorithm-5.

Algorithm-5: DHTS for travelling salesman problem

1. Initialize population size ($n$), Number of design variables ($m$)=number of cities, limits on design variables($L=1, U=m$), stopping criteria ($FE_{\text{max}}, g_{\text{max}}$)
2. $CDF$, COF, RDF
3. $HC_j = \text{random HC}, j \in [1,n]$ /* initialize population */
4. Calculate distance of all HC and arrange the HC in ascending order
5. While ($g < g_{\text{max}}$ and $FE < FE_{\text{max}}$) /* begin the optimization loop */
6. $R=\text{rand} \in [0,1]$ /* $R$ decides the probability for the selection of phases*/
7. for $j = 1 : n$
8. if $R \leq 0.3333$ /* update the population in the conduction phase */
9. Conduction algorithm
10. else if $0.3333 < R \& \& R \leq 0.6666$ /* update the population in the radiation phase */
11. Radiation algorithm
10. else /* update the population in the convection phase */
11. Convection algorithm
12. end if
13. end for
14. endwhile

To check the effectiveness of the proposed method, it is required to test it on benchmark traveling salesman problems. The detailed results and discussion are given in the next section.

5. Results and discussion. Since the question of ability is a deciding factor on a method’s efficiency, the DHTS is tested using 41 experimental cases of symmetrical TSP found in the TSPLIB library. Percentage error is reduced by running each case 20 times independently. MATLAB R2014a is utilized to code DHTS since it is a well-known program. For the test cases, population sizes between 15 and 200 generation are considered and the conduction factor is presumed to be 2. To ensure the existence of a two-edge exchange of HC the convection factor and radiation factor are also presumed to be 2. Down below, Table-1 showcases the numerical results obtained using the DHTS algorithm (bold numbers=test cases that DHTS reaches OHC). These results are obtained by calculating Euclidean distances between all vertices. The names of the individual symmetric TSP (ended by the number of cities in the problem) are in column one, the best result obtained using DHTS is shown in column 2, column 3 presents the average solution obtained over 20 runs, the fourth column indicates the standard deviation of acquired solutions over 20 runs, the 5th and 6th column present the percentage best solution length found over the 20 separate runs (PDbest% ) and the percentage average solution length found over 20 runs (PDavg% ), and the last column showcases the best known found in the TSPLIB.

To determine the percentage deviation of a solution to the optimal solution the following formula is used:

\[ PD_{solution}\% = \frac{solution length - optimal solution length}{optimal solution length} \times 100 \]  

In observance, one can determine that Table-1 illustrates the fact that the DHTS attains OHC for 24 TSP cases and gives near OHC for the remaining. These results also infer that the proposed procedure is capable of solving TSPs of various scales. To provide a visual presentation of the results, a bar graph is created (Figure-1) to show the differences between the acquired best solution with the optimum known. The bar graphs also uncover the fact that for problems less than 300 cities, the DHTS is capable of finding the optimum solutions successfully, while for anything larger it can still reach solutions near the optimal. Using the average value of the best solutions attained in the 20 runs, one can judge its ability in solving problems. This is only possible because the DHTS is a heuristic approach to TSP problems. The differences between the obtained mean and the known is illustrated in Figure-2 where it is observed that as the number of cities increases, the DHTS has the ability in solving decreases. The percentage deviation for the best and the mean solutions are presented in Figure-3, which also infers that the larger the number of cities, the lower the performance of the DHTS (though that does not change the fact that the DHTS can still reach near global solutions for the various TSP problems).

As it is extremely important to ensure quality, the results of the DHTS are compared with the results obtained using some of the other well-known state-of
Table-1: Results of the DHTS algorithm for symmetric TSP instances from TSPLIB

| TSP    | best | worst | mean | std  | pBest% | pAvg%  | Optimal |
|--------|------|-------|------|------|--------|--------|---------|
| n151   | 426  | 426   | 426  | 0.07017 | 0.11731 | 426    |
| berlin52 | 7542 | 7542  | 7542 | 0     | 0      | 7542   |
| a28   | 675  | 675   | 675  | 0     | 0      | 675    |
| pr76   | 108159 | 108159 | 108159 | 0 | 0   | 108159 |
| eil51  | 538  | 538   | 538  | 1.02533 | 0.130112 | 538    |
| korA100 | 21282 | 21282 | 21282 | 0 | 0      | 21282 |
| korB100 | 22272 | 22272 | 22272 | 48.6091 | 0.181564 | 22141 |
| korC100 | 20749 | 20749 | 20749 | 6.310485 | 0.011567 | 20749 |
| korD100 | 21299 | 21299 | 21299 | 40.56088 | 0.164814 | 21294 |
| korE100 | 22146 | 22146 | 22146 | 35.72459 | 0.141432 | 22068 |
| eil101  | 659  | 641   | 651  | 3.928738 | 0.333863 | 629    |
| tim105  | 13179 | 14312 | 14312 | 6.927601 | 0.0153 | 14379 |
| pr107  | 44387 | 44387 | 44387 | 35.61791 | 0.037921 | 44398 |
| pr124  | 59246 | 59246 | 59246 | 42.48362 | 0.092061 | 59237 |
| hier127 | 118728 | 118728 | 118728 | 195.1689 | 0.185996 | 118392 |
| ch130  | 6111  | 6174  | 6137 | 15.78361 | 0.016367 | 6110   |
| pr136  | 97785 | 97341 | 97564 | 290.5734 | 0.059935 | 96772 |
| pr144  | 38590 | 38544 | 38567 | 17.02286 | 0.011958 | 58337 |
| ch150  | 6528  | 6555  | 6543 | 10.56251 | 0.240029 | 6528   |
| korA150 | 26524 | 26570 | 26535 | 51.37671 | 0.231111 | 26524 |
| korB150 | 26141 | 26239 | 26179 | 37.07485 | 0.040207 | 26130 |
| pr152  | 73818 | 73764 | 73786 | 70.2301 | 0.078831 | 73682 |
| tim195  | 2332  | 2344  | 2337 | 3.972135 | 0.38743 | 602669 | 2333 |
| d186  | 15789 | 13867 | 13832 | 19.94457 | 0.037034 | 329531 | 13780 |
| korA200 | 29483 | 29483 | 29483 | 39.05324 | 0.028385 | 272296 | 29388 |
| korB200 | 39470 | 39518 | 39532 | 42.52529 | 0.112104 | 280429 | 29437 |
| tr225  | 126845 | 127077 | 126802 | 138.4326 | 0.125787 | 126643 |
| tsp225  | 2016  | 2040  | 2026 | 7.877355 | 0.268131 | 2016   |
| pr226  | 80599 | 80599 | 80599 | 101.4062 | 0.091895 | 80369 |
| gil262  | 2380  | 2401  | 2390 | 6.733838 | 0.084104 | 517241 | 2378 |
| pr264  | 49155 | 49184 | 49184 | 103.3054 | 0.009725 | 49135 |
| a280  | 2579  | 2580  | 2583 | 8.894941 | 0.170609 | 2579  |
| pr299  | 48266 | 48266 | 48266 | 27.55376 | 0.155621 | 48191 |
| tim18  | 42711 | 42556 | 42511 | 111.7893 | 0.33782 | 42711 |
| nd600  | 15400 | 15517 | 15447 | 43.6379 | 0.777875 | 1.088944 | 15381 |
| os147  | 11876 | 11859 | 11869 | 6.580397 | 0.126465 | 1.218363 | 11861 |
| pr359  | 107982 | 107982 | 107982 | 95.25912 | 0.4337 | 481827 | 107271 |
| rve757  | 6845  | 6907  | 6852 | 18.87752 | 1.683044 | 2.123642 | 6773 |
| car83  | 8941  | 9056 | 8946 | 4.715224 | 1.533046 | 1.597774 | 8086 |
| run102  | 26385 | 26424 | 263159 | 636.4372 | 1.289351 | 1.583373 | 259045 |
| navi1379 | 57724 | 57950 | 57848 | 57.97662 | 1.917441 | 2.134256 | 56638 |

the art algorithms such as various versions of the ACO like ACO with; multiple ant clans (ACOMAC) and ACOMAC with the closest neighborhood, simulated annealing, 2-opt, hybrid ACO, Taguchi method, and at the end, ACO with ABC. Not to mention the ACO. But the algorithm is also contrasted with other methods such as hybrid PSO, hybrid GA, GA with ACO, RABNET, water flow algorithm, and IVRS. The average value of the solutions attained is shown in Table-2 from which one can infer that the proposed method exceeds the compared methods in all, but for 6 problems (even then, for those 6 they are only slightly inferior).

Besides the previous work done to affirm the proposed methods ability, this method is also compared with other works on a wide range of TSP ranging from 51
Figure 1. Comparison of obtained best solutions with the known optimum solutions

Figure 2. Comparisons of mean value to the known best solutions

Figure 3. The percentage deviation for the mean and the best solutions
cities -783 cities. Using and testing 38 different TSP problems, one can see that the DHTS outperforms most algorithms including the neuro-immune network [34], the improved BA [31], GCGA with local search [48], GSA ant-colony system with PSO [3], Massutti and Castro’s method [29], HGA [46], and ACE [9]. These results are presented in Table-3 where it is observed that DHTS provides finer results for 31 TSP instances.

DHTS shows better performance over numerous-immune network [34], GCGA with local search [48], Massutti and Castro’s method citeMC09, GSA ant-colony system with PSO [3] and improved BA [31]. In this comparison, the proposed method performs better or the TSP instances excluding pr136, ch130, tsp225, pr144, and pr264 (in which HGA [46] attains finer results) and d198, eil76, and pr264 (in which ACE [9] performs better).

Below in Table-4, the comparison between DIWO [51] and DHTS is illustrated. DIWO [51] presents the results procured from the various TSP instances(51-1002 cities) in term of a % PD average. The table also demonstrates the fact that the DHTS exhibits a superior performance in contrast to the DIWO. Figure-4 is a graphical representation of the comparison between the DHTS and DIWO [51].

Table-2: Comparison of average tours of DHTS with other state of the art algorithms for different TSP cases

| Cities | Proposed-DHTS | ACOMAC [47] | ACOMAC ->S [48] | RABNET-TSP [15] | Modified RABNET-TSP [20] | GSA ACO-PSO [3] | IVRS = 2opt [17] | ACO = 2opt [17] | MACO [37] | CGA [3] | WFA with 2Opt [32] | WFA with 1Opt [32] | ACO with TSPLIB [54] | ACO with KRC [53] | HGA+1 local [57] | PSO-ACO-3Opt [37] | ACE [9] | DIWO [51] |
|--------|---------------|--------------|------------------|-----------------|---------------------------|------------------|----------------|--------------|-------------|---------|----------------|----------------|----------------|-------------|---------------|----------------|---------|-----------|
|        |               |              |                  |                 |                           |                  |                |              |             |         |                |                |                |              |               |                |         |          |
Table 3: Comparison of DHTS with neuro-immune network [34], GCGA with local search [48], Massutti and Castro’s method [29], GSA ant-colony system with PSO [3], HGA [46], ACE [9] and improved BA [31].

| Algorithm          | Optimum | DHTS | neuro-immune network | GCGA with local search | Massutti and Castro’s method | GSA ant-colony system with PSO | HGA | ACE | improved BA |
|--------------------|---------|------|-----------------------|------------------------|-----------------------------|-------------------------------|-----|-----|-------------|
| ch10               | 614     | 613  | 613.7                 | 613.5                  | 612.4                       | 610.2                         | 609 | 610 | 610.3       |
| sol50              | 965     | 964  | 964.6                 | 964.5                  | 963.7                       | 961.3                         | 962 | 963 | 962.6       |
| ch50               | 698     | 698  | 698.1                 | 698.1                  | 697.2                       | 695.7                         | 697 | 697 | 697.1       |
| rct100             | 22182   | 22182| 22181.5               | 22181.5                | 22181.3                     | 22181.0                       | 22181| 22181| 22181.3     |
| rct500             | 22181   | 22181| 22180.9               | 22180.9                | 22180.8                     | 22180.6                       | 22180| 22180| 22180.6     |
| rct1000            | 22181   | 22181| 22181.1               | 22181.1                | 22181.1                     | 22181.1                       | 22181| 22181| 22181.1     |
| rct2000            | 22181   | 22181| 22180.9               | 22180.9                | 22180.8                     | 22180.6                       | 22180| 22180| 22180.6     |
| ch100              | 11882   | 11882| 11880.9               | 11880.9                | 11880.8                     | 11880.7                       | 11880| 11880| 11880.7     |
| rct30              | 614     | 614  | 613.7                 | 613.5                  | 612.4                       | 610.2                         | 609 | 610 | 610.3       |
| sol50              | 965     | 964  | 964.6                 | 964.5                  | 963.7                       | 961.3                         | 962 | 963 | 962.6       |
| ch50               | 698     | 698  | 698.1                 | 698.1                  | 697.2                       | 695.7                         | 697 | 697 | 697.1       |
| rct100             | 22182   | 22182| 22181.5               | 22181.5                | 22181.3                     | 22181.0                       | 22181| 22181| 22181.3     |
| rct500             | 22181   | 22181| 22180.9               | 22180.9                | 22180.8                     | 22180.6                       | 22180| 22180| 22180.6     |
| rct1000            | 22181   | 22181| 22181.1               | 22181.1                | 22181.1                     | 22181.1                       | 22181| 22181| 22181.1     |
| rct2000            | 22181   | 22181| 22180.9               | 22180.9                | 22180.8                     | 22180.6                       | 22180| 22180| 22180.6     |
| ch100              | 11882   | 11882| 11880.9               | 11880.9                | 11880.8                     | 11880.7                       | 11880| 11880| 11880.7     |

6. Conclusions and future work. To conclude, this work modifies the existing heat transfer search (HTS) algorithm to incorporate discrete variables thus turning it into an employable method to solve travelling salesman problems. The excellent performance of the discrete HTS (DHTS) is validated through the testing of the method on 41 different benchmark problems retrieved from the TSP library in which the proposed algorithm yields the optimal HCs for 24 of the 41 instances and gives near optimal HCs for the remaining tests. Additionally, besides using various benchmark problems, the ability of DHTS in solving TSP is compared with other states of art algorithms which are derived from methods such as the particle swarm optimization, the genetic algorithm, artificial bee colony optimization, ant colony optimization, and other methods within that platform. The various comparisons successfully illustrate the fact that the ability of DHTS in solving TSP is comparable if not better with other states of the art methods.

As future work, the authors plan to enhance our proposed algorithm to solve other combinatorial optimization problems such as vehicle routing [44], scheduling and mixed-integer programming problems. Moreover, we intend to generalize this work to solve some kinds of TSP problems such as asymmetric [4], spherical [33] and generalized TSP [41].
Table 4: DHTS comparison with DIWO [51].

|       | PD % average |
|-------|--------------|
|       | DIWO [51]    | Proposed DHTS |
| eil51 | 0.6999       | 0.117371      |
| berlin52 | 0.0313   | 0             |
| st70  | 0.3125       | 0             |
| kroA100 | 0.0375    | 0             |
| kroB100 | 0.181564  |               |
| pr107 | 0.4837       | 0.037921      |
| pr136 | 0.94         | 0.5886        |
| kroA150 | 0.231111  |               |
| kroB150 | 0.221584  |               |
| d198  | 0.6691       | 0.329531      |
| tsp225 | 2.3949     | 0.268131      |
| pr226 | 0.2238       | 0.093195      |
| rd400 | 2.4229       | 1.088934      |
| pr1002 | 3.1873     | 1.588373      |

Figure 4: The Graphical representation of the comparisons between the proposed DHTS and DIWO [51].

REFERENCES

[1] R. E. Bellman and S. E. Dreyfus, Applied Dynamic Programming, Princeton University Press, 2015.

[2] P. Berman and M. Karpinski, 8/7-approximation algorithm for (1, 2)-TSP, Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, Society for Industrial and Applied Mathematics, 2006, 641–648.

[3] S. M. Chen and C. Y. Chien, Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques, Expert Systems with Applications, 38 (2011), 14439–14450.
[4] J. Cirasella, D. S. Johnson, L. A. McGeoch, and W. Zhang, The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators, and Tests, Algorithm Engineering and Experimentation, 2153 (2011), 32–59.
[5] S. Climer and W. Zhang, Cut-and-solve: An iterative search strategy for combinatorial optimization problems, Artificial Intelligence, 170 (2006), 714–738.
[6] G. A. Croes, A method for solving traveling-salesman problems, Operations Research, 6 (1958), 791–812.
[7] S. O. Degertekin and I. Lamberti, Heat transfer search algorithm for sizing optimization of truss structures, Latin American Journal of Solids and Structures, 14 (2017), 373–397.
[8] G. Dong, W. W. Guo and K. Tickle, Solving the traveling salesman problem using cooperative genetic ant systems, Expert Systems with Applications, 39 (2012), 5006–5011.
[9] B. Escario, J. F. Jimenez and J. M. Giron-Sierra, Ant colony extended: Experiments on the travelling salesman problem, Expert Systems with Applications, 42 (2015), 390–410.
[10] P. Gang, I. Iimura and S. Nakayama, An evolutionary multiple heuristic with genetic local search for solving TSP, International Journal of Information Technology, 14 (2008), 1–11.
[11] M. Gunduz and M. S. Kiran, A hierarchic approach based on swarm intelligence to solve the traveling salesman problem, Turkish Journal of Electrical Engineering & Computer Sciences, 23 (2015), 103–117.
[12] T. Guo and Z. Michalewicz, Invor-over operator for the TSP-proceedings of the 5th parallel problem solving from nature conference, (1998), 1498–1520.
[13] F. Han, Q. H. Ling and D. S. Huang, An improved approximation approach incorporating particle swarm optimization and a priori information into neural networks, Neural Computing and Applications, 19 (2010), 255–261.
[14] K. Helsgaun, An effective implementation of the Lin Kernighan traveling salesman heuristic, European Journal of Operational Research, 126 (2000), 106–130.
[15] D. S. Huang and J. X. Du, A constructive hybrid structure optimization methodology for radial basis probabilistic neural networks, IEEE Transactions on Neural Networks, 19 (2008), 2099–2115.
[16] J. E. Hunt and D. E. Cooke, Learning using an artificial immune system, Journal of Network and Computer Applications, 19 (1996), 189–212.
[17] D. S. Johnson and L. A. McGeoch, Experimental analysis of heuristics for the STSP, The Traveling Salesman Problem and its Variations, Springer, Boston, MA, 12 (2002), 369–443.
[18] K. Jun-man and Z. Yi, Application of an improved ant colony optimization on generalized traveling salesman problem, Energy Procedia, 17 (2012), 319–325.
[19] W. Junqiang and O. Aijia, A hybrid algorithm of ACO and delete-cross method for TSP, Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on. IEEE, 2012, 694–1696.
[20] J. Junzhong, H. Zhen, L. Chunnian and D. Qiguo, An ant colony algorithm based on Multiple-Grain representation for the traveling salesman problems [J], Journal of Computer Research and Development, 3 (2010), 9.
[21] J. Kennedy and R. C. Eberhart, A discrete binary version of the particle swarm algorithm, Systems, Man, and Cybernetics, Computational Cybernetics and Simulation., 1997 IEEE International Conference on, IEEE, 5 (1997), 4104–4108.
[22] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, Science, 220 (1983), 671–680.
[23] E. L. Lawler and D. E. Wood, Branch-and-bound methods: A survey, Operations Research, 14 (1966), 699–719.
[24] S. Lin, Computer solutions of the traveling salesman problem, The Bell System Technical Journal, 44 (1965), 2245–2269.
[25] S. Lin and B. W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, Operations Research, 21 (1973), 498–516.
[26] X. Liu and C. Xiu, A novel hysteretic chaotic neural network and its applications, Neurocomputing, 70 (2007), 2561–2565.
[27] M. Mahi, K. Baykan and H. Kodaz, A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem, Applied Soft Computing, 30 (2015), 484–490.
[28] Y. Marinakis, M. Marinaki and G. Dounias, Honey bees mating optimization algorithm for the Euclidean traveling salesman problem, Information Sciences, 181 (2011), 4684–4698.
[29] T. A. S. Masutti and L. N. de Castro, A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem, Information Sciences, 179 (2009), 1454–1468.

[30] P. Merz and B. Freisleben, Genetic local search for the TSP: New results, Evolutionary Computation, 1997., IEEE International Conference on. IEEE, 159–164 1997.

[31] E. Osaba, X. S. Yang, F. Diaz, P. Lopez-Garcia and R. Carballedo, An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems, Engineering Applications of Artificial Intelligence, 48 (2016), 59–71.

[32] Z. A. Othman, A. I. Srouf, A. R. Hamdan and P. Y. Ling, Performance water flow-like algorithm for TSP by improving its local search, International Journal of Advancements in Computing Technology, 5 (2013), 126.

[33] X. Ouyang, Y. Zhou and Q. Luo, A novel discrete cuckoo search algorithm for spherical traveling salesman problem, Applied Mathematics & Information Sciences, 7 (2013), 777–784.

[34] R. Pasti and L. N. de Castro, A neuro-immune network for solving the traveling salesman problem, Neural Networks, 2006. ICNN’06, International Joint Conference on IEEE, (pp. 3760–3766, 2006.

[35] V. K. Patel and V. J. Savsani, Heat transfer search (HTS): A novel optimization algorithm, Information Sciences, 324 (2015), 217–246.

[36] M. Peker, B. EN, and P. Y. Kumru, An efficient solving of the traveling salesman problem: the ant colony system having parameters optimized by the Taguchi method, Turkish Journal of Electrical Engineering & Computer Sciences, 21 (2013), 2015–2036.

[37] A. Rodriguez and R. Ruiz, The effect of the asymmetry of road transportation networks on the traveling salesman problem, Computers & Operations Research, 39 (2012), 1566–1576.

[38] Y. Saji and M. E. Riffi, A novel discrete bat algorithm for solving the travelling salesman problem, Neural Computing and Applications, 27 (2016), 1853–1866.

[39] F. Samanlioglu, W. G. Ferrell Jr and M. E. Kurz, A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem, Computers & Industrial Engineering, 55 (2008), 439–449.

[40] J. Shu, Z. Zhao and Q. Dai, Genetic algorithm for TSP, Operations Research and Management Science, 1 (2004), 4.

[41] L. V. Snyder and M. S. Daskin, A random-key genetic algorithm for the generalized traveling salesman problem, European Journal of Operational Research, 174 (2006), 38–53.

[42] M. A. Tawhid and V. Savsani, $\epsilon$-constraint heat transfer search ($\epsilon$-HTS) algorithm for solving multi-objective engineering design problems, Journal of Computational Design and Engineering, 5 (2018), 104–119.

[43] G. Tejani, V. Savsani and V. Patel, Modified sub-population based heat transfer search algorithm for structural optimization, International Journal of Applied Metaheuristic Computing (IJAMC), 8 (2017), 1–23.

[44] P. Toth and D. Vigo, Vehicle Routing: Problems, Methods, and Applications, Society for Industrial and Applied Mathematics, 2014.

[45] C. F. Tsai, C. W. Tsai and C. C. Tseng, A new hybrid heuristic approach for solving large traveling salesman problem, Information Sciences, 166 (2004), 67–81.

[46] Y. Wang, The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem, Computers Industrial Engineering, 70 (2014), 124–133.

[47] J. Yang, X. Shi, M. Marchese and Y. Liang, An ant colony optimization method for generalized TSP problem, Progress in Natural Science, 18 (2008), 1417–1422.

[48] J. Yang, C Wu, H. P. Lee and Y. Liang, Solving traveling salesman problems using generalized chromosome genetic algorithm, Progress in Natural Science, 18 (2008), 887–892.

[49] W. Zhang and R. E. Korf, A study of complexity transitions on the asymmetric traveling salesman problem, Artificial Intelligence, 81 (1996), 223–239.

[50] Y. Q. Zhou, Z. X. Huang and H. X. Liu, Discrete glowworm swarm optimization algorithm for TSP problem, Dianzi Xuebao(Acta Electronica Sinica), 40 (2012), 1164–1170.

[51] Y. Zhou, Q. Luo, H. Chen, A. He and J. Wu, A discrete invasive weed optimization algorithm for solving traveling salesman problem, Neurocomputing, 151 (2015), 1227–1236

Received November 2017; revised February 2018.

E-mail address: poonam.savsani@gmail.com
E-mail address: Mtawhid@tru.ca