

2018 INFORMS International Conference
Taipei, Taiwan, June 17–20, 2018
© 2018 INFORMS| ISBN 978-0-9906153-1-6
https://doi.org/10.1287/infp.2018.XXXX
pp. 000–000

The Solution Attractor Theory of Local Search System: The Traveling Salesman Problem Case

Weiqi Li
University of Michigan-Flint, 303 E. Kearsley Street, Flint, MI 48439, USA, weli@umflint.edu

Abstract A local search algorithm can be treated as a discrete dynamical system, and therefore its search behavior can be studied from the perspective of dynamical systems. The attractor theory in dynamical systems provides the necessary and sufficient theoretical foundation to study the search behavior of local search systems. This paper introduces the solution-attractor theory of local search system. In a local search system, search trajectories converge into a solution attractor in the solution space. This solution-attractor theory not only provides a model to describe the search behavior of a local search system, but also offers an important method to solve the Traveling Salesman Problem (TSP) efficiently with optimality guarantee.

Keywords combinatorial optimization, global optimization, traveling salesman problem, local search, dynamical systems

1. Introduction

Although both the TSP and local search have huge literature, there is still a variety of open problems. The study of local search for the TSP continues to be an important problem in computer science, combinatorial optimization, and computational mathematics. In this paper, a local search algorithm is treated as a discrete dynamical system, and therefore its search behavior is studied from the perspective of dynamical systems.

The goal of dynamical systems analysis is to capture the distinctive properties of certain points or regions in the state space for a given dynamical system. The dynamical systems theory has discovered that many dynamical systems exhibit attracting behavior: all initial states tend to evolve towards a single final state or a set of states. An *attractor* is a single or a set of points in the state space, invariant under the dynamics, towards which neighboring states in a given basin of attraction asymptotically approach in the course of dynamic evolution [2, 9, 11, 13, 19, 23]. The attractor theory in dynamical systems provides the necessary and sufficient theoretical foundation to study the search behavior of local search systems.

This paper introduces the solution attractor theory of local search systems that describes the search behavior of a local search system. The TSP is used as a study problem. The fundamental concept within the solution attractor theory is the notion that a local search system has an attractor, which drives the search trajectories to converge to a small region in the solution space that contains the most promising solutions to the problem at hand.

In a local search system, all search trajectories will eventually converge to a limit set A that is invariant with respect to a certain dynamical search law $f(t)$. This limit set is called a *solution attractor* of the local search system for a given problem instance. For the TSP, let s denote a tour in the solution space S, $f_t(s)$ denote a local search system, A denote the solution attractor of $f_t(s)$, there exists a solution attractor $A \subset S$ that attracts all search

trajectories. That is, $f_t(s)$ tends towards A as search time $T \to \infty$. No matter where a search trajectory is initiated in the solution space, it must end up at one point of the limit set A. Let s' be a locally optimal tour and s^* a globally optimal tour. The edges contained in a globally optimal tour s^* are called G-edges, namely the set G. δ denotes a bonds distance between two tours, and g denotes a subset of G in a locally optimal tour s'. Thus, the solution attractor of a local search system for the TSP can be characterized by the following properties:

(1) *Attractivity*: i.e., $\forall s_i \in S$, $f_t(s_i) \in A$ for sufficiently long time t;
(2) *Invariance*: i.e., $\forall s'_i \in A$, $f_t(s'_i) = s'_i$ for all time t;
(3) *Compactness*: i.e., the distance between the globally optimal tour s^* and any tour $s_i \in A$ is less than or equal to δ;
(4) *Irreducibility*: i.e., the solution attractor A consists of limit number of invariant points;
(5) *Neighborhood topology*: i.e., all tours in A are neighbors by sharing some common edges;
(6) *Completeness*: i.e., for a sufficiently large K, all K locally optimal tours together contain the whole set of G; that is, $g_1 \cup g_2 \cup ... \cup g_K = G$.

The focus of this paper is not on formally proving these properties, but on a way of thinking about how using these properties to solve the TSP efficiently. So far there exist no efficient algorithms to solve the NP-hard TSP, only the brute-force search can provide a solution. Therefore, the general question this paper attempts to answer is "The TSP can be solved by a brute-force search algorithm, but can it be done efficiently?".

This paper is organized as follows. Section 2 gives a general definition of the TSP and describes the search behavior of a local search system on the TSP. Section 3 introduces the solution attractor theory of local search systems and describes some important properties of the solution attractor. This section also describes an important data structure, called *hit-frequency matrix E*, that is used to capture the convergent evolution of the search trajectories. Section 4 presents an attractor-based search system that can be used to solve the general TSP efficiently. The final section concludes this paper.

2. The TSP and Local Search System

In this paper, the TSP is defined in more general term as a complete graph $Q = (V, E, C)$, where $V = \{v_i : i = 1, 2, ..., n\}$ is a set of n nodes; $E = \{e(i, j) : i, j \in V, i \neq j\}$ is an $n \times n$ edge matrix containing the set of edges that completely connects the n nodes; and $C = \{c(i, j) : i, j \in V, i \neq j\}$ is an $n \times n$ symmetric cost matrix holding a set of costs associated with the set of edges. Usually the edge matrix E is not necessary to be included in the TSP definition because the TSP is a complete graph. However, the matrix E is included in our definition because it is a natural data structure that we use to study the asymptotic behavior of search trajectories through the temporal evolution of the edge configuration in the data structure. We assume that a TSP instance contains $h (h \geqslant 1)$ globally optimal tours in the solution space S and denote S^* as the set of h globally optimal tours. Therefore, the objective of the TSP is to find the whole set of optimal tours $S^* \subset S$:

$$S^* = arg\ \{\min_{s \in S} f(s)\} \tag{1}$$

Any optimal solution $s^* \in S^*$ optimizes the objective function $f(s)$ and satisfies a set of restrictions. The TSP in the classic definition is looking for any single best tour in S^*; while the TSP in the general definition is searching for all best tours in S^*, and the salesman wants to know what all best alternative tours are available. Usually, for a given TSP instance, we don't know the number of optimal tours in the solution space until we find them. Obviously, this general TSP becomes even more difficult to solve. To solve this general TSP, we need a search system that converges not just in value but also in solution. Convergence *in value*

means that a search system can find any one of the optimal solutions in the solution space eventually. Convergence *in solution* means that the search system can identify over and over the same optimal solutions in the solution space. Obviously, finding the same set of optimal solutions over and over is the essential requirement for global optimization.

A local search system iteratively explores the neighborhoods of solutions, trying to improve the current solution by local changes in the direction towards the optimal point. A local search process leads to a sequence of solutions $s_0, s_1, s_2, ..., s_\infty$, where s_0 is an arbitrary initial solution and s_∞ is the final solution at the end of search. This time series defines a search trajectory and represents the part of the solution space that is searched by the search trajectory.

In a local search system, there is a great variety of ways to construct initial points, choose candidate moves, and define criteria for accepting candidate moves. Most local search systems are based on randomization. The initial point is randomly constructed and the next move is chosen from multiple possibilities usually based on a random number. Therefore, local optimality depends on the initial point, the neighborhood function used in the search process, and the randomness in the search system. These factors determine the dynamical search law $f(t)$ in the search system. Randomization makes the search system become a randomized system. A pervasive theme about randomized system is that a small random sample generated by the randomized system is a good representative of the population as a whole. Randomization is also an effective way to achieving diversification in search. There are no two search trajectories that are exactly alike in such a search system. K search trajectories can explore a wide region of the solution space, and the locally optimal solutions generated by these search trajectories are good representative of the solutions in A. However, no matter where a search trajectory starts, all search trajectories tend towards the same small region in the solution space if the problem instance is unimodal, or several small regions if the problem instance is multimodal. Here unimodal means that the solution space contains only one optimal point, while multimodal means that the solution space has several optimal points.

In the past decades, many researchers have empirically studied the converging behavior of local search systems [5, 10, 12, 14, 16, 17, 18, 22, 24, 25]. They use "big-valley" landscape of the search space to describe the phenomenon that multiple search trajectories converge to a small region in the solution space. They observed that the locally optimal points occur relatively close to each other, and to the globally optimal point. For the TSP, the properties of the "big-valley" landscape were studied based on the analysis of overlaps and distance between the locally optimal tours. Here the overlap, called *bonds*, between two tours is defined as the fraction of edges that are common to both tours; and the *bonds distance* is defined as the difference between the number of nodes and the bonds. It was found that the bonds distance between any pair of locally optimal tours satisfies a normal distribution with very low average and the locally optimal tours consists most of short edges. From dynamical systems perspective, the "big-valley" structure actually is a solution attractor of the search system that drives the local search trajectories to converge into a small region in the solution space [15]. The long-term dynamics of a local search process is captured in a limit set. The significant contribution of this paper is to combine all these ideas into one central theory: the solution attractor of a local search system.

3. Solution Attractor of Local Search System

The theory of dynamical systems is an extremely broad area of study. A dynamical system can be considered to be a model of describing the temporal evolution of the system. A dynamical system consists of a state space S, a set of times T, and a rule f for evolution, $f : S \times T \to S$ that gives the consequents to a state $s \in S$. The mathematics typically specifies a dynamical law f that determines how the values of a set of states evolve through time,

and thus an attractor is defined as a set of numerical values toward which a system tends to evolve, for a wide variety of starting conditions of the system [2, 3, 4, 6, 7, 9, 11, 13, 19, 20, 23].

The solution attractor A of a local search system can be simply viewed as the set of solutions, to which all initial points evolve under iterates of the search function $f(t)$. In a local search system, a search trajectory usually stops at a point in the solution space, which is called a locally optimal point. A different search trajectory explores different region of the solution space and stops at a different locally optimal point. Therefore, a solution attractor A of a local search system is a subset of the solution space that contains the final points of all search trajectories for the given problem instance; that is, the solution attractor A contains all locally optimal solutions. A single search trajectory typically converges to either one of the points in A. The set A is asymptotically stable because all search trajectories approach this set as search time $t \to \infty$. Since a globally optimal solution s^* is a special case of locally optimal solutions, it is reasonable to believe that $s^* \in A$ and $\forall s_i \in A, s^* < s_i$. This property leads to the following proposition: A globally optimal solution s^* is the center of a solution attractor A, surrounded by locally optimal solutions within δ distance in the solution attractor.

A multimodal optimization problem has $h(h > 1)$ globally optimal solutions, and therefore a search system for a multimodal problem will have h solution attractors. A search trajectory converges to either one of these h solution attractors. According to dynamical systems theory, the closure of an arbitrary union of attractors is still an attractor. Therefore, the union of the h solution attractors is still a solution attractor; that is, $A_1 \cup A_2 \cup ... \cup A_h = A$. Each solution attractor A_i has its own basin of attraction and contains a globally optimal solution s_i^*. All search trajectories will be distributed to these h different solution attractors. FIGURE 1 visually summaries the concepts of search trajectories and solution attractors in a local search system for a multimodal optimization problem. Illustrating search trajectories, solution attractors and the solution space as a three-dimensional object is a valid metaphor for understanding how a local search system might proceed, how search trajectories converge to solution attractors, and how a solution attractor is formed.

Basically, a local search system lacks the information needed to find all G-edges because it does not have the data structure that can be used to collect such information. In the TSP, an edge is a connection between two nodes and many tours go through it, and thus it contains important information about many tours. The edge matrix E in the TSP definition

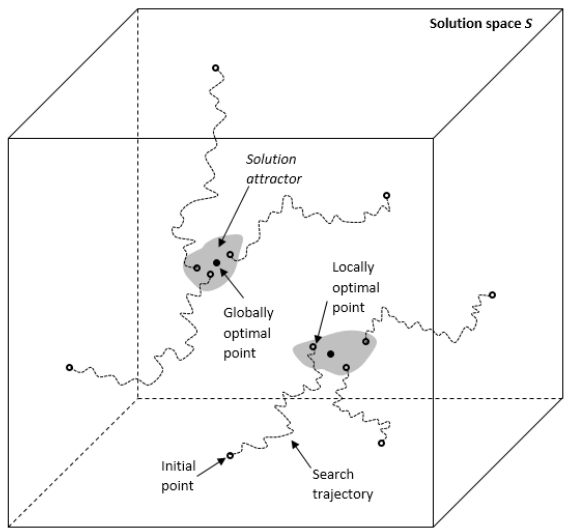


FIGURE 1. Illustration of search trajectories and solution attractors.

is an exploitable data structure. We define that, in the edge matrix E, the element $e(i,j)$ in the row i and column j contains value 1 if and only if a tour goes from node v_i directly to node v_j, otherwise it contains zero (or blank). For K tours, the matrix E can be used to record the number of times the edge $e(i,j)$ is hit by these K tours. The value in an element $e(i,j)$ of the matrix E is defined as the number of occurrence of the edges $e(i,j)$ in these K tours:

$$e(i,j) = \sum_{k=1}^{K} e^k(i,j) \tag{2}$$

In this way, the edge matrix E actually serves as a learning mechanism and memory architecture that can catch rich information about the search behavior of the search system, based on the hit-frequency values in its elements. Thus, the matrix E is also called *hit-frequency matrix E* when being used for this purpose.

FIGURE 2, using a 50-node example, visualizes the global dynamics of the search system through the convergence of edge configuration in the matrix E. The TSP instance is searched by 100 local search trajectories concurrently. The hit-frequency value in the element $e(i,j)$ of the matrix E is converted into a unit of half-tone information, a value that we interpret as a number H_{ij} somewhere between 0 and 1. The value 1 corresponds to black color and 0 to white color. In this way, the image in the matrix E can reveal the convergent state of the edge configuration in the search trajectories at any moment during the search process. FIGURE 2(a) shows the image of the edge configuration of the matrix E for the 100 random initial tours. Since each element has equal chance to be hit by these initial tours, almost all elements in the matrix E are hit by the 100 initial tours and all elements have very low half-tone values. FIGURE 2(b) and (c) shows the image of edge configuration of the matrix E when all search trajectories complete 2000 iterations and 5000 iterations, respectively. At 5000^{th} iteration, many elements of the matrix E become white color, while some other elements have much higher half-tone values. This example demonstrates that as search continues, the number of edges hit by the search trajectories becomes smaller and smaller. This phenomenon means that all search trajectories are moving closer and closer to each other, and they become increasingly similar. The distribution of the hit-frequency on the edges in the matrix E gives us a global description of the asymptotic behavior of the search system. The long-term dynamics of K search trajectories would always converge to a fixed set of edges. If K is sufficient large, the evolution of the edge configuration of the matrix E will be toward one fixed pattern that is independent from where and how these search trajectories proceed in the solution space.

Let $W = n(n-1)/2$ denote total number of edges in the matrix E, $\alpha(t)$ denote the number of the edges that are hit by all K search trajectories at time t, $\beta(t)$ denote the number of the edges that are hit by one or some of K search trajectories, and $\gamma(t)$ denote the number of the edges that have no hit, then we have, at time t,

$$W = \alpha(t) + \beta(t) + \gamma(t) \tag{3}$$

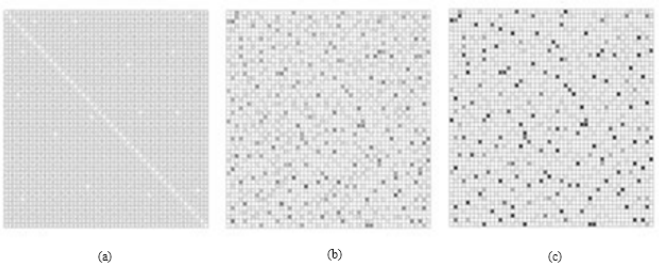


FIGURE 2. Visualizing the convergent dynamics of edge configuration in E

.

For a given TSP instance, W is a constant value. During the search process, the values for $\alpha(t)$, $\beta(t)$ and $\gamma(t)$ are constantly changing. The values for both $\alpha(t)$ and $\gamma(t)$ will increase, and the value for $\beta(t)$ will decrease. If K is sufficiently large, the values for $\alpha(t)$, $\beta(t)$ and $\gamma(t)$ will approach to constant values as the search time $t \to \infty$.

Now the essential question is: What is the relationship between the size of the solution attractor A and the size of the problem n? In the TSP, the computational complexity is associated with the combinatoiral explosion of potential solutions in the solution space. If the size of A does not grow exponentially as n increases, the computational complexity in the TSP can be effectively reduced. Our experiment results show that the relationship curve between the size of A and n is linear.

4. The Attractor-Based Search System

Our definition of solution attractor not only defines what a solution attractor *is*, but also describes what it *does* and what we can *do with it*. The solution attractor theory provides a new knowledge that can help us to develop an effective and efficient search strategy for solving the TSP. FIGURE 3 presents a novel search algorithm, called the *attractor-based search* algorithm. This algorithm is very straightforward: generating K locally optimal tours, storing them in the matrix E, and then finding all tours in the matrix E using an exhausted search procedure. This attractor-based search system combines beautifully two crucial aspects in search: *exploration* and *exploitation*. The philosophy behind the attractor-based search system is that: if we don't know how to find the globally optimal point, we create a region that is known to contain this point and then search that region completely.

The matrix E itself create a data structure that can be easily searched by an enumerative search technique that is designed especially for trees: *depth-first tree search*. This simple recursive search technique traverses a directed graph starting from a node and then searches adjacent nodes recursively. FIGURE 4 illustrates an example of how a matrix E being searched by the depth-first tree search algorithm. Suppose that the search system stores three locally optimal tours $s_1 = (1, 2, 3, 4, 5)$, $s_2 = (1, 5, 3, 2, 4)$, and $s_3 = (1, 5, 4, 2, 3)$ for a 5-nodes TSP in the matrix E, as shown in FIGURE 4(a). The depth-first tree search process starts at node v_1 and then goes to node v_2, as illustrated in FIGURE 4(b). During the construction of a tour, if adding a node leads an invalid tour, the search process discards the invalid tour and continues searching in next valid column in the row until the end of the row in the matrix E. The tree search process returns four tours: $s_1 = (1, 2, 3, 4, 5)$, $s_2 = (1, 5, 3, 2, 4)$, $s_3 = (1, 5, 4, 2, 3)$, and $s_4 = (1, 2, 4, 5, 3)$. Originally, we stored three tours in the matrix E. The exhausted search process found a new tour $s_4 = (1, 2, 4, 5, 3)$. This example

```
1  procedure TSP_Attractor(Q)
2  begin
3    Initialize E;
4    Trajectories = 0;
5    repeat
6       s_i = Initial_Tour();
7       s_j = Local_Search(s_i);
8       E = Update(E, s_j);
9       Trajectories = Trajectories + 1;
10   until Trajectories = K
11   S* = Exhausted_Search(E);
12   Output S*;
13 end
```

FIGURE 3. The attractor-based search procedure for the TSP

.

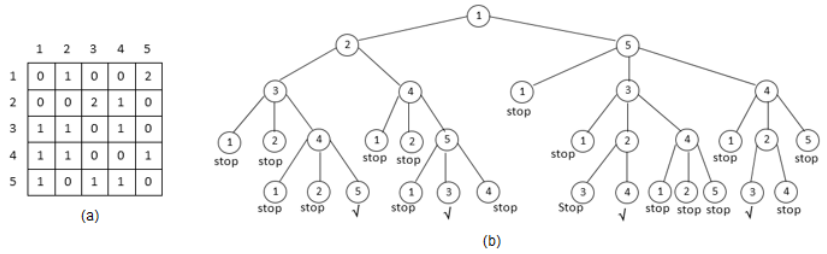


FIGURE 4. An example of depth-first tree search in the matrix E

.

not only demonstrates how the tree-search procedure constructs tours without considering the zero-hit edges, but also reveals the fundamental motivation of using the matrix E in the attractor-based search system: from a set of K locally optimal tours, the matrix E can help us find more other locally optimal tours and the globally optimal tours.

5. Summary

In the local search process, the search trajectories replace bad edges with better ones. The search trajectories actually perform the task of pruning out the edges that cannot possibly be included in the globally optimal tours. When one edge is removed, all tours that pass the edge will be removed from the consideration. When majority of the edges are removed, majority of connections between nodes are also removed, and thus the huge number of possible tours in the solution space are removed from the later exhausted search process. If we describe the solution space as a search tree, huge amount of useless branches of the tree have been cut off by the local search trajectories, and the final edge configuration in the matrix E becomes a well-cut tree, which leads to a considerable reduction in search time. In this way, the computational complexity of the problem is significantly reduced. Large solution space is reducd to an extremely small region - a solution attractor.

From a computational complexity stance, the TSP is an *intractable* problem; that is, for such a problem there exists no efficient algorithm to solve it. Most intractable problems have only one algorithm - the brute-force search algorithm - that really provides a solution. There are some algorithms such as the branch-and-cut [21] can solve to optimality many real-world instances in reasonable time; however, there is no fundamental progress in the area of exhaustive search for the TSP. In computer science, P versus NP problem is about how fast can we search through a huge number of possibilities. There are a huge number of possibilities, we don't always need to explore them all. P versus NP problem asks us if we ever need to explore all the possibilities of any search problem.

For the TSP, when a problem instance is large, the number of possible tours in the solution space is so large as to forbid an exhaustive search for the optimal tour. The attractor-based search system consists of two major computing phases. The first phase uses a local search procedure to search the solution space and generates K locally optimal tours. Local search converges very quickly, within low-order polynomial time [1]. This phase produces a solution attractor, a much smaller region for further complete search in the second phase. The second phase completely search the solution attractor using tree-search process, which can be done in $O(n^2)$. Therefore, the attractor-based search system finds the optimal solutions with computing complexity of $O(n^2)$. In this sense, the TSP may not be as complex as we might have expected.

References

[1] Aart E, Lenstra JK (2003) *Local Search in Combinatorial Optimization* (Princeton: Princeton University Press).

[2] Alligood KT, Sauer TD, York JA (1997) *Chaos: Introduction to Dynamical System* (New York: Springer-Verlag).
[3] Arbieto A, Obata D (2015) On attractor and their basins. *Involve* 8(2):195-209.
[4] Auslander J, Bhatia NP, Seibert P (1964) Attractors in dynamical systems. *Boletin De La Sociedad Matematics Mexicana* 9:55-66.
[5] Boese KD, Kahng AB, Muddu S (1994) A new adaptive multi-start technique for combinatorial global optimization. *Operations Research Letters* 16:101-113.
[6] Buescu J (1991) *Exotic Attractors* (Basel, Switzerland: Birkhauser).
[7] Denes A, Makey G (2011) Attractors and basis of dynamical systems. *Electronic Journal of Qualitative Theory of Differential Equations* 20:1-11.
[8] Fisher RA (1956) *Statistical Methods and Scientific Inference* (Landon: Oliver Boyd)
[9] Fogelberg H (1992) The phase space approach to complexity. *Journal of Statistical Physics* 69:411ff.
[10] Freisleben B, Merz P (1996) A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. *Proceedings of IEEE International Conference on Evolutionary Computation 1996* 616-621 (New York: IEEE Press)
[11] Grassberger P, Procaccia I (1983) Characterization of strange attractor. *Physical Review Letters* 50(5):346-349.
[12] Hains DR, Whitley LD, Howe AE (2011) Revisiting the big valley search space structure in the TSP. *Journal of the Operational Research Society* 62:305-312.
[13] Hilborn RC (1994) *Chaos and Nonlinear Dynamics: An Introduction for Science and Engineers* (New York: Oxford University Press).
[14] Kirkpatrick S, Toulouse G. (1985) Configuration space analysis of traveling salesman problems. *Journal de Physique* 46(8):1277-1292.
[15] Li W, Feng M (2013) Solution attractor of local searh in traveling salesman problem: concept, construction and application. *International Journal of Metaheuristics* 2(3): 201-233.
[16] Lin S (1956) Computer solution of the traveling salesman problem. *Bell System Technical Journal* 44:2245-2269.
[17] Merz P, Freisleben B (1998) Memetic algorithms and the fitness landscape of the graph bipartition in problem. *Parallel Problem Solving from Nature - PPSN V* LNCS 1498:765-774 (Berlin: Springer).
[18] Mezard M, Parisi G (1986) A replica analysis of the traveling salesman problem. *Journal de Physique* 47:1285-1296.
[19] Milnor J (1985) On the concept of attractor. *Communications in Mathematical Physics* 99:177-195.
[20] Milnor J (2010) *Collected Papers of John Milnor VI: Dynamical Systems (1953-2000)* (American Mathematical Society).
[21] Padberg M, Rinaldi G (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review* 33(1):60-100.
[22] Reeves CR (1998) Landscapes, operators and heuristic search. *Annals of Operations Research* 86:473-490.
[23] Ruelle D (1981) Small random perturbations of dynamical systems and the definition of attractor. *Communications in Mathematical Physics* 82:137-151.
[24] Statistical mechanics and the traveling salesman problem. *Europhysics Letters* 2:919-923.
[25] Zhang W, Looks M (2005) A novel local search algorithm for the traveling salesman problem that exploits backbone. *Proceedings of* 19^{th} *International Joint Conference on Artificial Intelligence* 343-348.