Chapter

Solution Attractor of Local Search System: A Method to Reduce Computational Complexity of the Traveling Salesman Problem

Weiqi Li

Abstract

The traveling salesman problem (TSP) is presumably difficult to solve exactly using local search algorithms. It can be exactly solved by only one algorithm—the enumerative search algorithm. However, the scanning of all possible solutions requires exponential computing time. Do we need exploring all the possibilities to find the optimal solution? How can we narrow down the search space effectively and efficiently for an exhausted search? This chapter attempts to answer these questions. A local search algorithm is a discrete dynamical system, in which a search trajectory searches a part of the solution space and stops at a locally optimal point. A solution attractor of a local search system for the TSP is defined as a subset of the solution space that contains all locally optimal tours. The solution attractor concept gives us great insight into the computational complexity of the TSP. If we know where the solution attractor is located in the solution space, we simply completely search the solution attractor, rather than the entire solution space, to find the globally optimal tour. This chapter describes the solution attractor of local search system for the TSP and then presents a novel search system—the attractor-based search system—that can solve the TSP much efficiently with global optimality guarantee.

Keywords: local search, global optimization, computational complexity, dynamical system, combinatorial optimization, solution attractor

1. Introduction

What it is that makes the TSP difficulty? The difficulty of the TSP is associated with the combinatorial explosion of potential solutions in the solution space. When a TSP instance is large, the number of possible tours in the solution space is so large as to forbid an exhausted search for the optimal tour. Numerous approaches to solving the TSP have been published. Some algorithms such as enumerative search, branch-and-bound search, and linear programming are exact approaches but lack efficiency. Other approximate algorithms, based on heuristics, are quick to find a good tour but lack effectiveness and robustness. Modern approximate algorithms, with today's fast computers, can find good solutions for extremely large TSP instances within a reasonable time, which are with a high probability just 2–3% away from the optimal tour [1–3].

Most approximate algorithms have been based on or derived from a general search technique known as *local search*. Local search algorithms iteratively explore the neighborhoods of solutions trying to improve the current solution by local changes. However, the scope of a single search trajectory is limited by the neighborhood definition. Both the TSP and local search have been hot research topics for decades, and many aspects of them have been studied. However, there is still a variety of open questions. The study of local search for the TSP continues to be a vibrant, exciting, and fruitful endeavor in combinatorial optimization, computational mathematics, and computer science.

A local search algorithm is essentially in the domain of dynamical systems. The goal of a dynamical system analysis is to capture the distinctive properties of certain points in the state space for a given dynamical system. The attractor theory of dynamical systems is a natural paradigm that provides the necessary and sufficient theoretical foundation to study the convergent behavior of a local search system. The TSP is believed to be NP-hard because we do not have an efficient enumerative search system for the problem. Do we need to examine all possibilities in order to solve the problem? Can we quickly narrow down the search space to a small region in which the optimal solution is located and then search that small region completely to find the optimal solution? This chapter attempts to use the solution attractor concept to answer these questions. If we can quickly identify that small region, the solution attractor, and then search that region thoroughly in reasonable time, the computational complexity of the problem can be dramatically reduced or may not exist. This chapter introduces the solution attractor concept, which not only helps us understand the behavior of a local search system for the TSP but also offers an important method to solve the problem efficiently with global optimality guarantee. This chapter presents a novel search algorithm—the attractor-based search system (ABSS)—that is a simple and quick global search system for the TSP.

2. Reframing the TSP definition

A problem is the frame into which the solutions fall. By changing the frame, we can change the range of possible solutions and scope of the optimal solutions. The classic TSP is defined as a complete graph $Q = (V, E, C)$, where $V = \{v_i : i = 1, 2, \dots, n\}$ is a set of n nodes, $E = \{e(i,j) : i,j = 1, 2, \dots, n; i \neq j\}$ is an $n \times n$ edge matrix containing the set of edges that completely connects the n nodes, and $C = \{c(i,j) : i,j = 1, 2, \dots, n; i \neq j\}$ is an $n \times n$ cost matrix holding a set of costs between nodes. A tour $s \in S$ is a closed tour that visits every node exactly once and returns to the starting node at the end. The solution space S contains a finite set of all feasible tours. The goal of the TSP is to find a tour s^* with minimal cost:

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

Obviously, this definition requires a search algorithm to find any single optimal tour in the solution space for a given instance. However, many real-world optimization problems are inherently multimodal. They may contain multiple optimal solutions in their solution spaces. Finding all optimal solutions is the essential requirement for global optimization. In practice, knowledge of multiple optimal solutions is essentially helpful, providing the decision-maker with multiple best options. We assume that a TSP instance contains h $(h \geq 1)$ optimal tours in the solution space S and denotes S^* as the set of h optimal tours. Under global optimization frame, the objective of the TSP is to find the set of optimal tours $S^* \subset S$:

$$S^* = \arg\left[\min_{s \in S} f(s)\right] = \left[s_1^*, s_2^*, \dots, s_h^*\right] \tag{2}$$

For a given TSP instance, we do not know the number of optimal tours in the solution space until we find all of them. Obviously, this reframed TSP definition becomes even more difficult to solve. To solve this reframed TSP, we need a search algorithm 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 the same set of optimal solutions in the solution space over and over again.

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 a powerful data structure that can shift our point of view so that we can uncover alternative approaches. One factor contributing to algorithmic difficulty is that we lack a data structure that links the structure of the problem and the behavior of the search algorithm and that can make the complex search space traceable and tractable. It may be unreasonable to expect a search algorithm to be able to solve any problem without taking into account the structure and properties of the problem. Local search algorithms may not require much problem-specific knowledge in order to generate good solutions. However, in order to solve a problem exactly, we should design a search algorithm that is based on the structure of the problem at hand.

3. Solution attractor of local search system for TSP

A dynamical system is a model to describing the temporal evolution of a system in its state space [4–9]. The theory of dynamical system is an extremely broad area of study. The study of dynamical systems has discovered that many dynamical systems exhibit attracting behavior in the system trajectories. In such a system, all initial states tend to evolve toward a single final state or a set of final states. This single state or a set of states is called *attractor*. A heuristic local search system essentially is a discrete dynamical system and therefore natural in the domain of dynamical systems.

A local search system has a solution space S, a set of times T (iterations of search), and a search function $f : S \times T \to S$ for temporal evolution that gives the consequent to a solution $s \in S$. A search trajectory is the sequence of solutions of a local search system at successive time steps in the form $s(t+1) = f(s(t))$. The behavior of a search trajectory can be understood as a process of iterating a function $f(s)$. Questions about the behavior of a local search system over time are actually the questions about its search trajectories. Let us denote s_0 as an initial point of a search trajectory, f^t as the f^{th} iterate of the function $f(s)$, and a locally optimal solution s' as the limit of the convergent search trajectory $s_0, f(s_0), f^2(s_0), \dots, f^t(s_0), \dots$; then

$$f(s') = f\left(\lim_{t\to\infty} f^t(s_0)\right) = \lim_{t\to\infty} f^{t+1}(s_0) = s' \tag{3}$$

For the TSP, a search trajectory leads to a sequence of tours $s_0, s_1, s_2, \dots, s_t$, where s_0 is an arbitrary initial tour and s_t is the final tour at the end of search after t iterations. This time series represents a part of the solution space searched by this search trajectory. The globally optimal tour s^* is the target point of a search trajectory. Due to the constraint of the neighborhood search structure, a search trajectory rarely reaches the target point and eventually stops at a locally optimal tour s'. In a heuristic local search system, different initial points and randomness in the search process lead to a complex search behavior and generate different search trajectories. There are no two search trajectories that are exactly alike. Different search trajectories explore different regions of the solution space and stop at different final points. Since all search trajectories have the same target point, they move toward the same direction and finally stop in the same target region in the solution space. This target region is called the *solution attractor*, denoted as A. Roughly speaking, the solution attractor of a local search system is a closed region of the solution space toward which a search trajectory tends to evolve regardless of the starting point. A solution attractor is the equilibrium level of the system dynamics. At this level, all search trajectories will stop moving, and therefore the solution attractor consists of all locally optimal tours. A single search trajectory typically converges to either one of the points in the solution attractor. Since the globally optimal tour is a special case of locally optimal tours, it is undoubtedly embodied in the solution attractor, that is, $s^* \in A$ and $A \subset S$. **Figure 1** summaries the concepts of search trajectories and solution attractor in a local search system. Illustrating search trajectories and solution attractor of a local search system as 2-D object is a valid metaphor for understanding how a local search system might proceed. The solution attractor A of a local search system has the following properties [10–12]:

- Invariance, i.e., $\forall s' \in A, f^t(s') = s'$ and $f^t(A) = A$ for all time t.
- Attractiveness, i.e., $\forall s_i \in S, f^t(s_i) \in A$ for sufficient time t.
- Convexity, i.e., all locally optimal tours in A are gathered in an extremely small region of the solution space.
- Centrality, i.e., the best of these locally optimal tours (the globally optimal tour) is located centrally with respect to the other locally optimal tours.
- Irreducibility, i.e., the solution attractor A contains a limit number of invariant locally optimal tours.

In general term, for a TSP instance with h $(h \geq 1)$ optimal tours, the local search system will have h solution attractors $(A_1, A_2, \dots, A_h)$ that attract all search trajectories. Each of the solution attractors has its own set of locally optimal tours,

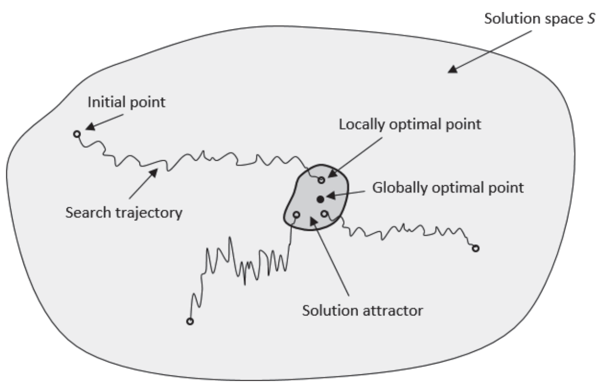

Figure 1.
Search trajectories and solution attractor in a local search system.

surrounding a globally optimal tour s_i^* $(i = 1, 2, \ldots, h)$. The search trajectories will explore many different regions of the solution space and converge to these solution attractors. A particular search trajectory will converge into one of these solution attractors. The set of locally optimal tours generated by all search trajectories will be distributed to these h solution attractors. According to dynamical systems theory [9], the closure of an arbitrary union of attractors is still an attractor, that is, the attractor of a local search system for a multimodal TSP is a complete collection of solution attractors $A = A_1 \cup A_2 \cup \ldots \cup A_h$ and $A \subset S$.

4. The attractor-based search system for TSP

Figure 2 presents the attractor-based search system (ABSS) for the TSP. In this algorithm, Q is a given TSP instance. K is the number of search trajectories used to generate K locally optimal tours. E is the edge matrix used to store the K locally optimal tours. s_i is an initial tour generated by the function `Initial_Tour()`, which can use any technique to construct the initial tour. s_j is a locally optimal tour generated by the function `Local_Search()`, which can use any local search technique. The function `Update()` updates the edge matrix E by recording the edge configuration of tour s_j into E. Finally, the function `Exhausted_Search()` searches the matrix E completely using any enumerative search technique and outputs the set of the found globally optimal tours, S^*. The search strategy behind the ABSS is simple and effective: we first identify the small regions—the solution attractors—in which the globally optimal tours are located, and then we search these small regions completely to find the globally optimal tours. In this strategy, we avoid searching the large unnecessary region of the solution space so that the search time is dramatically reduced. The ABSS shows strong features of effectiveness, flexibility, adaptability, and scalability. It can be implemented in many different ways: serial or parallel. The computational model in ABSS is inherently parallel and can support the exploitation of massive parallelism. If the ABSS is implemented using proper number of concurrent processors, it can deal with dynamic TSP in real time:

The critical element in the ABSS is the edge matrix E. Few search algorithms have used the edge matrix E in their search processes. An edge is the most basic element in a tour. It is a connection between two nodes and contains pieces of information about $(n-2)!$ tours that go through it. A tour is a list of ordered nodes and has an edge configuration in the matrix E, as an example illustrated in **Figure 3**. Each edge has an implicit probability to be selected by a locally optimal tour. The edges in the matrix E can be divided into three groups: G-edges, globally superior

```
ABSS_Algorithm(Q)
begin
  initialize $E$;
  NumberOfTrajectories = 0;
 repeat
    $s_i$ = Initial_Tour();
    $s_j$ = Local_Search($s_i$);
    $E$ = Update($E$, $s_j$);
    NumberOfTrajectories = NumberOfTrajectories + 1;
 until NumberOfTrajectories = $K$
 $S^*$ = Exhausted_Search(E);
 end
```

Figure 2.
The ABSS algorithm for the TSP.

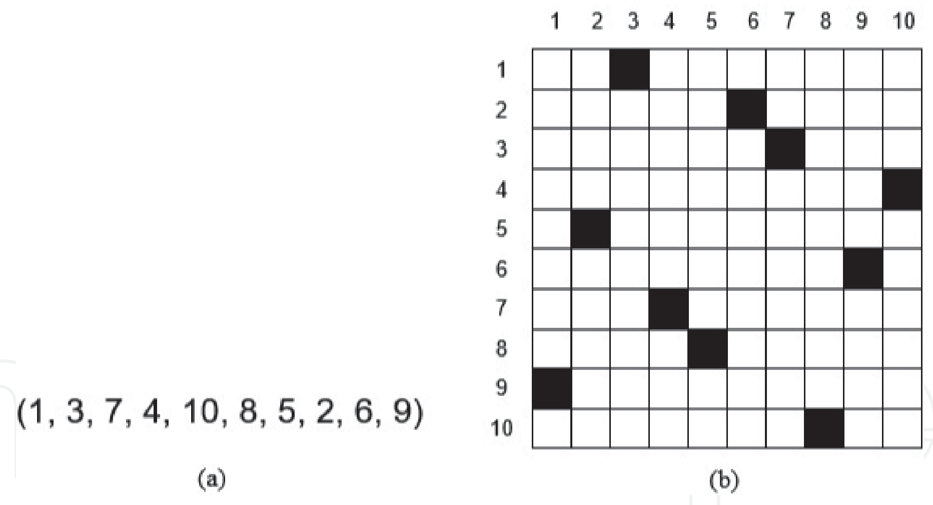

Figure 3.
(a) Shows a 10-node tour and (b) shows its edge configuration in the matrix E.

edges, and bad edges. The edges that are contained in a globally optimal tour are *G*-edges. A globally superior edge is the edge that is hit by many locally optimal tours. Although each of these locally optimal tour selects this edge based on its own neighborhood function and search path, the edge is globally superior because it is selected by these tours from different search paths that go through different regions of the solution space. Bad edges are the edges that are eventually discarded by all search trajectories or selected by only few locally optimal tours. The edge configuration of a locally optimal tour consists of some *G*-edges, some globally superior edges and a few bad edges. Therefore, the edge matrix *E* is an exploitable data structure that plays the following roles in the ABSS:

- It is a natural data structure that can store the edge configurations of search trajectories and thus can visually demonstrate the asymptotic behavior of the search trajectories during the search. When the search trajectories reach their final points, it records the frequency of occurrence of each of the edges in the locally optimal tours.

- It is an instrument that can alter the state of what we measure for the TSP. We can change a tour-search process into an edge-search process, and thus the problem of finding the optimal tour is converted into the problem of finding a set of edges. The edge space represented by the edge matrix *E* is much simpler and smaller than the solution space represented by the tours.

- It is a mechanism that can transform non-deterministic local search to deterministic global search. Through the matrix *E*, we can see that the search trajectories actually perform the process of edge inclusion and exclusion, and the temporal evolution of the edge configuration matrix *E* generated by different sets of *K* search trajectories always converges to the same small set of edges.

A search trajectory changes its edge configuration during the search process. Let W be the total number of edges in the matrix E, $\alpha(t)$ the number of the common edges that are hit by all search trajectories at time t, $\beta(t)$ the number of the edges that are hit by one or some of the search trajectories, and $\gamma(t)$ the number of the edges that have no hit from the search trajectories. Then at any time t, we have

$$W = \alpha(t) + \beta(t) + \gamma(t) \tag{4}$$

For a given TSP instance, W is a constant value $n(n-1)/2$ for a symmetric instance or $n(n-1)$ for an asymmetric instance. We can expect that, as local search process continues, the values for both $\alpha(t)$ and $\gamma(t)$ will increase and value for $\beta(t)$ will decrease. Our experiments confirmed this inference about $\alpha(t)$, $\beta(t)$, and $\gamma(t)$. **Figure 4** illustrates the curve patterns of $\alpha(t)$, $\beta(t)$, and $\gamma(t)$. These curves cannot increase or decrease forever, and they approach to constant values as the search time continues, that is,

$$W = \lim_{t\to\infty}\alpha(t) + \lim_{t\to\infty}\beta(t) + \lim_{t\to\infty}\gamma(t) = \mathrm{A} + \mathrm{B} + \Gamma \qquad (5)$$

This indicates that at certain point of time, the union of the edge configurations of the search trajectories will become fixed. This aggregate edge configuration will be the edge configuration of the solution attractor at limit.

When the matrix E records the edge configurations of K locally optimal tours, the edges are partitioned into two sets: the edges with hit (hit edges) and the edges without hit (non-hit edges). The hit edges include all globally superior edges, all G-edges, and some bad edges. **Figure 5** shows the composition of edges in the matrix E after the edge configurations of K locally optimal tours are stored in it. The local search process can quickly make large number of edges become the non-hit edges. In our experiments, we found that the ration $\gamma(t)/W$ exceeds 75% easily with short search time for the symmetric TSP. This fact indicates that the edge configuration of the solution attractor contains very small percentage of the edges. Therefore, compared to the full solution space, the solution attractor is extremely small.

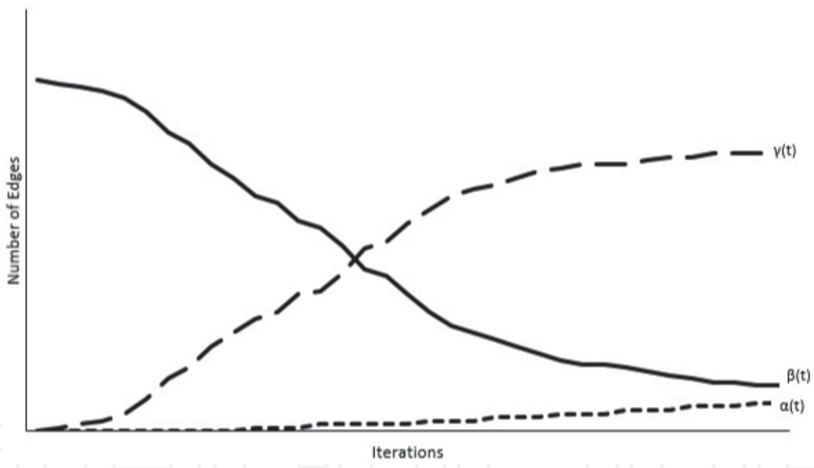

Figure 4.
The α(t), β(t), *and* γ(t) *curves with search iterations.*

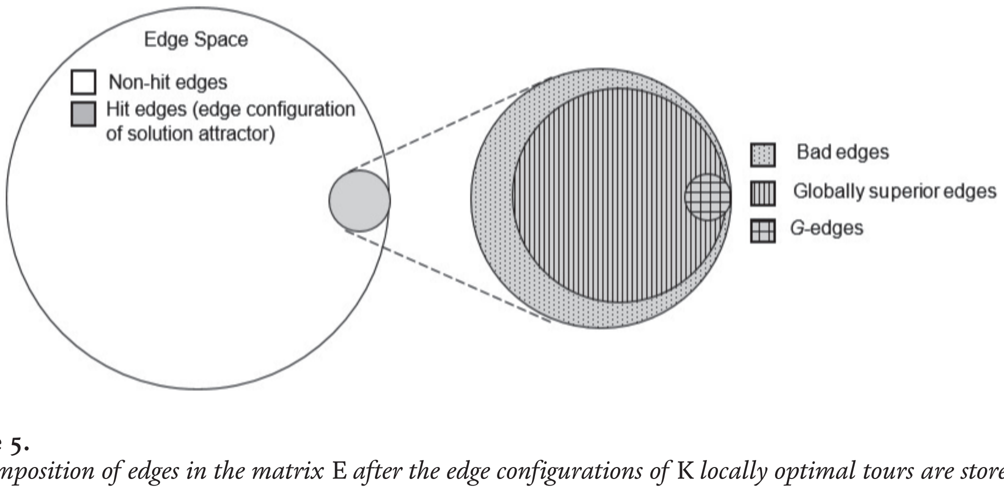

Figure 5.
The composition of edges in the matrix E *after the edge configurations of* K *locally optimal tours are stored.*

Different sets of K search trajectories will generate a little different edge configuration in the matrix E. However, the underlying edge configuration of the solution attractor in the matrix E is structurally stable because small differences in the final edge configurations generated by different sets of K search trajectories do not mean the qualitative difference in the dynamical behavior of search trajectories. The core structure of the edge configuration of the solution attractor keeps unchanged. In our experiments, we observed that in the aggregated edge configurations of the different sets of K locally optimal tours, the set of globally superior edges and the G-edges is always the same. This empirical fact indicates that a local search system actually is a deterministic system. Although a single search trajectory appears stochastic, there is an important aspect of order hidden in the local search system that makes all different sets of K search trajectories converge to the same set of core edges.

5. Global optimization and computational complexity of ABSS

In order to make sure that the ABSS is an effective and efficient search system, we should answer the following fundamental questions:

1. "How can we construct the edge configuration of the solution attractor without large number of search trajectories?" that is, "What is a proper size of K?"

2. What is the relationship between the size of the constructed solution attractor and the size of the TSP instance?

3. How does the ABSS meet the requirements of a global optimization system?

4. Is the best tour in the solution attractor the best tour in the solution space?

It is easy to verify that the edge configuration of a true solution attractor can be obtained if all search trajectories are performed and all search trajectories reach their real locally optimal points. In other words, the probability of finding all globally optimal points is one if all possible search trajectories are performed. However, the required search effort may be very huge—equivalent to enumerating all possibilities in the solution space. In fact, we can construct the edge configuration of the solution attractor with a limited number of K locally optimal tours. In a heuristic local search system, K search trajectories start a sample of initial points from a uniform distribution over the solution space S and generate a sample of locally optimal points uniformly distributed over the solution attractor A. The fundamental theory behind using K search trajectories is the information theory. According to the information theory [13], each solution point in the solution space contains some information about its neighboring points that can be modeled as mapping $\Omega_{s_i} : s_i \rightarrow \mathfrak{R}$, called *information* or *influence* function, which is a decreasing function of the spatial distance to the solution point s_i in the solution space. The information function value of s_i is maximum at the point and decreases gradually with the distance from that point. The notion of influence function has been used extensively in data mining, data clustering, and pattern recognition. In a local search system for the TSP, as one search trajectory is approaching to a locally optimal tour, it shares more and more edges with other search trajectories and thus collects more and more information about the other locally optimal tours and the globally optimal tour. When K search trajectories reach their end points and record their edge configurations in the matrix E, the aggregate edge configuration in the matrix E is not just a countable union of the edge configurations of the K locally optimal tours but also includes the edge configurations of all other locally optimal tours. The essential motivation behind using the edge matrix E is that a collection of K locally optimal tours is able to provide whole information about all locally optimal tours and the matrix E is a tool that put all pieces of puzzles together to reveal the edge configuration of the solution attractor. What is the proper number for K? In our experiments, we found that $K = 6n$ is the magic number. The union of the edge configurations of at most $6n$ random initial tours can generate the edge configuration of the entire solution space (i.e., all cells of the matrix E can be hit by these initial tours). The core structure (the set of the globally superior edges and the G-edges) of the edge configuration of the constructed solution attractor becomes unique and fixed when the number of search trajectories $K \geq 6n$.

Another related question is "how many moves a local search trajectory has to make before it reaches a real locally optimal tour?" So far we do not have an answer to this question. We even do not know any nontrivial upper bounds on the number of moves that may be needed to reach local optimality [14–17]. In practice, we are rarely able to find a true locally optimal point because we simply do not allow the local search process run enough long time. We usually let a search trajectory run a predefined number of iterations, accept whatever solution it generates, and treat it as a locally optimal solution. Therefore, the size of the constructed solution attractor depends not only on the problem structure and the neighborhood function used in the local search process but also on the amount of search time invested in the local search process. If we spend more time in the local search process ($t_2 > t_1$), the resulting constructed solution attractor should be smaller ($A_2 < A_1$), as illustrated in **Figure 6**.

Let M_S be the edge configuration of the solution space S, M_A the edge configuration of the true solution attractor A, and f^t the t-iterate of the search function f in the local search process on K search trajectories, and then it follows easily that M_A is equal to the intersection of the nested sequence of forward edge sets:

$$M_S \supset f(M_S) \ldots \supset f^t(M_S) \ldots \supset M_A \qquad (6)$$

Therefore, at any search time t before the K search trajectories reach their true end points, the edge configuration of the true solution attractor M_A is always a subset of the edge configuration of the constructed solution attractor $f^t(M_S)$, and thus the constructed solution attractor is always larger than the true solution attractor.

What is the relationship between the size of the constructed solution attractor and the size of the given problem? So far there is no theoretical or analytical tool available in the literature that can be used to answer this question. We have to depend on empirical results to lend some insights. If the size of the constructed attractor increases exponentially with the size of the problem increases, the ABSS still does not fundamentally reduce the computational complexity of the problem.

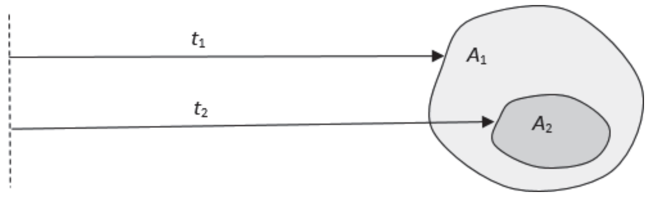

Figure 6.
The size of a constructed solution attractor is also determined by the time spent in the local search process.

The ABSS consists of two search phases: the local search phase that construct the solution attractor (from line 5 to line 10 in the ABSS algorithm) and the exhausted search phase that find the best tour in the solution attractor (line 11). For the TSP, the solution space can be represented by a search tree. The local search phase actually performs the task of pruning off the edges that cannot possibly be included in the globally optimal tours. When the first edge is discarded by all K search trajectories, $(n-2)!$ tours that go through this edge are excluded in the search space of the exhausted search phase. Each time an edge is removed, the search space of the exhausted search phase is reduced by a factor. In such a way, the number of combinatorial branching possibilities for the exhausted search can be exponentially reduced. Decades of research and empirical evidence have found that heuristic local search algorithms converge very quickly, within low-order polynomial time [14]. When majority of the edges are removed, a huge number of possible tours in the solution space are removed from consideration in the exhausted search phase. In this way, the computational complexity of the problem is significantly reduced. In our experiments, the local search process can remove over 70% of edges in the matrix E in a number of iterations bounded by a linear polynomial time. Therefore, the local search phase in the ABSS can be done in $O(n^2)$. **Figure 7** shows the result of one of our experiments. All other similar experiments reveal the same pattern. All our experiments used the 2-opt local search technique because the 2-opt has the smallest expected number of local optima [14]. The experiments were carried out on a PC with 2.60 GHz Intel® Core(TM)i7-3687U CPU, running under Microsoft Windows 7 Enterprise. The ABSS algorithm was coded in Microsoft Visual Basic 2012. In this experiment, we generated 10 unimodal TSP instances in the size from 1000 to 10,000 nodes with 1000-node increment. For each instance, the search system generated $k = 6n$ search trajectories. First, we let each search trajectory stop when no improvement was made during 10,000 iterations, no matter the size of the problem (viz., fixed search time). We counted the number of tours in the constructed solution attractor for each instance. Next we ran the search system again on these instances. This time we made each search trajectory stop when no improvement was made during $10n$ iterations (varied search time 1) and $100n$ iterations (varied search time 2), respectively. Then we counted the number of

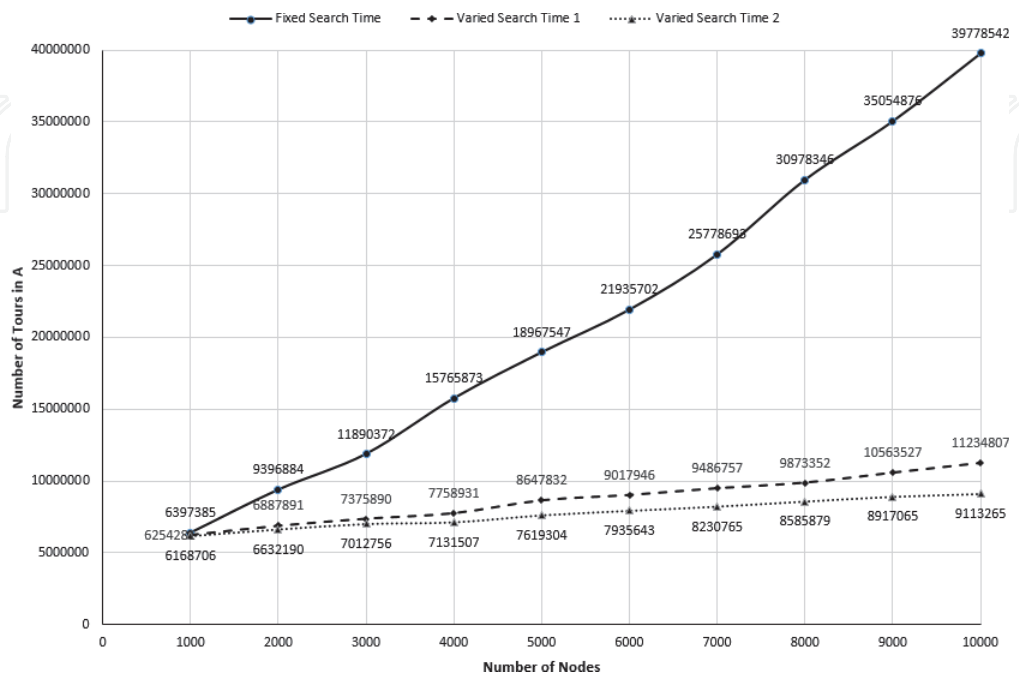

Figure 7.
The relationship between the size of the constructed solution attractor and the size of the problem.

tours in the constructed solution attractor for each instance. As illustrated in the chart of **Figure 7**, all curves appear to be linear, and the varied-search-time curves have much flatter slope because longer local search time leads a smaller solution attractor.

After the local search phase, majority of unnecessary branches have been cut off from the search tree. Usually, when using tree search enumerative algorithm, the effective branching factor is used to measure the computing complexity of the algorithm. An *effective branching factorb** is the number of successors generated by a typical node for a given search tree problem. We use the following definition to calculate effective branding factor b^* in the exhausted search phase:

$$N = b^* + (b^*)^2 + \dots + (b^*)^n \tag{7}$$

where N is total number of nodes generated from the origin node and n is the size of the TSP instance, representing the depth of the tree. We conducted several experiments on different TSP instances. The tree search process always starts from node 1 (the first row of the matrix E). N is the total number of nodes that are processed to construct all valid and invalid tours in the matrix E from the node 1. N does not count the node 1 (the origin node), but includes node 1 as the end node of a valid tour. **Figure 8** shows the result of one experiment, using the same instances and setting reported in **Figure 7**. The effective branching factors in all our experiments are very small, all less than 2. This result indicates that the edge configuration of the solution attractor presents a tree with extremely sparse branches, and the degree of sparseness does not change as the problem size increases if we properly increase local search time for a larger instance. It also indicates that the exhausted search phase is polynomial time if we polynomially increase local search time for larger instances. Therefore, the tree represented by the edge configuration of the constructed solution attractor has a manageable size that can be searched completely in $O(n^2)$.

The ABSS is a global optimization system. The goal of a global optimization system is to find all absolute best solutions in the solution space. There are two major tasks in a global optimization system: (1) finding all globally optimal points in the solution space and (2) making sure that they are globally optimal. To complete these tasks, the global optimization system should meet the following requirements: (1) its search behavior should be globally convergent, (2) it should be deterministic

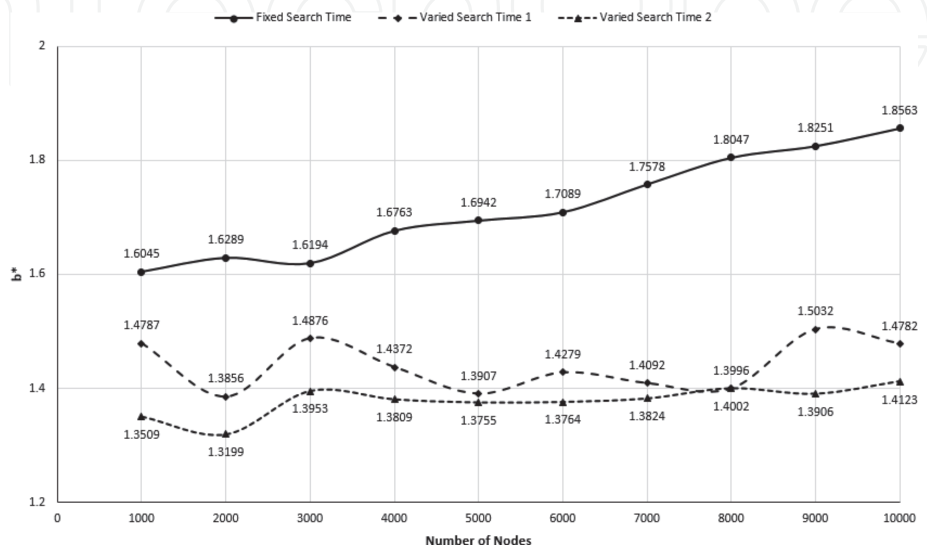

Figure 8.
The b* *values for different problem size* n.

and has a rigorous guarantee for finding all globally optimal solutions without excessive computing burden, and (3) it should have a self-evident optimality criterion.

In the ABSS, two different search phases have different search objectives. The objective of the local search phase is "searching for most promising tours in the solution space." It tries to provide an answer to the question "In which small region of the solution space is the best tour located?" The objective pursued by the exhausted search phase is "finding the best tour among the most promising tours." It tries to provide an answer to the question "In this small region, which tour is the best one?" Putting these two objectives together, the ABSS tries to provide an answer to the question "Which tour is the best tour in the solution space?"

The ABSS combines beautifully two crucial aspects in search: exploration and exploitation. In the local search phase, K search trajectories explore the solution space independently and individually to collect the edges for constructing the solution attractor. The K search trajectories create and maintain diversity from beginning to the end. Randomization in the local search process makes the local search process become a randomized process. A search trajectory changes its edge configuration according to the objective function and its neighborhood structure. The local search phase actually uses the Monte Carlo simulation to sample locally optimal tours. Monte Carlo simulation is defined as simulations used to model the probability of different outcomes in a process that cannot easily be predicted due to intervention of random variables. The essential idea of Monte Carlo method is to use randomness to solve problems that might be deterministic in principle. In the ABSS, K search trajectories start a sample of initial tours from uniform distribution over the solution space and, through a randomized local search process, generate a sample of locally optimal tours that are uniformly distributed in the constructed solution attractor. Therefore the edge configuration of the solution attractor is constructed through this Monte Carlo sampling process. The distribution of the hit edges in the matrix E converges to a small set of edges, and the set of the edges is statistically fixed. This fixed edge configuration is not sensitive to the selection of K search trajectories. Convergence and stability are two desirable properties of the solution attractor: all search trajectories will converge to the solution attractor and remain there forever. The ability of K search trajectories to explore the entire solution space and thus collect all globally superior edges and G-edges can help the ABSS achieve its required function—finding all globally optimal tours.

The global convergence and deterministic property of the search trajectories make the ABSS converge in solution, that is, the ABSS always find the same set of the best tours. This argument was empirically confirmed in our experiments. For a given TSP instance, we repeated the same search process on the same instance many times, each time using a different set of K search trajectories, and the search system always generates the same set of the best tours in all trials. **Table 1** shows the result of one experiment. This experiment generated two TSP instances Q_1 and Q_2 with $n_1 = 1000$ and $n_2 = 10000$ nodes. The ABSS ran each instance 15 times, each time using a different set of $K = 6n$ search trajectories. The ABSS found the same single best tour in all 15 trials for Q_1 and the same set of three best tours in all 15 trials for Q_2. The three best tours for Q_2 have the same cost value but with different edge configurations. It is clear that Q_1 is a unimodal TSP instance and Q_2 is a multimodal instance having three optimal tours in its solution space. If any trial had generated a different set of the best tours, we could immediately make a conclusion that the best tours in the constructed solution attractor may not be the globally optimal tours. From the experimental and practical perspective, the fact that the same set of the best tours was detected in all trials provides a significant empirical evidence of the optimality of these tours.

Trial #	Number of tours in A	Range of tour cost	Number of best tours in A
1000 nodes (Q_1)	**(6000 initial tours)**		
1	5,703,833	[3926, 4437]	1
2	5,703,785	[3926, 4521]	1
3	5,703,479	[3926, 4509]	1
4	5,703,829	[3926, 4495]	1
5	5,703,868	[3926, 4540]	1
6	5,703,499	[3926, 4500]	1
7	5,703,253	[3926, 4556]	1
8	5,703,791	[3926, 4488]	1
9	5,703,742	[3926, 4498]	1
10	5,703,990	[3926, 4551]	1
11	5,703,637	[3926, 4526]	1
12	5,703,457	[3926, 4536]	1
13	5,703,642	[3926, 4534]	1
14	5,703,626	[3926, 4546]	1
15	5,703,727	[3926, 4522]	1
10,000 nodes (Q_2)	**(60,000 initial tours)**		
1	9,428,645	[81,967, 85,287]	3
2	9,428,571	[81,967, 84,979]	3
3	9,428,032	[81,967, 85,286]	3
4	9,429,004	[81,967, 85,365]	3
5	9,428,625	[81,967, 85,348]	3
6	9,428,819	[81,967, 85,345]	3
7	9,428,815	[81,967, 85,232]	3
8	9,429,021	[81,967, 85,254]	3
9	9,428,950	[81,967, 85,320]	3
10	9,428,847	[81,967, 85,286]	3
11	9,428,749	[81,967, 85,036]	3
12	9,428,978	[81,967, 85,248]	3
13	9,428,767	[81,967, 85,076]	3
14	9,428,933	[81,967, 85,223]	3
15	9,428,799	[81,967, 85,337]	3

Table 1.
Tours in solution attractor for 1000-node and 10,000-node TSP instances.

One factor that makes the TSP difficult to solve is that we have not found a simple optimality criterion to decide whether or not a locally optimal tour is also a globally optimal tour. Selecting the best tour among a set of tours and knowing it is the best one are the full challenges of the TSP. A brute-force algorithm that sorts through all tours in the solution space can be certain that it meets the challenge. However, it lacks practical efficiency. For a TSP instance, there are an unknown number of globally and locally optimal tours. The ABSS uses a simple and practical optimality criterion: the best tours in the set of all locally optimal tours are the globally optimal tour. In fact, this criterion is the necessary and sufficient condition for a locally optimal tour to be a globally optimal tour. In the ABSS, the local search phase identifies the solution attractor, and no tour outside the solution attractor can be better than any tour inside. Then the exhausted search phase examines all tours in the solution attractor and finds the best tours. In fact, this optimality criterion describes how the ABSS models and solves the TSP.

For a tour $s_i \in S$, its neighborhood $N(s_i) \subset S$ is defined, consisting of all tours that can be reached from s_i in one single transition. A locally optimal tour s' satisfies $f(s') \le f(s)$ for all $s \in S \cap N(s')$. A solution attractor A consists of all locally optimal tours. A best tour s^* in a solution attractor satisfies $f(s^*) < f(s')$ for all $s' \in A$. The best tour $s^* \in A$ satisfies the following conditions, which allow the propagation of the minimum properties of s^* in the solution attractor A to the whole solution space S, that is, $f(s^* \in A) < f(s)$ for all $s \in S$:

1. $f(s') \leq f(s)$ for all $s \in S \cap N(s')$

2. $A \ni s'$ for all $s' \in S$

3. $f(s^* \in A) < f(s')$ for all $s' \in A$

4. $\min_{s \in S} f(s) = \min_{s \in A} f(s)$

5. $\lim_{t \to \infty} f(s^*) = s^*$

6. Conclusions

For the TSP, the computational complexity is associated with the combinatorial explosion of potential solutions in the solution space. If we accept the argument that the number of tours in the solution space indicates the difficulty of the TSP, then the fact that the solution space can be significantly reduced to a small solution attractor means that the difficulty of the TSP can be dramatically reduced. The novel perspective of solution attractor in a local search system for the TSP gives us an opportunity to overcome combinatorial complexity. The solution attractor shows us where the best tour can be found in the solution space. If we concentrate the exhausted search effort in this much smaller region, the number of possibilities in search space is no longer prohibitive. Our experiments showed that the ABSS can significantly reduce the computational complexity for the TSP and thus can solve the TSP much efficiently with global optimality guarantee. The ABSS is an obvious finite algorithm in computing complexity of $O(n^2)$ and space requirement of $O(n^2)$ for the TSP. This suggests that the TSP might not be as complex as we might have expected.

The edge matrix E is the data structure that is defined by the TSP naturally and is used in the ABSS to separate the solution attractor from the entire solution space. In the ABSS, the combination of an efficient local search process, a powerful data structure (the matrix E), and an exhausted search process provides a highly effective and efficient search system. If some other NP-hard problems have the same nice data structure that can be used to reduce the search space, these problems can also be solved in polynomial time.

This chapter focuses on the solution attractor of the local search system for the TSP. Does it appear to be technical archetypes for other combinatorial optimization problems? Each optimization problem has its own specifics and data structure. In order to fully understand the search process for a particular problem, we must put our attention to the data structure that is defined by the problem. The combination of a proper data structure and simple search strategy can make the highly complex solution space become tractable and lead to more knowledge about the problem and provide opportunities for new algorithmic designs.

The TSP is the most prominent problem in NP-hard problems. It is hoped that this chapter will serve as a pioneer in this field and bring more and better works from other researchers and practitioners. The ultimate goal of this chapter is to encourage readers to take up their own pursuit of interesting problem-by-problem methods for attacking diverse optimization problems.

The solution attractor theory provides some important insights into the power of efficient computations and a line of reasoning that may lead to a proof in the near future about P vs. NP problem. The P vs. NP problem is an important computational issue in nearly every scientific discipline [18]. It is about how efficient we can search through a huge number of possibilities. Computational complexity theory suggests that there are limits of the power of general-purpose optimization techniques. Majority of people are in favor of $P \neq NP$ because we totally lack fundamental progress in the area of enumerative search [19]. What are these limits? If we design a search algorithm that fully utilizes the natural structure of the problem, like the edge matrix E of the TSP, we may be able to remove some constraint on our road.

Author details

Weiqi Li
University of Michigan – Flint, Flint, USA

*Address all correspondence to: weli@umich.edu

References

[1] Ausiello G, Crescenzi P, Kann V, Marchetti-sp G, Spaccamela M. Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties. New York: Springer; 2003

[2] Rego C, Gamboa D, Glover F, Osterman C. Traveling salesman problem heuristics: Leading methods, implementations and latest advances. European Journal of Operational Research. 2011;**211**:427-441

[3] Korte B, Vygen J. Combinatorial Optimization: Theory and Algorithms. New York: Springer; 2012

[4] Alligood KT, Sauer TD, Yorke JA. Chaos: Introduction to Dynamical System. New York: Springer; 2000

[5] Brin M, Stuck G. Introduction to Dynamical Systems. Cambridge: Cambridge University Press; 2016

[6] Brown RA. Modern Introduction to Dynamical System. New York: Oxford University Press; 2018

[7] Dénes A, Makay G. Attractors and basins of dynamical systems. Electronic Journal of Qualitative Theory of Differential Equations. 2011;**20**:1-11

[8] Milnor J. On the concept of attractor. Communications in Mathematical Physics. 1985;**99**:177-195

[9] Milnor J. Collected Papers of John Milnor VI: Dynamical Systems (1953–2000). Washington, DC: American Mathematical Society; 2010

[10] Li W. Dynamics of local search trajectory in traveling salesman problem. Journal of Heuristics. 2005;**11**: 507-524

[11] Li W, Feng M. The solution attractor of local search in traveling salesman problem: Concept, construction and application. International Journal of Metaheuristics. 2013;**2**:201-233

[12] Li W, Li X. The solution attractor of local search in traveling salesman problem (part 2): Computational study. International Journal of Metaheuristics. 2019;**7**:93-126

[13] Shannon CE. A mathematical theory of communication. Bell System Technical Journal. 1948;**27**:623-656

[14] Aarts E, Lenstra JK. Local Search in Combinatorial Optimization. Princeton: Princeton University Press; 2003

[15] Chandra B, Karloff H, Tovey C. New results on the old k-opt algorithm for the traveling salesman problem. SIAM Journal on Computing. 1999;**28**: 1998-2029

[16] Fischer ST. A note on the complexity of local search problems. Information Processing Letters. 1995;**53**: 69-75

[17] Grover LK. Local search and the local structure of NP-complete problems. Operations Research Letters. 1992;**12**:235-243

[18] Fortnow L. The Golden Ticket – P, NP, and the Search for the Impossible. Princeton: Princeton University Press; 2013

[19] Fortnow L. The status of the P versus NP problem. Communications of the ACM. 2009;**52**:78-86