Stochastic Local Search Using the Search Space Smoothing Meta-Heuristic：A Case Study*

Sheqin Dong Fan Guo Jun Yuan Rensheng Wang Xianlong Hong

Department of Computer Science & Technology, Tsinghua University, Beijing China

Abstract

In this paper, two smoothing effects are firstly pointed out by analysis and by experiment on Traveling Salesman Problem(TSP) instances. We design a novel algorithm which runs stochastic local search under the SSS framework. The function determining the accepting probability of uphill moves is designed so that the algorithm can take advantage of the local smoothing effect ignored in original SSS. Experimental results on TSPLIB instances demonstrated that the performance of the new algorithm is much superior to traditional SSS approach.

Keywords: Solution Space Smoothing, Stochastic Local Search, TSP

1. Introduction

Search Space Smoothing is a meta-heuristic in combinatorial optimization first proposed by Jun Gu in 1994 [1]. Unlike traditional iterated local search method, the problem instance, so the search space, is altered during every iteration. This alteration is called smoothing operation. The ruggedness of the search landscape is significantly lowered after a smoothing operation is performed, while the neighborhood graph remains the same. A parameter α is introduced in the smoothing formula, and it controls the smoothing strength, i.e. how much the ruggedness of search landscape is lowered. Local search under a series of smoothed search spaces which corresponds to a decreasing sequence of α can achieved better results than the original local search algorithm. Gu attributes this improvement to the fact that final solution of the previous search space could guide local search in the current search space. Schneider et al. [7] designed several new smoothing function for SSS. They conducted experiments on two larger instances of 442 and 532 cities each (The largest number of cities of test instances in [1] is 100) and listed the results obtained by different smoothing functions. The solutions of minimum length for "hyperbolic" and "logarithmic" functions are within 0.5% excess of optimal. However, no running time information is provided in this paper.

Two kinds of smoothing effects are identified and discussed in this paper. After a smoothing operation, probably the absolute value of energy difference of two neighborhood solutions is lowered, with the sign of the value unchanged. Probabilistic acceptance rather than the greedy, deterministic one described in [7] can make use of this kind of smoothing effect. Previous publication uses similar techniques to solve placement problem in VLSI physical design [2][3]. In this paper, we provide a systemic discussion on the design of SSS based algorithm for Traveling Salesman Problem. Our experiments is also TSP-based so that we can compare our results with the heuristics in [1].

The remainder of this paper is organized as follows. In Section 2, basic ideas of search space smoothing algorithm is introduced. In Section 3, we give formal definitions of global smoothing and local smoothing, and show the existence of them with numerical experiments. In Section 4, we discuss how to implement and design an efficient SSS variant. In Section 5, experimental data on random and TSPLIB instances were summarized. Finally in Section 6, we give our conclusion remarks.

2. Basic idea of SSS

The general procedure of SSS algorithm in [1] could be stated as follows:

```
Procedure SSS()
Begin
  distancematrix d,d1;
   readInstance(d);
  tour t= initTour(d);
  float α=α0;
  //Search
  while (α > 0) do begin
          d1= smooth(d, α);
          t  = localSearch(d1,t);
          α  = nextAlfa(α);
  end;
  t= localSearch(d1,t);
End;
```

Smoothing is operated on the distance matrix, which represents a unique search space. The procedure first generates a starting tour (usually by running a tour construction heuristic). Then local search is performed in each search space: the starting tour is the final one in last iteration. Here α is called *smoothing factor*, which corresponds to the strength of the smoothing operation. For convenience, we could think that the larger the factor α is, the

*This paper is supported by the NSFC 60473126

smoother the search space becomes, and the search space restores to the original one when $\alpha = 0$.

3. The Smoothing Effect

3.1 Smoothing for Traveling Salesman

The Traveling Salesman Problem is one of the most well-known combinatorial optimization problems. It has been long served as a standard testbed for algorithmic ideas. The problem can be simply stated as follows: given a collection of cities, find the shortest loop that visits each city exactly once.

TSP is known to be NP-hard. Its search space, i.e. all the solutions (with different energy values) and their neighborhood structure, is of exponential size regarding to the number of cities.

One of the most simple yet efficient methods for solving TSP are local search with exchange neighborhood. Such as 2-exchange (also 2-opt) move[1]. [7] also includes more complicated 3-opt moves. However, we find that accepting probability of 3-opt moves is less than one tenth of 2-opt moves. And in practice, eliminating such moves in the algorithm leads to little degeneration of the tour quality while saving a lot running time. In our implementation, only 2-opt moves are considered.

3.2 Global and Local Smoothing

We define the two kinds of smoothing effect formally and perform numerical experiments to show their existence. Here a solution (feasible tour) is usually represented by s with proper subscript. The search landscape is L := (S, N, g), S is the set of all solutions, N is neighborhood function and g is the objective function which evaluates the "energy" of a solution. For a smoothed search landscape, S and N remains the same, the objective function is denoted by g_α.

Global Smoothing

Definition 1: A solution s_0 is a **local minimum**, if and only if

$$(\forall s)\left(s \in N(s_0) \rightarrow g(s_0) \le g(s)\right)$$

After the smoothing operation, the local minimum s_0 is **eliminated**, if and only if

$$(\exists s)\left(\left(s \in N(s_0)\right) \wedge \left(g_\alpha(s_0) > g_\alpha(s)\right)\right)$$

Definition 2: If there is any local minima s_0 that is eliminated in the smoothed search space, the search space is **globally smoothed at s_0**.

To demonstrate the existence of global smoothing effect, we perform smoothing operations on small and large instances. We randomly generate 10 instances with 12 cities and find all the local minima by examining all the permutation of cities, then compare the number and quality of local minima before and after smoothing for each instance. Results are listed in Table 1 below. The smoothing function is the exponential one. "excess" is the average percent excess of all local minima over the global minimum.

Table 1. Statistics of local minima for small instances

Smoothing Factor	0		1	
Instance	# local minima	excess	# local minima	excess
1	5	8.8	2	0.2
2	2	0.3	2	0.1
3	5	3.2	2	1.0
4	3	5.5	2	1.8
5	5	1.8	5	0.7
6	3	4.6	2	0.4
7	1	0.0	1	0.0
8	3	2.0	4	1.1
9	10	5.5	10	0.5
10	8	11.4	12	2.5

For larger instances, it is not realistic to find all local minima. The global smoothing effect is indirectly shown as follows: randomly generate 100 solutions and use 2-opt local search to find the nearest local minimum under the 2-exchange neighborhood. Calculate the average pairwise distance for all solutions which approximates the density of local minima. The distance between two solutions is the total number of edges subtracted by the number of edges that are shared by the two solutions. For instances with the same number of cities, smaller value for local minima density implies that the number of local minima is also smaller. We performed experiments on pcb442, a well-known TSPLIB[5] instance. The result is in Table 2:

Table 2. Avg pairwise distance of 100 local minima for pcb442

Smoothing Factor	0	6
Avg distance	182.87	204.76

Local Smoothing

Definition 3: For a solution s_0, denote the size of its neighborhood set by $|N(s_0)|$, thn the **local ruggedness at s_0** is described by

$$LR(s_0) = \sqrt{\frac{1}{|N(s_0)|}\sum_{s \in N(s_0)}\left(\left(g(s) - g(s_0)\right)^2\right)}$$

Definition 4: Suppose the average energy of the neighborhood of a solution s_0 is

$$AVGE(s_0) = \frac{1}{|N(s_0)|+1}\sum_{s \in \{s_0\} \cup N(s_0)} g(s)$$

For the smoothed search space under parameter α, the local ruggedness and average energy of the

neighborhood at s_0 are denoted by adding a subscript α, if the following formula holds,

$$\frac{LR_\alpha(s_0)}{AVGE_\alpha(s_0)} < \frac{LR(s_0)}{AVGE(s_0)}$$

then the search space is **locally smoothed at s_0**.

To demonstrate the existence of local smoothing effect, we calculate the $LR_\alpha(s_0)/AVGE_\alpha(s_0)$ values for pcb442 when α = 0,1 and 2. The values listed in Table 3 is the average value for 100 randomly generated tours("random solution") and their nearest local minimum under 2-exchange neighborhood ("random minima") respectively. The values decrease sharply as smoothing factor increases, indicating significant local smoothing effect.

Table 3. Normalized local ruggedness for pcb442

Smoothing Factor	0	1	2
random solution	$1.9*10^{-3}$	$2.1*10^{-4}$	$2.7*10^{-5}$
random minima	$6.0*10^{-2}$	$8.5*10^{-4}$	$7.6*10^{-5}$

3.3 Stochastic Local Search for SSS

$$A_{ij}^{(t)} = \begin{cases} 1 & g(s_j) - g(s_i) \le 0 \\ \exp\left(\dfrac{g(s_j) - g(s_i)}{kt}\right) & g(s_j) - g(s_i) > 0 \end{cases}$$

The formula above is the accepting probability of the transition $s_i \to s_j$ of Simulated Annealing[7], t is the temperature and k is the constant to normalize different energy levels of various search spaces. Such stochastic local search can make use of local smoothing effect. By replacing the greedy accepting strategy of [7] by the following function, with the temperature t substituted by smoothing factor α, we can introduce probability to original SSS so that it is sensitive to both global and local smoothing effect. This is the basic idea of our SSS variant. Details about the algorithm design are proposed in the next section.

4. Algorithm Design

we employ exponential smoothing function which is sensitive to distances with small deviation from the average distance even for a small smoothing factor. Other three elements of algorithm design is discussed in the remainder of this section.

Neighborhood Structure and Pruning

We choose 2-exchange neighborhood for the neighborhood structure. Its greatest advantage is low time cost in generating a neighborhood solution. Although for 3-exchange or more complicated neighborhood there are less local minima in a search space, the average level of the barrier around a local minimum is much higher, leading to a significant lower acceptance probability. Therefore, complicated neighborhood is not always efficient.

Previous SSS implementations actually do not have a sampling rule and pick up a solution from the whole neighborhood set. We employ the neighborhood pruning introduced in [4] (originally it is a speedup technique for Simulated Annealing, which includes the basic idea of fast 2-opt implementation, refer to pp. 268). The subset selected from the neighborhood set contains all the improving neighborhood solution and its size is $\Theta(N)$ compared to the $\Theta(N^2)$ 2-exchange neighborhood. As a result, the probability of acceptance after neighborhood pruning is higher. So the *max_count* variable, i.e. the total number of times of repeating the "pick up", can have a smaller value, saving the running time for SSS. We set *max_count* to be 20N in our implementation, and N is the number of cities.

Probability Accepting Function

The probability accepting function is similar to the formula in Section 3.4 but is in a more general form as follows:

$$A_{ij}(\alpha) = \begin{cases} 1 & g(s_j) - g(s_i) \le 0 \\ \exp\left(\dfrac{g(s_j) - g(s_i)}{k\alpha gp(\alpha)}\right) & g(s_j) - g(s_i) > 0 \end{cases}$$

The term k is to normalize different level of energies for different instances with different smoothing factors. A good estimation is

$$k = k_0 g \frac{g_\alpha(s_0)}{g_0(s_0)}$$

k_0 estimates the energy level of original search space and is proportional to the tour length of a 2-opt with a random nearest neighbor start. $g_\alpha(s_0)$ and $g_0(s_0)$ are the length of the starting tour of local search in the search space smoothed under the parameter α and that of the same tour in the original search space respectively. $g_\alpha(s_0)/g_0(s_0)$ estimates the difference of energy level between the smoothed search space and the original one.

Although the idea of probabilistic acceptance is inspired by Simulated Annealing, it is not necessarily similar to the case of SA. For SSS, adding a $p(\alpha)$ term to the power of α is able to increase the accepting probability when α is small. In our implementation, it is a multi-segmental function and the value of k also varies with α to make the function continuous.

5. Experimental Results

In the experiment, we demonstrate performances of both the original SSS algorithm with 2-opt as the local search algorithm(SSS-1) and the variant

proposed in Section 4 (SSS-2) for random Euclidean TSP instances and those from TSPLIB95[5]. All computations of the algorithm written in C++ are compiled by g++ 3.3.3 and performed on an Intel Pentium IV 3.2G PC with 2GB main memory) with Windows XP and cygwin. Random instances are generated using the "portgen" utility downloadable from SGI challenge page[5].

All the SSS implementations use exponential smoothing. For SSS-1, α is initially set to 6 and reduced to zero at a step of 0.1. More subtle evolving schedule of the smoothing factor is unnecessary because the original SSS is rather robust in this aspect. The initial solutions are obtained by nearest neighbor tour construction heuristic with the starting city randomly chosen.

Table 4 summarized the average tour quality and running time of SSS algorithms(SSS-1, SSS-2) and Simulated Annealing (SA, with low temperature start and neighborhood pruning, α). The tour quality is measured by percent excess over Held-Karp lower bound. Running Time in Seconds is normalized to be the predicted user seconds on Compaq ES40 6/500 alpha machine, 500 Mhz Processor, 2Gb of RAM. Tour quality and running time data for SA is from pp. 270 in [4]. For SSS-1 and SSS-2, 5 random instances are generated for each problem size and 5 runs are conducted for each instance.

Table 4. Experimental results for SSS algorithms(1)

Random Euclidean Instances						
	Average Percent Excess			Running Time in Seconds		
Algorithm	10^2	$10^{2.5}$	10^3	10^2	$10^{2.5}$	10^3
SA_2 α=100	1.1	1.3	1.6	56	$2.6*10^2$	$7.6*10^2$
SSS-1	8.1	8.0	5.4	0.54	5.8	89
SSS-2	3.1	1.5	1.2	16	$1.5*10^2$	$2.0*10^3$

Table 5 below summarized the average tour quality and normalized running time data for six TSPLIB instances. Their problem sizes range also range from 10^2 to 10^3. 5 runs are conducted for each instance for SSS-1 and SSS-2. The tour quality is measured by percent excess over optimality. Running Time in Seconds is the original data without normalization. We see a similar trend on these real-world instances, i.e. SSS-2 takes more time but gets much better tours. It is also interesting to see that SSS-2 performs better in TSPLIB instances than random Euclidean ones.

6. Conclusion

Based on the observation that both local smoothing and global smoothing effects exist in the meta-heuristic Search Space Smoothing, an efficient algorithm integrating stochastic local search is proposed in this paper. By introducing improving techniques for neighborhood generation, the new algorithm is able to take the advantage of smoothing effect efficiently. Experimental results on random Euclidean and six TSPLIB instances demonstrated that proposed approaches achieve better tour quality than original SSS algorithm at the expense of increased running time.

Table 5. Experimental results for SSS algorithms (2)

Six TSPLIB Instances						
	Average Percent Excess					
Algorithm	kroA100	lin318	pcb442	att532	rat783	dsj1000
SSS-1	15.2	6.8	5.9	10.6	8.2	9.3
SSS-2	0.9	1.6	0.7	0.9	1.3	1.3
	Running Time in Seconds					
Algorithm	kroA100	lin318	pcb442	att532	rat783	dsj1000
SSS-1	0.2	0.7	2.9	5.3	12	26
SSS-2	4.6	43	85	130	270	550

7. References

[1] J. Gu and X. Huang. Search Space Smoothing: A Case Study of the Traveling Salesman Problem (TSP). *IEEE Trans. Systems, Man, and Cybernetics*, 24(5):728-735, 1994.

[2] S. Dong et al. VLSI module placement with pre-placed modules and considering congestion using solution space smoothing. In *Proc. 2003 Asia and South Pacific Design Automation Conference(ASPDAC'03)*, pages 741-744, 2003.

[3] S. Dong et al. Solution Space Smoothing With Five Smoothing Function for VLSI Module Placement. In *Proc. 5th Interational Conf. on ASIC(ASICON'03)*, pages 132-135, 2003.

[4] D. S. Johnson and L. A. McGeoch. Experimental analysis of heuristics for the STSP. In G. Gutin and A. P. Punnen, editors, *The Traveling Salesman Problem and Its Variations*, pages 369-443, Kluwer Academic Publishers, Boston, 2002

[5] D. S. Johnson. *8th DIMACS Implementation Challenge: The Traveling Salesman Problem*. http://www.research.att.com/~dsj/chtsp/index.html. Latest Update: 16 November 2004.

[6] G. Reinelt. *The Traveling Salesman Problem: Computational Solutions for TSP Applications*, Lecture Notes in Computer Science 840, Springer-Verlag, Berlin, 1994.

[7] J. Schneider et al. Search-space smoothing for combinatorial optimization problems. *Physica A*, 243(1):77-112, 1999