A Mushy State Simulated Annealing

K. Shojaee G.[i], H. Shakouri G. [ii] * and M.B. Menhaj[iii]

Received 06 Mar 2010; received in revised 15 Jun 2011; accepted 24 Oct 2011

ABSTRACT

It is a long time that the Simulated Annealing (SA) procedure has been introduced as a model-free optimization for solving NP-hard problems. Improvements from the standard SA in the recent decade mostly concentrate on combining its original algorithm with some heuristic methods. These modifications are rarely happened to the initial condition selection methods from which the annealing schedules or the time schedule itself start. There are several parameters in the process of annealing, the adjustment of which affects the overall performance. This paper focuses on the importance of initial temperature and then proposes a lower temperature with low energy to speed up the process, using an auxiliary memory to buffer the best solution. Such an annealing indeed starts from a "mushy state" rather than a quite liquid molten material. The mushy state characteristics indeed depends upon the problems that SA is being applied to solve for. In this paper, the Mushy State Simulated Annealing (MSSA) is fully developed and then applied to the popular Traveling Salesman Problem (TSP). The mushy state may be obtained by some simple methods like crossover elimination. A very fast version of a Wise Traveling Salesman, who starts from a randomly chosen city and seeks for the nearest one as the next, is also applied to initiate SA by a low-energy, low-temperature state. This fast method results in quite accurate solutions compared to the methods recently cited in the literature.

KEYWORDS

Combinatorial Optimization, Traveling Salesman Problem, Simulated Annealing, Initial Condition.

1. INTRODUCTION

Simulated Annealing (SA) is one of the earliest methods for derivative-free optimization such as Tabu Search (TS) [1]. Although it was introduced first to solve combinatorial discrete problems [2], it has recently shown a high attitude for solution of continuous problems as well [3]-[5]. SA is derived from physical behaviour of molten metal when the temperature is slowly falling to forms a regular crystalline solid structure. There are two key parameters in the cooling process that determine how firm or amorphous will be the result for the metal in its frozen state. The first one is the initial temperature from which the cooling starts; and the second is the rate by which the temperature is falling.

Concerning the rate of decay, it should be low enough to allow the atoms in the molten metal to line them up and give enough time to form a crystal lattice with the minimum internal energy. Evidently, a slow decay will lead to a long time for the solidifying process. To reduce the time, one may think of a low initial temperature. However, on the other hand, if the initial temperature is not high enough, atoms of the molten metal would not have enough freedom to rearrange their positions in a very regular minimum energy structure.

Although there are some theoretical limits and formulations to choose a proper cooling rate [6]-[22], there is not any deterministic criterion in the literature to set the initial pseudo-temperature. For instance, applying the standard SA to the travelling salesman problem (TSP), one may set it to 0.5 and change it by 10% at each iteration [9], while some others prefer the initial temperature of 1000 and reducing it by 1%, i.e. a factor of 0.99 [10]. Moreover, the concept is a case dependent one and even may not fit to a bounded range, e.g. in some papers it is even initialized in a range from 0.001 to 100 [11].

There are a few research papers that suggest a formulation to relate the initial temperature to particular characteristics of the problem. Pao et al. considered an initial temperature such that the initial acceptance rate is about 70% [12]. Feng-Tse Lin, et al. proposed an Annealing-Genetic approach and used the following formula [13]:

[i] Low-Power High-Performance Nanosystems Laboratory, School of Electrical and Computer Engineering, University of Tehran (k.shojaee@ece.ut.ac.ir)

[ii]* Corresponding Author, Industrial Engineering Department, University of Tehran, Tehran, Iran (hshakouri@ut.ac.ir)

iii Electrical Engineering Department, Amirkabir University of Technology, Tehran, Iran.

$T_0 = \Delta E$ / (Population Size/2),

where ΔE is the difference between the highest cost and the lowest one found for the first generation of the randomly generated population.

Thompson and Bilbro set the initial temperature by defining a probability function for energy change in continuous problems. The probability of accepting a higher cost solution was set to 0.75. The following probability distribution is then solved to find T_0 [14]:

$p = \exp(\Delta E / T)$,

where ΔE is the average cost of the random solutions plus its standard deviation.

Hao Chen et al. set the initial temperature such that the initial acceptance probability for an average uphill move becomes 0.97 [15].

Although SA algorithms are conceptually simple, it is by no means simple or straightforward to find optimal parameters such as initial temperature, the annealing schedule, the acceptance function parameters, etc. The reasons may be stated as, 1) setting parameters for SA is problem dependent, and it is best accomplished through trial and error, 2) many studies have demonstrated that SA algorithms are very sensitive to parameters, and their performances are largely dependent on fine tuning of its parameters [16].

The problem dependent nature of setting parameters for SA and its sensitivity to parameters limit the effectiveness and robustness of SA algorithms. SA possesses a formal proof of convergence to the global optimum. This convergence proof relies on a very slow cooling schedule of setting the initial condition to a sufficiently large temperature and let it decay by:

$$T_k = T_0 / \log(k),$$

where k is bounded by the number of iterations [17]. While this cooling schedule is impractical, it identifies a useful trade-off where longer cooling schedules tend to lead to better quality solutions.

Furthermore, the stochastic simulated annealing (SSA) [7] tends to find a global optimum if the annealing process is carried out sufficiently slowly. It means that SSA is able to find high-quality solutions (global optimum or near-global-optimum), if the temperature is reduced exponentially but with a sufficiently small rate of decay. For many applications, this may lead to a prohibitively long relaxation time in order to find solutions of acceptable quality, and conversely, reasonably long periods of time may still result in poor solutions. Lipo Wang et al. have used chaotic neural networks in combination with the best features of SSA and have shown experimentally the effectiveness of this new stochastic chaotic simulated annealing (SCSA) [18]. However, there is not any especial idea on the initializing or the cooling schedule in their approach. It is woth mentioning that Yuyao had also applied earlier a chaotic noise to a Hopfield neural network and had set the annealing process such that the chaotic noise gradually reduced. Though the process was initially chaotic, eventually it was convergent, leading to a richer and more flexible dynamics [19].

In summary, we observe that there is a trade-off between choosing a high initial temperature (or choosing a low rate of cooling), and gaining a short processing time (or finding the minimum energy structure). A similar trade-off exists when applying SA to any optimization problem like TSP. Assuming the objective function of an optimization problem to be an energy function, and the initial guesses for the unknown variables to be the original problem, the aforementioned trade-off surely appears as depicted below:

Initial temperature ↑ $\Rightarrow$ Optimization time ↑
Initial temperature ↓ $\Rightarrow$ Final energy ↑ (Local minima)
Rate of cooling ↑ $\Rightarrow$ Final energy ↑ (Local minima)
Rate of cooling ↓ $\Rightarrow$ Optimization time ↑

It is easy to deduce that selection of a proper set of optimization parameters for SA is itself a multi-objective decision making (optimization) problem. In this paper, we will investigate the first one, i.e. the initial temperature, and propose an approach to speed up the algorithm, while obtaining accurate solutions for the popular case study of TSP.

It is usual to select a very high temperature that provides a suitable initial condition with enough mobility for the atoms to move freely to new locations far away enough in order to form as possible as minimum-energy structures. A certain criterion is to set the initial temperature large enough that almost any trial point (state) will be admissible. This may cause the SA process to experiment new admissible points with even higher energy states. As the temperature decays, the probability to accept states that do not reduce the energy decreases.

Since the cooling process that starts from a high temperature in a liquid-like state is time consuming, this paper proposes to start annealing from a state in a lower temperature with a lower internal energy. Such a state is called a “mushy state”, rather than a liquid state. In such reduced temperatures with low energy, the ratio of admissible states to the total trials may be less than 10%, compared to that of usual high temperatures.

After that the state is set to a lower energy state in a lower initial temperature, the annealing process can bring us the benefit of a faster local search and find the optimal state with the minimum energy. Starting from a very high temperature the metal should be cooled slowly, otherwise the atoms do not have time to orient themselves into a regular structure; however, if the initial state is imposed to the atoms in a low energy low temperature, we can

adjust the cooling rate to be faster.

Perhaps there are many optimization methods, even direct (random search methods) that can be applied as a prelude for SA. A simple algorithm that is used in this paper is to eliminate all intersecting paths in an initially selected random tour. A second simple method is also applied in this paper to show independence of MSSA to the method that the initial tour is found.

The rest of the paper is organized in five sections. The standard SA and TSP are briefly introduced in sections 2 and 3, respectively. Section 4 describes first how to choose the initial conditions and how to schedule the annealing, and then presents the MSSA algorithm. The results obtained applying the proposed method are given in section 5, where we have compared the best, the worst and the average error in the final solution (if available) with some recent studies. Finally, section 6 concludes the paper.

2. A SHORT OVERVIEW ON THE SA

Rather than giving a detailed description of SA, herein the fundamental terminology of SA is explained shortly [8],[21]. The method consists of four main parts.

A. Objective function

An objective function $f(.)$ is a mapping from an input vector $\underline{x}$ into a scalar E:

$$E = f(\underline{x}), \tag{1}$$

where each $\underline{x}$ is assumed as a point in an input vector space. The SA is to sample the input space effectively to find an $\underline{x}$ that minimizes E. The input vector may be the structure of the atoms and/or their movements limited to that structure, and E may be the internal energy of the metal. In TSP, $\underline{x}$ is the tour sequence and E is the total cost (distance) of travelling.

B. Generating function

A generating function $g(., .)$ specifies the probability density function of the difference between the current point and the next point to be visited. Specifically, $\Delta\underline{x} = \underline{x}_{new} - \underline{x}$ is a random variable with probability density function $g(\Delta\underline{x}, T)$, where T is the pseudo-temperature. If E is the internal energy of the metal, T is the real temperature; however, in TSP it can be interpreted as percentage of the new points leading to reduction of the total cost. Clearly, if the number of intersecting paths in a tour $\underline{x}$ is large, we can assume that the pseudo-temperature is high. Usually $g(., .)$ is independent of the temperature. However, in the conventional SA, also known as Boltzmann machines, the generating function is a Gaussian probability density function:

$$g(\Delta\underline{x}, T) = (2\pi T)^{-n/2} \exp[-|\Delta\underline{x}|^2 / (2T)], \tag{2}$$

where n is the dimension of the space under exploration. One may replace the Cauchy distribution with a fatter tail for the Gaussian distribution to give the chance to explore new points in the space farther from the current point while searching the space.

For discrete or combinatorial optimization problems, like TSP, each $\underline{x}$ is not necessarily an n-vector with unconstrained values. Instead, each $\underline{x}$ is restricted to be one of N points that comprise the solution space or the input space. Usually N is very large but finite so that to reduce probability of a time consuming search without any result. Since, adding randomly generated $\Delta\underline{x}$ to a current point $\underline{x}$ may not generate another legal point in the solution space, instead of using generating functions, *a move set* is usually defined to find the next admissible point, denoted by $M(\underline{x})$. This creates the set of admissible points available for exploration after $\underline{x}$. Usually the move set, i.e. $M(\underline{x})$, is chosen in the sense that the objective function at any point of the move set, i.e. a set of *neighbouring* points $\underline{x}+\Delta\underline{x}$, will not differ too much from the objective function at $\underline{x}$. The definition of the move set is problem dependent. For TSP there are at least three kinds of move sets that are properly defined and frequently used by researchers: *Inversion*, *Translation*, and *Switching* [21]. An especial variant of inversion is the simple idea of *Crossover Elimination*.

Once the move set is defined, $\underline{x}_{new}$ is usually selected at random from the move set, such that all neighbouring points have an equal probability of being chosen. In this paper, we have fixed the move set to the inversion, which has shown better performance compared to that of the others.

C. Acceptance function

After that the objective is evaluated for a new point $\underline{x}_{new}$, SA decides whether to accept or reject it based on the value of an acceptance function $h(.,.)$. The most frequently used acceptance function is the *Boltzmann probability distribution*:

$$h(\Delta E, T) = \frac{1}{1 + \exp(\Delta E/(cT))} \tag{3}$$

where c is a constant, T is the temperature, and ΔE is the energy difference between $\underline{x}_{new}$ and $\underline{x}$:

$$\Delta E = f(\underline{x}_{new}) - f(\underline{x}) \tag{4}$$

Usually x_{new} is accepted with probability $h(\Delta E, T)$. If ΔE is negative, SA tends to accept the new point to reduce the energy. Nevertheless, if ΔE is positive SA may also accept the new point and move to a higher energy state. It means, SA can move uphill or downhill; but the lower the temperature, the less likely to accept any significant upward change.

There are several alternatives for the acceptance function. A simple alternative with approximately the same behaviour is:

$$h(\Delta E, T) = \exp(-\frac{\Delta E}{cT}) \quad (5)$$

where there is no need to check for the sign of ΔE. Instead, if $h(\Delta E, T)$ is greater than a uniformly distributed random number, the new point is accepted. A deterministic alternative method is to use *Threshold Accepting*, where $\underline{x}_{new}$ is accepted just if $\Delta E < cT$ [22].

D. Annealing schedule

An annealing or cooling schedule regulates how rapidly the temperature T goes from high to low values, as a function of time or iteration counts. There are not so many studies discussing the initial temperature selection or even the cooling schedule. Indeed, the exact interpretation of *high* and *low* and the specification of a good annealing schedule require certain problem-specific physical insights and/or trial-and-errors. The easiest way of setting an annealing schedule is to decrease the temperature T by a certain percentage at the k^{th} iteration:

$$T_{k+1} = \alpha\, T_k \quad (6)$$

where $0 < \alpha < 1$ is an adjusting parameter. It is proved that a *Boltzmann machine* using the aforementioned generating function can find a global optimum of $f(\underline{x})$ if the temperature T is reduced not faster than $T_0 / \log(k)$ [17]. Researchers have used various cooling strategies, among which we choose the following:

$$T_{k+1} = T_k / \log(k^{1/D}), \quad (7)$$

where D is set to 2.

3. TRAVELING SALESMAN PROBLEM (TSP)

Travelling Salesman Problem (TSP) seems to be the most well-known typical NP-hard problem. Given a set of nodes and a set of weights specifying cost to travel between each two nodes, the optimal solution is to find a closed loop of the paths with minimal total weights in a finite complete graph.

Let us denote the set of "cities" in TSP as $C = \{c_1, c_2, ..., c_n\}$ accompanying a matrix D_T, an element of which is called d_{ij} that gives the distance or cost function (weight) for going from t_i, to t_j. In real problems, usually the coordinates of the cities are given, by which the matrix D_T can be easily computed.

The path linking the two cities here is called a "link". A sequence of cities $C^* = [c_{s1}, c_{s2}, \ldots, c_{sn}]$ denotes an admissible solution of TSP (the salesman must visit each city once and only once), where $\{s_1, s_2, ..., s_n\}$ is a sequence of $\{1, ..., n\}$. Then, the Travelling Salesman Problem's optimal goal can be expressed as minimizing the following objective function interpreted as the so-called energy function:

$$E(C) = d_{s_1 s_n} + \sum_{i=1}^{n-1} d_{s_i s_{i+1}} \,. \quad (8)$$

where $C = [s_1, s_2, ..., s_n]$ is the travelling tour. If all the costs between any two cities are equal in both directions, i.e. D_T is a symmetric matrix, the problem is called symmetric TSP; otherwise, it is called asymmetric [23].

Sometimes D_T is calculated based on the coordinates of the cities that may generate real numbers. Normally d_{ij}'s are rounded to integer numbers to standardize the results according to the standard code proposed in [23]. Generation of the distance matrix, D_T, given the coordinates is a straightforward procedure. However, the reverse process is not possible for all cases.

Suppose the coordinates are given by two vectors namely X and Y. There are $2n$ elements in the vectors X and Y, while a symmetric D_T contains $\frac{1}{2}\, n \times (n-1)$ elements. Each equation can be written as:

$$(x_i - x_j)^2 + (y_i - y_j)^2 = d_{ij}^2; \quad i = 1, \ldots, n;\ j = i+1, \ldots, n-1; \quad (9)$$

where xi , yi , xj and yj are the ith and jth elements in X and Y, respectively. Therefore, the solution of $n\,(n-1)/2$ nonlinear equations available from D_T requires:

$$2n \le \tfrac{1}{2}\, n \times (n-1),$$

or equivalently:

$$n \ge 5$$

in order to find the $2n$ unknown coordinates in X and Y.

Thus, there will not be a unique solution for cases with more than 5 cities. However, a feasible solution will suffice to apply the above modification to all cases with $n \ge 5$. Note that the nonlinear equations should be solved just once to find the feasible solution for the coordinates X and Y. It is obvious that for an asymmetric D_T there is no solution without any extra information.

4. INITIALIZING AND ANNEALING SCHEDULE

As mentioned before, there are not many papers discussing about the initial condition issue when the annealing process starts. The main idea proposed in this paper is originated from the behaviour of the metal during the annealing process. The cooling schedule is an exponential shape function of the time that can be divided into three parts. The first part is a rapidly decaying curve with a high slope in average and the last one is the ending part of the exponential function with an almost frozen state. The first part should be passed as fast as possible, while complying the lower bound on the rate of cooling, noting that the last part has almost no significant effect on the final result. Therefore, none of these two states are of interest in this paper. The initial temperature is proposed to be selected within the middle of the curve, as shown in Figure 1.

To assign a proxy for the temperature, we may use a ratio named γ defined as:

γ = the ratio of accepted new points to the total trials.

If the ratio is high, it means that the internal energy is

high enough to move many of atoms so that the energy reduction becomes possible. The ratio should be close to one, say 0.9. Conversely, if the ratio is low, say 0.1, the metal is nearby to become frozen; there are not so many new structures that reduce the energy. We propose to start annealing from such initial conditions in the middle zone, i.e. a mushy or doughty state, rather than in liquid or firm states.

E. Initiating temperature and energy

We have applied two different methods to initialize both the temperature and energy in the mushy state. The first one is a simple algorithm for crossover elimination and the second one is an efficient fast simple method derived from a rough behaviour of a Wise Travelling Salesman (WTS), who seeks for the next nearest city. The following subsections describe these two methods.

1) Crossover Elimination

In the special case of TSP, knowing that the optimal tour will not contain any intersection of the paths, a simple yet fast algorithm of intersection detection and elimination is applied. Starting from a randomly generated initial tour, every couple of links with crossover should be deleted and replaced by swapping the two overlapping links. Figure 2 easily illustrates this idea.

To do so, without loss of generality, let us continue describing the algorithm for the case in which the input data is given in terms of its co-ordinates. Based on this assumption, we assume that the co-ordinates are arranged in two vectors named X and Y. Now, let assume that the initial random tour is named C_0. This vector is a sequence of the city indices:

$C_0 = [1, \ldots, c_i, c_{i+1}, \ldots, c_j, c_{j+1}, \ldots]$.

Suppose the subscripts of the elements of the sequence be the same as shown in Fig. 2. Therefore, the line equations for all links of the tour can be calculated, by which it is possible to check that if each of the two paths are intersected or not. We need to solve $\frac{1}{2} n \times (n - 3)$ linear equations, where any valid solution should be in the range of the coordinates X and Y, subsequently requiring:

$x_{\min} < x_c < x_{\max}$

$y_{\min} < y_c < y_{\max}$

where, x_c and y_c are coordinates of the intersection point of each pair of links with non-common ends, and $x_{\min}$, $x_{\max}$, $y_{\min}$ and $y_{\max}$ are respectively the minimum and maximum coordinates found on the two links. Note that checking one of the above two conditions suffice to ensure intersection occurrence. Then, if there is a cross over, like in Fig. 2 (a), the sequence should be modified to generate T_m as:

$C_m = [1, \ldots, c_i, c_j, \ldots, c_{i+1}, c_{j+1}, \ldots]$.

And as depicted in Fig. 1 (b). It means that the $(i+1)^{th}$ element should be exchanged with the j^{th} element. Then, it is clear that the total cost will be reduced, i.e.:

$E(C_m) < E(C_0)$.

Once that crossover occurrence is checked for $\frac{1}{2} n \times (n - 3)$ times in a tour and resolved by swapping the links, there may be new intersections generated. Therefore, the algorithm is iterated until no crossover is found in the final tour.

The algorithm is now summarized below:

1) Generate an initial tour randomly;
2) Check for intersection of each path with all other non-neighbouring paths;
3) If there is a crossover, remove it by swapping the paths;
4) Repeat steps (b) and (c) until there is no crossover in the tour.

2) Wise Traveling Salesman (WTS)

The main thought beyond this algorithm is the way that a normal wise person may roughly decide on its next destination at each current position. For the first time that a normal person starts his/her travel, he/she may guess that a good path perhaps can continue from the nearest city. Indeed, at each step, the next city to travel is chosen among the nearest cities to the current city. Besides this original idea, one may apply a random walk process to let the traveler experience new experiments while using his/her wisdom to choose its next destination. It means that he/she examines other paths in the next experimental tour by changing some of the cities in the sequence randomly. Finally, he/she will give a weighting for his/her previously experienced tours in the next travel. However, in this paper we have not applied these two factors for initializing SA. Thus, the simplest version of this greedy method, which is discussed in details in [30], can be summarized by the following steps:

1) Select the starting city randomly;
2) Compute the costs from the current city to all unvisited cities in a vector;
3) Sort the resulting vector elements and choose the next city with the least cost;
4) Calculate the total cost (distance) after completing the tour.

This way, the initial energy, and consequently the initial temperature when starting SA, will be much lower than its usual values.

F. Annealing schedule

The proposed method also includes modifications to the annealing schedule to compensate for side effects of shortening the annealing process. The first modification is to repeat generation of new points at each temperature

until the acceptance becomes A_{Max}. The maximum iterations, sometimes called the Markov chain length, is initially considered such that $A_{Max} = 100 \times n$, where n is number of the cities in TSP. Since the acceptance probability exponentially decreases at each temperature, we may reduce A_{Max} by a constant ratio, say 0.9. However, in very low temperatures the acceptance probability is very low and there should be a criterion to agree that it is the freezing temperature. In this research, if the number of total iterations at each temperature, i.e. the Markov chain length, exceeds $A_{Max} \times n$, we stop the annealing process and call the current temperature as the freezing point.

Furthermore, in a normal SA, if the optimum solution is lost once that the algorithm faces with an uphill acceptance there is enough opportunity to recover it in the future steps. When starting the annealing from low temperatures, the expected time to find the global optimum becomes shorter. Therefore, the first modification is a memory to keep the latest minimum energy solution found during all steps passed in the annealing. The best tour found is saved and inserted once in the process at each temperature in order to be used later in case to be compared with the current situation. This will resolve the probability of missing the best previously obtained solution.

This way the annealing process is summarized as:

1) Set the initial temperature to a mushy state temperature, by employing a simple locally optimizing algorithm, where the acceptance ratio, γ, is about 10%;
2) Set the maximum acceptance to $A_{Max} = 100 \times n$;
3) Start the annealing as usual, while:
   Changing A_{Max} to $0.9 \times A_{Max}$;
   Saving the best solution found up to now;
   Inserting the best solution to each Markov chain;
4) Stop the annealing process if the current Markov chain length is greater than $A_{Max} \times n$.

5. SIMULATION RESULTS

The proposed initializing method, so called Mushy State Simulated Annealing (MSSA) [31], is applied to many benchmarks listed in TSPLIB [23]. MSSA is run 40 times: 20 runs with an initial condition obtained by crossover elimination and 20 runs initialized by WTS. The initial conditions, i.e. the initial pseudo-temperature and the initial energy are case dependent parameters. As mentioned in Section 4, we used the ratio of accepted motions (new points), which are found based on an acceptance function like eqn. (5), to the total number of tried motions (γ). If this ratio tends to zero, the case is in its solid state, and if it is very high (near to 1), then it is in the liquid state.

Figure 3 shows the relationship between the pseudo-temperature and the acceptance-trial ratio for the case of eli51 (a benchmark given by TSPLIB). It is seen that in high temperatures the ratio saturates to about 1, and in low temperatures the case has reached to its solid state with the minimum energy. Therefore, the best initial condition to start annealing is a temperature close to the melting point or the take off point in the curve, which is about the pseudo-temperature of 30 in this case.

The results for cases with less than 1432 cities are given below in Table 1, through which it will be easy to realize that the method has significantly improved SA. The optimal values given by the TSBLIB site, for each case are listed in the second column of the table. We have compared the best, the worst and the average of the errors in the results obtained by the proposed approach with those of given by recent novel studies (if the best and/or the worst cases are available). The error percentage is calculated by:

$$\delta = 100 \, (E - E^*) / E^*$$

where E^* is the optimal (minimum) energy.

The first method chosen for comparison is a Constructive Optimizing Neural Network (CONN) proposed in [24], for which it is claimed that all runs has led to the same results, so that the best, the worst and the average of the solutions are the same. The second one is a Kohonen-Like decomposition method [25], in its three different versions abbreviated by KD, KL and KG. The third is a Genetic Algorithm-Based Clustering [26]. Four variants of this method are introduced and tested, the results of which are given as EER, SE, ECER and SP. The fourth collection of the methods compared in a table is categorized under the column entitled Self-Organizing neural networks. Four versions are given in the table namely KNIES-global, KNIES-local, Budinich and ESOM [27]. The results for the normal SA are also taken from the same reference. A set of enhanced methods called Self-Organizing Map designed by Genetic Algorithms is the fifth set. There also four columns quoted for this category from [16][28]. Finally, we have compared our results with the best and the average error percentages of the results given in [29] for its memetic neural networks.

It is easy to deduce that MSSA using both initializing methods has led to very accurate results, with slightly weaker characteristics for WTS as a cost of speeding up the algorithm. The proposed method has shown superiority to all other competing methods in Table 1, though their methods are not tested for the last benchmark, u1432, which perhaps will lead to more inaccurate results, if tested. To accomplish the comparison study, we have added another set of methods from [8], in which 11 methods are run on 30 benchmarks from lin105 to u1432. For the sake of brevity, the problems are categorized into 3 groups, namely: small, medium and large size benchmarks. The results are given

in Table 2, where the mean of the average errors in each group is shown; see [8] for a detailed explanation of each method.

TABLE 1: COMPARISON BETWEEN MSSA AND OTHER NEW METHODS FOR 24 BENCHMARKS

TSP Benchmark	Optimal Solution	MSSA By Crossover Elimination			MSSA By WTS			CONN [24][25]			KOHONEN-LIKE Decomposition [25] (Average δ)		
		Best δ	Average δ	Worst δ	Best δ	Average δ	Worst δ	Best δ	Average δ	Worst δ	KD	KL	KG
eil51	426	0	0.36	0.94	0	0.79	1.88	2.58	2.58	2.58	3.50	2.86	2.86
st70	675	0	1.02	1.63	0	0.90	1.48	2.96	2.96	2.96	3.67	1.51	2.33
eil76	538	0	0.82	1.67	0	0.70	2.04	5.02	5.02	5.02	6.49	4.98	5.48
gr96	514	0	0.83	1.36	0	0.53	1.17	3.61	3.61	3.61	-	-	-
kroA100	21282	0	0.52	0.93	0.28	0.65	2.04	2.57	2.57	2.57	-	-	-
rd100	7910	0.01	1.40	1.96	0.01	1.42	3.12	3.59	3.59	3.59	4.89	2.09	2.62
eil101	629	0	0.90	1.91	0	1.97	3.34	4.61	4.61	4.61	6.84	4.66	5.63
lin105	14379	0	0.52	1.00	0	0.33	0.77	0.38	0.38	0.38	2.18	1.98	1.29
pr107	44303	0	0.14	0.30	0	0.13	0.30	2.77	2.77	2.77	10.83	0.73	0.42
pr124	59030	0	0.25	0.60	0	0.21	0.45	1.74	1.74	1.74	3.22	0.08	0.49
bier127	118282	0.12	0.37	0.68	0.04	0.58	1.12	2.45	2.45	2.45	5.82	2.76	3.08
pr136	96772	0.35	1.05	1.97	0.55	1.43	2.89	2.27	2.27	2.27	1.93	4.53	5.15
gr137	698	0.14	0.63	1.29	0	0.92	2.00	4.69	4.69	4.69	-	-	-
kroA150	26524	1.36	2.18	3.37	0.48	1.47	2.22	4.78	4.78	4.78	-	-	-
kroA200	29368	0.48	1.22	2.50	1.03	1.49	2.30	4.40	4.40	4.40	5.66	5.72	6.57
pr226	80369	0.90	1.48	2.85	1.34	1.96	2.21	1.93	1.93	1.93	-	-	-
pr264	49135	0.85	2.91	5.55	1.66	2.83	4.49	3.58	3.58	3.58	-	-	-
lin318	42029	0.53	1.34	1.90	0.70	1.79	2.56	-	-	-	-	-	-
rd400	15281	1.90	2.48	2.85	2.30	2.54	2.80	5.77	5.77	5.77	-	-	-
pcb442	50778	2.04	2.26	2.68	1.78	2.46	3.09	5.56	5.56	5.56	8.00	11.07	10.45
att532	87550	1.56	1.96	2.37	1.92	2.52	3.37	5.66	5.66	5.66	6.15	6.74	6.80
rat783	8806	1.46	1.90	2.43	1.44	2.27	3.45	7.59	7.59	7.59	9.11	-	9.53
pr1002	259045	2.07	2.27	2.48	2.33	2.57	2.73	6.94	6.94	6.94	7.08	-	7.60
u1432	152970	2.04	2.99	3.96	1.97	3.18	4.28	-	-	-	-	-	-
Average		0.66	1.33	2.05	0.74	1.48	2.34	4.00	4.00	4.00	5.69	3.82	4.69
*MSSA By Crossover Elimination		-	-	-	0.66	1.33	2.05	0.66	1.33	2.06	1.10	0.94	1.10
*MSSA By WTS		0.74	1.48	2.34	-	-	-	0.75	1.47	2.33	1.32	1.15	1.32

TABLE 1: COMPARISON BETWEEN MSSA …. (CONT'D)

Genetic Algorithm-Based Clustering [26][26] (Average δ)				Self-Organizing neural networks [27] (Average δ)					Self-Organizing Map designed by Genetic Algorithms [27][28] (Average δ)				Memetic neural network [29]	
									Enhanced SOMs					
EER	SE	ECER	SP	KNIES-global	KNIES-local	SA	Budinich	ESOM	CEN	Budinich	ESOM	EISOM	Best δ	Average δ
1.16	7.19	0.23	0.23	2.86	2.86	2.33	3.10	2.10	1.88	2.48	0.93	1.97	1.64	2.14
-	-	-	-	2.33	1.51	2.14	1.70	2.09	-	-	-	-	0.59	0.99
4.27	3.41	0.92	2.18	5.48	4.98	5.54	5.32	3.89	-	-	-	-	2.04	2.88
10.62	8.07	3.09	2.46	-	-	4.12	2.09	1.03	4.39	0.46	0.46	0.53	-	-
16.54	6.70	3.81	2.41	-	-	5.94	3.68	1.01	1.60	0.93	0.81	0.54	0.24	1.14
11.10	8.93	3.23	3.81	2.62	2.09	3.26	3.16	1.96	-	-	-	-	0.99	2.65
17.99	8.71	4.55	4.12	5.63	4.66	5.74	5.24	3.43	2.07	4.31	2.72	2.92	2.07	3.15
22.55	2.85	2.31	3.10	1.29	1.98	1.87	1.71	0.25	-	-	-	-	0.00	0.34
20.46	5.79	2.11	2.98	0.42	0.73	1.54	1.32	1.48	-	-	-	-	0.14	0.67
30.51	3.75	2.93	3.02	0.49	0.08	1.26	1.62	0.67	-	-	-	-	0.26	1.52
9.49	4.39	3.56	2.21	3.08	2.76	3.52	3.61	1.70	-	-	-	-	1.25	2.78
26.50	12.54	12.43	6.19	5.15	4.53	4.90	5.20	4.31	-	-	-	-	0.73	3.10
23.85	6.58	2.54	4.22	-	-	8.45	8.61	4.27	3.29	4.51	2.52	2.18	-	-
29.15	7.10	8.01	5.17	-	-	-	-	-	2.90	2.23	1.69	1.26	1.64	2.73
40.97	10.46	7.72	5.91	6.57	5.72	5.61	6.13	2.91	3.22	2.67	1.96	1.21	1.08	2.20
57.09	5.52	4.71	5.05	-	-	-	-	-	-	-	-	-	-	-
54.08	10.26	10.49	9.37	-	-	-	-	-	-	-	-	-	-	-
53.18	12.06	12.49	12.99	-	-	7.56	8.19	4.11	4.67	2.81	2.89	1.96	3.63	5.51
56.83	17.25	14.33	15.95	-	-	-	-	-	-	-	-	-	-	-
60.29	14.11	19.80	12.76	10.45	11.07	9.15	8.43	7.43	5.31	6.88	5.11	5.67	3.57	6.08
67.58	17.72	16.18	18.41	6.80	6.74	5.38	5.67	4.95	5.81	4.76	3.54	2.39	3.29	4.21
72.88	19.89	25.96	23.28	-	-	-	-	-	-	-	-	-	5.46	5.95
-	-	-	-	-	-	6.03	8.75	5.93	6.99	7.44	5.07	4.01	4.75	6.11
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
32.72	9.20	7.69	6.94	4.09	3.82	4.69	4.64	2.97	3.83	3.59	2.52	2.24	1.94	3.04
1.22	1.22	1.22	1.22	0.94	0.94	0.99	0.99	0.99	1.32	1.32	1.32	1.32	0.63	1.24
1.38	1.38	1.38	1.38	1.15	1.15	1.19	1.19	1.19	1.56	1.56	1.56	1.56	0.66	1.41

The last two rows contain average of the cases for which the corresponding method is run and the results are given.

TABLE 2: COMPARISON OF MSSA WITH OTHER METHODS GIVEN IN [8] FOR 29 BENCHMARKS (THE AVERAGE OF THE AVERAGE ERROR IN 25 RUNS)

Algorithms in [7]	Average Error in			Total Average Error
	Small Size	Medium Size	Large Size	
SA (Simulated Annealing)	2.76	3.25	3.7	3.09
TA (Threshold Accepting)	5.37	4.18	9.95	5.75
RRT (Record-to-Record Travel)	4.22	6.79	13.96	6.78
BD (Bounded Demon)	5.26	4.44	7.73	5.4
RBD (Randomized Bounded Demon)	4.33	9.38	13.59	7.66
AD (Annealed Demon)	3.24	3.27	10.4	4.49
RAD (Randomized Annealed Demon)	2.82	4.38	10.94	4.76
ABD (Annealed Bounded Demon)	2.65	2.77	9.15	3.81
RABD (Randomized Annealed Bounded Demon)	2.63	3.64	4.13	3.24
ADH (Annealed Demon Hybrid)	2.97	2.95	9.19	4.03
ABDH (Annealed Bounded Demon Hybrid)	2.69	2.89	8.52	3.76
MSSA (the proposed method)				
By Crossover Elimination	1.10	2.22	2.63	1.63
By WTS	1.24	2.42	2.88	1.81

To have a better feeling from the speed of the algorithms, let us first have a look at the figures given below.

As explained before, if annealing starts from a very high temperature, say 500 or more for the case of eli51, with γ bigger than 90% (or an alternative parameter like the initial acceptance probability for an average uphill move [15]), it may take more than 8000 evaluations of the energy function to reach the minimum, while starting from a mushy state will lead to convergence in less than 2000 iterations. This means 4 times faster, as observed in Figure 4, where it is shown how the annealing schedule would be and from which point it is started in this method. It should be added here that the total calculation time to find a mushy state with about γ =10%, done by any algorithm like the crossover elimination or the WTS, is less than one tenth of the total iteration time needed for SA to slow down its initially high temperature within 6000 = 8000 – 2000 iterations.

As the final point, it is worth mentioning that comparison of the calculation speed for different methods is not accurate unless the methods are run on one computer using the same conditions. Since the speed of an algorithm depends on the properties of the computer specification by which the algorithm is being run, the number of *floating point operation* (*fpo*) is a proper alternative to compare the speeds. However, for the fact of randomness, it is almost impossibleto compute and compare the right number of *fpo* for each algorithm. As it is observed in this paper, we compared the proposed method with the standard SA and approved analytically that MSSA is much faster. For the purpose of comparison of the proposed method with the other methods listed in Table 2, merely it would be good enough to refer to the average (minimum/maximum) error in the final results of each algorithm.

6. CONCLUSION

Simulated annealing is one of the top ten methods of model-free optimization methods, various versions of which were proposed by researchers during the two last decades. Focusing on the initial condition by which the annealing starts, this paper proposed a novel variant of the original SA named mushy state simulated annealing (MSSA). In this method we start annealing not from a high temperature in a liquid state, but from a low temperature in a mushy state. Moreover, we use a memory to save the best solution found previously. This technique has speeded up the optimization process while receiving to quite accurate optimum solutions. For the case study of TSP, two simple algorithms including crossover elimination and the shortly introduced method of WTS are used to initiate MSSA. Simulation results were compared to many recent new optimization methods that were applied to solve TSP. Despite of its higher speed compared to the normal SA, superiority of

the proposed method was observed in all cases with less than 1432 cities. The average error obtained by MSSA for the 24 benchmarks was much less than all other methods.

Appendix

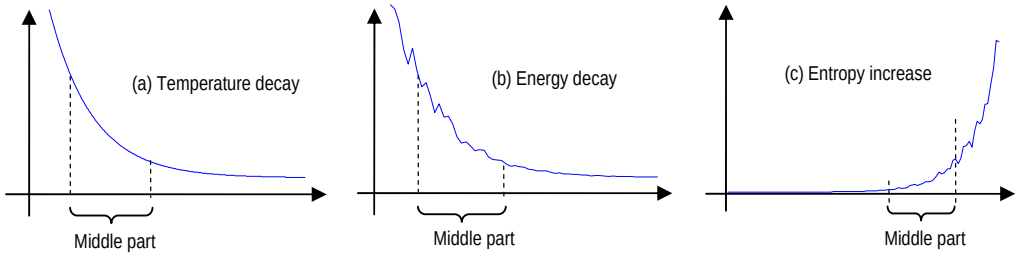

Figure 1: Typical behavior of an annealing schedule; the mushy state falls in the middle;
(a) Temperature decay, (b) Energy decay, (c) Entropy increase

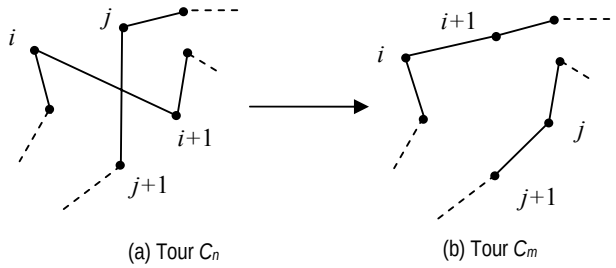

Figure 2: Intersected Links Elimination

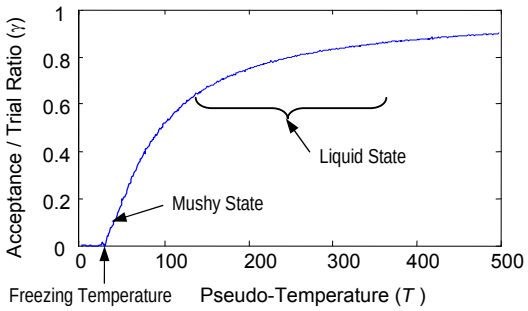

Figure 3: Relation between the Pseudo-Temperature and the Acceptance-Trial Ratio, for the case of eli51.

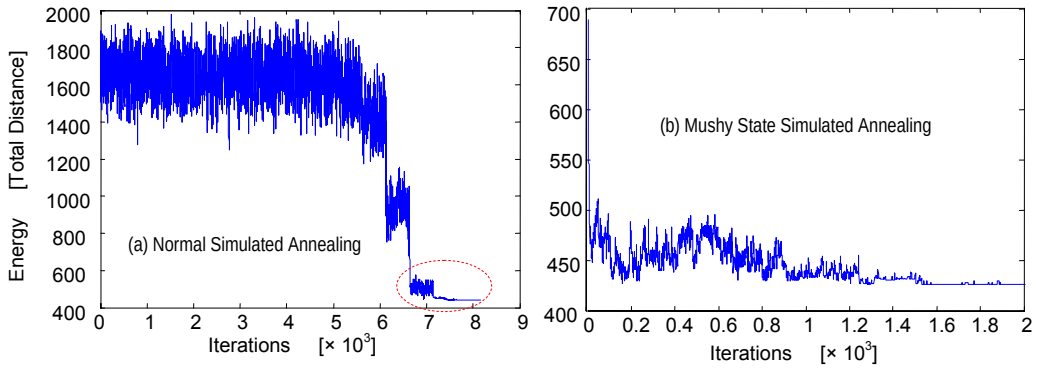

Figure 4: Energy decay in the annealing process for eli51;
(a) Normal Simulated Annealing, (b) Mushy State Simulated Annealing

7.REFERENCES

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing”, Science, vol. 220, pp. 671–680, 1983.

[2] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", New York: Freeman, 1979.

[3] Herbert H. Tsang, and Kay C. Wiese, “The Significance of Thermodynamic Models in the Accuracy Improvement of RNA Secondary Structure Prediction Using Permutation-based Simulated Annealing”, IEEE Congress on Evolutionary Computation, (CEC 2007).

[4] Ming-Hao Hung, Li-Sun Shu, Shinn-Jang Ho, Shiow-Fen Hwang, and Shinn-Ying Ho, “A Novel Intelligent Multiobjective Simulated Annealing Algorithm for Designing Robust PID Controllers”, IEEE Transactions on Systems, Man, and Cybernetics—PART A: Systems and Humans, Vol. 38, No. 2, pp. 319-330, (Mar. 2008).

[5] Kevin I. Smith, Richard M. Everson, Jonathan E. Fieldsend , Chris Murphy, and Rashmi Misra, “Dominance-Based Multiobjective Simulated Annealing”, IEEE Transactions on E Evolutionary Computation, Vol. 12, No. 3, pp. 323-341, (June 2008).

[6] S. A. Kravitz and R. A. Rutenbar, “Placement by simulated annealing on a multiprocessor,” IEEE Trans. Computer-Aided Design Integr. CircuitsSyst., vol. 6, no. 4, pp. 534–549, Jul. 1987.

[7] E. Aarts and J. Korst, "Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing", New York: Wiley, 1989.

[8] Joshua W. Pepper, Bruce L. Golden, and Edward A. Wasil, “Solving the Traveling Salesman Problem With Annealing-Based Heuristics: A Computational Study”, IEEE Transactions on Systems, Man, and Cybernetics—PART A: Systems and Humans, Vol. 32, No. 1, (Jan. 2002).

[9] Hyeon-Joong Cho, Se-Young Oh and Doo-Hyun Choi, “Population-oriented simulated annealing technique based on local Temperature concept”, ELECTRONICS LETTERS Vol. 34 No. 3 pp.312-313 (5th February 1998).

[10] Percy P. C. Yip, and Yoh-Han Pao, “Combinatorial Optimization with Use of Guided Evolutionary Simulated Annealing”, IEEE Transactions on Neural Networks, Vol. 6. No. 2, pp. 290-295 (March 1995).

[11] Andrew Soh, “Parallel N-aray Speculative Computation of Simulated Annealing”, IEEE Transactions on Parallel and Distributed Systems, Vol. 6, No. 10, pp. 997-1005 (Oct. 1995).

[12] D. C. W. Pao, S. P. Lam and A. S. Fong, “Parallel implementation of simulated annealing using transaction processing”, IEE Proc-Computer Digit. Tech., Vol. 146, No. 2, pp. 107-113, (March 1999).

[13] Feng-Tse Lin, Cheng-Yan Kao, and Ching-Chi Hsu, “Applying the Genetic Approach to Simulated Annealing in Solving Some NP-Hard Problems”, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23. No. 6, (Nov./Dec. 1993)

[14] Dale R. Thompson and Griff L. Bilbro, “Sample-Sort Simulated Annealing”, IEEE Transactions on Systems, Man, and Cybernetics—PART B: Cybernetics, Vol. 35, No. 3, pp. 625-632 (Jun. 2005).

[15] Hao Chen, Nicholas S. Flann, and Daniel W. Watson, “Parallel Genetic Simulated Annealing: A Massively Parallel SIMD Algorithm”, IEEE Transactions on Parallel and Distributed Systems, Vol. 9, No. 2, pp.126-136 (February 1998).

[16] K.L. Wong A.G.Constantinides, “Speculative parallel simulated annealing with acceptance prediction”, Electronic Letters, Vol. 34, No. 3, (February 1998), pp. 312-313.

[17] L. Ingber and B. Rosen, “Genetic Algorithms and Very Fast Simulated Reannealing: A Comparison”, Mathematical Computer Modeling, vol. 16, no. 11, (1992), pp. 87-100.

[18] Lipo Wang, Sa Li, Fuyu Tian, and Xiuju Fu, “A Noisy Chaotic Neural Network for Solving Combinatorial Optimization Problems: Stochastic Chaotic Simulated Annealing”, Transactions on Systems, Man, and Cybernetics—PART B: Cybernetics, Vol. 34, No. 5 pp. 2119-2125, (Oct. 2004).

[19] Yuyao He, “Chaotic Simulated Annealing With Decaying Chaotic Noise”, IEEE Transactions on Neural Networks, Vol. 13, No. 6, (November 2002), pp. 1526-1531.

[20] Sitao Wu and Tommy W. S. Chow, “Self-Organizing and Self-Evolving Neurons: A New Neural Network for Optimization”, IEEE Transactions on Neural Networks, VOL. 18, NO. 2, (March 2007), pp. 385-396.

[21] J. Jang, C. Sun, E. Mizutani, “Neuro-Fuzzy and Soft Computing”, Proc. of the Prentice Hall 1997.

[22] G. Dueck and T. Scheuer, “Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing,” J. Computer. Phys., vol. 90, 1990, pp. 161–175.

[23] G. Reinelt. Tsplib95, 1995. Available at: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.

[24] M. Saadatmand-Tarzjan, M. Khademi, M. R. Akbarzadeh-T., and H. Abrishami Moghaddam, “A Novel Constructive-Optimizer Neural Network for the Traveling Salesman Problem” IEEE Transactions on Systems, Man, and Cybernetics—PART B: Cybernetics, Vol. 37, No. 4, (Aug. 2007).

[25] Necati Aras, I. Kuban Altınel, and John Oommen, “A Kohonen-Like Decomposition Method for the Euclidean Traveling Salesman Problem Knies_Decompose”, IEEE Transactions on Neural Networks, Vol. 14, No. 4, (July 2003).

[26] Chun-Hung Cheng, Wing-Kin Lee, and Kam-Fai Wong, “A Genetic Algorithm-Based Clustering Approach for Database Partitioning”, IEEE Transactions on Systems, Man, and Cybernetics—PART C: Applications and Reviews, Vol. 32, No. 3, (Aug. 2002).

[27] Kwong-Sak Leung, Hui-Dong Jin, and Zong-Ben Xu, “An expanding Self-Organizing neural network for the traveling salesman problem**”,** Neurocomputing, Vol. 62, pp. 267-292, (Dec. 2004).

[28] Hui-Dong Jin, Kwong-Sak Leung, Man-Leung Wong and Zong-Ben Xu, “An Efficient Self-Organizing Map Designed by Genetic Algorithms for the Traveling Salesman Problem”, IEEE Transactions on Systems, Man, and Cybernetics—PART B: Cybernetics, Vol. 33, No. 6, (Dec. 2003), pp. 877-888.

[29] J. C. Creput, A. Koukam, “A memetic neural network for the Euclidean traveling salesman problem”, Neurocomputing Accepted 22 (January 2008).

[30] Hamed Shakouri G., Kambiz Shojaee, Mojtaba Behnam T. " The Wise Experiencing Traveling Salesman (WETS): Introduction to a simple evolutionary solution for the problem" IEEE Congress on Evolutionary Computation CEC 2009 18-21 (May 2009) Trondheim, Norway.

[31] Hamed Shakouri G., kambiz shojaee, " Investigation on the choice of the initial temperature in the Simulated Annealing: A Mushy State SA for TSP " 17th Mediterranean Conference on Control Automation, MED'09 Thessaloniki, Greece on June 24-26, 2009.