2003 International Symposium on Advanced Intelligent Systems
September 25-28, 2003, Jeju, Korea

Automatic Parameter Tuning for Simulated Annealing based on Threading Technique and its Application to Traveling Salesman Problem

Fangyan Dong*, Eduardo Masato Iyoda*, Kewei Chen**, Hajime Nobuhara*, and Kaoru Hirota*
*Department of Computational Intelligence & Systems Science, Tokyo Institute of Technology
4259 Nagatsuta, Midori-ku, Yokohama 226-8502, Japan
e-mail:{fydong, iyoda,nobuhara,hirota}@hrt.dis.titech.ac.jp, **chenkewei@nifty.com

Abstract - **In order to solve the difficulties of parameter settings in SA algorithm, an improved practical SA algorithm is proposed by employing the threading techniques, appropriate software structures, and dynamic adjustments of temperature parameters. Threads provide a mechanism to realize a parallel processing under a disperse environment by controlling the flux of internal information of an application. Thread services divide a process by multiple processes leading to parallel processing of information to access common data. Therefore, efficient search is achieved by multiple search processes, different initial conditions, and automatic temperature adjustments. The proposed methods are evaluated, for three types of Traveling Salesman Problem (TSP) (random-tour, fractal-tour, and TSPLIB test data) are used for the performance evaluation. The experimental results show that the computational time is 5% decreased comparing to conventional SA algorithm, furthermore there is no need for manual parameter settings. These results also demonstrate that the proposed method is applicable to real-world vehicle routing problems.**

I. Introduction

It is well known that Simulated Annealing (SA) algorithm is efficient and readily applicable for solving large-scale Combinatorial Optimization Problems (COP) [1]. But in many practical applications, the methods present some limitations, e.g., the choice of the initial state and the initial temperature may affect the accuracy of the final solution and the operation cost required. Moreover, formal procedures to define these parameters have not been established.

In order to solve the difficulties of parameter settings in SA algorithm, an improved practical SA algorithm is proposed by employing the operating system threading techniques, appropriate software structures, and dynamic adjustments of temperature parameters. Threads provide a mechanism to realize a parallel processing under a disperse environment by controlling the flux of internal information of an application. Thread services divide a process by multiple processes leading to parallel processing of information to access common data. Therefore, the efficient search is achieved by multiple search processes, different initial conditions, and automatic temperature adjustments. The proposed methods are evaluated, for three types of Traveling Salesman Problems (TSP's) (random-tour, fractal-tour, and TSPLIB test data) are used for the performance evaluation.

In II, threading technique and its distributing calculation is illustrated, and how to apply the advanced technique to meta-heuristic for COP problems is presented. In III, the property of SA algorithm is analyzed and an improved Multi-threading SA with Multi-Start and Intellectual Parameters Setting is proposed. In section IV, a variety of data based on TSP is used to evaluate the proposed method.

II. Threading & Distributing Calculation

Threading technique is used to construct an advanced and complexity software model in the filed of operating system. How to make it into meta-heuristic algorithm to realize the distributing calculation for practice solution is mentioned in the followings.

A. Threading Models and Its Advantages

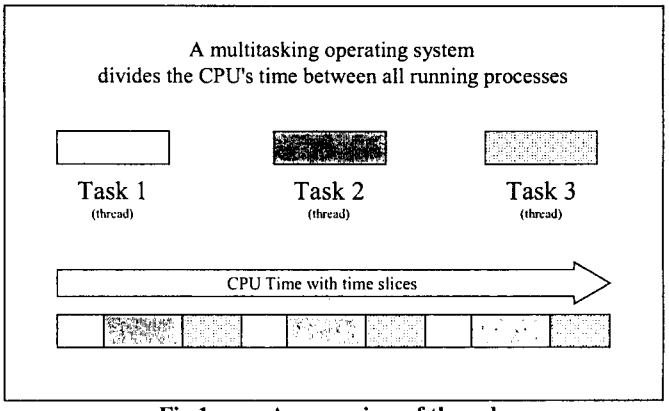

Fig.1 An overview of threads

In Fig.1, an overview of threads is depicted. A thread is a process or a part of an application. Threads provide a way to

realize parallel processing under a disperse environment, by controlling the flux of internal information of an application. Thread services divide a large process in multiple processes, leading to parallel processing of information. To access common data, data access synchronization is possible.

A thread resembles a CPU task, for each Application Interface (API), the CPU executes a thread for a periodical time. As illustrated in Figure 2, according to the priority level (real time, high priority, normal priority, idle priority), an appropriate amount of time is allotted to CPU.

Threads can assign time process to the CPU, from minimal units or substantial ones. Usually, a program is a single thread, for various API this approach is sufficient.

Multithreading is a mechanism that divides a task in 2 or more threads. The OS distributes CPU time in threads, so multithreading requires more CPU time.

By making use of multithreading OS, the basis for fast, good response algorithms can be implemented.

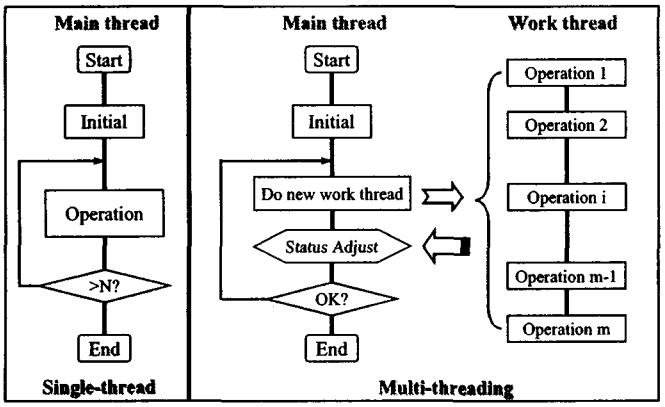

Fig.2 Single-thread and Multi-threading

B. Distributing Calculation with multi-threading

To improve practical meta-heuristics, splitting in work thread (mainly heuristics search) and main thread (state regulation and termination condition evaluation) is used. Traditional *N*-step probabilistic search is divided into *N/M* steps, in a way that *M* threads realize the search, resulting in a more efficient and easy controllable method (Fig.2). The main advantages are:

- Increased speed and CPU usage ratio.
- Since the application is composed of main thread and small-scale work threads, parameter adjustment is simple, and efficiency is increased.
- By instantaneously using the feedback information from the work thread, a global improvement is assured.
- Due to its parallel computing property, it can be implemented in multi-platform environment.
- Introducing threads and priority level control, for real API applications, a simple componentization and efficient functions are possible

III. Improved SA with Multi-threading

In order to solve the difficulties of parameter settings in SA algorithm, an improved practical SA algorithm is proposed by employing the Operating System threading techniques, appropriate software structures, and dynamic adjustments of temperature parameters.

A. SA: Simulated Annealing Algorithm

Annealing algorithm is a stochastic optimization strategy for solving the combinational optimization problem. It simulates the experimental annealing procedure of metal or glass to get a good enough solution.

There are two general phases in the annealing process: One is the heating phase and another is the slowly cooling down phase.

Feasible Solution: $S=\{S_1,\ldots,S_n\}$

Object Function: $C: S \rightarrow R^+$

$$^{\exists}S^* \in S, \Rightarrow C(S^*)=\min C(S_i),\ ^{\forall}S_i \in S$$

Procedure Simulated Annealing (S_0, T_0)

$\{S_0$: initial state, T_0 : parameter of temputer, $C_0 = C(S_0)\}$

(1) $S := S_0$; $k = 0$ {S :current state}

(2) REPEAT

(3) REPEAT

(4) $S_j := generate(S)$;

(5) IF $C_j \le C^*$ THEN $S := S_j$;

(6) ELSE IF $accept(j, S)$ THEN $S := S_j$;

(7) UNTIL 'inner-loop stop criterion'

(8) $T_{k+1} = update(T_k)^{\#}$; $k := k+1$

(9) UNTIL 'final stop criterion'

Procedure $accept(j, S)$

(1) IF $\exp\{-(C_j - C^*)/kT_k\} > random(0,1)$

(2) THEN $accept := true$

(3) ELSE $accept := false$

$$\#: T \xleftarrow[\lambda \in [0.2,0.9]]{} \lambda T, \quad 0 \prec \lambda \prec 1$$

Fig.3 Simulated Annealing Algorithm

Fig.3 shows the classical simulated annealing algorithm. The main steps of the algorithm are summarized as:

(1). Suppose S is a randomly initiated solution of the combinational optimization problems. Obtain the highest temperature: T0.

(2). Let T=T0

(3). Randomly turn S to its neighboring state: Ś. Objective value deviation: ΔH=H(S)-H(Ś)

(4). Determine whether Ś is accepted (i.e., S ← Ś) or rejected according to *Metropolis probability:*

(5). Decrease the temperature gradually (e.g., T=T*0.8)

(6). Go to step (3) until the temperature T is low enough or lower than a threshold.

(7). Stop

The SA is the most efficient method for COP, and its convergence is guaranteed.

In the implementation of the algorithm, the choice of initial state solution (S_0) and initial temperature (T_0) influence the precision of the final solution. The fixed decreasing coefficient is also not good for search proposes.

In real applications, the problem size can change daily, because of which it is difficult to guarantee good solutions within a fixed operation cost. One of the objectives of our research is the question of how to avoid the selection of S_0 and T_0.

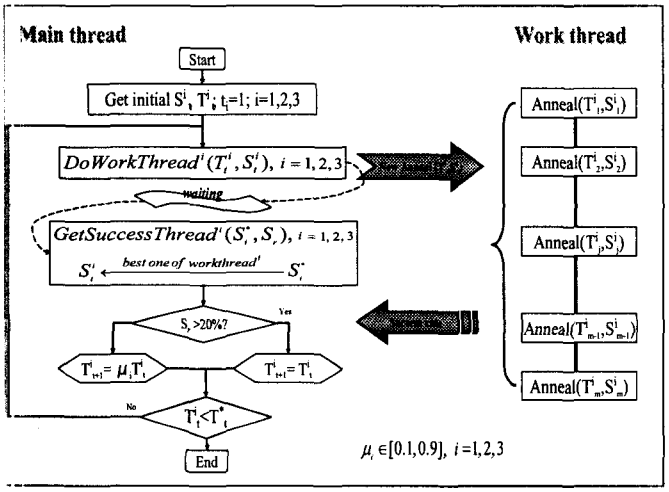

Fig.4. Distributing SA with Multi-start & Parameter hiding

B. Multi-Start and Intellectual Parameters Setting

To respond to the demand for easy parameter adjustment and fixed computational cost and, at the same time, to preserve the precision of the final solution, some essential conditions are necessary. Success of SA search depends on many factors like clear covering of parameter resetting and flexibility. Figure 4 shows the ***I**mproved **D**istributing **SA*** (realized by multi-threading) with multi-start and parameter hiding (intellectual adjustment).

The main steps of the IDSA algorithm are:

(1). Set $\{S_t^i, T_t^i\}$ $(i = 1,2,3; t = 0)$ by randomly initiated solution and relative highest temperature.

(2). Start SA work threads (i=1,2,3).

(3). Get the feedback of success rate (S_r) from work thread i.

(4). $S_{t+1}^i \xleftarrow{thread^i} S_t^*$ as next new solution.

(5). $If\,(S_r \le 0.2) T_{t+1}^i = \mu_i T_t^i, else T_{t+1}^i = T_t^i$ as new temperature.

(6). Go to step (2) until temperature: T^i is low enough or lower than a threshold.

(7). Stop

The properties of the improved SA are:

- With few SA search processes, as well as control and distribution of multiple executions, an efficient search is achieved.
- During the change of temperatures from high to low, every state is uniformly preserved (SA internal process and external main adjustment included), so that temperature parameter is also automatically adjusted (in the internal process, initialization is not necessary).
- Each annealing thread starts with a different initial condition S_i, resulting in better solutions.
- Each work thread has fixed calculation number to avoid useless and duplication search (When S_r is high, it can continue at the same state).

IV. Experiment for Performance of IDSA

Two kinds of experiments have been done with the concepts of Steadiness and Robustness.

A. Steadiness Defined by Meta-Algorithm

Under the conditions with fixed computational environment (CPU type, memory, etc.) and limited operation cost (timer, resource, etc.), the solution precision can be fixed invariably above a reserved index (0.9 recommended). Then, the meta-algorithm should be considered as the steadiness.

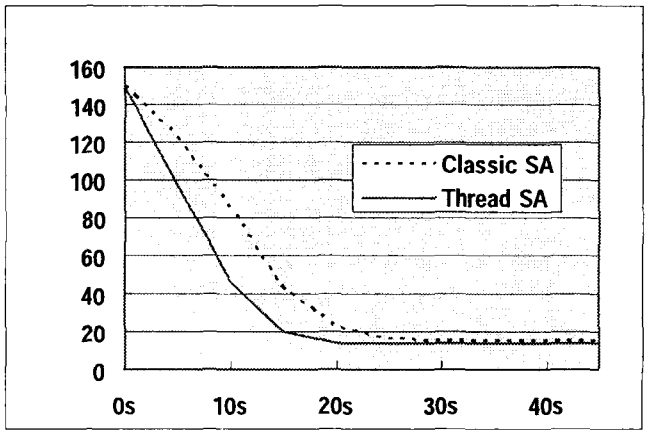

Fig.5-1 City Number 300

Fig.5-1,2 show the results for Standard-SA and Thread-SA with 300 and 500 city number. The steadiness and the fast convergence are confirmed.

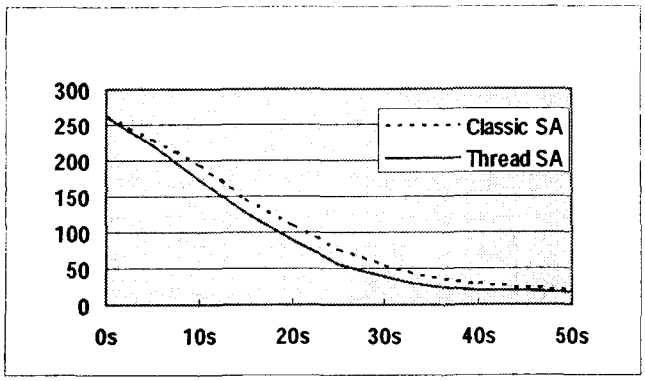

Fig.5-2 City Number 500

B. Robustness Defined by Meta-Algorithm

If the average test results are more than required precision under the range variation (from 10% to 120%) of problem scale N with specify computational environment and by using all sorts of test data, then the meta-algorithm is able to achieve the robustness.

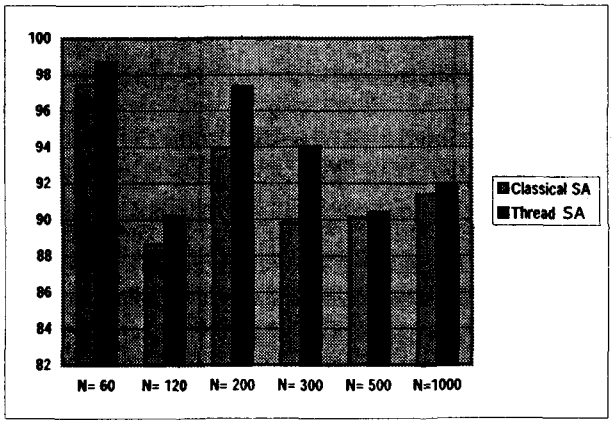

Fig.6 Robustness Evaluation

Fig.6 shows search precision for Classical-SA and Thread-SA with the cities number varied from 60 to 1000. The robustness and a better result of Thread-SA are confirmed.

V. Conclusions

Multi-threading technique is introduced and distributing calculation is also mentioned for meta-heuristic algorithm. Since the primary factors (high speed, high precision, no-parameter, knowledge inference, and controllability) are basic requirements for solving the real COP problem. How to use various advanced software models into the meta-heuristics algorithm and how to improve their performances are important and interesting subjects.

The Algorithm called *I*mproved *D*istributing *SA* is proposed, that is realized by multi-threading and multi-start and parameter hiding (intellectual adjustment). The properties are summarized as follows:

(1). Using thread technique, the search operations are distributive, control-able and efficient.

(2). Using multi-start with a different initial condition, each annealing thread can work well and result in better solutions.

(3). For the changes of temperatures from high to low, the information got from finished work thread can be checked and temperature parameter is intellectual adjusted to let thermostatic declining propose fast or low.

(4). Each work thread has fixed calculation number to avoid useless and duplication search.

Based on the principles of steadiness and robustness, experiments have been done to confirm the performance of proposed algorithm. The results show that it is better than classical one. The Thread-SA is also simplicity to construct and distributed execution to increase calculation efficiency and catch more intermediate information for intellectual adjustment.

Meta-heuristics can be interpreted in a number of ways, but basically they are tools (solving method, software structure, techniques for practical application, performance evaluation based) for finding good quality solutions by using limited operation resources.

References

[1] Mutsunori Yagiura, Toshihide Ibaraki: On Meta-heuristic Algorithms for Combinatorial Optimization Problem, Asakura Publ. Co., pp.97-102, 2001.

[2] Kenneth D. Boese: Models for Iterative Global Optimization, UCLA Computer Science Department, Ph.D. Thesis, 1996.

[3] J.L. Bentley, Experiments on traveling salesman heuristics, In First Annual ACM-SIAM Symposium on Discrete Algorithms, pp.91-99, San Francisco, CA, January 1990.

[4]S.Kirkpatrick, C.D.Gelatt, Jr, and M.P. Vecchi: Optimization by simulated annealing, Science, vol.220, pp.671-680, 1983.

[5] V.Cerny: Thermodynamical approach to the traveling salesman problem: an efficient simulated annealing algorithm, J.Optimization Theory and Applications, vol.45, pp.41-51, 1985.

[6] Sadiq M. Salt, and habib Youssef : Iterative Computer Algorithms with Applizations in Enginering Solving Combinatorial Optimization Problem, Maruzhen Publ. Co., pp.43-81, 2001.

[7] http://www.math.princeton.edu/tsp/