IMPROVED SIMULATED ANNEALING, BOLTZMANN MACHINE, AND ATTRIBUTED GRAPH MATCHING †

Lei Xu †† and Erkki Oja

Lappeenranta University of Technology, Department of Information Technology
BOX 20, 53851 Lappeenranta, Finland

Abstract. By separating the search control and the solution updating of the commonly used simulated annealing technique, we propose a revised version of the simulated annealing method which produces better solutions and can reduce the computation time. We also use it to improve the performance of the Boltzmann machine. Furthermore, we present a simple combinatorial optimization model for solving the attributed graph matching problem of e.g. computer vision and give two algorithms to solve the model, one using our improved simulated annealing method directly, the other using it via the Boltzmann machine. Computer simulations have been conducted on the model using both the revised and the original simulated annealing and the Boltzmann machine. The advantages of our revised methods are shown by the results.

1. Introduction. Simulated Annealing (SA) has been widely used to solve various combinatorial optimization problems such as TSP, VLSI design [1,2] as well as clustering and attributed graph matching [3]. It has also been used in the Boltzmann Machine (BM) [4, 5-7].

By separating the Metropolis Sampling (MS) process which is a major part of the SA process, into a search control process and a solution updating process, one of the present authors proposed an Improved Simulated Annealing (ISA) method [8,9] which will be reviewed and further analyzed in this paper. This method can be guaranteed to always yield a better solution than SA. It is useful especially in the following cases, which are often encountered in actual applications:

(1) The time spent on each MS process is not long enough to let the process reach the equilibrium state.

(2) The speed of annealing is too fast.

(3) The temperature specified for stopping the annealing process is not low enough.

In these cases, SA usually finds a bad solution, but ISA can still obtain a better solution. In addition, ISA also has a simple but effective way to decide when a MS process can be finished to start another MS process, and when the whole annealing process can be stopped in such a way that the time cost is reduced but the solution is still satisfactory.

Furthermore, in this paper we will use the basic idea of ISA to improve the performance of BM. Advantages similar to the ones given above are again obtained.

Our work is motivated by the computer vision problem. Attributed graphs have turned out to be very useful data structures when used for image representation and understanding in computer vision systems [10-12]. They have also been successfully

† This work was supported by Tekes Grant 4196/1988 under Finsoft project

†† Permanent address: Dept. of Mathematics, Peking University, P.R.China

used to handle optimal task assignments in distributed computer systems [13]. In [3], one of the authors proposed a way to use SA to implement attributed graph matching. This paper will give another solution: a general attributed graph matching problem is turned into a model of combinatorial optimization which is different from that in [3]. With this model, we can either directly use SA or use symmetrically interconnected neural networks via the BM to obtain the solution of attributed graph matching.

In sec. 2, the commonly used SA algorithm is analyzed. In sec. 3, the ISA is given, its advantages are discussed, and it is applied on the BM. In sec. 4, a general combinatorial optimization model for attributed graph matching is presented and solved by ISA and the revised BM. Finally, in sec. 5, the advantages of ISA are shown through computer simulations with performance comparisons on both the optimality of solution and the time costs.

2. Analysis of Simulated Annealing. In problem-solving with SA, each combinatorial state s_i is regarded as a configuration state of a physical system, the objective function $E(s_i)$ as the system energy, and a parameter T is used to imitate temperature. For a given T, MS is used to simulate the thermodynamical equilibrium at which the Boltzmann distribution

$$f(s_i) = \frac{e^{-E(s_i)}}{\sum_i e^{-E(si)}} \tag{1}$$

can properly describe the probability of the energy at each state. As T gradually decreases, $f(s_i)$ becomes sharp around the state of global minimum energy and will be fixed at the state as $T \to 0$, i.e., the global optimization solution can be obtained.

Generally, the commonly used SA could be described as follows:

Initialization: Generate a random state s as the present solution, and initialize $E(s)$ and $T = T^{(0)}$;

step 1 : Randomly make a small perturbation Δs to get a new state $s + \Delta s$ with the energy increment $\Delta E = E(s + \Delta s) - E(s)$;

step 2 : If $\Delta E < 0$ goto step 3, otherwise, generate a random number ξ by sampling a uniform distribution over [0,1] and if $e^{-\Delta E/T} \leq \xi$, goto step 1;

step 3 : $s + \Delta s$ replaces s as the new present solution, and $E := E + \Delta E$;

step 4 : Check whether the MS at the present T reached its equilibrium; if not, goto step 1;

step 5 : Reduce T into $T' < T$ by some means (e.g.,$T := \lambda T$). Check whether the annealing process has terminated (e.g., $T < T_{min}$); if yes, the present s with its $E(s)$ is taken as the final solution, stop; otherwise, goto step 1.

There steps 1,2,3,4 implement the MS process and step 5 imitates the annealing procedure. The effectiveness of SA depends on: **(1)** Whether each MS process reaches its equilibrium, i.e., how long each MS process should take at each T. **(2)** Whether $T^{(0)}$ is high enough , T_{min} is low enough. and T decreases slowly enough. A low $T^{(0)}$, high T_{min}, sharply decreasing T and short time implementation for each MS process will lead to low computer cost, but the solution is not so good. In contrast, very high $T^{(0)}$, very low T_{min}, and slowly decreasing T will always result in an optimal or near-optimal solution, but usually the cost is substantially high. Although there are several investigations (including some theoretical analysis) on how to select the above parameters, they are either preliminary or theoretical. The common way is still simply to give $T^{(0)}, T_{min}$, a fixed number of steps for each MS process in a heuristic way, and let T decrease exponentially by $T := \lambda T (0 < \lambda < 1)$ [1][7].

The above procedure contains a drawback related to the sequence of solutions. Let $s_{i,j}$ denote the present solution of the j-th iteration at T_i, one iteration being one cycle from step 1 to step 5. Then all the present solutions produced during the implementation of SA will form an updating sequence $\{(s_{i,j}, j = 0, 1, \ldots), i = 0, 1, \ldots\}$. The sequence records the updating history of the present solution, as well as of the track that shows how the search is controlled. The relation between $s_{i,j}$ and $s_{i,j+1}$ has three possibilities:

(a). $E(s_{i,j+1}) < E(s_{i,j})$, if $\Delta E < 0$ in step 2

(b). $E(s_{i,j+1}) > E(s_{i,j})$, if $\Delta E > 0$ and $e^{-\Delta E/T_i} > \xi$ in step 2

(c). $E(s_{i,j+1}) = E(s_{i,j})$, if $\Delta E > 0$ and $e^{-\Delta E/T_i} \leq \xi$ in step 2.

Obviously, $\{(E(s_{i,j}), j = 0, 1, \ldots), i = 0, 1, \ldots\}$ is not a monotonically unincreasing sequence.

Possibility (b) allows the present solution to escape from local minima. This is the key point of the MS process in SA. But it also allows the possibility that the final solution $E(s_{q,p})$ (suppose SA stopped after p steps at T_q) may be worse than some earlier solutions $E(s_{i,j}), j < p, i < q$. This may happen especially if the replacement of T_i by T_{i+1} occurs before the MS process at T_i reaches its equilibrium, or $T^{(0)}$ is not high enough and T_{min} not low enough, or T decreases too fast. As stated in sec.1, these cases are often encountered in practice since there is still no appropriate way to choose these parameters.

The above analysis explains why the solution by simulated annealing is sometimes even worse than that by some conventional heuristic methods, and it also explains the curve phenomenon in Fig.7d in ref. [2].

3. ISA and a Related Improvement on BM. Although necessary for the search control to escape from local minima, possibility (b) above impacts a bad influence on the updating sequence of the present solution. The contradiction can be solved by separating the search track and the updating sequence of the present solution. We only retain $\{(s_{i,j}, j = 0, 1, \ldots), i = 0, 1, \ldots\}$ as the search track, and construct a new sequence $\hat{s}_{i,j}$ as the updating sequence of the present solution:

$$\hat{s}_{0,0} = s_{0,0}; if\ E(s_{i,j+1}) < E(s_{i,j}) \quad then \quad \hat{s}_{i,j+1} = s_{i,j}; \quad otherwise \quad \hat{s}_{i,j+1} = \hat{s}_{i,j}, \tag{2.a}$$

and if $s_{i,k}$ is the last state at T_i and $s_{i+1,0}$ is the first state at T_i, then

$$if\ E(s_{i+1,0}) < E(s_{i,k}) \quad then \quad \hat{s}_{i+1,0} = s_{i,k}; \quad otherwise \quad \hat{s}_{i+1,0} = \hat{s}_{i,k}. \tag{2.b}$$

This modification results in ISA which has the following three advantages:

(1). All the good features of SA can be retained (since the search track is unchanged), but a better final solution can be obtained since $\{(E(\hat{s}_{i,j}), j = 0, 1, \ldots), i = 0, 1, \ldots\}$ is now monotonically unincreasing.

(2). At a given T_i, if in q successive states $\hat{s}_{i,j} = \hat{s}_{i,j+1} = \ldots = \hat{s}_{i,j+q}$ holds, and q is large enough, it could be considered that the equilibrium has been approximately reached. So, a simple and effective way to check whether MS process could be stopped at temperature T_i is to check whether $q > q_0$ (a given threshold).

(3). Let $\hat{s}_{i,k_i}$ denote the last present solution at T_i. If in p successive temperatures $\hat{s}_{i,k_i} = \hat{s}_{i+1,k_{i+1}} = \ldots = \hat{s}_{i+p,k_{i+p}}$ holds, and p is large enough, it could be considered that any further reduction of T is useless. So, a way to check whether the whole annealing process could be stopped is to check whether $p > p_0$ (a given threshold).

The first advantage makes the solution obtained by ISA better than that obtained by SA at nearly the same computing cost. The last two advantages can considerably reduce the computing cost but still keep a good solution.

The algorithm for implementing ISA is given as follows:

```
Initialization: Generate a random state $s$ as the present solution, and initialize $E(s)$ and $T = T^{(0)}$; set $T_{min}$, $j_{max}$, $q_0$, $p_0$; Let $p = 0$, $q = 0$, $j = 0$, $\hat{s} = s$, $E = E(s)$, $E_{\hat{s}} = E$, $E_t = E$;
  step 1 : If $j > j_{max}$, goto step 6; otherwise, $j := j + 1$, randomly make a small perturbation $\Delta s$ to get a new state $s + \Delta s$ with the energy increment $\Delta E = E(s + \Delta s) - E(s)$;
  step 2 : If $\Delta E > 0$, generate a random number $\xi$ by sampling a uniform distribution over [0,1]. If $e^{-\Delta E/T} \leq \xi$, goto step 1;
  step 3 : $s + \Delta s$ replaces $s$ as the new present solution, and $E := E + \Delta E$;.
  step 4 : If $E < E_{\hat{s}}$ then $E_{\hat{s}} := E$ and $\hat{s} := s + \Delta s$ , $q := 0$; Otherwise $q := q + 1$;
  step 5 : If $q < q_0$, goto step 1;
  step 6 : If $E_t < E_{\hat{s}}$ then $E_t := E_{\hat{s}}$ and $p := 0$; Otherwise $p := p + 1$;
  step 7 : If $p < p_0$ and $T > T_{min}$ , reduce $T$ into $T' < T$ by some means(e.g.,$T := \lambda T$), $q := 0$ and $k := 0$ and goto step 1; Otherwise, the present $\hat{s}$ with its $E(\hat{s})$ is taken as the final solution, stop.
```

There T_{min}, j_{max}, respectively, are given thresholds for the minimum temperature and for the maximum time of the MS process at each T_i. The two thresholds together with q_0 and p_0 control when the MS process at each T_i finishes and when the whole annealing stops.

The same method can be applied to the BM in a straightforward way. The BM is a symmetrical interconnected neural network with energy expression:

$$E = -0.5 \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} c_i c_j + \sum_{i=1}^{n} \theta_i c_i \tag{3a}$$

where θ_i is a threshold associated to neuron c_i. The basic solution step is as follows: At a given T, select a neuron c_i, calculate its energy gap ΔE_i (i.e., the difference between the energy of the two global states, one with neuron c_i off ($c_i = 0$) and the other with neuron c_i on ($c_i = 1$)) by

$$\Delta E_i = \sum_{i=1, j \neq i}^{n} w_{ij} c_j - \theta_i \tag{3b}$$

and let the neuron c_i take the value $c_i = 1$ with probabilty

$$p_i = 1/(1 + exp(-\Delta E_i/T)). \tag{4}$$

Then, select another neuron and repeat the same process until the equilibrium is reached. The whole process starts at a high temperature T , and gradually decreases T once the equilibrium is reached, until a low enough T value is reached.

The BM can be revised in a similar way as SA in sec. 3. What we need to do is to also set up a new updating sequence of the present solutions. This can be done as follows: In addition to a binary array A which indicates the current global state of the network, another binary array B is used to record the global state of the current minimal energy. Initially, both arrays $A = B$ record a randomly chosen global state. Then A will be updated once each neuron c_i changes its value according to probability p_i given by Eq.(4), but B is updated by $B = A$ only when the energy of the current global state is lower than the energy of the state recorded in B. In this way, a monotonically unincreasing energy sequence of the current solution is obtained. When the whole process is stopped, the current state recorded in B could be taken as the final solution.

4. Attributed Graph Matching by ISA and the Improved BM. Attributed Graphs have shown superior adequacy when used for image representation and understanding in computer vision [10-12]. They have also been successfully used e.g. to handle optimal task assignment in distributed computer systems [13].

To fix the notation, assume an attributed graph $G = [(V, V_a), (E, E_a)]$. There $V = (v_1, v_2, ..., v_n)$ is the node (vertex) set and $V_a = (av_1, av_2, ..., av_n)$ its node attribute set, and $E = (e_1, e_2, ..., e_p)$ is the edge set and $E_a = (ae_1, ae_2, ..., ae_p)$ its edge attribute set. Both for nodes and edges, each attribute may be either a symbolical or numerical variable. The problem of matching two attributed graphs, $G = [(V, V_a), (E, E_a)]$ and $G' = [(V', V'_a), (E', E'_a)]$ with node sets $V = (v_1, v_2, ..., v_n)$, $V' = (v'_1, v'_2, ..., v'_m)$, means setting up a one-to-one correspondence between the nodes of a subset of G and G'. Generally speaking, there will be $m!/(m-n)!$ (assume $m \geq n$) combinations for constructing one-to-one correspondences between nodes of a subset of G and G'. Specifically, when $n = m$ the problem is called attributed graph matching; when $n < m$, the problem is called attributed subgraph matching.

Usually, there is some cost associated with constructing a one-to-one correspondence between a node $v \in V$ and a node $v' \in V'$ and between an edge $e \in E$ and an edge $e' \in E'$; denote the costs by $d(v, v')$ and $d(e, e')$ respectively. Specifically, $d(v, v')$, $d(e, e')$ can be calculated from the differences between the attributes of v and v' (e and e'). e.g, for numerical attributes, $d(v, v')$ may be the square distance between the attributes of v and v'. Thus, for each combination, we can sum up all the $d(v, v')$ and $d(e, e')$ to get a total cost $D(G, G')$. The goal of attributed graph matching is to choose among all the possible combinations one that has the minimum value of $D(G, G')$ for the optimal match between G and G', or a near minimal value of $D(G, G')$ for a good match between G and G'.

To form the objective function in practice, define a binary $n \times m$ matrix $[U_{ij}]$, in which each row i corresponds to a node v_i of G and each column j corresponds to a node v'_j of G'. If a one-to-one correspondence is assigned to a pair of nodes v_i, v'_j, then element $U_{ij} = 1$, otherwise $U_{ij} = 0$. For a feasible matching possibility, the sum of all elements U_{ij} equals n (here, suppose $n \leq m$), and there should be one and only one element $U_{ij} = 1$ in each row and at most one element $U_{ij} = 1$ in each column. Thus, there should be one and only one element $U_{ij} = 1$ both in each row and each column when $n = m$. The value of the whole matrix is considered as a combinatorial state.

A combinatorial optimization model is proposed as follows: among all the combinatorial states choose one which makes the following objective function take the minimal value:

(i) For $n = m$

$$E = a\sum_{i=1}^{n}(\sum_{j=1}^{n} U_{ij} - 1)^2 + a\sum_{j=1}^{n}(\sum_{i=1}^{n} U_{ij} - 1)^2$$

$$+b\sum_{i=1}^{n}\sum_{j=1}^{n}(av_i - av'_j)^2 U_{ij} + c\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1,l\neq i}^{n}\sum_{k=1,k\neq j}^{n}(ae_{ij} - ae'_{lk})^2 U_{ij}U_{lk} \tag{5a}$$

(ii) For $n < m$

$$E = a\sum_{i=1}^{n}(\sum_{j=1}^{m} U_{ij} - 1)^2 + a(\sum_{j=1}^{n}\sum_{i=1}^{m} U_{ij} - n)^2$$

$$+b\sum_{i=1}^{n}\sum_{j=1}^{m}(av_i - av'_j)^2 U_{ij} + c\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1,l\neq i}^{n}\sum_{k=1,k\neq j}^{m}(ae_{ij} - ae'_{lk})^2 U_{ij}U_{lk} \tag{5b}$$

where ae_{il}, ae'_{jk} denote the attribute vectors of the edges $e(v_i, v_l)$ and $e(v'_j, v'_k)$, respectively. The first two terms attempt to insure a one-to-one correspondence between V of G and V' of G'. The 3rd and 4th terms attempt to minimize the total costs of all $d(v, v')$ and $d(e, e')$, respectively. The coefficients a, b, and c are weight parameters. It is now possible to use the SA or ISA to minimize these objective functions.

As one of the present authors pointed out in [3], the two key points in using SA to solve the graph matching problem are how to perturb the present state (i.e, from present state s to a new state $s + \Delta s$) and how to locally calculate the corresponding energy increment ΔE. For the model of Eq.(5), we treat the two tasks in a simple way of randomly inversing an element U_{ij} into $U_{ij} := 1 - U_{ij}$ and then locally calculating its resulted ΔE by

(i) For $n = m$

$$\Delta E(U_{ij}, 0 \to 1) = 2a\sum_{k=1,k\neq j}^{n} U_{ik} + 2a\sum_{k=1,k\neq i}^{n} U_{kj} - 2a$$

$$+b(av_i - av'_j)^2 + 2c\sum_{r=1,r\neq i}^{n}\sum_{q=1,q\neq j}^{m}(ae_{ri} - ae'_{qj})U_{rq} \tag{6a}$$

and $\Delta E(U_{ij}, 1 \to 0) = -\Delta E(U_{ij}, 0 \to 1)$

(ii) For $n < m$

$$\Delta E(U_{ij}, 0 \to 1) = 2a\sum_{k=1,k\neq j}^{n} U_{ik} + 2a\sum_{r=1,r\neq i}^{n}\sum_{q=1,q\neq j}^{m} U_{rq} - 2aN$$

$$+b(av_i - av'_j)^2 + 2c \sum_{r=1, r\neq i}^{n} \sum_{q=1, q\neq j}^{m} (ae_{ri} - ae'_{qj})U_{rq} \tag{6b}$$

and $\Delta E(U_{ij}, 1 \to 0) = -\Delta E(U_{ij}, 0 \to 1)$

By using the procedures of SA and ISA proposed in sec. 3, the following algorithms are obtained directly for solving the model of Eq.(5):

Algorithm AGM-SA

Initialization: Set an $N \times M$ Matrix $[U_{ij}]$ by randomly deciding each of its elements to be 1 or 0; initialize E by Eq.(5), and $T = T^{(0)}$; set T_{min}, k_{max}; let $k = 0$;

```
step 1 : If $k > k_{max}$ goto step 4; $k := k + 1$; Choose with equal probabilities an
         integer $i$ among [1,2,...,N] and an integer $j$ among [1,2,...,M]; Compute $\Delta E$
         by Eq.(6);
step 2 : Generate a random number $\xi$ by sampling a uniform distribution over [0,1];
         If $exp[-\Delta E/T]) \leq \xi$ goto step 1;
step 3 : $U_{ij} := 1 - U_{ij}$ and $E := E + \Delta E$;
step 4 : If $T > T_{min}$ then $T := \lambda T$ $(0 < \lambda < 1)$ and $k := 0$ and goto step 1; otherwise,
         the present $[U_{ij}]$ with value $E$ is taken as the final solution and stop.
```

Algorithm AGM-ISA

Initialization: Set an $N \times M$ Matrix $[U_{ij}]$ by randomly deciding each of its elements to be 1 or 0, and let another $N \times M$ Matrix $[U'_{ij}]$ take the same values as $[U_{ij}]$; Initialize $T^{(0)}$,T_{min}, k_{max}, q_{max}, p_{max}; Let $p = 0$, $q = 0$, $k = 0$, $\hat{s} = s$, $E = E(s)$, $E_{\hat{s}} = E$, $E_t = E$;

```
step 1 : $k := k + 1$; Choose with equal probabilities an integer $i$ among [1,2,...,N] and
         an integer $j$ among [1,2,...,M]; Compute $\Delta E$ by Eq.(6);
step 2 : If $\Delta E < 0$ goto step 4, otherwise, generate a random number $\xi$ by sampling
         a uniform distribution over [0,1]; If $exp[-\Delta E/T]) > \xi$ goto step 4;
step 3 : If $k > k_{max}$, goto step 7, otherwise goto step 1;
step 4 : $U_{ij} := 1 - U_{ij}$ and $E := E + \Delta E$;
step 5 : If $E < E_{\hat{s}}$ then $E_{\hat{s}} := E$ and $[U'_{ij}] := [U_{ij}]$ and $q := 0$; Otherwise $q := q + 1$;
step 6 : If $q < q_{max}$, goto step 1;
step 7 : If $E_t < E_{\hat{s}}$ then $E_t := E_{\hat{s}}$ and $p := 0$; Otherwise $p := p + 1$;
step 8 : If $p < p_{max}$ and $T > T_{min}$ then $T := \lambda T$ $(0 < \lambda < 1)$and $q := 0$ and $k := 0$
         and goto step 1; Otherwise, the present $[U'_{ij}]$ is taken as the final solution,
         and use Eq.(5) to calculate its $E$ and stop.
```

When we wish to solve the same problem with BM or Improved BM, Eq.(5) can be further rewritten into the following form:

(i) For $n = m$

$$E = -0.5 \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{n} \sum_{k=1}^{n} w_{ij,lk} U_{ij} U_{lk} - \sum_{i=1}^{n} \sum_{j=1}^{n} U_{ij} \theta_{ij},$$

$$w_{ij,lk} = -2a(\delta_{il} + \delta_{jk}) - 2c(1 - \delta_{il})(1 - \delta_{jk})(ae_{ij} - ae'_{lk})^2,$$

$$\theta_{ij} = -4a + b(av_i - av'_j)^2 \tag{7a}$$

where δ_{ij} is the Kronecker delta;

(ii) For $n < m$

$$E = -0.5 \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{l=1}^{n} \sum_{k=1}^{m} w_{ij,lk} U_{ij} U_{lk} - \sum_{i=1}^{n} \sum_{j=1}^{m} U_{ij} \theta_{ij},$$

$$w_{ij,lk} = -2a(\delta_{il} + 1) - 2c(1 - \delta_{il})(1 - \delta_{jk})(ae_{ij} - ae'_{lk})^2,$$

$$\theta_{ij} = -2a(n+1) + b(av_i - av'_j)^2. \tag{7b}$$

Compare Eq.(7) with the energy expression Eq.(3); it is not difficult to see that U_{ij}, U_{lk} correspond to neurons s_i, s_j, respectively, $w_{ij,lk}$ corresponds to connections w_{ij}, and θ_{ij} to θ_i. If the edges of G and G' are undirectional, i.e., $e_{il} = e_{li}$ and $e'_{jk} = e'_{kj}$, and if the distance function $d(.,.)$ satisfies $d(x,y) = d(y,x)$, then it is easy to see that $w_{ij,lk} = w_{lk,ij}$, i.e., $w_{ij} = w_{ji}$. Therefore, if each element U_{ij} is regarded as a neuron, then matrix $[U_{ij}]$ constitutes a symmetrically interconnected neural network that BM can work on.

As a result, BM and The improved BM can also be directly used to solve the model of Eq.(5). Due to the limited space, we do not here introduce the details which can be found in [14].

5. Computer Simulations. One of our attributed graph matching problems is shown in Fig.1. There are two attributed graphs G, G', each having 8 nodes and 10 edges. On each node and edge there is an attribute. Each node attribute is denoted by a digit and each edge attribute is denoted by a circled digit. G' has the same topological structure as G, but its node labels are all wrong. The cost of one-to-one correspondence of nodes (edges) between G and G' are defined by the square distance between their corresponding attributes, i.e., $d(v_i, v'_j) = (av_i - av_j)^2$ and $d(e_{ij}, e'_{ij}) = (ae_{ij} - ae'_{ij})^2$ where av_i (av'_j) is the attribute of node v_i (v'_j) and ae_{ij} (ae'_{ij}) is the attribute of edge e_{ij} (e'_{ij}), for edges $e_{ij} = e(v_i, v_j)$, $e'_{ij} = e(v'_i, v'_j)$. Specifically, for e_{ij}, e'_{ij}, if one is a pseudo-edge (i.e., there is no real edge between the two nodes) and the other is not, we define $d(e_{ij}, e'_{ij}) = 10^6$ to penalize the correspondence; but if both are pseudo-edges, we define $d(e_{ij}, e'_{ij}) = 0$. These costs are used for computing the total matching cost E by Eq.(5) and ΔE by Eq.(6). In these formulas, the parameters are $a = 1000.0$, $b = 1.0$ and $c = 1.0$.

(1) First, AGM-SA was used. For a group of parameters $T = 2000$, $T_{min} = 500$, $k_{max} = 500$, and $\lambda = 0.95$, the initial match was randomly generated as given in Fig.2a, which is a very bad match and totally unfeasible. After 14028 perturbations (iterations), the algorithm stopped since the present $T = 475.65 < T_{min}$. The final solution $E = 3010.25$ is bad and not feasible as shown in Fig.2(b). Then we kept the other parameters unchanged, but decreased T_{min} to 10.0. After 52105 peturbations, the algorithm stopped at $T = 9.644 < T_{min}$. The global solution $E = 0.5$ was obtained as given in Fig.2(c), but with the extra cost of 38077 perturbations. Hereafter, we still kept the other parameters unchanged, but continuosly reduced T_{min} to 1.0. After 74649 perturbations, the algorithm stopped at $T = 0.9591$. However, due the drawback pointed out in sec.2.2, the solution mistakenly jumped from the optimal one to an unfeasible one with $E = 1025.5$. Then as we continued to lower T_{min}, the solution never returned to the optimal one; even after 119194 perturbations , the solution is still a bad and unfeasible one, as given in Fig.2(d). These experiments also revealed that for AGM-SA, T_{min} not only greatly influences the time cost but also influences the optimality of the solution.

(2) Second, AGM-ISA was used. The same conditions (including the randomized condition and the initial match) and the same group of parameters as those in experiment (1) were used. In the first trial with $q_{max} = +\infty$ and $p_{max} = +\infty$, also after 14028 perturbations, the algorithm stopped at the global optimal solution as given in Fig.2c. This is obviously better than that obtained by AGM-SA both in the time cost (38077 perturbations saved) and the optimality of solution. Forthermore, the optimal solution will never be lost by further lowering T_{min}. In another trial, we let $q_{max} = +250$, $p_{max} = 15$, but kept all the other parameters unchanged. The algorithm stopped after only 7384 perturbations. The global optimal solution was still obtained.

6. Summary. The search track and the updating sequence of the present solution in SA has been separated by constructing a new present solution updating sequence. As a result, an improvement has been made to SA, on both the optimality of solution and the computing cost. We have also shown that a similar improvement can be made on BM, too.

After modeling the attributed graph matching problem by the objective function of a combinatorial optimization problem, SA and BM as well as their corresponding improved versions were used to solve the model. The advantages of the improved versions over their original forms were shown through computer simulations with performance comparisons on both the optimality of solution and the time cost.

REFERENCES

[1] S.Kirkpatrick et al, Science, Vol.220, No. 4598, 1983, pp671-680.
[2] E.H.L.Aarts et al, Philips J Res. Vol.40, No.4, 1985, pp193-226.
[3] L. Xu, Proc. of 9th ICPR, Vol. II, Rome, Italy, Nov. 1988, pp1040-1042.
[4] G.E.Hinton et al, "Boltzmann Machines : Constraint Satisfaction Networks that learn", Tech. Rep. No. CMU-CS-84-119, Carnegie-Mellon Univ.
[5] J.Hopfield et al, Biological Cybernetics, Vol.52, 1985, pp141-152.
[6] J. Ramanujam, Proc. of 1988 IEEE ICNN, Vol.II, pp325-332. San Diego, California, 1988.
[7] Y.P.Simon Foo, i.b.i.d., pp275-290.
[8] L. Xu, "An Improved Simulated Annealing method and Its Application to Clustering Analysis", to appear on Information and Control, a Journal of Chinese Society of Automation (in Chinese).
[9] L. Xu, "An Improvement on Simulated Annealing and Boltzmann Machine", Proc. of IJCNN'90, Washington D.C., Jan. 1990.
[10] W.H.Tsai and K.S.Fu, IEEE Trans. Vol. SMC-9, No.12., 1979
[11] W.H.Tsai and K.S.Fu, IEEE Trans. Vol. SMC-13, No.1., 1983, pp48-62.
[12] M.You and K.C.Wang, , Proc. of 7th IJCPR, Vol.1, 1984, pp314-319.
[13] C.C.Shen and W.H.Tsai, IEEE Trans. Vol. C-34, No.3, 1985, pp197-203.
[14] L.Xu and E.Oja, "Neural Networks for Attributed Graph Matching in Machine Vision", Tech. Rep., Lappeenranta Univ. Tech., Oct., 1989.

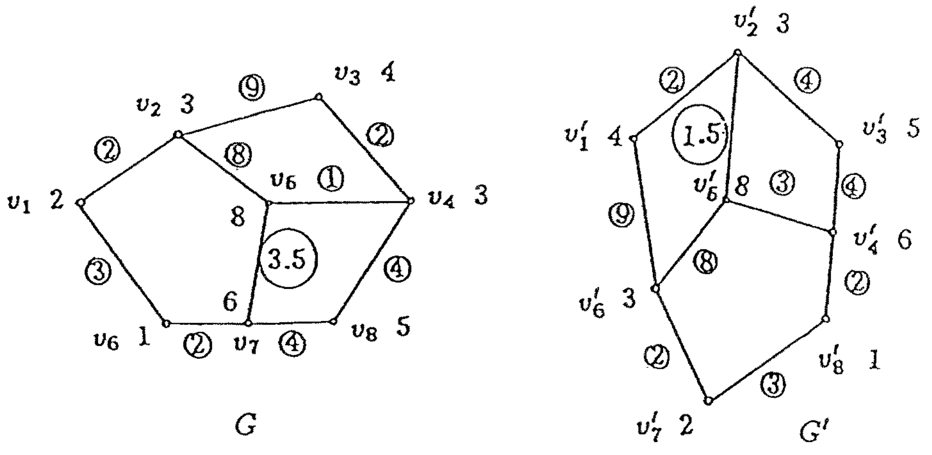

Fig.1 Two Attribued Graphs

$$\begin{array}{c|cccccccc}
 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \\
\hline
v'_1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
v'_2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
v'_3 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
v'_4 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
v'_5 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
v'_6 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
v'_7 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
v'_8 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0
\end{array}$$

E = 1.13046e+008.

Fig.2(a)

$$\begin{array}{c|cccccccc}
 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \\
\hline
v'_1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
v'_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
v'_3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
v'_4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
v'_5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
v'_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
v'_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
v'_8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

The final E = 3010.25
The Matched Matrix

Fig.2(b)

$$\begin{array}{c|cccccccc}
 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \\
\hline
v'_1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
v'_2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
v'_3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
v'_4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
v'_5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
v'_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
v'_7 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
v'_8 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{array}$$

The final E = 0.5
The Matched Matrix

Fig.2(c)

$$\begin{array}{c|cccccccc}
 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \\
\hline
v'_1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
v'_2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
v'_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
v'_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
v'_5 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
v'_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
v'_7 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
v'_8 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{array}$$

The final E = 3099.75
The Matched Matrix

Fig.2(d)