A New Adaptive Move Type Selection in Simulated Annealing

invited paper

Srdjan Lj. Milenković, Vančo B. Litovski and Zoran Obradović[1]

Abstract - **Simulated annealing, a general powerful optimization technique for solving a class of problems with combinatorial nature, is described. One (and only one, we think) of the disadvantages this algorithm posses, is its slow convergence. Many hardware and software techniques developed in order to speed up the algorithm are reported in the literature, but low attention is paid to the possibility that the speed could be increased by the new state generation strategy. One such strategy based on adaptive move type selection (when many move types are available) will be proposed. Adaptive move type selection strategy activates every move only in the temperature range where its acceptance ratio has reasonable value (not extremely high nor low). Outside of this temperature range move type is inactive, i.e. there is no chance to be performed. This will prevent move types that cause low changes in the cost function to be used at high temperature, as well as, those moves that cause high cost changes at low temperature. The amounts of rejected moves, and consequently, computation time, are expected to be reduced.**

I. INTRODUCTION

Simulated annealing [Kir,83] is an iterative procedure using the analogy between the cooling process of physical system and the optimization problem. Let us imagine some arbitrary physical system at high temperature. If it is high enough, the system will be far from its minimum energy (ground) state. During the cooling process, where the temperature is being decreased, the system energy will be decreased, as well. At extremely low temperatures the most probable state reached by the system is its ground state. If the objective function of the optimization problem is considered as the system energy, previously described cooling process could be used to obtain the solution with minimal energy, i.e. objective function. Generally speaking, this idea could be used to solve any problem with finite count of possible states and some function that assigns the number as a solution quality measure to the every state. That function is called *objective* or *cost* function.

Annealing process, mentioned above, could be simulated by the computer using the procedure depicted on Figure 1.a (C like pseudo code is used).

---

[1]The authors are with the Faculty of Electronic Engineering, University of Niš, Beogradska 14, 18000 Niš, Yugoslavia. Zoran Obradović is with the School of Electrical Engineering and Computer Science, Washington State University, Pullman WA 99164-2752, USA and is also affilated with the Mathematical Institute, Belgrade, Yugoslavia.

```
InitialSolution();
T = InitT();
C = Cost();
while("freezing point is not reached") {
        while("equilibrium is not reached") {
                move = SelectMoveType();
                Generate(move);
                Cnew = Cost();
                if( Accept(C,Cnew,T) == FALSE) {
                        Reject(move);
                }
        }
        T = Update(T);
}
```

(a)

```
Accept(C,Cnew,T) {
        ΔC = Cnew - C;
        r = Random(0,1);
        y = min(1,exp(-ΔC/T));

        if(r < y ) return(TRUE);
        else return(FALSE);
}
```

(b)

Figure 1

The algorithm starts by initial solution determination and temperature initialization. Those steps are done by **InitialSolution** and **InitT** functions from Figure 1.a. Initial solution is usually randomly generated because the final solution does not depend on it [Gid,85]. The temperature is assigned the value high enough so, at the beginning of the process, every move to be accepted with high probability. It is worth to mention here that, in the field of optimization, temperature has no meaning and represents the control parameter only, but following the analogy, it has kept the name. Initialization process is finished by initial solution cost function calculation. After that, iterative process could be invoked. There are two iterative loops. Inside the inner loop the temperature remains constant and is changed by **Update** function when the inner loop convergence is achieved. In order to have cooling process, the temperature is being decreased inside outer simulated annealing loop. The inner annealing loop consists of few successive steps. First, a new solution is generated by small changes of the current one. Those small changes are called *moves* or *perturbations*. Usually, all moves are partitioned into subsets so the moves with the similar nature belong to the same set. Such sets define the *move types*. If the annealing is defined to have more than one move type, new state generation is performed first, selecting one of the possible move types and then applying it to the current solution. **SelectMoveType** and **Generate** function perform these actions, respectively. After the generation phase, a cost function of the newly generated solution is determined. **Accept** function decides whether the new solution will be accepted as the current one or not. It is obvious from Figure 1.b, which gives a simplified form of **Accept** function, that the new solution will be accepted if its cost function is less than the cost of the current one. Even if the cost function of the new solution is greater than the cost of current one, there is chance the new solution to be accepted but with the probability equal to $\exp(-\Delta C/T)$ which is temperature controlled. If the solution is rejected, **Reject** function restores the data structures to the previous state. Inner loop iterates until the system reaches the steady state (equilibrium). After that, as mentioned before, the temperature is decreased and

iteration process is continued until the outer loop convergence. The outer loop converges at very low temperatures so its convergence is known as freezing point detection.

As **Accept** function is temperature dependent one may observe the following: At high temperature ($T \to \infty$), new states that are increasing the cost function are accepted with probability equal to one, as well as the states that are decreasing the cost function. That means, the system will be far from ground state. If the temperature is decreased, the states that are increasing the cost function will be accepted with lower probability, while the acceptance of the states decreasing the cost stays constant (equal to one) during the whole process. The result of such activity is that the energy, or cost function, is decreasing. At the temperature close to zero, only the solution with the cost function lower than the cost of the previous one, will be accepted so the annealing becomes similar to the greedy algorithms. The solution of such process converges to the state with minimal cost function similarly to the physical system during the cooling process. This similarity is confirmed by the fact that the probability the system, at the temperature T, is in state S_i with cost C_i is given by:

$$(1) \qquad p_i(T) = P(S = S_i) = \frac{\exp(-C_i / T)}{\sum_j \exp(-C_j / T)} ,$$

that defines probability distribution similar to the Boltzmann one, and the cost function expected value at T is:

$$(2) \qquad < C(T) > = \frac{\sum_i C_i \exp(-C_i / T)}{\sum_j \exp(-C_j / T)} .$$

The energy of some physical system at the temperature T is given by the equation very similar to (2) (Boltzmann constant is omitted in (1) and (2)). Of course, in order that (1) and (2) to be applicable, simulated annealing must satisfy some constraints (see for example [Gid,85] and [Mit,85]).

Simulated annealing differs from similar algorithms by accepting the solutions of lower quality. That prevents the convergence to local optimum, and the process reaches the global optimum (actually near optimum) rather than the local one. Due to that reason, simulated annealing belongs to Hill-Climbing class of algorithms. Because the current state of the optimization is random variable, annealing is Monte-Carlo, or Stochastic method, as well.

Although the simulated annealing has very simple structure, its realization is not so simple and is closely related to the following questions: How to initialize the temperature? How to generate the new solutions? How to decrease the temperature? How to detect the equilibrium and freezing point? The best results are obtained using adaptive strategies as answers to the mentioned questions [Cat,88], [Hua,86], [Kir,83], [Lam,88], [Mit,85], [Ott,88], [Sec,86], [Sec,88a], [Sec,88b], [Mil,92].

Although the annealing posses good performances considering the solution quality, the algorithm has one disadvantage, that is slow convergence. Great effort has been made to overcome this problem. There are two basic strategies for this purpose - software and hardware accelerations. In that sense, the mentioned adaptive methods for cooling process control are introduced as software accelerations. Realization of the annealing algorithm on parallel computers represents its hardware acceleration.

The aim of this paper is to propose another alternative for simulated annealing efficiency increase. It belongs to the class of software accelerations and is based on carefully constructed move type selection strategy that will be described in the following chapters. Chapter II gives an overview of the existing move type selection methods while chapter III describes our new one. Experimental results that show the proposed method quality are given in chapter IV.

II. EXISTING MOVE TYPE SELECTION METHODS OVERVIEW

As we have mentioned above, the new states generation strategy, in the case where more than one move type is available, consists of move type selection and its application to the current solution. The problem of move type selection, to our knowledge, is not well addressed in the literature. Just few papers, [Sec,88a] and [Sec,88b], deal with this problem. Here, the annealing is applied to the placement problem in integrated circuits layout design. There are many move types like "*pairwise interchange*", "*single module displacement*", "*pin swapping*", etc. The authors of the mentioned papers, have declared two move types as the master types (pairwise interchange and single cell displacement). In the new state generation phase, one of them is selected randomly so the ratio of their probabilities to be selected is equal to R, where R is experimentally determined parameter. R has a value that favors single module displacement move type because it causes lower cost changes. Other move types are performed if the master move types are rejected.

This selection strategy has few disadvantages. First, the selection probability of some move types is conditional. That causes the states transition probability matrix (the probability the system reaches new state S_i if the current one is S_j, in one step) to have elements that represent conditional probability, as well. The mathematical model of the simulated annealing with such transition matrix is not known so the behavior of such optimization process is not well defined. Next, parameter R is problem dependent, and finally, R stays constant during the whole optimization process. That means the single cell displacement move type is forced at high temperatures, as well as, in other temperature ranges. It should not be, because small improvements in cost function obtained applying single module displacement at high temperature will be discarded by greater changes caused by pairwise interchange move type whose acceptance probability is still high. Similar situation happens at low temperatures. The algorithm will perform many pairwise interchange move types that will be rejected. The point is that the computation time is unnecessary increased.

Having in mind the previous discussion it could be concluded that some adaptive move type selection strategy, that is not problem dependent with unconstrained transition probability matrix is desired. One such strategy is described in the following chapter.

III. A NEW ADAPTIVE MOVE TYPE SELECTION

In order to define a new adaptive move type selection strategy that satisfies the constraints noted in the previous chapter, let us introduce some notations. Let us denote the possible move types by π_i, $i=1,2, \ldots,n_\pi$. Let ρ_i represents a number of possible moves of type π_i. Move type π_i causes an average cost change $\Delta_i C$ with standard deviation $\sigma(\Delta_i C)$.

The simulated annealing employing new adaptive move type selection strategy slightly differs from the classical one in **InitT,** and drastically in **SelectMoveType** functions from Figure 1. Adopting a procedure that is reminiscent to the one described in [Hua,86], we first determine initial temperature for any possible move type. For that purpose, a number of virtual moves are performed and statistical changes in cost function are collected. All moves are accepted because it is assumed that the system is at infinite temperature. Values $\Delta_i C$ and $\sigma(\Delta_i C)$ are calculated for every π_i. After that, initial temperature T_i of move type π_i is determined so that the solutions with positive cost changes:

$$\Delta C = \Delta_i C + 3\sigma(\Delta_i C) , \tag{3}$$

are accepted with high probability, that leads to the following equation for T_i:

$$T_i = 6.66\left(\Delta_i C + 3\sigma(\Delta_i C)\right) . \tag{4}$$

The initial temperature of the whole process is then given by:

$$T_{\text{inf}} = \max T_i . \tag{5}$$

The idea from [Hua,86] had to be changed because it gives the initial temperature of the whole process only, while here we need initial temperature of every move type separately. But, having in mind that the same philosophy is used, (5) gives approximately the same initial temperature as the method given in [Hua,86].

During the cooling process, at temperature T, **SelectMoveType** function randomly selects one move type with probability proportional to its selection factor that is given by:

$$(6)\qquad \alpha_i(T)=\begin{cases}0, & \text{if } T>T_i\\ \dfrac{\rho_i}{\rho_{\max}}\eta_i(T), & \text{if } T\le T_i \text{ and } \eta_i(T)\ge\eta_{\min}\\ 0, & \text{if } \eta_i(T)<\eta_{\min}\end{cases},$$

where $\eta_i(T)$ is acceptance ratio of move type π_i at the temperature T, i.e.:

$$(7)\qquad \eta_i(T)=\frac{n_{ia}(T)}{n_i(T)}.$$

Here, $n_i(T)$ and $n_{ia}(T)$ represent number of generated and accepted moves of type π_i at T, respectively.

The selection factors (6), are normalized by $\rho_{\max}$, while $\eta_{\min}$ defines the stop criterion of the outer annealing loop. The cooling process is finished when the acceptance ratios of all move types become lower than $\eta_{\min}$.

As it can be seen, the proposed move type selection strategy does not depend on the number of possible move types, i.e. n_π may be any number. All move types have the same priority, there are not master move types. All data needed for selection are collected on the example is being solved so that the strategy is problem and example independent. One move type is inactive at those temperatures where its selection factor is equal to zero. This happens when $T>T_i$, where its acceptance ratio is extremely high, and when $\eta_i(T)<\eta_{\min}$, where the probability of its acceptance is very small. This eliminates obsolete attempts to improve the solution using that move type. In the temperature range where the move type is active, its selection factor is proportional to its acceptance ratio and to the number of possible moves of that type. The last fact avoids multiple usage of move types with high acceptance ratio having small number of possible moves. At given temperature, the selection strategy forces those move types with higher probability to be accepted. Furthermore, the move types causing greater cost changes will have higher initial temperature so at high temperature those move types will be used more frequently than those causing smaller cost changes. On the contrary, the acceptance ratio of the move types causing greater cost changes at low temperature is smaller in comparison with the acceptance ratio of the move types causing smaller changes so here the last ones will be forced. This mechanism leads to rejected moves number decrease, and consequently, to the speed up the cooling process.

Finally, one may conclude that the proposed selection strategy avoids problems mentioned in the previous section.

IV. EXPERIMENTAL RESULTS

In order to examine the properties of the adaptive move type selection strategy, two versions of the annealing algorithm are developed and applied to the multi-variable function optimization problem. One of them uses traditional new

state generation strategy with move rang limiting as a speed up technique [Sec,86]. In the following text, this version is referred to as RL annealing. The second one employs the generation strategy with previously described adaptive move type selection and will be referred to as AMTS annealing.

It is assumed that the unknown variable values are bounded, so the search for a solution in bounded intervals is performed.

RL annealing uses new states generation strategy that is illustrated on Figure 2.a. In the generation phase, one of the variables is randomly selected and changed. If x_a and x_b are the bounds of the selected variable, and x_c is its current value, this variable is assigned to the new value randomly selected from the interval defined by the window, as illustrated on Figure 2.a. The introduced window is also known as *rang limiter*. Window size, or rang limiter, is temperature dependent. At the beginning of the process, window size is equal to the interval size, and is being decreased with temperature decreasing (for more details see [Sec,86], [Sec,88a], [Sec,88b]).

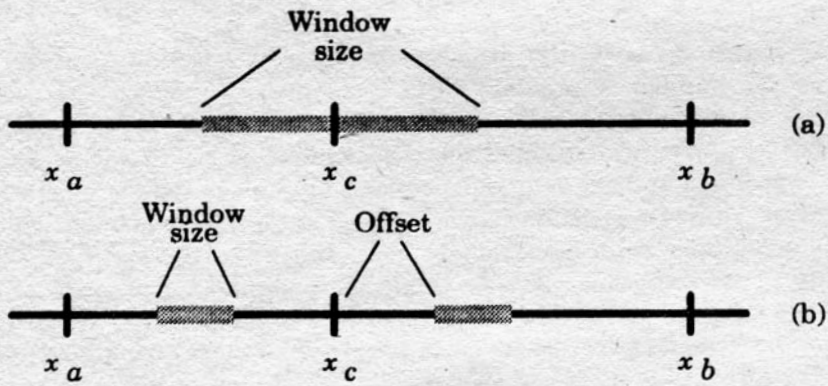

Figure 2

AMTS annealing uses similar mechanism wile producing the new solution. The variable to be changed is randomly selected, as well. New value is randomly chosen from one of two intervals defined by *window size* and *offset*, as shown on Figure 2.b. Unlike the window size in RL annealing, window size and offset in AMTS annealing are constant during the process. Here, a set of move types is defined using different window size and offset. The window size and offset are in correlation so that smaller window size is related to the smaller offset. Between the move types, these values differ from each other for at least one order of magnitude. The move types constructed in such way will produce drastically different average cost changes and standard deviations. This is usual situation in real life problems. When applied in such environment, good performances of the proposed selection strategy are expected.

A set of examples are tested and some interesting results are shown on Figure 3, 4 and 5, and in Table I. Figure 3 depicts a typical plot of the cost function versus the temperature. Evidently, it is very similar to the plot of the physical system energy. Figure 4 compares global acceptance ratios in RL and AMTS annealing for the same example. AMTS annealing acceptance ratio is higher in

wider temperature range, as mentioned before. Figure 5 shows move type selection factors in normalized form, as functions of temperature. One may see that there are different temperature ranges with different active move types confirming that their use was optimized. Here, four different move types are used.

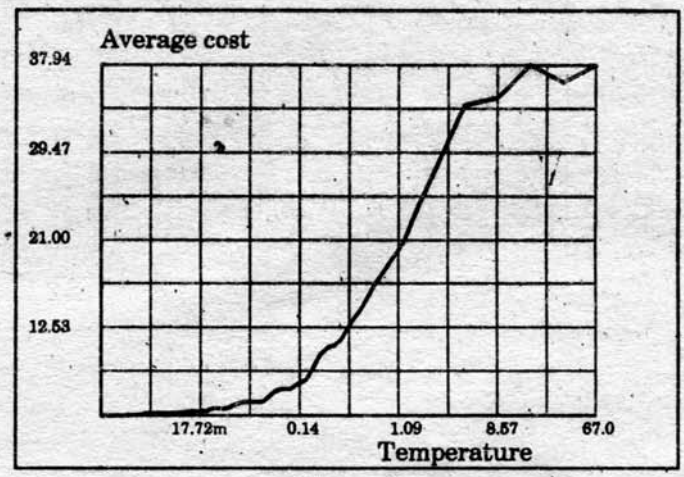

Figure 3

The main results are given in Table I. The CPU time and minimum values reached by RL and AMTS annealing for a set of examples are given. Those data are average values obtained in five successive executions on Silicon Graphics Iris Indigo work station. Also, the values of exact minimum are given, if known. AMTS annealing is faster and gives better solutions. The last example seems as an exception of the above conclusion but look at the solution quality. The lower minimum obtained compensates for the time spent. Having in mind that 'test4' is an electronic filter design problem where the cost function is given in normalized form, we think that AMTS spent more time searching for a significant better solution (regardless the minima reached differ in the third significant digit).

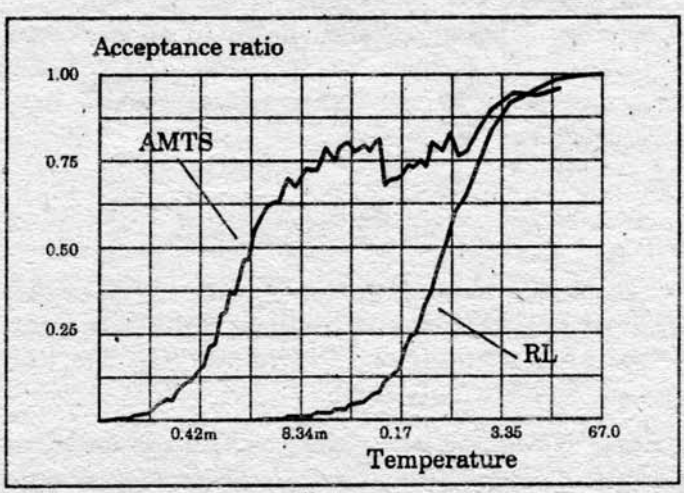

Figure 4

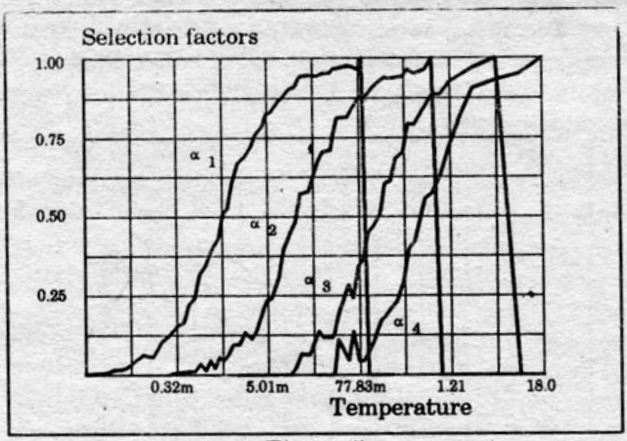

Figure 5

Table I

Example	Exact minimum	RL		AMTS	
		CPU [sec]	minimum	CPU [sec]	minimum
test1	4.0	21.636	4.000172	7.384	4.0
test2	4.0	60.058	4.064574	45.804	4.000722
test3	2.71828	2.892	2.178204	0.434	2.71828
test4	*	51.78	0.114080	81.192	0.110262

V. CONCLUSION

Simulated annealing is presented as a general method that could be applied to the wide class of problems with combinatorial nature. New possibility for algorithm efficiency increase was observed and developed. A new adaptive move type selection strategy is proposed and shown to be successful. This strategy is general purpose so it is problem independent and overcomes the problems traditional strategies suffer from. Experimental results have shown expected behavior and efficiency.

VI. REFERENCES

[Cat,88] F. Cathoor, H. de Man, J. Vandewalle, "**SAMURAI: A General end Efficient Simulated - Annealing Schedule with Fully Adaptive Annealing Parameters**", *INTEGRATION the VLSI journal*, Vol. 6, No. 2, pp. 147-178, July 1988.

[Gid,85] B. Gidas, "**Nonstationary Markov Chains and Convergence of the Annealing Algorithm**", *Journal of Statistical Physics*, Vol. 39, No. 1/2, pp. 73-131, 1985.

[Hua,86] M.D. Huang, F. Romeo, A.S. Vincentelli, "**An efficient General Cooling Schedule for Simulated Annealing**", *Proc. of International Conf. on Computer Aided Design*, pp. 381-384, 1986.

[Lam,88] J. Lam, J.M. Delsome, "**Performance of a New Annealing Schedule**", *Proc. of 25th Design Automation Conf.*, pp. 306-311, 1988.

[Kir,83] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, "**Optimization by Simulated Annealing** ", *Science*, Vol. 220, No. 4598, pp. 671-680, May 1983.

[Mil,92] S.L. Milenković, "**Algorithms and Methods for Automatic Module Placement in electronic circuit design**", *in CADEC 2*, edited by V.B. Litovski, pp. 255-392, Nauka, Beograd 1992, in Serbian.

[Mit,85] D. Mitra, F. Romeo, A.S. Vincentelli, "**Convergence and Finite - Time Behavior of Simulated Annealing**", *in Proc. 24th Conf. on Decision and Control*, pp. 761-767, December 1985.

[Ott,88] R.H.J.M. Otten, L.P.P.P. van Ginneken, "**Stop Criteria in Simulated Annealing**", *Proc. of International Conf. on Computer Design*, pp. 549-552, 1988.

[Sec,86] C. Sechen, A.S. Vincentelli, "**TimberWolf3.2: A New Standard Cell Placement and Global Routing Package**", *in Proc. 23rd Design Automation Conf.*, pp. 432-439, 1986.

[Sec,88a] C. Sechen, D. Braun, A.S. Vincentelli, "**ThunterdBird: A Compete Standard Cell Layout Package**", *IEEE Journal of Solid - State Circuits*, Vol. 23, No. 2, pp. 410-420, April 1988.

[Sec,88b] C. Sechen, "**Chip - Planning, and Global Routing of Macro/Custom Cell Integrated Circuits Using Simulated Annealing**", *Proc. of 25th Design Automation Conf.*, pp. 73-80, 1988.