International Journal of Soft Computing 3 (2):134-138, 2008
ISSN: 1816-9503


# An Adaptive Approach for Parallel Simulated Annealing

[1]A. Iyemperumal and [2]S.P. Rajagopalan
[1]Department of Mathematics, [2]School of Computer Science and Engineering,
Dr. M.G.R. University, Chennai-95, India

**Abstract:** In this study, we analyse alternatives for the parallelization of the simulated annealing algorithm when applied to the placement of modules in a VLSI circuit considering the use of PVM on an Ethernet cluster of workstations. It is shown that different parallelization approaches have to be used for high and low temperature values of the annealing process. The algorithm used for low temperatures is an adaptive version of the speculative algorithm. Within this adaptive algorithm, the number of processors allocated to the solution of the placement problem and the number of moves evaluated per processor between synchronization points change with the temperature. At high temperatures, an algorithm based on the parallel evaluation of independent chains of moves has been adopted. It is shown that results with the same quality of those produced by the serial version can be obtained when shorter length chains are used in the parallel implementations.

**Key words:** Adaptive algorithm, parallelization, speedup analysis, PVM, VLST

## INTRODUCTION

We propose a new adaptive technique for the parallelization of the Simulated Annealing (SA) algorithm applied to the placement problem in VLSI design, considering the use of PVM (Geist *et al*., 1994) and a dedicated Ethernet cluster of homogeneous workstations. SA was initially proposed by Kirkpatrick for the solution of complex optimization problems and has shown to be effective in the solution of a large set of applications (Huang *et al*., 1986; Laarhoven and Aarts, 1987).

Within SA, the temperature parameter can deeply affect the algorithm behaviour during its execution. In this research it is shown that such dynamic behaviour offers new adopted parallelization approach consists of changing the used algorithm according to the optimization process phase. In fact, different parallelization techniques are used for high and low temperature values. This aspect can be regarded as the first level of adaptation introduced in the algorithm.

The algorithm used for low temperatures itself is an adaptive version of the speculative algorithm proposed by Sohn (2005). In this adaptive algorithm, the number of processors allocated to the parallel machine and the number of moves evaluated per processor between synchronizations change with the temperature.

At high temperatures, an algorithm based on the parallel evaluation of independent chains of module moves has been adopted. It is shown that results with the same quality as those produced by the serial version can be obtained when shorter length chains are used in the parallel implementation.

## THE SPECULATIVE ALGORITHM

Let S be the set of parallel moves and $S^n \subset S$ be the subset of all non-conflicting moves belonging to S. The application of concurrent moves in $S^n$ to an initial configuration must produce the same final result as the application of any sequence of these moves. Therefore, it is always possible to serialize the moves in $S^n$. The speculative algorithm proposed by Sohn (2005) is based on the idea of suing serializable sets of moves to avoid the generation of parallel conflicting moves.

In the implementation proposed by Sohn, the sequence of moves processed by the serial algorithm is strictly reproduced in the parallel version.

To ensure that the parallel version will generate the same decision sequence as the corresponding serial version, the processors use the same speed to generate the sequence of pseudo random numbers. Figure 1 shows the speculative algorithm running 11 (not 16) iterations in 3 steps. The first step shows N = 7 processors running 7 simultaneous iterations. In this step, processors 4, 5 and 7 accept their moves (shaded boxes indicate accepted states). In this case a processor 4 acceptance is taken and the decisions taken processors 5, 6 and 7 are mistaken. When proposing a move, each processor speculates that

**Corresponding Author:** A. Iyemperumal, Assistant Professor Department of Mathematics, Dr. M.G.R. University, Chennai-95, India

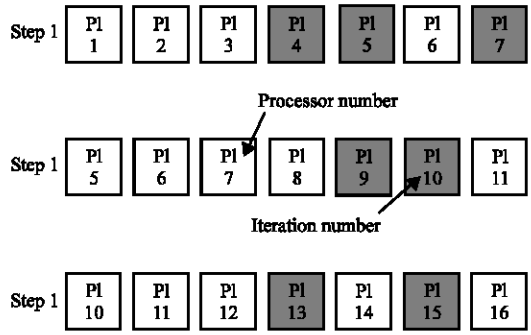


Fig. 1: The speculative algorithm

all previous moves in the same step will be rejected. To follow the serial algorithm path, the moves generated by processors 5-7 should have been applied to the data base with the modifications introduced by the accepted move generated by processor 4.

Since the iterations 5-7 of the first step are discarded, in the second step the seven processors evaluate iterations 5-11. The processors receive the index of the accepted move and based on that information, update their local database and evaluate the indexes of the next. Note that at each step all the processors generate the complete sequence of pseudo random numbers which can locally modify the data base by receiving only the index of the accepted move at the previous step.

Let N be the number of processors and M the size of the serializable set S at each step. At temperature T, the acceptance probability of an individual move is given by a(T). The probability Pr(M = m) for the set S to have size m, is given by,

$$Pr(M = m) = a(T)*[1-a(T)]^{m-1}*m \quad (1)$$

The expected size of the serializable set S is given by:

$$E[S] = [1 - a(T)]^{N} * N + \sum_{m=1}^{N} a(T)*[1 - a(T)]^{m-1} * m \quad (2)$$

## THE ADAPTIVE ALGORITHM

The technique proposed to achieve an efficient parallel implementation of the SA algorithm in low temperatures is an adaptive version of the speculative algorithm. In high temperatures, a technique based on assigning independent move chains to different processors has shown to be more effective. The number of accepted moves gives a clue on the ideal point where a change of approach should be performed.

The restriction imposed by Sohn's algorithm, concerning the generation of exactly the same sequence of moves produced by the serial version, is too strong. The important issue is in fact to keep the same proportion of accepted and rejected moves which occurs in the serial algorithm. In this way, the first modification introduced in the original speculative algorithm was to relax the need for the parallel algorithm to execute the same sequence of moves that the serial algorithm does. With this modification, each processor generates its own independent sequence of pseudo-random numbers. At each step, the winner processor sends to all others its accepted move.

The second modification is related to the process of deciding when to reduce the temperature during the algorithm iterations. In the original speculative algorithm the master has a counter which stores the summation of the sizes of the serializable sets at each step. When a sufficient number of states is detected, the master reduces the temperature and broadcasts a message for all slaves to do the same. The modification introduced is based on the determination of the expected size of the serializable set at each step. Let nsteps be the number of moves that should be tested at each temperature to reach an equilibrium condition. Equation 2 gives the value of E[S] at each step. Therefore, if it is possible to determine the acceptance probability a(T), the number of steps ($\eta$) that need to be processed at each temperature can be evaluated as:

$$\eta = \frac{nsteps}{E[S]} \quad (3)$$

Whenever a new temperature value is determined, the value of h is evaluated by the master and sent to the slaves. The value of a(T) is obtained by extrapolating the known values corresponding to the last processed temperatures. In addition, the policy of validate the accepted move proposed by the processor with the lowest index has been relaxed. The first accepted more notified to the master by a slave is validated. With this scheme, all subsequent acceptance messages can be discarded by the master.

A first version of the algorithm with the introduction of these modifications has been implemented. The quality of the solutions produced by the parallel algorithm was similar to that achieved with the serial version. However, very disappoint results were achieved for the execution time, confirming that the communication cost within the environment of an Ethernet cluster of workstations is not negligible at all. In fact, for small circuits, the times spend on the proposition and evaluation of a new move is certainly much smaller than the time spent on message communication. Therefore, to achieve an effective speedup, it is necessary to analyse the adaptive algorithm taking the communication cost into account.

## SPEED ANALYSIS

Let $t_s$, be the time spent on the processing of the serial SA algorithm at a single temperature value. $t_s$ is given by:

$$t_s = nsteps * t_1 \quad (4)$$

$t_1 \Rightarrow$ Processing time for a single move.

The time spent by the parallel version of the algorithm ($t_p$) is given by:

$$\eta = \frac{nsteps}{E[S]}(t_1 + t_2 + Pr(M \neq 0)t_1) \quad (5)$$

$t_2 \Rightarrow$ Communication time between the master and the slaves.

$Pr(M\neq0) \Rightarrow$ Probability for at least one of the N processors to accept a move.

$$Pr(M^1 0)\ [1-(1-a(T))^N]$$

The last term in Eq. 5 is included to take into consideration the need for all the processors to update their copies of the data base when at least one move is accepted. Therefore, the speedup is given by:

$$speedup = \frac{t_S}{t_P} = \frac{t_1 * E[S]}{t_1 + t_2 + Pr(M \neq 0) * t_1} \quad (6)$$

$$If\ t_2 = K * t_1 \quad (7)$$

$$speedup = \frac{E[S]}{1 + K + Pr(M \neq 0)} \quad (8)$$

If the communication cost is negligible, the value of K is zero and the speedup will be in the range given by E[S]/2 at high temperatures and by E[S] at low temperatures.

## SPEEDING-UP THE ALGORITHM

For the environment under consideration in this study, where the communication cost can be high compared with the cost of proposing and evaluating a move, there is a potential great benefit in increasing the number of moves evaluated between synchronizations. An alternative to achieve this effect is to increase the number of processors in the parallel machine. However, this should be wiser to increase the number of moves evaluated per processor between synchronizations. This solution does not increase the value of K (Eq. 7, 8) since each processor will still send a single accepted move (if any) at each step.

Let n be the number of evaluated moves per processor between synchronizations. Eq. 5 can be rewritten as:

$$t_P = \frac{nsteps}{E'[S]}(n * t_1 + t_2 Pr'(M \neq 0) * t_1) \quad (9)$$

$Pr'(M\neq0) \Rightarrow$ Probability that at least one of the N processors accepts a move,

$$Pr'(M \neq 0) = \left[1 - (1 - a(T))^{n*N}\right]$$

and

$$E'[S] = [1 - a(T)]^{n*N} * n * N + \sum_{m=1}^{n*N} a(T) * [1 - a(T)]^{m-1} * m \quad (10)$$

The speedup equation can then be written as:

$$speedup = \frac{t_S}{t_P} = \frac{t_1 * E'[S]}{n * t_1 + t_2 + Pr'(M \neq 0) * t_1} \quad (11)$$

or

$$speedup = \frac{E[S]}{1 + K + Pr(M \neq 0)} \quad (12)$$

It is important to note that for very small values of K, the speedup tends to be reduced. Therefore, it is not effective to have each processor evaluating more than one move between synchronizations in these cases. On the other hand, if K>>n, the speedup value increases linearly with E'[S].

From Eq. 12 we can see that, for a constant K, the optimal value of n is dynamic and changes with the acceptance probability a(T). This conclusion suggests the sue of an adaptive scheme where the value of n changes with the temperature.

Since, E'[S] and n are functions of the acceptance probability, it is reasonable to think that the speedup can be optimized by changing the value of K. Since, only a single message can flow through the Ethernet at any particular time instant, the value of K is clearly a function of the number of processors exchanging messages. Therefore, Eq. 7 can be rewritten as:

$$K[N] = \frac{t_2[N]}{t_1} \quad (13)$$

Table 1: Results for the circuit with K = 50

| | Serial | | Adaptive | |
|---|---|---|---|---|
| | Cost | Time(s) | Cost | Time(s) |
| Average | 1915 | 63.78 | 1942 | 86.02 |
| Std. Deviation | 28 | 0.60 | 20 | 7.46 |

Table 2: Results for the circuit with K = 3,8

| | Serial | | Adaptive | |
|---|---|---|---|---|
| | Cost | Time(s) | Cost | Time(s) |
| Average | 61929 | 10.343,17 | 62261 | 2671.00 |
| Std. Deviation | 1193 | 107.84 | 1288 | 82.01 |

The measured values of $t_2[N]$ for N between 2 and 10 have shown that the communication time varies linearly with the number of processors in the virtual machine.

Since, the Ethernet cluster used in the experiments consists of 10 workstations, the optimization of the speedup value has been calculated exhaustively for all values of N between 1 and 10 and for all values of n between 1 and 20.

The following Table 1 and 2 show the results achieved with the optimal choice of N and n after several runs of the SA algorithm for the circuits with K = 50 (100 modules, 300 nets) and K = 3,8 (1024 modules, 3000 nets). These results refer only to the domain of utilization of the proposed adaptive algorithm (a(T)<0,2).

Results presented in Table 1 and 2 confirm that the method is not effective for small circuits, even though an improvement by a factor of 5 has been achieved in relation to the non-adaptive speculative algorithm.

## THE ALGORITHM FOR HIGH TEMPERATURES

The adaptive algorithm described is inefficient for high temperatures because, at this phase of the annealing the cardinality of the serializable set is too small. To overcome this difficulty, for each temperature, each processor runs the full serial annealing algorithm on its local database. After computing the different chains of moves, a global synchronization is performed and the lowest cost chain is chosen as the starting point for the next temperature value. The expectation to achieve speedup is based on the assumption that, within the parallel algorithm, shorter move chains can be used at each processor without impairing the solution quality.

In all previously described algorithms, the equilibrium at a particular temperature is achieved through the processing of a big enough number of proposed moves. This number (nsteps) is directly proportional to the number of modules in the circuit to be paced (nmodules).

$$nsteps = KSTEP * nmodules \quad (14)$$

Table 3: Results for the parallel chain method

| | Circuit_1 | | Circuit_2 | |
|---|---|---|---|---|
| | Serial time(s) | Parallel time(s) | Serial time(s) | Parallel time(s) |
| Average | 26.16 | 14.96 | 2329.94 | 808.47 |
| Std. Devin. | 0.54 | 0.26 | 37.75 | 10.68 |

The constant KSTEP has been determined through the analysis of the quality of the generated solution in a large number of runs of the algorithm. To the circuits tested, the value KSTEP = 15 has been selected as a good trade-off between solution quality and the computational cost of the algorithm. The possibility of reducing the value of KSTEP is the basis for expecting some speedup in the parallel version of the algorithm. Experimental results have shown that this hypothesis is true: results similar to the serial version were achieved using KSTEP = 5.

Table 3 shows the results achieved with the application of the parallel chain method for 2 circuits: CIRUCIT_1 with 100 modules and 300 nets and CIRCUIT_2 with 1024 modules and 3000 nets. The values shown are the average results after 20 runs of the algorithm considering the use of KSTEP = 5, aand N = 10. The quality of the results achieved has always been similar to that obtained with the serial version of the algorithm. From Table 3 we can see that, for large circuits, the speedup gain is very close to the adopted reduction factor applied to KSTEP.

## CONCLUSION

The parallelization of the SA algorithm is non-trivial and considering its use to solve the placement problem on an Ethernet cluster of workstations, produces reasonable speedup only for large circuits.

The temperature parameter has a big impact on the behaviour of the SA algorithm; for an effective implementation of a parallel version of the algorithm different approaches had to be used for high and low temperatures.

The adopted approach for low temperatures is based on an adaptive version of the speculative algorithm proposed by Sohn (2005). Within this adaptive algorithm, the number of processors allocated to solve the problem and the number of moves evaluated per processor change with the temperature. In the practical experiments performed, the adaptive algorithm started to be used when the acceptance probability fell below 20%. Considering the use of 10 processors within the Ethernet cluster and the placement of a circuit consisting of 1024 modules and 3000 nets, a speedup of 3,87 has been achieved. For small

circuits the speedup achieved was below 1, but it improved by a factor of 5 the speedup obtained with the non-adaptive speculative algorithm.

The algorithm used in high temperatures processes independent chains of module moves in parallel. The speedup resulted from the use of shorter chain lengths than would be necessary to achieve results of the same quality with the serial version of the algorithm. Within its domain of operation (acceptance probability greater than 20%), a speedup factor of 2,88 has been produced considering the use of 10 processors and the placement of a circuit with 1024 modules and 3000 nets.

## REFERENCES

Geist, A. *et al*., 1994. PVM3 User's Guide and Reference Manual. Oak Ridge National Laboratory.

Huang, M.D., F. Romeo and A.S. Vincentelli, 1986. An Efficient General Cooling Schedule for Simulated Annealing. IEEE International Conference on CAD.

Laarhoven, P.J.M. and E. Aarts, 1987. Simulated Annealing. Theory and Applications, D. Reidel Publishing Company.

Sohn, A., 2005. Parallel N-ary Speculative Computation of Simulated Annealing. IEEE. Trans. Parallel Distribu. Sys., 6: 10.