1996

A heat quench algorithm for the minimization of multiple-valued programmable logic arrays

Dueck, Gerhard W.; Butler, Jon T.

A heat-quench algorithm for the minimization of multiple-valued programmable logic arrays", Computer and Electrical Engineering Journal, Vol. 22, No. 2, 1996, pp. 103-107 1995

http://hdl.handle.net/10945/35753

Pergamon

0045-7906(95)00036-4

Computers Elect. Engng Vol. 22, No. 2, pp. 103–107, 1996
Published by Elsevier Science Ltd
Printed in Great Britain

A HEAT QUENCH ALGORITHM FOR THE MINIMIZATION OF MULTIPLE-VALUED PROGRAMMABLE LOGIC ARRAYS

GERHARD W. DUECK and JON T. BUTLER

Department of Mathematics and Computing Sciences, St Francis Xavier University, Antigonish, N.S. B2G 2W5, Canada and Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA 93943-5004, U.S.A.

(Received for publication 21 November 1995)

Abstract—Simulated annealing has been used extensively to solve combinatorial problems. Although it does not guarantee optimum results, results are often optimum or near optimum. The primary disadvantage is slow speed. It has been suggested [1] that quenching (rapid cooling) yields results that are far from optimum. We challenge this perception by showing a context in which quenching yields good solutions with good computation speeds. In this paper, we present an algorithm in which quenching is combined with rapid heating. We have successfully applied this algorithm to the multiple-valued logic minimization problem. Our results suggest that this algorithm holds promise for problems where moves exist that leave the cost of the current solution unchanged.

Key words: Multiple-valued logic, logic minimization, simulated annealing, heat quench, heuristic, sum-of-products, logic design.

1. INTRODUCTION

Simulated annealing is a means of finding good solutions to combinatorial optimization problems [1]. The basic operation in this technique is a *move*. A move is a transition from one solution in the solution space to another. Each move affects the cost of the solution. Intuitively, one favors cost decreasing moves, since a solution with minimum, or near-minimum, cost is the objective. However, by allowing only such moves, it is likely that the final solution is a local minimum, rather than the absolute minimum. To escape from a local minimum, cost increasing moves must be made.

In simulated annealing, prospective moves are chosen at random. If a move decreases the cost, it is accepted. Otherwise, the move is accepted with probability $P(\Delta E) = e^{-\Delta E/T}$, where T is the temperature and ΔE is the increase in cost that would result from this prospective move. Initially, T is large, and virtually all moves are accepted. Gradually, T is decreased, thus decreasing acceptance of moves that increase the cost. Eventually, the system will reach a state in which very few moves are accepted. That is, there are little or no cost-decreasing moves and most cost-increasing moves are rejected. In this case, the system is said to be *frozen*. The sequence of decreasing temperatures is called the *annealing schedule*. That is, at some temperature T_n, we count the number of attempted moves $N_{\text{attempted}}$ and a number of completed moves $N_{\text{completed}}$. Let $\text{Max}_{\text{attempted}}$ be the limit of the number of attempted moves at each temperature and let $\text{Max}_{\text{completed}}$ be the limit for the number of completed moves at each temperature. To maintain the current temperature, we require that both of the following inequalities be satisfied:

$$N_{\text{attempted}} \leqslant \text{Max}_{\text{attempted}}$$

$$N_{\text{completed}} \leqslant \text{Max}_{\text{completed}}.$$

When one inequality is not satisfied, the temperature is changed. The schedule used by us is described by $T_{n+1} = \alpha T_n$, where $\alpha < 1$ is the cooling rate. Typical values for α are in the range from 0.8 to 0.95. Lower values for α yield more rapid decrease in temperature a condition known as *quenching*.

A major drawback of simulated annealing is that good solutions require a slow cooling schedule, which is time consuming. That is, the system remains at each temperature until a certain number of moves have been attempted or completed. At high temperatures, accepted moves are found quickly. However, as the temperature decreases, fewer and fewer moves are accepted. Before a prospective move is rejected, its cost must be calculated. These calculations account for a large proportion of time required by simulated annealing. In this paper, we present a new algorithm that will heat and quench the solution at hand repeatedly. Note that simulated annealing is a search through the solution space. Indeed, it is a search in which relatively small changes are made to the quality of the solution. Clearly, cost-decreasing moves are needed. But, cost-increasing moves are also needed. That is, if only cost-decreasing moves are made, the search ends at a local minima, a point in the solution space where there are no cost-increasing moves. If there is only one local minima (the global minima) or if all local minima represent a solution that is close to the global minima, then there is little penalty to excluding cost-increasing moves. However, our experience shows that this is not the case for the minimization of multiple-valued functions. See the discussion of this in the next section.

However, due to special characteristics of the multiple-valued logic minimization problem (see next section) quenching produces good results in reasonable times. In addition, we show a deterministic quenching algorithm that requires significantly less cpu time than quenching with simulated annealing. Results obtained from quenching a solution far from optimum can be improved by repeatedly heating and quenching the previous solution. A further advantage comes from the fact that the heat/quench process can be performed in parallel, since heat/quench cycles are independent.

2. MULTIPLE-VALUED LOGIC MINIMIZATION

We illustrate the use of the heat/quench algorithm on the problem of minimizing a multiple-valued logic (MVL) function. Our description of the problem is informal. For a formal description, see [2,3].

Any MVL function can be expressed as a sum-of-products. The aim of MVL minimization is to find an expression with the minimum number of product terms. The only known way to solve this problem is by exhaustive search. However, the excessive computation makes this approach impractical. Therefore, we are interested in heuristic algorithms that produce near-minimal results. Our heuristic is a direct cover method. With this approach, a minterm is selected and then an implicant is chosen to cover the selected minterm. This process is repeated until all minterms are covered. Dueck and Miller [4] propose a direct cover algorithm that produced better results than other direct cover algorithms [5]. A disadvantage of the direct cover method is that it must start with all minterms.

Recently, Dueck *et al.* [2] proposed an algorithm that manipulates sum-of-product expressions directly, using product terms instead of minterms. The process is guided by simulated annealing. That is, two adjacent product terms are chosen at random. If the two product terms can be combined into a single term, the two product terms are replaced by their equivalent. This is a cost decreasing move. In this problem, the cost of a solution is the number of product terms in the expression. Two types of cost increasing moves were investigated. In the first, a product term of the chosen pair is divided into two; that is, one of the pair is randomly chosen and then randomly divided. Such move will always increase the cost by one. This method is known as *cut-and-combine*. The other type of move is called *reshape*. In reshape, a new term is formed containing parts of both terms (the chosen pair). The cost increase can be zero, one, or larger. In general, simulated annealing with reshape moves produces better results in less cpu time than with the cut-and-combine moves. Yildirim *et al.* [6] showed that a combination of the two moves yields even better results.

3. HEAT-QUENCH MINIMIZATION

We minimized 10 randomly-generated 4-variable 4-valued functions. Each function was initially given as a list of minterms. The results are summarized in Table 1. The programs that produced this data were written in C and run on a SUN workstation. The first row shows the results obtained

Table 1. Results from minimizing 10 four-valued four-variable functions

Heuristic	Average product terms	Average cpu time
Dueck and Miller	90.0	33.0
Simulated annealing (reshape)	83.9	75.6
Cut-and-combine (quench)	116.5	246.5
Reshape (quench)	93.1	15.9
Greedy	129.8	0.1
Deterministic quench	91.5	1.5

Table 2. Average number of moves of a given cost

Cost	Moves
0	22.6
1	132.8
2	184.7
3	90.0
4	22.8
5	2.7
6	0.1

by Dueck and Miller's direct cover method [4]. Significantly better results are obtained using simulated annealing with reshape moves, as the table shows. To experiment with quenching, the functions were minimized using a fast cooling rate; i.e. $\alpha = 0.3$. Note that the results obtained by the cut-and-combine method under quenching rate are far from minimal. On the other hand, we observe that simulated annealing with reshape and quenching produced results that are comparable to the results obtained by Dueck and Miller [4], but with only half the computation time. We have also implemented a greedy algorithm that combines all pairs of adjacent product terms that are combinable into a single product term. This is essentially simulated annealing with only cost-decreasing moves. That is, adjacent pairs of product terms are combined until no more pairs can be combined. Since product terms are not divided and recombined, there are no cost-increasing moves. It is clear that such an algorithm is fast. Indeed, as shown in Table 1, there is a speed reduction of two orders of magnitude over the previously discussed algorithms. However, it produces poor solutions. On the average, the greedy algorithm yields 129.8 product terms compared to 83.9 product terms for the best algorithm, simulated annealing (reshape). This substantiates our statement earlier about the existence of local minima far from global minima.

The question arises: why are the results obtained from quenching superior to those obtained by the greedy algorithm? We conjecture that this is because there are prospective zero cost moves. Such moves prevent the solution from being trapped in a local minima with high cost. We analyzed the cost of possible moves during minimization, using the average number of moves over the 10 functions described above. Initially, there are 466.0 possible moves on the average, where 154.5 yield a cost change of -1 (i.e. terms may be combined) and 311.5 yield zero cost. Since the initial function consists only of minterms, moves yielding a higher cost are not possible.

The costs of moves at the END of simulated annealing (using reshape and a cooling rate α of 0.3) are summarized in Table 2. On average, there are 455.6 possible moves. None of the moves result in a cost decrease; this is expected, since there are no cost decreasing moves at a local minima. Note that approx. 5% of the moves results in no increase in cost. This suggests there is merit to examining the potential of zero cost moves. After performing a zero cost move, a newly created term may combine with one of its neighbors. This prompted us to develop a deterministic quenching algorithm described below:

Algorithm: Deterministic Quenching

```
Step 1:   for each pair of adjacent cubes
              combine then if possible
          end for

Step 2:   for each pair of adjacent cubes
              If they can be reshaped at cost zero
                  reshape them
                  combine the resulting cubes with other
                  adjacent cubes (if possible)
              end if
          end for
```

iterate Step 2 until three consecutive iterations yield no further improvement

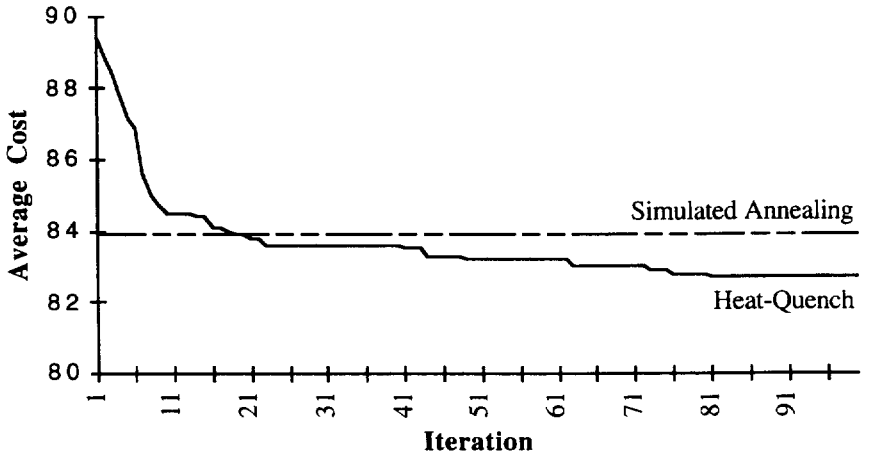


Fig. 1.

Table 3. Average results of heat-quench minimization

Iterations	Average No. of terms	cpu time
20	83.9	15.99
50	83.2	36.07
100	82.7	69.86

Deterministic quenching produced results that were better than those obtained with simulated annealing using reshape when used with quenching. Further, only a fraction of the cpu time was required (see Table 1). Unfortunately, the results are not as good as those obtained with simulated annealing (without quenching). This is expected because it is impossible to escape from local minima. The absence of cost increasing reshape moves compounds this difficulty, since there is no hope of moving without increases in cost to regions where a better solution may be found.

To escape from such local minima, we propose a process where the solution is passed to the simulated algorithm with high heat for a few iterations. Afterward, the quenching process is applied. With experimentation, we found that good results are obtained by heating the solution until the cost increases by 10%. We call this the *heating* phase. The solution is repeatedly heated and quenched. After iterating the heat-quench procedure 20 times, an average result that was as good as the one obtained from simulated annealing was obtained. However, the cpu time required was less than one quarter. In Table 3, we summarize the results obtained by further iterating the heat-quench procedure. Figure 1 shows the plot of the average of the best solution found thus far. Significant improvements are obtained in the first 10 iterations. After 100 iterations, there is very

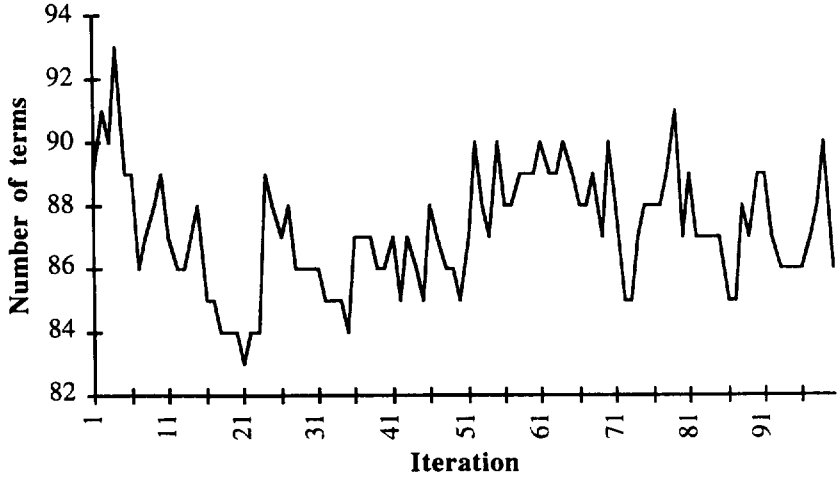


Fig. 2.

little improvement. Figure 2 shows the cost of a single function at the end of each heat-quench iteration.

4. CONCLUSION

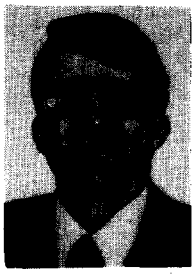
This paper has demonstrated the benefits of the heat-quench approach to solving the MVL minimization problem. That is, instead of a slow progression from an initial (perhaps poor) solution, as in classical simulated annealing, heat-quench alternatively heats the solution for a brief time and quenches for a brief time. The heat part is chosen so that the solution is degraded only slightly (only 10%). In this way, more time is devoted to exploring the solution space near minimal solutions. This is contrasted with simulated annealing, which spends time both far from the minimal solution and at the (near) minimal solution. Our results show either comparable solutions to simulated annealing, at a fraction of the computation time (one-fourth) or better solutions (by 1.4%) at the same computation time.

Our expectation is that the MVL minimization problem is not unique, and heat-quench can yield comparable improvements in other problems. Indeed, the disadvantages of simulated annealing cited above with regard to inefficiency in searching the solution space are algorithm specific, not problem specific.

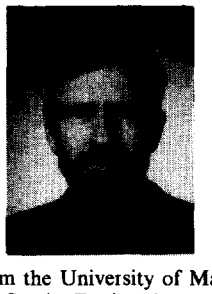
REFERENCES

1. S. Kirkpatrick, C. D. Gelatt Jr and M. P. Vecchi, Optimization by simulated annealing. *Science* **220,** 671–680 (1983).
2. G. W. Dueck, R. C. Earle, P. Tirumalai and J. T. Butler, Multiple-valued logic array minimization by simulated annealing. *Proc. 22nd Int. Symp. Multiple-valued Logic*, pp. 66–74 (1992).
3. J. M. Yurchak and J. T. Butler, HAMLET—an expression compiler/optimizer for the implementation of heuristics to minimize multiple-valued programmable logic arrays. *Proc. 21st Int. Symp. Multiple-valued Logic*, pp. 147–155 (1991).
4. G. W. Dueck and D. M. Miller, A direct cover MVL minimization using the truncated sum. *Proc. 17th Int. Symp. Multiple-valued Logic*, pp. 221–226 (1987).
5. P. P. Tirumalai and J. T. Butler, Minimization algorithms for multiple-valued programmable logic arrays. *IEEE Trans. Computers* **40,** 167–177 (1991).
6. C. Yildirim, J. T. Butler and C. Yang, Multiple-valued programmable logic arrays minimization by concurrent multiple and mixed simulated annealing. *Proc. 23rd Int. Symp. Multiple-valued Logic*, pp. 17–23 (1993).

AUTHORS' BIOGRAPHIES



Jon T. Butler received the Ph.D. degree from Ohio State University in 1973. Since 1987, he has been a Professor at the Naval Postgraduate School, Monterey, Calif. Having served as an Editor of the IEEE Transactions on Computers and as Editor-in-Chief of Computer, he is presently Editor-in-Chief of the Computer Society Press—Advances. He is a Fellow of the IEEE.



Gerhard W. Dueck received the Ph.D. degree from the University of Manitoba in 1988. He is an Associate Professor at St Francis Xavier University in Antigonish, Nova Scotia. During the academic year 91/92 he was research associate at the Naval Postgraduate School, Monterey, Calif. He has served as program chair for the International Symposium on Multiple-Valued Logic. He is a member of the IEEE.