Approximate Algorithms for Minimization of Binary Decision Diagrams on the Basis of Linear Transformations of Variables

A. Kolpakov and R. Kh. Latypov

Kazan University, Kazan, Russia

Received December 17, 2003

Abstract—Algorithms for an approximate minimization of binary decision diagrams (BDD) on the basis of linear transformations of variables are proposed. The algorithms rely on the transformations of only adjacent variables and have a polynomial complexity relative to the size of the table that lists values of the function involved.

1. INTRODUCTION

One of the main problems in the design of the equipment involving digital devices is the problem of optimizing the sizes of circuits. A decrease in the sizes of circuits at the stage of the design of devices makes it possible, on the one hand, to save means at the stage of production of circuits and can lead, on the other hand, to an appreciably increased rate of the computation process on account of a decrease in the delay time.

The idea of the linear transformation of input variables in the problem of minimization of circuits, which is put forward in [1], lies in the fact that the calculation of the initial output function is broken up into two steps. First, instead of the initial output function, the circuit calculates a certain “changed” function and then the second step is taken to carry out the linear transformation of variables so as to derive a “correct” output value. Thus, the value of the initial function is equal to \( f(x_1, \ldots, x_n) = L(\tilde{f}(x_1, \ldots, x_n)) \), where \( L \) is the linear transformation of input variables and \( \tilde{f} \) is the function calculated by the circuit. Various aspects of the linear transformations of variables are set out in [2, 4]. Another approach based on the linear presentation of the structure of data is evolved in [5–8].

The practical implementation of a linear transformation makes sense where the linear transformation \( L \) and the function \( \tilde{f} \) are performed in a simpler way than the initial function; the minimization problem consists in the search for the simplest function \( \tilde{f} \) in the sense of its implementation. Whereas the effective implementation of the linear transformation on the basis of binary adders has a satisfactory solution [9], the exact minimization of circuits on the basis of linear transformations is unfortunately made difficult in view of a large number of possible transformations, which grows fast with an increase in the sizes of circuits. For this reason, the problem of the search for approximate minimization algorithms is urgent.

The binary decision diagrams (BDD) offer an effective method of handling large data structures in an effort to optimize and verify them and calculate various characteristics [10–14]. The BDD structure directly maps into the architecture of hardware tools, for example, programmable logic units, and finds use at the stage of synthesis of these units [14–16].

The linear transformations for the BDD minimization represent a reassuring concept because data structures proper remain invariable and only input variables undergo changes (recoding).
Instead of the operations performed on the table of the function, only transformations of input variables are produced, for which reason use is made of \( n \times n \) matrices rather than matrices of dimension \( 2^n \times 2^n \), where \( n \) is the number of variables. The simplest linear transformation is the renaming or the permutation of variables. The complexity of the search for the optimal permutation of variables is displayed in [17, 18]. It amounts to \( O(n^2 \times 3^n) \), where \( n \) is the number of variables. The use of this method reduces the time it takes to seek the optimal result, on the average, by 50%. But in the sense of complexity, \( O(3^n) \) is a very large value. The algorithm is set out in [19] that serves to find an optimal linear transformation of variables of the Boolean function with the aim to minimize exactly a requisite BDD. It is shown that the BDD can markedly be cut down using the linear transformations. In view of a large computation complexity, this algorithm is suitable only for functions that have no more than 6 incoming variables. In [20], the BDDs of incompletely prescribed functions are minimized with the use of a strict symmetry: the symmetric blocks are fixed and the common permutations are used for the remaining portion. The method is sufficiently effective, but it is proved that the results will be worse than in the case of common permutations, although the processing time decreases.

The analysis of the works reveals that the effective minimization algorithms are set up only for particular cases of linear transformations. An exact minimization is laborious. In this work, we suggest algorithms for an approximate minimization of BDDs. As the basis of the algorithms, we use transformations from a more common class in comparison with the linear transformation class, namely, the class of affine transformations of adjacent variables. Experimental results are laid down that confirm the effectiveness of the suggested algorithms.

The work presents the issues under discussion in the following sequence. First, we describe basic definitions of the linear and affine transformations and the binary decision diagrams. Then, we investigate the effect of linear and affine transformations on BDDs and single out the idea of minimization algorithms. Further, we present algorithms for an approximate minimization of BDDs and some experimental results.

2. BINARY DECISION DIAGRAMS

We will consider a Boolean function \( f: B^n \rightarrow B \) of variables \( x_1, \ldots, x_n \). It is possible to correlate to any Boolean function a binary tree (Fig. 1) in which each nonterminal vertex is denoted by a variable \( x_i \) and two branches emerging from it identify the decomposition of the function into two cofunctions \( f = \overline{x_i} f_{x_i=0} \lor x_i f_{x_i=1}, 1 \leq i \leq n \).

In developing BDDs from a binary tree, use is made of the reduction of excessive subgraphs. We will consider the basic types of applied reductions.

Type I. Reduction of isomorphic subgraphs. It is only one of all the isomorphic subgraphs that is left intact. The input of the cancelable subgraph is redirected to the input of the isomorphic subgraph (the example of this reduction is shown in Fig. 2a).