Exact BDD Minimization for Path-Related Objective Functions

Rüdiger Ebendt Rolf Drechsler

Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
Email: {ebendt,drechsle}@informatik.uni-bremen.de

Abstract

In this paper we investigate the exact optimization of reduced ordered Binary Decision Diagrams *(BDDs) with respect to path-related objective functions. We aim at a deeper understanding of the computational effort of exact methods targeting the new objective functions. This is achieved by an approach based on* Dynamic Programming *which generalizes the framework of Friedman and Supowit. A prime reason for the computational complexity can be identified using this framework.*

For the first time, experimental results give the minimal expected path length of BDDs for benchmark functions. They have been obtained by an exact Branch&Bound *method which can be derived from the general framework. The exact solutions are used to evaluate a heuristic approach. Apart from a few exceptions, the results prove the high quality of the heuristic solutions.*

1 Introduction

Reduced ordered *Binary Decision Diagrams* (BDDs) were introduced in [7] and are well-known from logic synthesis and hardware verification.

Run time and space requirement of BDD-based algorithms depend on the size of the BDD. However, this size is very sensitive to a chosen variable ordering [7]. In general, determining an optimal variable ordering is a difficult problem. It has been shown that it is NP-complete to decide whether the number of nodes of a given BDD can be improved by variable reordering [5]. Therefore, heuristic methods have been proposed, based on structural information or on dynamic reconstruction [22]. But for applications like logic synthesis using multiplexor-based BDD circuits also exact methods are needed: here a reduction in the number of BDD nodes directly transfers to a smaller chip area. Evaluation of heuristic solutions showed that they are often far away from the best known solution. Moreover, exact methods can provide the basis for such an evaluation.

Similar questions arise for *alternative, path-related* objective functions. The optimization with respect to the *number of paths* in a BDD has been studied in [14]. It is motivated by a number of applications in different areas, e.g. testing of BDD circuits, and minimization of DSOPs which are used in the calculation of spectra of Boolean functions [13] or as starting point for the minimization of *Exclusive-Sum-Of-Products* (ESOPs), see e.g. [24]. The optimization with respect to the *Expected Path Length* (EPL) has been studied in [11, 18, 20] and is motivated by DD-based functional simulation [1, 19]. Minimization of EPL as well as of the *Maximal Path Length* (MPL) in BDDs is also motivated by logic synthesis targeting the delay of the resulting circuits. The minimization of MPL has been studied in [11, 21]. To evaluate the quality of heuristics, again a comparison with exact solutions is needed.

In this paper we analyze the computational hardness of the respective exact optimization problems. A known approach to sequencing optimization problems [3, 16] based on *Dynamic Programming* (DP) is generalized. By this, a much less restrictive framework is obtained. Next, this framework is used as a formal tool to analyze the given problems. First, the problems of exact BDD node minimization as well as of EPL-minimization can be solved with DP-based approaches for *Branch&Bound* (B&B) derived by this framework. Second, we show that the problems of minimizing the number of paths in BDDs and of MPL-minimization can not be solved easily. A prime reason for this can be identified, the violation of *Bellmann's principle* [2].

Experiments show that, apart from a few exceptions, the results of a heuristic approach to minimize the EPL in BDDs are of the same quality as exact solutions.

2 Preliminaries

2.1 BDDs

Reduced ordered *Binary Decision Diagrams* (BDDs) are directed acyclic graphs where a Shannon decomposition

$$f = x_i f_{x_i} + \overline{x}_i f_{\overline{x}_i} \quad (1 \le i \le n)$$

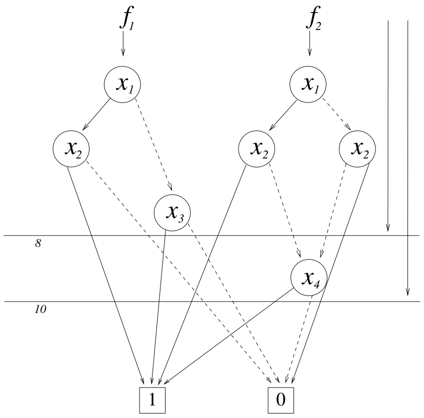
is carried out with each node. Nodes v are labeled with variables in $X_n = \{x_1, \ldots, x_n\}$ (denoted by $\mathrm{var}(v)$), edges are 1- or 0-edges, leading to one of the two child nodes denoted $\mathrm{then}(v)$ and $\mathrm{else}(v)$. The variables are bound to values in $\mathbf{B} := \{0, 1\}$. They are encountered at most once and in the same order, the "variable ordering" denoted π, on every path from the root to one of the two terminal nodes **1** and **0**. Formally, variable orderings map level numbers to variables. The set of all orderings is denoted Π. For a BDD F, we also use π as a prefix operator (i.e. πF) to express that F respects the ordering π. The term $\mathrm{nodes}(F, x_i)$ denotes the set of nodes in the x_i-level of F and $\mathrm{label}(F, x_i)$ abbreviates $|\mathrm{nodes}(F, x_i)|$.

Note that reduced diagrams are considered, derived by removing redundant nodes and merging isomorphic subgraphs. In the following we assume shared BDDs with *Complement Edges* (CEs) [6] without mentioning it further (and without using CEs in the illustrations). Note that all results reported here directly transfer to BDDs without CEs. For examples of shared BDDs, see Fig. 1, for more details see [7].

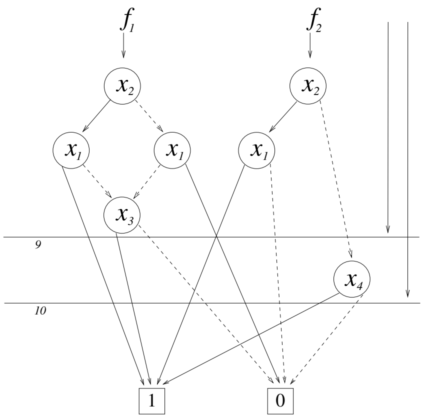
2.2 Cuts

Cuts through the BDD at depth k split up the graph into an upper and a lower part. Cuts can be characterized by a set of nodes which are on a borderline of the cut, e.g. a set of lower nodes which are adjacent to upper nodes or vice versa.

For a BDD F over X_n, let $\mathcal{C}(F, k)$ denote the set of nodes in levels below the k-th level of F (including



(a) An α-minimal ordering for $\{x_1, x_2, x_3\}$.



(b) A suboptimal ordering for $\{x_1, x_2, x_3\}$.

Figure 1. Two BDDs for $f_1 = x_1 \cdot x_2 + \overline{x}_1 \cdot x_3$ and $f_2 = x_1 \cdot x_2 + \overline{x}_2 \cdot x_4$.

the terminal nodes) that are either externally referenced (i.e. represent user functions), or are referenced directly from the nodes in levels $1, \ldots, k$ of F. Let $\mathcal{C}(F, 0)$ denote the set of externally referenced nodes, i.e. the set of nodes which represent user functions. The set $\mathcal{C}(F, 0)$ is equal to the set of output nodes in F.

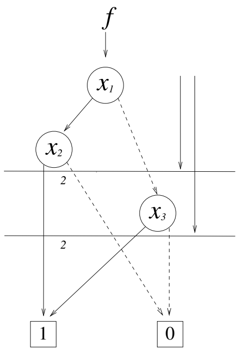
The nodes on the borderline of an *edge cut* at depth k are $\mathcal{C}(F, k)$. By the definition of $\mathcal{C}$, every path starting at an output node and ending at a terminal node must traverse a node in $\mathcal{C}(F, k)$. This definition of a cut is especially useful when considering path related objective functions. We will also need the notation $\mathcal{K}(F, k) = \mathcal{C}(F, k) \setminus \{\mathbf{1}, \mathbf{0}\}$. For examples of edge cuts, see Figs. 1 and 2.

The nodes on the borderline of a *horizontal cut* at depth k are the nodes situated at the k-th level $\mathrm{nodes}(\pi F, \pi(k))$.

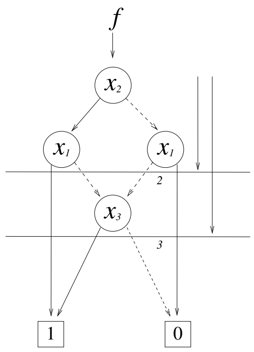
The function CUT is used as a general notation of cuts at depth k. CUT returns a set of nodes situated at the borderline, i.e. the choices for $\mathrm{CUT}(\pi F, k)$ are among $\{\mathrm{nodes}(\pi F, \pi(k)), \mathcal{C}(F, k)\}$.

2.3 Path-Related Objective Functions

We start by the EPL: let F be a BDD and let v be a node in F. The following definition of $\epsilon(v)$, i.e. the expected number of variable tests needed to evaluate an input assignment along a path from v to $\mathbf{1}$ follows [18]. The



(a) A μ-minimal ordering for $\{x_1, x_2, x_3\}$.



(b) A suboptimal ordering for $\{x_1, x_2, x_3\}$.

Figure 2. Two BDDs for $f = x_1 \cdot x_2 + \overline{x}_1 \cdot x_3$.

probability that a variable x is assigned to a value $b \in \mathbf{B}$ is denoted by $\mathrm{pr}(x = b)$.

$$\epsilon(v) = \begin{cases} 0, & v \in \{\mathbf{1}, \mathbf{0}\} \\ 1 \ + \mathrm{pr}\big(\mathrm{var}(v) = 1\big) \cdot \epsilon(\mathrm{then}(v)), & \text{else} \\ \quad + \mathrm{pr}\big(\mathrm{var}(v) = 0\big) \cdot \epsilon(\mathrm{else}(v)) & \end{cases} \quad (1)$$

The EPL of F, denoted $\epsilon(F)$, is based on this definition: in case of a single-rooted BDD it is simply the ϵ-value of the root node, otherwise it is the average of the according values for all roots. Moreover, let $\omega_\epsilon(v)$ denote the probability that an evaluation of input assignments which starts at an output node traverses v.

Other path-related objective functions for BDDs are the number of paths and the maximal path length: let $\alpha(v)$ denote the number of paths from v to a terminal node, and let $\alpha(F)$ denote the number of paths from an output node to a terminal node, respectively. Let $\mu(v)$ denote the maximal length of a path from v to a terminal node, and let $\mu(F)$ denote the maximal length of a path from an output node to a terminal node, respectively. The length of a path is the number of inner nodes on the path.

For a node v, let $\omega_\alpha(v)$ denote the number of paths from an output node to v and let $\omega_\mu(v)$ denote the maximal length of a path from an output node to v, respectively. Further, $\mu\_\mathrm{via}(v)$ denotes the maximal length of a path via v.

2.4 Miscellaneous

Sequences s are denoted using brackets, e.g. $s = \langle e_1, \ldots, e_k \rangle$. By $s \circ e$ we denote the concatenation of s with e to $\langle e_1, \ldots, e_k, e \rangle$. Further, let last: $\mathbb{R}^n \to \mathbb{R}$; $\mathrm{last}(x_1, \ldots, x_n) = x_n$ for all $x_1, \ldots, x_n \in \mathbb{R}$.

We also make use of the following notations: let $I \subseteq X_n$. Throughout the paper, $\Pi(I)$ denotes the set of all orderings whose first $|I|$ positions constitute I. Let cost be a cost function on BDDs, e.g. for BDD size, $\mathrm{cost}(F, X_n) = |F|$. If $\mathrm{cost}(F, X_n) = \kappa(F)$ for an objective function κ, we have a *cost function for* κ. Then

$$\mathit{min\_cost}_I = \min_{\pi \in \Pi(I)} \mathrm{cost}(\pi F, I)$$

denotes the minimal cost under all orderings in $\Pi(I)$. In the case of a cost function for κ, we call the ordering π leading to this minimum a κ-minimal ordering for I. We write Π_I for the set of all κ-minimal orderings for I. Note that $\mathit{min\_cost}_{X_n} = \min_{\pi \in \Pi} \kappa(\pi F)$.

3 Previous Work

To keep the paper self-contained, we briefly review previous work related to our studies. Our analysis is founded on results from two fields of research: the first field is sequencing optimization by DP, the second is BDD optimization. This paper presents research in the intersection of both fields.

3.1 Sequencing Optimization

Aiming at exact optimization with reasonable run times, it is mandatory to keep the size of the search space within sane limits: an exhaustive search essentially would compare every single input datum to every other input datum to find the solution. Hence, an exhaustive search requires $n!$ operations on the data. More mature methods manage to reduce the size of the search space to one of only 2^n states. Moreover, this space can often be pruned by B&B. Following this general outline, the framework for exact BDD minimization [15] was based on a more general DP-approach to solve *sequencing optimization problems* [4,16]. It makes use of *Bellmann's principle* [2]:

> *An optimal sub-sequence $e_1, \ldots, e_k$ must be part of the overall optimal sequence $e_1, \ldots, e_k, \ldots, e_n$ via e_k.* (2)

In [4, 16], n-element sequencing problems were solved with recurrent equations for partial solution costs. These are derived by repeatedly applying (2) to m-element starting sequences ($1 \leq m \leq n$) with a fixed last element (an example will be given at the end of the section).

The tackled problems all respect the following sufficient condition for the validity of (2):

> *The cost of the overall sequence is the sum of the elements costs. For all partial sequences $e_1, \ldots, e_k$, the cost caused by e_k must depend only on what elements are preceding e_k (i.e. it must be independent of their order).* (3)

As a consequence of (2), it is not necessary to construct all of the $n!$ orders for the n elements of the sequence.

As an illustrating example, next it is described how this idea has been used for exact node minimization in [15]. In brief, the optimal variable ordering is computed iteratively by computing for increasing k's min_cost_I for each k-element subset I of X_n, until $k = n$: then, the BDD has a variable ordering yielding a BDD size of $min\_cost_{X_n}$. This is an optimal variable ordering.

This is done by a gradual schema of continuous minimum updates, using the following reccurrent equation [15]. Let F be a BDD.

$$\begin{aligned} & min\_cost_{I'} \\ & = \min_{x_i \in I'} \left[min\_cost_{I' \setminus \{x_i\}} + \mathrm{label}(\pi_i F, x_i) \right] \end{aligned} \quad (4)$$

where π_i is a variable ordering in $\Pi(I' \setminus \{x_i\})$ such that $\pi_i(|I'|) = x_i$. The starting value is $min\_cost_\emptyset = 0$.

This recurrence is based on the principle expressed in (2). The optimal order for an $|I'|$-element sub-sequence of variables is determined by minimizing over all possible last variables x_i. By (2), for every such variable the optimal sub-sequence of the first $(|I'|-1)$ variables must be part of the optimal sub-sequence for all $|I'|$ elements via x_i ("via" here means ending with x_i).

In essence, (2) holds as a direct consequence of the following: the term $\mathrm{label}(\pi_i F, x_i)$ only depends on which variables occur before x_i in the ordering. This has been shown in [15] and is a sufficient condition following (3).

The state space considered here is 2^{X_n} which is of a size growing much slower with n than $n!$. By the use of B&B with lower and upper bounds on BDD size, it can be further reduced [9, 10]. But also recent approaches like the A^*-based approach in [12] still depend on the use of a smart state encoding.

4 Generalized Cost Function for Path-Related Objective Functions

Let a function acc map series with at most n elements to $\mathbb{R}$ and let it respect the following condition:

$$\mathrm{acc}(c_1, \ldots, c_k) = \mathrm{acc}(\mathrm{acc}(c_1, \ldots, c_{k-1}), c_k) \ (1 \leq k \leq n)$$

Then, for $I \subseteq X_n$, a general form of a cost function that is appropriate for a recursion schema is:

$$\begin{aligned} \mathrm{cost}(\pi F, I) &= \mathrm{acc}(c_1, \ldots, c_{|I|}) \text{ where} \\ c_k &= \bigodot_{v \in \textsc{Cut}(\pi F, k)} C(v) \quad (1 \leq k \leq |I|) \end{aligned}$$

Since a cost function can be uniquely determined by the choices of $\mathrm{acc}, \odot, \textsc{Cut}$, and C, it is convenient to give cost functions by tuples $(\mathrm{acc}, \odot, \textsc{Cut}, C)$, e.g. $\mathrm{cost\_size} = (\sum, \sum, \mathrm{nodes}, 1)$. For all nodes v, the contribution is $1(v) = 1$. By this, in the k-th summand of acc, only the nodes in the k-th level are counted, respectively. Depending on the choice of acc and $\odot$, more complex cost functions can be expressed.

5 Computational Effort of Exact Optimization Targeting Path-Related Objective Functions

All path-related BDD optimization problems are special sequencing problems. This raises the question whether DP-based B&B optimization methods using the framework outlined in Section 3 can be found. This is particularly promising since a B&B method for node minimization already is known (see Section 3).

5.1 Exact Minimization of Path Related Objective Functions

Before a sound DP-method following the framework of Section 3 for the exact optimization of α and μ can be obtained, (2) must be proven valid. Unfortunately, both for α and μ, there is only little hope to find a sufficient condition following (3):

Let F be a BDD with an underlying DAG $G = (V, E)$. First, equations must be found that describe the contribution of a single node v to $\alpha(F)$ or $\mu(F)$. We give the following equations describing this interrelation: let $0 \leq k \leq n$. For α, it is

$$\alpha(F) = \sum_{v \in \mathcal{C}(F,k)} \alpha(v) \cdot \omega_\alpha(v), \quad (5)$$

$$\alpha(F) = \sum_{v \in \mathcal{C}(F,n)} \omega_\alpha(v). \quad (6)$$

For μ, it is

$$\mu(F) = \max_{v \in V} \mu\_\mathrm{via}(v), \text{ or, more specific,} \quad (7)$$

$$\mu(F) = \max_{v \in \mathcal{C}(F,k)} \mu\_\mathrm{via}(v), \quad (8)$$

and

$$\mu(F) = \max_{v \in \mathcal{C}(F,n)} \omega_\mu(v). \quad (9)$$

It is straightforward to see this since every path from an output node to a terminal node must traverse a node in an edge cut (see also Section 2.2). The more general equations are (8) and (5). Second, (3) must hold, i.e. the contribution of v must be invariant with respect to reordering of the BDD part above the level the node v is situated at. Unfortunately, all of the above node contributions depend on the ordering of variables before (and some also on that after) the position of v in the ordering.

Still, this does not give strong evidence that sound DP-approaches would not exist: note that (3) is a *sufficient* but *not* a *necessary* condition for the validity of Bellmann's principle.

In the following section, the remaining possibilities to obtain the desired DP-approaches are discussed. Obviously, it is necessary to *generalize* the schema of recursion applied in (4).

5.2 Generalized Dynamic Programming Framework

For a generalization, a *necessary and sufficient* condition is formulated which in fact is equivalent to the principle of Bellmann (2) itself. Hence a much less restrictive and more general schema is yielded. A benefit from the following definition in comparison to (2) is the increased *operationality*, i.e. it is easier to detect whether a given sequencing problem respects the condition or not.

Let s_1, s_2 be two sequences (orders) of the elements in $\{e_1, \ldots, e_{k-1}\}$ and let s_1 be an optimal sequence. Let $\mathrm{cost}(s)$ denote the cost of a sequence s. Then it must be

$$\mathrm{cost}(s_1) = \mathrm{cost}(s_2) \Rightarrow \mathrm{cost}(s_1 \circ e_k) = \mathrm{cost}(s_2 \circ e_k) \quad (10)$$
$$\mathrm{cost}(s_1) < \mathrm{cost}(s_2) \Rightarrow \mathrm{cost}(s_1 \circ e_k) < \mathrm{cost}(s_2 \circ e_k) \quad (11)$$

In essence, Conditions (10) and (11) prevent an optimal sub-sequence from being replaced by a suboptimal one while at the same time the optimality of the overall sequence would be preserved. Equivalence to Bellmann's principle can be shown straightforwardly.

Next, the schema of recursion is given, together with sufficient and necessary conditions following (10) and (11). Thereby, we focus on the problem of BDD optimization, giving the schema for BDDs right away. However, note that it is straightforward to transfer the idea to (any) other sequencing problem.

Theorem 5.1 *Let κ be an objective function for BDDs and let F be a BDD. Let $x_i \in I' \subseteq X_n$. Let $\mathrm{cost} = (\mathrm{acc}, \odot, \mathrm{CUT}, C)$ be a cost function for κ. Further, let $\pi_i^* \in \Pi_{I' \setminus \{x_i\}}$ such that $\pi_i^*(|I'|) = x_i$.*

Assume that the following conditions are respected:

1) *For $v \in \mathrm{CUT}(\pi_i^* F, |I'|)$, $C(v)$ does not depend on the last $n - |I'|$ positions in π_i^*.*

2) *Let $I_1, I_2 \subseteq X_n$, $x_j \notin I_1$, $I_2 = I_1 \cup \{x_j\}$, $\pi_1, \pi_2 \in \Pi(I_1)$ where $\pi_1(|I_2|) = \pi_2(|I_2|) = x_j$, and let π_1 be κ-minimal for $|I_1|$.*

   For shorter notation,

$$\mathrm{coll}_1(\pi_1 F, |I_2|) = \bigodot_{v \in \mathrm{CUT}(\pi_1 F, |I_2|)} C(v) \text{ and}$$
$$\mathrm{coll}_2(\pi_2 F, |I_2|) = \bigodot_{v \in \mathrm{CUT}(\pi_2 F, |I_2|)} C(v).$$

   It must be

$$\begin{aligned}
&\mathrm{cost}(\pi_1 F, I_1) = \mathrm{cost}(\pi_2 F, I_1) \Rightarrow \\
&\quad \mathrm{acc}(\mathrm{cost}(\pi_1 F, I_1), \mathrm{coll}_1(\pi_1 F, |I_2|)) \\
&\quad = \mathrm{acc}(\mathrm{cost}(\pi_2 F, I_1), \mathrm{coll}_2(\pi_2 F, |I_2|)), \\
&\mathrm{cost}(\pi_1 F, I_1) < \mathrm{cost}(\pi_2 F, I_1) \Rightarrow \\
&\quad \mathrm{acc}(\mathrm{cost}(\pi_1 F, I_1), \mathrm{coll}_1(\pi_1 F, |I_2|)) \\
&\quad < \mathrm{acc}(\mathrm{cost}(\pi_2 F, I_1), \mathrm{coll}_2(\pi_2 F, |I_2|)).
\end{aligned}$$

Let $min\_cost_\emptyset = \mathrm{cost}(F, \emptyset)$. Then the following recurrent equation for min_cost

$$min\_cost_{I'} = \min_{x_i \in I'} \left[\mathrm{acc}(min\_cost_{I' \setminus \{x_i\}}, \bigodot_{v \in \mathrm{CUT}(\pi_i^* F, |I'|)} C(v)) \right] \quad (12)$$

holds and we have

$$min\_cost_{X_n} = \min_{\pi \in \Pi} \kappa(\pi F).$$

Further, a DP-method to compute $min\_cost_{X_n}$ exists. It is operating on the state space 2^{X_n}.

Condition 1) ensures that the recursion is well-defined. As before, the correctness of the schema follows from Bellmann's principle. In the following, the schema is applied to various problems of BDD minimization.

5.2.1 Application to Node Minimization

The cost function for the number of nodes is $\mathrm{cost} = (\sum, \sum, \mathrm{nodes}, 1)$, see Section 4. The term $min\_cost_\emptyset = 0$ is the starting value of the recursion. It is trivial to show that Conditions 1) and 2) are respected. Hence $min\_cost_{X_n} = \min_{\pi \in \Pi} |\pi F|$.

By that, essentially the same schema as in (4) is obtained (with the minor specialization that π_i is chosen as π_i^*).

This DP-approach can be turned into a B&B method by the use of lower bounds. In [9], the lower bound

$$l\_b = min\_cost_I + \max\{|\mathcal{K}(F, |I|)|, n - |I|\} + 1 \quad (13)$$

has been proposed. At the end of a recursion step of the outlined DP-approach, all data for a subset I for which the lower bound exceeds or equals the current upper bound (which is updated to the minimal BDD size seen so far with every intermediate BDD constructed), can safely be excluded from further consideration. This is because any ordering in $\Pi(I)$ must yield BDD sizes larger than the smallest BDD encountered.

5.2.2 Application to Minimization of Expected Path Length

The cost function for the expected path length is $\mathrm{cost} = (\sum, \sum, \mathrm{nodes}, \omega_\epsilon)$, the starting value is $min\_cost_\emptyset = 0$. Using results of [17] and [11] it is straightforward to show that Conditions 1) and 2) are respected. Note that the idea of (13) directly transfers to EPL-minimization. Here, it is possible to use the lower bound

$$l\_b = min\_cost_I + \sum_{v \in \mathcal{K}(F, |I|)} \omega_\epsilon(v). \quad (14)$$

In the remainder of the section, the schema is applied to the objective functions α and μ.

5.2.3 Application to Minimizaton of Number of Paths

The node contribution must be based on the cost function in (6), as all other equations define node contributions which depend on the lower part of the BDD (and thus this would violate Condition 1)). Consequently, the only choice for the cost function that respects Condition 1) is

$$\text{cost} = (\text{last}, \sum, \mathcal{C}, \omega_\alpha)$$

First, clearly $\omega_\alpha(v)$ does not depend on the part of the ordering after the position of $\text{var}(v)$, thus Condition 1) is respected. Second, for a BDD πF, it is

$$\begin{aligned} \text{cost}(\pi F, X_n)) &= \text{last}(\ldots, \sum_{v\in\mathcal{C}(\pi F,n)} \omega_\alpha(v)) \\ &= \alpha(\pi F). \end{aligned}$$

because of (6). We can choose an arbitrary value as the starting value of the recursion because the accumulation function is the function last. This yields the recurrence:

$$\mathit{min\_cost}_{I'} = \min_{x_i\in I'} \left[\sum_{v\in\mathcal{C}(\pi_i^* F,|I'|)} \omega_\alpha(v) \right] \quad (15)$$

where $\pi_i^* \in \Pi_{I'\setminus\{x_i\}}$ such that $\pi_i^*(|I'|) = x_i$ is derived. Note that the equation is recurrent although no terms $\mathit{min\_cost}_{I'\setminus\{x_i\}}$ do occur since π_i^* results from previous steps. In particular notice that the first condition of the general recursion schema already forces these choices. But what about the second condition, expressing the key of Bellmann's principle? Does it hold?

Next this is disproven by giving a counter-example (see Fig. 1). It shows that Condition 2) may be violated.

In Fig. 1(a), the ordering $\pi_1 = x_1, x_2, x_3, x_4$ for a BDD $\pi_1 F$ is α-minimal for $I = \{x_1, x_2, x_3\}$. This can be seen by inspecting all $3! = 6$ possible permutations of I (due to space limitation we cannot depict all these orderings). We have $\text{cost}_\alpha(\pi_1 F, I) = 8$. In Fig. 1(b), the ordering $\pi_2 = x_2, x_1, x_3, x_4$ for a BDD $\pi_2 F$ representing the same function causes a cost of 9 for I. Now let $I' = \{x_1, x_2, x_3, x_4\}$. It is $\text{cost}_\alpha(\pi_1 F, I') = \text{cost}_\alpha(\pi_2 F, I') = 10$, i.e. a suboptimal sub-ordering does not lead to higher "future" costs. This violates the second implication of Condition 2).

5.2.4 Application to Minimizaton of Maximal Path Length

The consideration is analogous to Section 5.2.3, essentially just $\sum$ is replaced by $\max$ and ω_α is replaced by ω_μ. Again a counter-example shows that Condition 2) may be violated (the other condition again holds), see Fig. 2. In Fig. 2(a) the ordering $\pi_1 = x_1, x_2, x_3$ for a BDD $\pi_1 F$ is μ-minimal for $I = \{x_1, x_2\}$: since the function essentially depends on x_1, x_2, at least one path going through two nodes, one labeled x_1, the other x_2, must exist. This path is of minimal length 2. The ordering x_2, x_1, x_3 in Fig. 2(b) for a BDD $\pi_2 F$ representing the same function also causes a cost for I of 2. However, the cost for $I = \{x_1, x_2, x_3\}$ is 3, whereas it is only 2 in the BDD $\pi_1 F$. This violates the first implication of Condition 2).

6 Experimental Results

In this section, experimental results are presented. All algorithms have been applied to circuits of the LGSynth93 benchmark set [8]. The tested methods target the two objective functions that allow a DP-based B&B-approach following the framework presented in this paper. This includes the exact B&B method for EPL minimization outlined in Section 5.2.2 (called ϵXACT)[1] as well as the approach to EPL-sifting described in [11]. For a comparison, also the best B&B method for exact node minimization called JANUS [10] has been applied.

To put up a testing environment, all algorithms have been integrated into the CUDD package [23]. By this it is guaranteed that they run in the same system environment. A system with an Athlon processor running at 2.2 GHz, with a main memory of 512 MByte and a run time limit of 36,000 CPU seconds has been used for the experiments.

In a series of experiments, all methods have been applied to the benchmark functions given in Table 1. In the first column the name of the function is given. Column *in* (*out*) gives the number of inputs (outputs) of a function. The next two columns *time* and *space* give the run time in CPU seconds and the space requirement in MByte for the approach JANUS, respectively. The next column *opt. #* shows the minimal numbers of nodes for a BDD representing the respective function. In the next two columns the same quantities run time and space requirement are given for the method ϵXACT, respectively. The next column *opt.* ϵ gives the optimum ϵ-value for a BDD representing the respective benchmark function. The next two columns show the run time and the space requirement for the approach to EPL-sifting. The last column $\hat{\epsilon}$ gives the heuristic ϵ-value as determined by EPL-sifting, respectively.

The results show that the run times of ϵXACT are generally larger than that of the exact node minimization method JANUS. There are two reasons for that: the BDDs created in intermediate steps during operation of ϵXACT can be significantly larger than those in the size-driven method JANUS. Moreover, ϵXACT needs to maintain an additional node attribute (the ω_ϵ-value) with time-consuming hash table accesses during variable swap operations.

For the first time, results of an exact approach to EPL-minimization are given. This allows for the evaluation of the previous heuristic approach called EPL-sifting which shows that it performs much faster (up to five orders of magnitude). Most of the time it achieves almost optimal results. However, it can also be observed that the results obtained by ϵXACT show an improvement in the ϵ-value of 9.6% on average. In some cases (see *comp*, *sct*, *cordic*, *t481*, and *vda*) the gain is significant and it can be more than 50% (see *comp*).

7 Conclusions

We investigated the exact optimization of BDDs with respect to path-related objective functions. Deeper understanding of the computational effort of exact optimization methods targeting the new objective functions has been obtained. For this purpose, the framework of Friedman and Supowit has been reformulated using less restrictive conditions, obtaining a more general framework for approaches based on *Dynamic Programming* and founded directly on *Bellmann's principle*. It was shown that neither the number of paths nor the maximal path length in BDDs can fulfill all requirements of the framework. This result limits the hope for efficient methods targeting this optimization task.

On the other hand we successfully derived a new exact algorithm for the expected path length in BDDs. It is a

[1] Instead of (14) only $\mathit{min\_cost}_I$ has been used as lower bound since otherwise the extra effort of computing the lower bound exceeded the gain in run time for all but the smallest benchmark functions.

Table 1. Results

name	in	out	JANUS			ϵXACT			EPL-sifting		
			time	space	opt. #	time	space	opt. ϵ	time	space	$\hat{\epsilon}$
cc	21	20	81s	36M	46	939s	50M	1.78	0.03s	<1M	1.78
cm150a	21	1	277s	37M	33	785s	23M	3.50	0.03s	<1M	3.50
cm163a	16	5	0.9s	<1M	26	4.5s	<1M	2.34	0.03s	<1M	2.34
cmb	16	4	0.3s	<1M	28	0.2s	<1M	2.00	0.03s	<1M	2.00
comp	32	3	3287s	130M	95	9419s	108M	4.00	0.13s	<1M	9.28
cordic	23	2	1.9s	<1M	42	50s	2M	4.73	0.03s	<1M	6.28
cps	24	102	2359s	61M	971	26335s	96M	2.31	0.10s	<1M	2.31
i1	25	16	20s	10M	36	232s	23M	1.72	0.03s	<1M	1.72
lal	26	19	450s	79M	67	10023s	310M	2.06	0.03s	<1M	2.08
mux	21	1	278s	36M	33	786s	22M	3.50	0.03s	<1M	3.50
pcle	19	9	5.2s	3M	42	169s	10M	2.50	0.03s	<1M	2.50
pm1	16	13	0.6s	<1M	40	1.6s	<1M	1.74	0.03s	<1M	1.75
s208.1	18	9	5.3s	2M	41	177s	10M	2.69	0.03s	<1M	2.69
s298	17	20	8.7s	3M	74	59s	5M	2.10	0.03s	<1M	2.10
s344	24	26	847s	111M	104	24872s	347M	2.22	0.03s	<1M	2.22
s349	24	26	851s	111M	104	24932s	347M	2.22	0.03s	<1M	2.22
s382	24	27	416s	75M	119	14831s	347M	2.15	0.04s	<1M	2.16
s400	24	27	413s	75M	119	14793s	347M	2.15	0.03s	<1M	2.16
s444	24	27	462s	82M	119	14637s	347M	2.15	0.04s	<1M	2.19
s526	24	27	833s	111M	113	16755s	347M	2.21	0.04s	<1M	2.21
s820	23	24	1080s	59M	220	9374s	93M	2.54	0.04s	<1M	2.54
s832	23	24	1127s	59M	220	9660s	93M	2.54	0.04s	<1M	2.55
sct	19	15	6s	3M	48	191s	10M	2.25	0.03s	<1M	2.36
t481	16	1	0.4s	<1M	21	4.5s	<1M	8.25	0.03s	<1M	9.00
tcon	17	16	0.6s	<1M	25	25s	5M	1.50	0.03s	<1M	1.50
ttt2	24	21	521s	82M	107	16189s	347M	2.55	0.03s	<1M	2.55
vda	17	39	30s	3M	478	512s	6M	4.39	0.05s	<1M	4.43

DP-based B&B method that can be obtained by the general framework.

Experimental results showed the feasibility of the exact approach. For the first time it became possible to evaluate heuristic approaches to EPL minimization.

References

[1] P. Ashar and S. Malik. Fast functional simulation using branching programs. In *Int'l Conf. on CAD*, pages 408–412, 1995.

[2] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.

[3] R. Bellmann. Combinatorial processes and dynamic programming. In *Proc. of Symp. in Applied Mathematics of the American Mathematical Society*, 1960.

[4] R. Bellmann. Dynamic programming treatment of the traveling salesman problem. *J. Assoc. Comput. Mach.*, (9):61–63, 1962.

[5] B. Bollig and I. Wegener. Improving the variable ordering of OBDDs in NP-complete. *IEEE Trans. on Comp.*, 45(9):993–1002, 1996.

[6] K. Brace, R. Rudell, and R. Bryant. Efficient implementation of a BDD package. In *Design Automation Conf.*, pages 40–45, 1990.

[7] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[8] Collaborative Benchmarking Laboratory. *1993 LGSynth Benchmarks*. North Carolina State University, Department of Computer Science, 1993.

[9] R. Drechsler, N. Drechsler, and W. Günther. Fast exact minimization of BDDs. *IEEE Trans. on CAD*, 19(3):384–389, 2000.

[10] R. Ebendt, W. Günther, and R. Drechsler. An improved branch and bound algorithm for exact BDD minimization. *IEEE Trans. on CAD*, 22(12):1657–1663, 2003.

[11] R. Ebendt, W. Günther, and R. Drechsler. Minimization of the expected path length in BDDs based on local changes. In *Asian and South Pacific Design Automation Conf.*, pages 866–871, 2004.

[12] R. Ebendt, W. Günther, and R. Drechsler. Combining ordered-best first search with branch and bound for exact BDD minimization. *IEEE Trans. on CAD*, 2005.

[13] B. Falkowski and C.-H. Chang. Paired Haar spectra computation through operations on disjoint cubes. In *IEEE Proceedings on Circuits, Devices and Systems*, pages 117–123, 1999.

[14] G. Fey and R. Drechsler. Minimizing the number of paths in BDDs - theory and algorithm. *IEEE Trans. on CAD*, 2005.

[15] S. Friedman and K. Supowit. Finding the optimal variable ordering for binary decision diagrams. *IEEE Trans. on Comp.*, 39(5):710–713, 1990.

[16] M. Held and R. Karp. A dynamic programming approach to sequencing problems. *J. Soc. Indust. Appl. Math.*, 10(1), 1962.

[17] Y. Iguchi, T. Sasao, and M. Matsuura. Evaluation of multiple-output logic functions using decision diagrams. In *Asian South Pacific Design Automation Conf.*, pages 312–315, 2003.

[18] Y. Y. Liu, K. H. Wang, T. T. Hwang, and C. L. Liu. Binary decision diagram with minimum expected path length. In *Design, Automation and Test in Europe*, pages 708–712, 2001.

[19] P. McGeer, K. McMillan, A. Saldanha, A. Sangiovanni-Vincentelli, and P. Scaglia. Fast discrete function evaluation using decision diagrams. In *Int'l Conf. on CAD*, pages 402–407, 1995.

[20] S. Nagayama, A. Mishchenko, T. Sasao, and J. Butler. Minimization of average path length in BDDs by variable reordering. In *Proc. of International Workshop on Logic and Synthesis*, 2003.

[21] S. Nagayama and T. Sasao. On the minimization of longest path length for decision diagrams. Proc. of International Workshop on Logic and Synthesis, 2004.

[22] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Int'l Conf. on CAD*, pages 42–47, 1993.

[23] F. Somenzi. *CU Decision Diagram Package Release 2.4.0*. University of Colorado at Boulder, 2004.

[24] M. Thornton, R. Drechsler, and D. Miller. *Spectral Techniques in VLSI CAD*. Kluwer Academic Publisher, 2001.