Efficient Approximate Decomposition Solver using Ising Model

Weihua Xiao
UM-SJTU Joint Inst.
Shanghai Jiao Tong University
Shanghai, China
019370910014@sjtu.edu.cn

Tingting Zhang
Department of Electrical & Computer Engineering
University of Alberta
Edmonton, Canada
ttzhang@ualberta.ca

Xingyue Qian
UM-SJTU Joint Inst.
Shanghai Jiao Tong University
Shanghai, China
qianxingyue@sjtu.edu.cn

Jie Han
Department of Electrical & Computer Engineering
University of Alberta
Edmonton, Canada
jhan8@ualberta.ca

Weikang Qian
UM-SJTU Joint Inst.
Shanghai Jiao Tong University
Shanghai, China
qianwk@sjtu.edu.cn

Abstract

Computing with memory is an energy-efficient computing approach. It pre-computes a function and store its values in a lookup table (LUT), which can be retrieved at runtime. Approximate Boolean decomposition reduces the LUT size for implementing complex functions, but it takes a long time to find a decomposition with a minimal error. As a parallel algorithm developed for the Ising model, simulated bifurcation (SB) shows a potential as a high-performance approach for combinatorial optimization. In this paper, we propose an efficient SB-based approximate function decomposition approach. Specifically, a new approximate disjoint decomposition method, called column-based approximate disjoint decomposition, is first proposed to fit the Ising model. Then, it is adapted to the Ising model-based optimization solver. Moreover, two improvement techniques are developed for an efficient search of the approximate disjoint decomposition when using SB. Experimental results show that compared to the state-of-the-art work, our approach achieves a 11% smaller mean error distance with an average 1.16× speedup when approximately decomposing 16-input 16-output Boolean functions.

Keywords

Approximate decomposition, Simulated bifurcation, Approximate lookup table, Ising model, Ising machine

1 Introduction

Computing with memory is one of the most effective low-power techniques for building hardware accelerators for computational-intensive applications. For this approach, the frequently used Boolean functions are first computed, and then the results are stored in lookup tables (LUTs). The Boolean function can be computed at runtime by reading the LUT depending on the inputs [1]. However, when implementing complex functions, the exponential increase of the LUT sizes with the number of input bits incurs a high hardware cost. To address this issue, a low-hardware-cost design paradigm, approximate computing, has been applied in implementing complex functions in LUTs with a reduced size at a cost of acceptable errors for error-tolerant applications [2].

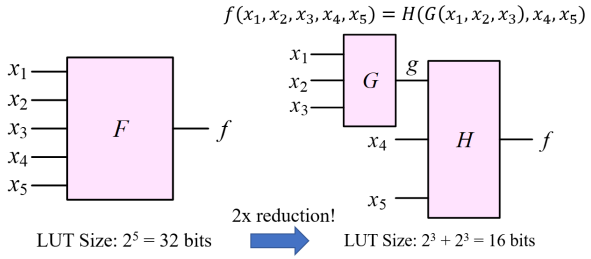

Figure 1: Reducing LUT size based on disjoint Boolean decomposition.

Existing approximate LUT designs can be categorized into three classes. The first class relies on Taylor approximation of functions [3, 4]. However, it cannot deal with non-continuous functions. The second class is based on approximate input pattern matching [5, 6]. However, additional hardware is required to obtain exact results when input pattern matching fails. The third one is developed based on *disjoint Boolean decomposition*, or *disjoint decomposition* for brevity [7]. For example, as shown in Fig. 1, a 32-bit LUT is needed to store a Boolean function f with 5 inputs. Suppose that a disjoint decomposition (details will be introduced in Section 2.2) can be applied to f, which decomposes f into two smaller functions G and H such that $f(x_1, x_2, x_3, x_4, x_5) = H(G(x_1, x_2, x_3), x_4, x_5)$. With such a decomposition, we can store f in two smaller LUTs with 16 bits in total, which leads to 2× reduction in the size of LUTs. However, the disjoint decomposition can only be applied to a Boolean function satisfying some special conditions [7]. The approximate disjoint decomposition is proposed to introduce approximation into an unsatisfying Boolean function, such that it can have a disjoint decomposition [8, 9].

In [9], a framework *DALTA* is proposed for approximate disjoint decomposition of multi-output Boolean functions. In [10], *DALTA* is improved based on the *simulated annealing* algorithm and the framework is extended to support the non-disjoint decomposition. Both frameworks rely on solving a key combinatorial optimization problem (COP) that aims at minimizing the introduced error due to approximate disjoint decomposition. Meng *et al.* formulate the COP as an integer linear programming (ILP) problem and further solve it by an ILP solver [9]. However, it suffers from a poor scalability as the solution space grows exponentially with the number of input bits

of the function, a common problem in solving general COPs [11]. To address the scalability issue, a heuristic method is proposed for solving the COP [9]. However, the method sacrifices the optimality of the solution. Thus, how to search the large solution space of this COP to obtain a good trade-off between the efficiency and solution quality remains as an open problem.

For efficient search in the solution space of COPs, *Ising model*-based solvers have recently attracted a growing interest. The Ising model mathematically emulates the energy of a physical system constructed by magnetic spins with two states, denoted as -1 and 1 [12]. The interactions among the spins and bias on the spins affect the spin states. The spin states that lead to the minimized energy of the system provide the solution of a given COP. Solving a COP using the Ising model involves two key steps: (1) Ising formulation, which maps the given COP into the Ising model, and (2) solution search, which finds the spin states by decreasing the energy of the system. For the second step, various algorithms have been developed [13], including simulated annealing [14] and *simulated bifurcation* (SB) [15]. Compared with simulated annealing, which requires the sequential update of the connected spins, the SB algorithm updates the spin states in parallel [15]. This advantage motivates research on the applications of SB in real-time optimization systems, such as stock trading systems [16], routing, and scheduling [17].

In this work, we propose a high-performance Ising model-based approximate disjoint decomposition approach to aid the design of approximate LUTs. The main contributions are as follows.

(1) A new approximate disjoint decomposition approach, referred to as *column-based approximate disjoint decomposition*, is proposed to suit the Ising model;
(2) *Ising formulations* of the column-based approximate disjoint decomposition in two decomposition modes are developed to adapt to the second-order Ising model;
(3) Two advanced strategies are developed for approximate disjoint decomposition using SB: *dynamic stop criteria* to identify if the system becomes steady by monitoring the variance of the system's energy during the search, and a heuristic to improve the search quality by pre-calculating some variable values before each update iteration of SB.

The experiment results show that compared to the state-of-the-art method [9], this approach achieves a 11% smaller mean error distance with an average 1.16× speedup when approximately decomposing 16-input 16-output Boolean functions.

In the remainder of this paper, Section 2 introduces some preliminaries. The Ising model-based approximate decomposition is discussed in Section 3. Then, the experiment results are reported in Section 4. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Ising Model and Simulated Bifurcation

The Ising model describes the energy of a physical system with N spins. The states of the N spins are denoted by a vector $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_N)$, where $\sigma_i \in \{-1, +1\}$ $(1 \le i \le N)$ denotes the state of the i-th spin. The energy of the system, denoted by $E(\boldsymbol{\sigma})$, is calculated by the following second-order polynomial using *spin variables* (by $\boldsymbol{\sigma} \in \{-1, +1\}^N$), given by [18]:

$$E(\boldsymbol{\sigma}) = -\sum_{i=1}^{N} h_i \sigma_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} J_{i,j} \sigma_i \sigma_j, \tag{1}$$

where h_i $(1 \le i \le N)$ is the bias on σ_i and $J_{i,j}$ $(1 \le i, j \le N)$ is the coefficient between σ_i and σ_j. Note that $J_{i,i} = 0$ and $J_{i,j} = J_{j,i}$.

An efficient Ising model solver is based on SB, which simulates the behavior of each spin in Ising model by an oscillator. It numerically describes the bifurcation phenomena and adiabatic processes in an oscillator network [15]. SB benefits from massive parallelism in computation, which simultaneously updates all the spins. The approximate solution, i.e., the *spin state*, is obtained by solving pairs of differential equations related to the positions and momenta of oscillators [15]. Euler integration is applied to solve the differential equations, which iteratively updates the position values [19]. After a fixed number of iterations, the spin state indicated by the sign of position values provides a solution. This paper applies the recently proposed high-performance ballistic SB (bSB) [20] to solve Ising model-based problems.

2.2 Row-based Disjoint Decomposition

Definition 1. *Let g be a Boolean function of n input bits, denoted by $\boldsymbol{X} = (x_1, \cdots, x_n)$. Let $w = \{\boldsymbol{A}, \boldsymbol{B}\}$ be a partition on $\boldsymbol{X}$, such that $\boldsymbol{A} \cup \boldsymbol{B} = \boldsymbol{X}$ and $\boldsymbol{A} \cap \boldsymbol{B} = \emptyset$. For simplicity, we call w the input partition. The function is said to have a disjoint decomposition over the input partition $w = \{\boldsymbol{A}, \boldsymbol{B}\}$ if there exist functions ϕ and F, such that $g(\boldsymbol{X}) = F(\phi(\boldsymbol{B}), \boldsymbol{A})$, where $\boldsymbol{A}$ is called the free set and $\boldsymbol{B}$ is called the bound set.*

In [7], a necessary and sufficient condition is proposed for a Boolean function $g(\boldsymbol{X})$ to have a disjoint decomposition over a given input partition $w = \{\boldsymbol{A}, \boldsymbol{B}\}$. The condition is proposed based on a matrix representation of the Boolean function's truth table under a given input partition, called the *Boolean matrix*, which uses the variables in $\boldsymbol{A}$ (*resp.* $\boldsymbol{B}$) to define the rows (*resp.* columns). The condition is as follows [7]:

Theorem 1. (Row-based Decomposition) *A Boolean function f has a disjoint decomposition over an input partition $\{A, B\}$, if and only if the corresponding Boolean matrix has at most the following four distinct types of rows: 1) a pattern of all* 0*s; 2) a pattern of all* 1*s; 3) a fixed pattern $\boldsymbol{V}$ of* 0*'s and* 1*'s; 4) the complement of $\boldsymbol{V}$.*

To find a disjoint decomposition of a Boolean function $g(\boldsymbol{X})$ is equivalent to determining three factors [9, 10]: 1) the input partition $w = \{\boldsymbol{A}, \boldsymbol{B}\}$; 2) the fixed row pattern $\boldsymbol{V}$, which consists of $2^{|\boldsymbol{B}|}$ bits; 3) the row type vector $\boldsymbol{S}$, which has $2^{|\boldsymbol{A}|}$ elements, and each element $S_i \in \{1, 2, 3, 4\}$ $(1 \le i \le 2^{|\boldsymbol{A}|})$ represents the type of the i-th row in the corresponding Boolean matrix. We denote the tuple $(w, \boldsymbol{V}, \boldsymbol{S})$ as the setting of a row-based disjoint decomposition. The functions ϕ and F can be obtained from the vectors $\boldsymbol{V}$ and $\boldsymbol{S}$, respectively.

Example 1. *Given an input partition with $\boldsymbol{A} = \{x_1, x_2\}$ and $\boldsymbol{B} = \{x_3, x_4\}$, the corresponding Boolean matrix of Boolean function f is shown in Fig. 2. It has four different types of rows with $\boldsymbol{V} = (1, 1, 0, 0)$ and $\boldsymbol{S} = (3, 1, 2, 4)$. Thus, its corresponding Boolean function has a disjoint decomposition with free set $\{x_1, x_2\}$ and bound set $\{x_3, x_4\}$. The truth table of function ϕ is given by $\boldsymbol{V}$ and thus $\phi(x_3, x_4) = \bar{x_3}$. The function F is derived from $\boldsymbol{S}$ as $F(\phi(x_3, x_4), x_1, x_2) = \phi\bar{x}_1\bar{x}_2 + x_1\bar{x}_2 + \bar{\phi}x_1x_2$.*

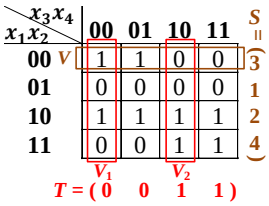

Figure 2: A Boolean matrix of a Boolean function with a disjoint decomposition.

2.3 Row-based Approximate Disjoint Decomposition

However, the conditions in Theorem 1 is hard to be satisfied for an arbitrary Boolean function $g(\boldsymbol{X})$ over a given input partition w. To tackle the unsatisfactory case, researchers approximate $g(\boldsymbol{X})$ into $\hat{g}(\boldsymbol{X})$, such that $\hat{g}(\boldsymbol{X})$ has a disjoint decomposition [8–10]. More generally, a multi-output Boolean function $\boldsymbol{G}(\boldsymbol{X}) = (g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X}))$ can be approximated into $\hat{\boldsymbol{G}}(\boldsymbol{X}) = (\hat{g}_1(\boldsymbol{X}), \ldots, \hat{g}_m(\boldsymbol{X}))$, such that each *component function* $\hat{g}_k(\boldsymbol{X})$ $(1 \le k \le m)$ has a disjoint decomposition. Thus, we have to determine the setting $(w_k, \boldsymbol{V}_k, \boldsymbol{S}_k)$ for each function $g_k(\boldsymbol{X})$. However, different settings correspond to different approximations of Boolean functions, leading to different errors. The approximate disjoint decomposition is to derive the optimal setting for each component function to minimize the overall error. In this work, two types of metrics are used to measure the error: *error rate* (*ER*), which is the probability that an input pattern produces a wrong output for the approximate function, and *mean error distance* (*MED*), which measures the average deviation of the output binary encoding, calculated as:

$$MED(\boldsymbol{G}, \hat{\boldsymbol{G}}) = \sum_{\boldsymbol{X}} p_{\boldsymbol{X}} |Bin(\boldsymbol{G}(\boldsymbol{X})) - Bin(\hat{\boldsymbol{G}}(\boldsymbol{X}))|, \tag{2}$$

where $p_{\boldsymbol{X}}$ is the occurrence probability of input pattern $\boldsymbol{X}$, and $Bin(\boldsymbol{W})$ is the binary number encoded by the Boolean vector $\boldsymbol{W}$.

2.4 Core COP of Row-based Approximate Disjoint Decomposition

We focus on the more general multi-output Boolean functions in this paper. In [9], a framework *DALTA* is proposed for the approximate disjoint decomposition over multi-output Boolean functions, on which our proposed approximate decomposition solver is based. Instead of simultaneously deriving the optimal settings $(w_k, \boldsymbol{V}_k, \boldsymbol{S}_k)$'s for all component functions g_k's, it optimizes the setting of each individually. This process is carried out sequentially, starting from the most significant bit to the least significant bit, and it is repeated for R rounds. However, the searching space of the setting $(w_k, \boldsymbol{V}_k, \boldsymbol{S}_k)$ is large, which is hard to be optimized directly. *DALTA* randomly generates P candidate partitions w_k's and then optimizes $\boldsymbol{V}_k$ and $\boldsymbol{S}_k$ for each partition to minimize the introduced error. Thus, the core problem of the framework is a COP that optimizes $\boldsymbol{V}_k$ and $\boldsymbol{S}_k$ given a partition w_k, called *row-based core COP*. There exist two modes for solving row-based core COP:

- Separate mode: finding a setting $(\boldsymbol{V}_k, \boldsymbol{S}_k)$ to minimize the introduced ER for the current component function;
- Joint mode: finding a setting $(\boldsymbol{V}_k, \boldsymbol{S}_k)$ to minimize the introduced MED of all Boolean functions, which is computed by fixing all the other Boolean functions to their accurate versions if they have not been optimized yet in the first round, or their latest approximated versions otherwise.

3 Ising Model-based Approximate Disjoint Decomposition

In this section, we develop an Ising model-based solver for the core COP in the approximate disjoint decomposition. Instead of the row-based approximate disjoint decomposition, we propose a column-based approximate disjoint decomposition in Section 3.1, which is more friendly to the Ising model. Moreover, we introduce the *column-based core COP*. Section 3.2 then presents how to map the column-based core COP into an Ising model. For efficiently solving the COP by bSB, Section 3.3 proposes two improvement techniques.

3.1 Column-based Approximate Disjoint Decomposition

In [9], the *row-based core COP* is formulated as a constrained ILP problem. However, this formulation has a bad scalability. To solve the above issue, we resort to Ising model-based solver. However, our study finds that to solve a row-based core COP by an Ising model requires a third-order Ising model, which is more complex than the second-order Ising model shown in Eq. (1). To address this issue, we resort to another necessary and sufficient condition for disjoint decomposition as follows [7]:

Theorem 2. (Column-based Decomposition) *Let $\{\boldsymbol{A}, \boldsymbol{B}\}$ be a partition on $\boldsymbol{X}$. A Boolean function f has a disjoint decomposition over $\{\boldsymbol{A}, \boldsymbol{B}\}$, if and only if its Boolean matrix with variables in sets $\boldsymbol{A}$ and $\boldsymbol{B}$ defining rows and columns, respectively, has at most two types of columns.*

For example, the Boolean matrix in Fig. 2, whose corresponding Boolean function has a disjoint decomposition shown in Example 1, has two types of columns, $(1, 0, 1, 0)$ and $(0, 0, 1, 1)$.

Based on Theorem 2, we propose a column-based approximate disjoint decomposition. Consider a multi-output Boolean function, and assume that the k-th $(1 \le k \le m)$ component function g_k is represented by a Boolean matrix with $r = 2^{|A|}$ rows and $c = 2^{|B|}$ columns. Denote the exact (*resp.* approximate) value at the i-th $(1 \le i \le r)$ row and the j-th $(1 \le j \le c)$ column in the Boolean matrix of the k-th component function as O_{kij} (*resp.* $\hat{O}_{kij}$).

To find a column-based disjoint decomposition for component function g_k is equivalent to determining a setting represented as $(w_k, \boldsymbol{V}_{k1}, \boldsymbol{V}_{k2}, \boldsymbol{T}_k)$, where w_k is an input partition, $\boldsymbol{V}_{k1} = (V_{k11}, \ldots, V_{k1r}) \in \{0, 1\}^r$ is the column pattern 1, $\boldsymbol{V}_{k2} = (V_{k21}, \ldots, V_{k2r}) \in \{0, 1\}^r$ is the column pattern 2, and $\boldsymbol{T}_k = (T_{k1}, \ldots, T_{kc}) \in \{0, 1\}^c$ is the column type vector. Note that $T_{kj} = 0$ (*resp.* 1) means that the j-th column equals column pattern 1 (*resp.* 2). For example, for the Boolean matrix in Fig. 2, we have $\boldsymbol{V}_{k1} = (1, 0, 1, 0)$, $\boldsymbol{V}_{k2} = (0, 0, 1, 1)$, and $\boldsymbol{T}_k = (0, 0, 1, 1)$.

For the column-based approximate disjoint decomposition, our target is to find an optimized setting $(w_k, \boldsymbol{V}_{k1}, \boldsymbol{V}_{k2}, \boldsymbol{T}_k)$ for g_k to minimize the total error. The corresponding *column-based core COP* is to optimize $\boldsymbol{V}_{k1}$, $\boldsymbol{V}_{k2}$, and $\boldsymbol{T}_k$ under a given input partition w_k. Clearly, the column-based core COP has $(2r + c)$ binary variables.

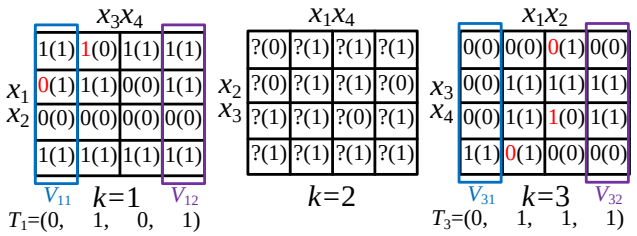

Figure 3: An example of the approximate disjoint decomposition for a 3-output Boolean function. A value inside (*resp.* outside) parentheses represents the exact (*resp.* approximate) value of the Boolean function.

The approximate value $\hat{O}_{kij}$ satisfies that

$$\hat{O}_{kij} = (1 - T_{kj})V_{k1i} + T_{kj}V_{k2i}. \tag{3}$$

Example 2. *In Fig. 3, it shows an example of the column-based disjoint decomposition for a 3-output Boolean function. The column-based decomposition settings of the first and third component functions have been determined. For example, w_1 of the first component function is $\{\{x_1, x_2\}, \{x_3, x_4\}\}$. In its corresponding Boolean matrix (leftmost), a value inside (resp. outside) parentheses at the i-th row and j-th column represents the exact (resp. approximate) value of the component function under the corresponding input pattern, i.e., O_{kij} (resp. $\hat{O}_{kij}$). Note that an $\hat{O}_{kij}$ is marked in red if $\hat{O}_{kij} \neq O_{kij}$. Its two column pattern vectors are $V_{11} = (1, 0, 0, 1)$ and $V_{12} = (1, 1, 0, 1)$, and the corresponding type vector is $T_1 = (0, 1, 0, 1)$. Similarly, V_{31}, V_{32}, and T_3 are shown for the third component function under the input partition $w_3 = \{\{x_3, x_4\}, \{x_1, x_2\}\}$. The approximate values are unknown of the second component function, marked by ?. We have to determine V_{21}, V_{22}, and T_2 under the partition $w_2 = \{\{x_2, x_3\}, \{x_1, x_4\}\}$.*

3.2 Ising Formulation of Column-based Core COP

This section shows the formulation of the column-based core COP as an Ising model. Sections 3.2.1 and 3.2.2 consider separate-mode and joint-mode approximate decompositions, respectively.

3.2.1 Ising Formulation of Column-based Core COP under the Separate Mode Under the separate mode, each of the m component functions is treated separately. Consequently, the target is to minimize the ER of the approximate disjoint decomposition for each function. Consider the approximate decomposition of the k-th component function as an example. Denote the probability of the input pattern corresponding to the i-th row and the j-th column in the Boolean matrix for the k-th component function as p_{kij}. The objective is to minimize the ER [9]:

$$\min \sum_{i=1}^{r} \sum_{j=1}^{c} p_{kij} |\hat{O}_{kij} - O_{kij}|, \tag{4}$$

where $\hat{O}_{kij}$ is given in Eq. (3).

To solve the above COP using the Ising model, Eq. (4) need to fit in the formulation as in Eq. (1). Let $ED_{kij} = |\hat{O}_{kij} - O_{kij}|$, where $1 \leq k \leq m$, $1 \leq i \leq r$, and $1 \leq j \leq c$. We first transform ED_{kij} to avoid the use of absolute operation. Since $\hat{O}_{kij}$ and the known O_{kij} are either 0 or 1, we have

$$ED_{kij} = \begin{cases} \hat{O}_{kij} & \text{if } O_{kij} = 0, \\ 1 - \hat{O}_{kij} & \text{if } O_{kij} = 1. \end{cases} \tag{5}$$

Thus, ED_{kij} can be rewritten as

$$ED_{kij} = (1 - O_{kij})\hat{O}_{kij} + O_{kij}(1 - \hat{O}_{kij}). \tag{6}$$

Hence, by replacing $|\hat{O}_{kij} - O_{kij}|$ in Eq. (4) by ED_{kij} in Eq. (6), Eq. (4) is rewritten as

$$\min \sum_{i=1}^{r} \sum_{j=1}^{c} p_{kij}(O_{kij} + (1 - 2O_{kij})\hat{O}_{kij}). \tag{7}$$

Note that the only unknowns in Eq. (7) are $\hat{O}_{kij}$, which, by Eq. (3), depends on variables T_{kj}, V_{k1i}, and $V_{k2i} \in \{0, 1\}$. In order to formulate the minimization problem as an Ising model, the binary variables T_{kj}, V_{k1i}, and V_{k2i} are converted to the *spin variables* $\bar{T}_{kj}$, $\bar{V}_{k1i}$, and $\bar{V}_{k2i}$ in $\{-1, +1\}$ using linear transformation, satisfying $T_{kj} = \frac{\bar{T}_{kj}+1}{2}$, $V_{k1i} = \frac{\bar{V}_{k1i}+1}{2}$, and $V_{k2i} = \frac{\bar{V}_{k2i}+1}{2}$. Then, Eq. (3) is reformulated using spin variables as

$$\hat{O}_{kij} = \frac{1}{2} + \frac{\bar{V}_{k1i}+\bar{V}_{k2i}-\bar{T}_{kj}\bar{V}_{k1i}+\bar{T}_{kj}\bar{V}_{k2i}}{4}. \tag{8}$$

In total, $N = 2r+c$ spin variables are required for each component function. Then, with the constant terms omitted, the COP in Eq. (7) is expressed by a second-order Ising formulation using the spin variables in Eq. (8) as follows:

$$\begin{aligned} & E(\{\bar{T}_{kj}\}, \{\bar{V}_{k1i}\}, \{\bar{V}_{k2i}\}) \\ &= \frac{\sum_{i=1}^{r}\sum_{j=1}^{c} p_{kij}(1-2O_{kij})(\bar{V}_{k1i}+\bar{V}_{k2i}-\bar{T}_{kj}\bar{V}_{k1i}+\bar{T}_{kj}\bar{V}_{k2i})}{4} \\ &= \sum_{i=1}^{r}\left(\sum_{j=1}^{c} \frac{p_{kij}(1-2O_{kij})}{4}\right)\bar{V}_{k1i} + \sum_{i=1}^{r}\left(\sum_{j=1}^{c} \frac{p_{kij}(1-2O_{kij})}{4}\right)\bar{V}_{k2i} \\ &- \sum_{i=1}^{r}\sum_{j=1}^{c} \frac{p_{kij}(1-2O_{kij})}{4}\bar{T}_{kj}\bar{V}_{k1i} + \sum_{i=1}^{r}\sum_{j=1}^{c} \frac{p_{kij}(1-2O_{kij})}{4}\bar{T}_{kj}\bar{V}_{k2i}. \end{aligned} \tag{9}$$

3.2.2 Ising Formulation of Column-based Core COP under the Joint Mode The separate mode ignores the different significance of output bits and hence, may cause a large error. To reduce the error, the joint-mode decomposition is proposed in [9], which considers the different significance of the component functions. This section discusses the more general cases except for the first round, i.e., all the component functions have been approximately decomposed. The treatment for the first round is similar with some changes to the coefficients.

Assume that the current optimization is performed on the Boolean matrix for the k-th component function. Since inputs can be partitioned in different ways among different component functions, for the l-th ($l \neq k$) component function, we use i_l and j_l to index the known Boolean value ($\hat{O}_{li_lj_l}$) for the input pattern corresponding to the i-th row and j-th column in the Boolean matrix for the k-th component function. The objective is to minimize the MED between the exact and approximate m-bit outputs as follows [9]:

$$\min \sum_{i=1}^{r} \sum_{j=1}^{c} p_{kij} ED_{kij}, \tag{10}$$

where

$$ED_{kij} = |2^{k-1}\hat{O}_{kij} + \sum_{l=1,l\neq k}^{m} 2^{l-1}\hat{O}_{li_lj_l} - \sum_{l=1}^{m} 2^{l-1}O_{li_lj_l}|, \tag{11}$$

where $\hat{O}_{kij}$'s are the only unknowns, which are given by Eq. (3).

Next, we use an example to illustrate how to compute ED_{kij}, where $1 \leq k \leq m$, $1 \leq i \leq r$, and $1 \leq j \leq c$.

Example 3. *Consider the example shown in Fig. 3. We aim at computing ED_{213} in the second Boolean matrix in Fig. 3. The corresponding input pattern is $(x_1, x_2, x_3, x_4) = (1, 0, 0, 0)$. According to the input pattern, we can derive $(i_1, j_1) = (3, 1)$ and $(i_3, j_3) = (1, 3)$. Thus, $ED_{213} = |2\hat{O}_{213} + (1 \cdot 0 + 4 \cdot 0) - (1 \cdot 0 + 2 \cdot 1 + 4 \cdot 1)| = |2\hat{O}_{213} - 6|$.*

Let $D_{kij} = \sum_{l=1,l\neq k}^{m} 2^{l-1}\hat{O}_{li_1j_1} - \sum_{l=1}^{m} 2^{l-1}O_{li_1j_1}$. Then, $ED_{kij} = |2^{k-1}\hat{O}_{kij} + D_{kij}|$. Similarly, in order to adapt the Ising formulation, ED_{kij} needs to be further simplified to avoid the use of absolute operation. We distinguish two cases on ED_{kij} as follows:

- When $-2^{k-1} \leq D_{kij} \leq 0$, we have

$$ED_{kij} = \begin{cases} -D_{kij}, & \text{if } \hat{O}_{kij} = 0, \\ 2^{k-1}\hat{O}_{kij} + D_{kij}, & \text{if } \hat{O}_{kij} = 1. \end{cases}$$

  Thus, we have

$$ED_{kij} = (2^{k-1}\hat{O}_{kij} + D_{kij})\hat{O}_{kij} - D_{kij}(1 - \hat{O}_{kij}). \tag{12}$$

  Since $\hat{O}_{kij}$ is a binary value, we have $\hat{O}_{kij}^2 = \hat{O}_{kij}$. Thus Eq. (12) can be further rewritten as

$$ED_{kij} = (2^{k-1} + 2D_{kij})\hat{O}_{kij} - D_{kij}. \tag{13}$$

- When $D_{kij} < -2^{k-1}$ or $D_{kij} > 0$, we have

$$ED_{kij} = \begin{cases} 2^{k-1}\hat{O}_{kij} + D_{kij} & D_{kij} > 0 \\ -2^{k-1}\hat{O}_{kij} - D_{kij} & D_{kij} < -2^{k-1} \end{cases}. \tag{14}$$

  Equivalently, we have

$$ED_{kij} = 2^{k-1}sgn(D_{kij})\hat{O}_{kij} + D_{kij}sgn(D_{kij}). \tag{15}$$

Thus, based on different values of D_{kij}, which is known, we can replace each ED_{kij} in Eq. (10) by either Eq. (13) or Eq. (15), which are linear equations on $\hat{O}_{kij}$. We further replace each $\hat{O}_{kij}$ by Eq. (8) and drop the constant terms. Finally, we can rewrite Eq. (10) as the following second-order Ising formulation with spin variables $\bar{T}_{kj}$, $\bar{V}_{k1i}$, and $\bar{V}_{k2i}$:

$$\begin{aligned} &E(\{\bar{T}_{kj}\}, \{\bar{V}_{k1i}\}, \{\bar{V}_{k2i}\}) \\ &= \frac{\sum_{i=1}^{r}\sum_{j=1}^{c} p_{kij}q_{kij}(\bar{V}_{k1i}+\bar{V}_{k2i}-\bar{T}_{kj}\bar{V}_{k1i}+\bar{T}_{kj}\bar{V}_{k2i})}{4} \\ &= \sum_{i=1}^{r}(\sum_{j=1}^{c}\frac{p_{kij}q_{kij}}{4})\bar{V}_{k1i} + \sum_{i=1}^{r}(\sum_{j=1}^{c}\frac{p_{kij}q_{kij}}{4})\bar{V}_{k2i} \\ &- \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{p_{kij}q_{kij}}{4}\bar{T}_{kj}\bar{V}_{k1i} + \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{p_{kij}q_{kij}}{4}\bar{T}_{kj}\bar{V}_{k2i}, \end{aligned} \tag{16}$$

where q_{kij} equals $(2^{k-1} + 2D_{kij})$ if $-2^{k-1} \leq D_{kij} \leq 0$, and $2^{k-1}sgn(D_{kij})$ otherwise.

3.3 Improvement Strategies for bSB-based Approximate Disjoint Decomposition

3.3.1 Dynamic Stop Criteria When using bSB to solve a COP, a question is how many iterations the Euler integration requires. The most common way is to choose a fixed number of iterations, as described in Section 2.1. However, since there is no explicit relationship found between solution convergence and the number of iterations, we cannot guarantee that each spin has reached its steady state after Euler integration using a given number of iterations. Therefore, a dynamic stop approach is proposed to instruct when to stop the Euler integration in bSB. It has two steps:

(1) Sample the energy for every f iterations, and at each sampling time, compute the variance on the last s sampled energies.
(2) Stop the Euler integration of bSB if the variance value is smaller than a predefined threshold ϵ; otherwise, continue the integration.

3.3.2 A Heuristic to Intervene State Update of bSB This section proposes a heuristic to intervene the state update of parts of spins. Its purpose is to improve the results of bSB. Before introducing the heuristic, we first have the following straightforward claim on solving a variant of the column-based core COP with given $\boldsymbol{V}_{k1}$ and $\boldsymbol{V}_{k2}$.

Theorem 3. *When solving a column-based core COP with given $\boldsymbol{V}_{k1}$ and $\boldsymbol{V}_{k2}$, the optimal $\boldsymbol{T}_k$ should satisfy that for each column, the column pattern vector with a smaller error is selected.*

The proposed heuristic is based on an observation that the column type vector $\boldsymbol{T}_k$ is often not optimal corresponding to the column pattern vectors in each sampled solution. Our heuristic is to reset the column type vector in each sampled solution as the optimal one given by Theorem 3, which is subsequently fed back to bSB for further optimization.

4 Experimental Results

This section presents the experimental results. The approximate disjoint decomposition methods are implemented in C++. The matrix and vector computation in the bSB solver are implemented using Eigen library in C++ [21]. We use Gurobi [22] as the ILP solver. All the experiments are conducted on a computer with a 16-core 1.9GHz AMD Ryzen 7 5800U processor and 16GB RAM.

The performance of approximate joint decomposition are evaluated over benchmarks in [9], including six continuous functions ($cos(x)$, $tan(x)$, $exp(x)$, $ln(x)$, $erf(x)$, $denoise(x)$) and four non-continuous functions for arithmetic circuits from AxBench [23] (Brent-Kung, Forwardk2j, Inversek2j, Multiplier). We consider two quantization schemes: (1) the number of inputs $n = 9$, the size of the free set as 4, and the size of the bound set as 5; (2) the number of inputs $n = 16$, the size of the free set as 7, and the size of the bound set as 9. The performance of different approaches is measured by MED and runtime.

The proposed method is compared to the state-of-the-art method in [9]. In that method, the number of tried input partitions is limited by $P = 1000$. The number of the iteration round is set as $R = 5$. For the ILP solver Gurobi, we set the runtime bound of solving a single ILP problem as 3600s. If the runtime reaches the bound, Gurobi returns the current best solution.

For our proposed Ising model-based method, the parameters for the stop criterion in Section 3.3.1 are set as $f = 20$ and $s = 20$ when $n = 9$, and $f = 10$ and $s = 10$ when $n = 16$. The threshold for energy variance is set as $\epsilon = 10^{-8}$.

4.1 Performance on Small-Scale Cases

We compare the performance of different approximate disjoint decomposition approaches on six continuous functions for separate and joint modes as shown in Table 1. The first quantization scheme is applied, i.e., $n = 9$. The number of outputs is $m = 9$. For the case of 9-input Boolean functions, the ILP-based method in [9], denoted as *DALTA-ILP*, can be applied for both separate and joint modes. The proposed Ising model-based method is compared with *DALTA-ILP* for separate and joint modes. Moreover, this work is also compared with the two heuristic methods for the joint mode, denoted as *DALTA* [9] and *BA* [10].

Table 1: Comparison of approximate disjoint decomposition using different methods for separate and joint modes with the number of inputs $n = 9$.

Benchmarks			Separate Mode				Joint Mode							
Function	Domain	Range	*DALTA-ILP* [9]		Prop.		*DALTA* [9]		*DALTA-ILP* [9]		*BA* [10]		Prop.	
			MED	Time(s)	MED	Time(s)	MED	Time(s)	MED	Time(s)	MED	Time(s)	MED	Time(s)
$cos(x)$	$[0, \frac{\pi}{2}]$	$[0, 1]$	11.64	258.37	8.33	0.56	2.96	3.06	2.48	3600	2.46	1.54	2.5	1.75
$tan(x)$	$[0, \frac{2\pi}{5}]$	$[0, 3.08]$	10.91	236.32	10.45	0.56	3.24	2.83	2.62		2.84	1.57	2.5	1.87
$exp(x)$	$[0, 3]$	$[0, 20.09]$	9.26	242.58	7.07	0.74	4.22	2.72	3.55		3.01	1.5	2.66	1.92
$ln(x)$	$[1, 10]$	$[0, 2.30]$	8.32	224.68	6.57	0.49	4.69	6.77	2.55		2.9	1.49	2.72	2.77
$erf(x)$	$[0, 3]$	$[0, 1]$	5.07	139.6	4.61	0.42	1.85	2.76	2.66		2.66	1.38	1.9	1.55
$denoise(x)$	$[0, 3]$	$[0, 0.81]$	10.91	229.25	9.69	0.46	4.75	2.81	3.38		4.27	1.51	2.8	1.51
Average of MED and Time			9.35	221.8	**7.83**	**0.53**	3.61	3.49	2.87	3600	3.02	**1.49**	**2.51**	1.89

The smallest average MED and the shortest average time are highlighted for separate and joint modes.

Since the separate mode ignores the significance of different Boolean functions, the approximate decomposition obtained under the separate mode is less accurate than that found under the joint mode. Under the separate mode, the proposed Ising model-based approach uses less than 0.6s to find an approximate decomposition with a MED less than 11. Compared with *DALTA-ILP*, our approach shows advantages in both accuracy and runtime. Using approximately 418× shorter time, it can find a solution with a 16% improvement in MED. For the joint mode, compared to *DALTA*, *DALTA-ILP* can find a better approximate decomposition with an average 20% decrease in MED but at a significant increase in runtime. Using *BA* approach achieves an average 16% smaller MED in a short time than using *DALTA*. However, the MED is noncompetitive to that obtained from using *DALTA-ILP*. The experimental results for all six benchmarks show that our approach outperforms the existing ones in MED. It can lead to a 12% smaller MED than *DALTA-ILP* on average. Although a relatively longer runtime is required compared with *BA*, the accuracy issue is solved, which is rather difficult for other approaches.

4.2 Performance on Large-Scale Cases

In this experiment, the inputs are quantized to 16 bits ($n = 16$) for all the benchmarks. The outputs for continuous functions are quantized to 16 bits ($m = 16$), and the outputs for non-continuous functions are adjusted accordingly ($m = 9$ for Brent-Kung and $m = 16$ for others). In this section, we only compare the proposed Ising model-based method over the joint mode with *DALTA*, as the final error will be large if using the separate mode for the 16-input Boolean functions. In the framework of *BA*, it will generate input partitions based on simulated annealing, different from those of *DALTA*. Thus, for fairness, we do not compare with *BA* in this section.

Fig. 4 plots the ratios of MEDs obtained from the Ising model-based method to those obtained from *DALTA* and also the ratios of runtimes of the former to those of the latter and also gives the MEDs and runtime of using *DALTA* as the baseline. A ratio less than 1 means that the Ising model-based method is better. As revealed in the figure, our method achieves an improvement in both MED and runtime for seven out of ten benchmarks. An average of 11% smaller MED is obtained with an average 1.16× speedup in runtime. These results indicate that the proposed Ising model-based approximate decomposition approach has a better performance.

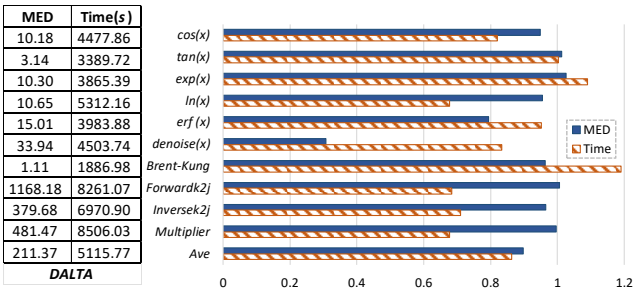

Figure 4: The performance of Ising model-based approximate disjoint decomposition vs. *DALTA* [9] for joint mode with the number of inputs $n = 16$.

5 Conclusion

In this paper, an efficient Ising model-based approximate disjoint decomposition approach is developed to aid the design of low-cost approximate LUTs. A column-based approximate disjoint decomposition approach is first proposed, which is mathematically friendly to a second-order Ising model-based COP solver. Then, how to avoid the use of absolute operations in the formulations is investigated for better adapting to the solver. Two improvement techniques are proposed to guide the update of spin states during the search for solutions using an SB-based Ising solver for approximate disjoint decomposition. The dynamic stop criteria dynamically determine the end of the search by identifying if the system becomes steady. For quality improvement, a heuristic is introduced to intervene the search by determining some variable values before the update. Extensive experiments on 16-bit Boolean functions show that the proposed Ising model-based solver outperforms the state-of-the-art methods in both accuracy and runtime.

Acknowledgment

This work at the University of Alberta was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Project Numbers: RES0048688, RES0051374 and RES0054326).

References

[1] J. Cong *et al.*, "Energy-efficient computing using adaptive table lookup based on nonvolatile memories," in *ISLPED*. IEEE, 2013, pp. 280–285.
[2] Q. Xu *et al.*, "Approximate computing: A survey," *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, 2016.
[3] M. J. Schulte and J. E. Stine, "Symmetric bipartite tables for accurate function approximation," in *ARITH*. IEEE, 1997, pp. 175–183.

[4] S.-F. Hsiao *et al.*, "Hierarchical multipartite function evaluation," *IEEE TC*, vol. 66, no. 1, pp. 89–99, 2016.
[5] M. Imani *et al.*, "Resistive configurable associative memory for approximate computing," in *DATE*. IEEE, 2016, pp. 1327–1332.
[6] A. Rahimi *et al.*, "Approximate associative memristive memory for energy-efficient GPUs," in *DATE*. IEEE, 2015, pp. 1497–1502.
[7] V.-S. Shen and A. C. Mckellar, "An algorithm for the disjunctive decomposition of switching functions," *IEEE TC*, vol. 100, no. 3, pp. 239–248, 1970.
[8] Y. Yao *et al.*, "Approximate disjoint bi-decomposition and its application to approximate logic synthesis," in *ICCD*, 2017, pp. 517–524.
[9] C. Meng *et al.*, "DALTA: A decomposition-based approximate lookup table architecture," in *ICCAD*. IEEE, 2021, pp. 1–8.
[10] X. Qian *et al.*, "High-accuracy low-power reconfigurable architectures for decomposition-based approximate lookup table," in *DATE*, 2023, pp. 1–6.
[11] B. H. Korte *et al.*, *Combinatorial optimization*. Springer, 2011, vol. 1.
[12] B. A. Cipra, "An introduction to the Ising model," *Am. Math. Mon.*, vol. 94, no. 10, pp. 937–959, 1987.
[13] T. Zhang *et al.*, "A review of simulation algorithms of classical Ising machines for combinatorial optimization," in *ISCAS*. IEEE, 2022, pp. 1877–1881.
[14] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *Journal of statistical physics*, vol. 34, no. 5, pp. 975–986, 1984.
[15] H. Goto *et al.*, "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems," *Sci. Adv.*, vol. 5, no. 4, p. eaav2372, 2019.
[16] K. Tatsumura *et al.*, "Real-time trading system based on selections of potentially profitable, uncorrelated, and balanced stocks by NP-hard combinatorial optimization," *arXiv preprint arXiv:2307.06339*, 2023.
[17] T. Zhang and J. Han, "Efficient traveling salesman problem solvers using the Ising model with simulated bifurcation," in *DATE*. IEEE, 2022, pp. 548–551.
[18] A. Lucas, "Ising formulations of many NP problems," *Front. Phys.*, 2014.
[19] T. Kanao and H. Goto, "Simulated bifurcation for higher-order cost functions," *Applied Physics Express*, vol. 16, no. 1, p. 014501, 2022.
[20] H. Goto *et al.*, "High-performance combinatorial optimization based on classical mechanics," *Sci. Adv.*, vol. 7, no. 6, p. eabe7953, 2021.
[21] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.
[22] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com
[23] A. Yazdanbakhsh *et al.*, "AxBench: A multiplatform benchmark suite for approximate computing," *IEEE IEEE Des. Test*, vol. 34, no. 2, pp. 60–68, 2016.