2015 International Symposium on Nonlinear Theory and its Applications
NOLTA2015, Kowloon, Hong Kong, China, December 1-4, 2015

Applications of Decision Diagrams in Information Networking

Takeru Inoue†

†NTT Network Innovation Laboratories, Japan
Email: takeru.inoue@ieee.org

Abstract—BDD, which is a compressed data structure used to represent a Boolean function compactly, has been widely applied to a variety of problems in information networking research: e.g., reliability evaluation, network optimization, configuration verification, packet classification, and so on. This paper, for the first time ever, presents a taxonomy of BDD applications and provides a basic framework of its use.

1. Introduction

Information networking technologies have been achieving tremendous growth as an indispensable infrastructure in our society. The technologies have been studied from a wide variety of perspectives such as optimization techniques on network design, verification methods on network configurations, efficient packet manipulation algorithms and so on. Recently, software-defined networking [1], which is a new paradigm to resuscitate rich and centralized management schemes, has asked researchers to investigate more sophisticated solutions for networking. In this research trend, a binary decision diagram or BDD [2,3] has been taking an important position in many topics: e.g., reliability evaluation [4,5], network optimization [6], packet classification [7], configuration verification [8], traffic measurement [9], policy enforcement [10] and publish/subscribe systems [11].

The reason why BDD has been used for such various applications is considered as follows: BDD is a powerful weapon to manipulate arbitrary discrete data. BDD is a data structure designed to represent a Boolean function, $f(\boldsymbol{x}) \in \{\top, \bot\}$, where $\boldsymbol{x} = x_1x_2\ldots$ is a vector of Boolean variables, and $\top$ and $\bot$ are true and false respectively. In BDD, a Boolean function is stored in a highly compressed manner, and arbitrary logic operations can be performed very efficiently without decompression; e.g., given two Boolean functions, f and g, AND of them, $f \wedge g$, can be obtained readiy. BDD was originally invented for VLSI logic design, but Boolean functions can represent several types of discrete data; e.g., for a family of sets, x_i can be considered as the existence of i-th set element. Although BDD was invented in 1986 [2], almost thirty years ago, there are still many interesting and exciting research topics related to it [12].

Despite of the growing popularity of BDDs in the networking research, there has been no literature that describes the application framework of BDDs to networking problems, to the best of our knowledge. This paper thoroughly discusses the use of BDDs in the networking research for the first time ever. There are two major streams in applying BDDs to the problems found in networking.

- **BDD as network graph**. A network, or a graph representing the network, is considered as a set of edges. A BDD is used to represent a family of subgraphs (or a family of edge subsets) that satisfy a given constraint. The BDD, therefore, maintains a feasible solution space, in which the optimal solution could be searched for or the feasible probability could be summed up. Since BDD itself cannot understand graph-theoretic properties such as loop-free nature or connectedness, BDDs used to be an unsuitable option to handle graph models. Recently, a novel algorithm that efficiently builds a BDD of given graph properties has been developed [13], and this stream now becomes an attractive approach to the networking community. Example applications of this stream include the network reliability evaluation and the network configuration optimization.
- **BDD as packet bits**. In the second stream, a packet, or a sequence of bits, is considered as a Boolean vector. A BDD represents a Boolean function of the bits, and is often used to test whether a given packet is matched to a condition. Since BDD only identifies binary states, i.e., $\top$ or $\bot$, its application was limited to firewall analysis [14], in which packets are classified into two classes, "accept" or "drop". Lately, BDDs have been successfully extended to represent multi-valued functions [7], and they are now able to classify packets based on several actions, such as next-hops or labeled paths. Example applications of this stream include packet classification and configuration verification.

2. BDD and Accompanying Techniques

A BDD is a graphical representation of a Boolean function. Fig. 1 shows examples of BDD. As shown in Fig. 1, a BDD is an acyclic directed graph with a single

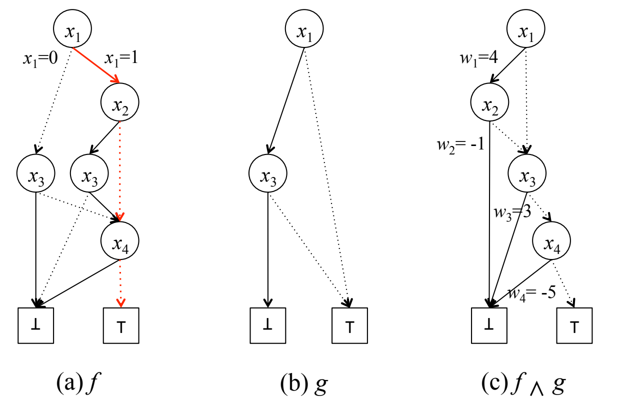


Figure 1: (a) BDD representing $f(\boldsymbol{x}) = (x_1 \wedge \neg x_2 \wedge \neg x_4) \vee (x_1 \wedge x_3 \wedge \neg x_4) \vee (\neg x_1 \wedge \neg x3 \wedge \neg x_4)$; a dotted arc indicates 0-child, while a solid arc is 1-child. (b) BDD of $g(\boldsymbol{x}) = \neg x_1 \vee (x_1 \wedge \neg x_2)$. (c) BDD of $f \wedge g$ with weights.

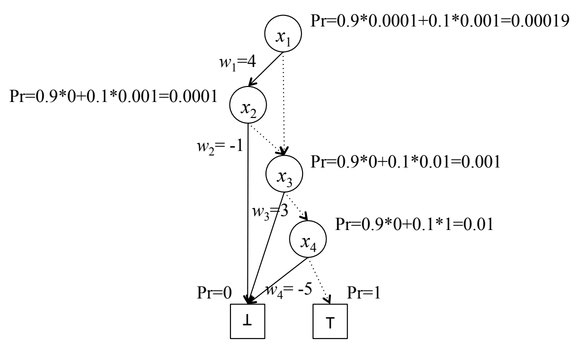


Figure 2: Calculation process of feasible probability. Given truth probabilities for each variable, $\Pr(x_i = 1) = 0.9 \ \forall i \in \{1, 2, 3, 4\}$, and assume that $\Pr(\top) = 1$ and $\Pr(\bot) = 0$ for terminal nodes. The probability of each BDD node is calculated from the terminals, and that of the whole feasible space is obtained at the root node.

root node, x_1, and two terminal nodes, $\bot$ and $\top$. Each non-terminal node is labeled as i-th variable, x_i, and it has two labeled arcs, 0-child and 1-child, each of which indicates whether x_i is 0 or 1. A path from the root to a terminal corresponds to a value of Boolean vector, $\boldsymbol{x}$, or a set of values when some variables are skipped (e.g., the red BDD path in Fig. 1a, on which x_3 is skipped, expresses the value of $\boldsymbol{x} = 10*0$, where $*$ means the value of corresponding variable is "don't care"). The variables must be in the same order from the root to a terminal on every path. The terminal node at the end of path indicates the value of $f(\boldsymbol{x})$. The height, or the path length, is never greater than the number of variables. Common prefix and suffix are shared among paths for compression (e.g., two paths, $10*0$ and 1110, share prefix $x_1 = 1$ and suffix $x_4 = 0$ in Fig. 1a). It is believed that BDDs are well compressed for most practical functions [15]. BDD size is defined as the number of non-terminal nodes in it, and is denoted by $||f||$ if the BDD represents function f (e.g., BDD of Fig. 1a has five non-terminal nodes).

BDD defines an efficient algorithm named Apply, which performs arbitrary logic operations over two operand BDDs and constructs the resulting BDD [2]. The operation is conducted very efficiently, because it is directly executed without decompressing the BDDs. The worst-case BDD size can be quite large, $||f \Diamond g|| \leq ||f|| \cdot ||g||$, given that $\Diamond$ is an arbitrary operator. However, the size is usually considerably smaller than this worst-case upper bound, closer to $||f|| + ||g||$ [16]. Fig. 1c shows an example of AND operation. Applying such logic operations over several BDDs, much more complicated BDDs can be constructed. This is a powerful tool used to solve many mathematical problems. Let's assume a constraint satisfaction problem, or an optimization problem with constraints. If the constraints are given by a combination of Boolean functions, the feasible space can be represented as a BDD by performing appropriate logic operations over BDDs of constraints. The BDD can be constructed even if the constraints form a non-convex space. Once a BDD of feasible space is obtained, several techniques can be conducted on it, as follows.

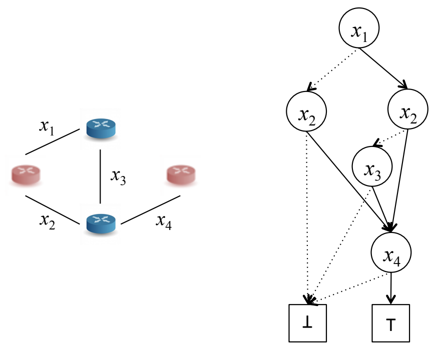
- **Feasibility check**. Given a solution, it is trivial to determine whether the solution is feasible by traversing the corresponding path on the BDD; e.g., in Fig. 1a, $\boldsymbol{x} = 1010$ is feasible, because the corresponding red path ends at the $\top$-terminal.

- **Optimization**. Given a linear objective function, the optimal solution is easily found by dynamic programming; e.g., for $\min_{\boldsymbol{x}} 4x_1 - x_2 + 3x_3 - 5x_4$ s.t. $f \wedge g = \top$ in Fig. 1c, we find $\boldsymbol{x}^\star = 0100$ as the shortest path to the $\top$-terminal.

- **Counting**. Given probabilities of becoming true for each variable, the probability of realizing feasible solutions can be calculated by summing up it from the bottom to the root, as shown in Fig. 2.

In addition to the traditional techniques described above, BDD has gained two more great advances lately.

- **Graph properties**. Consider a network is defined as a set of edges, and x_i indicates the existence of i-th edge; a BDD represents a family of subgraphs (a family of edge subsets). Frontier-based search [13] directly constructs a BDD of given graph properties, without repeatedly performing logic operations. Fig. 3 shows a BDD of all trees connecting specified vertices. Specifying a combination of several graph properties, many complicated graph types are supported, such as Steiner trees, spanning forests, Hamilton paths,

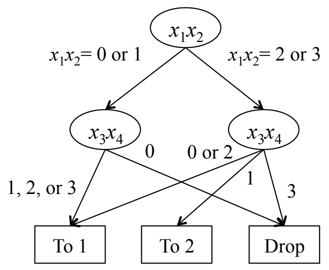


(a) Network (b) BDD

Figure 3: (a) Graph representing a network. The use of i-th edge is indicated by x_i. (b) BDD representing a set of trees connecting red vertices in the network.

x_1x_2	x_3x_4	Action
[0,1]	[1,3]	To 1
[1,1]	*	To 2
[2,3]	[0,2]	To 1
*	*	Drop

(a) Rules



(b) MDD

Figure 4: (a) Rule table. Each rule associates two header fields with actions. (b) MDD representing the rules. Since consecutive two-bits are aggregated in the MDD, non-terminal nodes are labeled by the two-bits while arcs are labeled 0- to 3-child.

Euler cycles, k-cliques, independent sets, connected components, and so on. The resulting BDD can be used to perform a logic operation or the traditional techniques above. This search method was independently developed for limited graph types [4, 16, 17], but they have been integrated as a unified framework in [13], and so it can be applied to various practical problems now.

- **Multi-valued functions**. Assume a BDD representing a set of packet bits; for instance, it is specified by a so-called "5-tuple" rules (i.e., source/destination IP prefixes, source/destination port ranges, and protocol). Multi-valued decision diagram, or MDD, maps the bits to more than two values; i.e., $F(\boldsymbol{x}) \in \{1, 2, \ldots\}$. The solution space can be "colored" by several subspaces, while the space is divided into only $\top$ and $\perp$ with BDD. MDD can be used not only for analyzing firewalls, but also for classifying a packet into several actions like determining next-hop switches or selecting a labeled path, as shown in Fig. 4. Moreover, the length of MDD path is shrunk by aggregating multiple bits into a single variable, and the corresponding color (action) is looked up very quickly. MDD was originally studied in the LSI-CAD community [18], and recently its construction algorithms tailored to packet manipulation have been proposed in [7].

3. Taxonomy of BDD Applications in Information Networking

This section describes four typical applications of BDDs studied in the networking community.

3.1. BDD as Network Graph

- **Network reliability** is defined as the probability with which specified vertices are connected under possible edge failures. This is a straight-forward application of BDD, and it can be simply calculated as follows. First, all subgraphs connecting the specified vertices are represented as a BDD by the frontier-based search. Secondly, the feasible probability is obtained on the BDD like Fig. 2. Reference [4] showed that the network reliability can be calculated in a few minutes with a network of hundreds edges. Reference [5] extends this method by logic operations, so as to support vertex failures as well as edge failures. BDD is considered as the most efficient method to evaluate the network reliability.

- **Network optimization** also can be performed by combining some techniques described in Section 2. First, BDDs representing each constraint are constructed separately by the frontier-based search and/or logic operations (e.g., assume f_i represents i-th constraint). Secondly, the AND of all the constraints is calculated (e.g., $\bigwedge_i f_i$). Finally, the optimal solution is searched for on the ANDed BDD, as described in Section 2. Reference [6] found the optimal network configuration (optimal subgraph) in a smart grid network with nearly five hundreds power-lines. This is the first work to successfully optimize such a large-scale power system.

3.2. BDD as Packet Bits

- **Packet classification** is a functionality that determines the action taken on a packet based on multiple header fields. Assume that a set of prioritized rules is given, as shown in Fig. 4a. First, BDDs corresponding to each rule are constructed (e.g., f_i for i-th rule). Since i-th rule is masked by the upper rules, the overlapped region must be subtracted (e.g., $f'_i = f_i \wedge \neg \bigvee_{j<i} f_j$). Secondly, BDDs are converted into MDDs by replacing the $\top$-terminal with the corresponding action. Finally, all the MDDs are unified with an OR-like operation, and the MDD representing the whole rules is obtained, such as Fig. 4b; consecutive bits are aggregated if needed. Reference [7] proposed a more sophisticated classification method designed for the software-defined networking; the method classifies 10 million packets per second (roughly 10 Gbps) against hundreds of thousands of rules.

- **Configuration verification** examines forwarding rules of network devices in a network, in order to check the conformity with a network policy, such as loop-free, no blackhole, and waypointing. Assume that BDDs of each rule, f'_i, has been calculated, similarly with packet classification. Given a Boolean function representing a set of packets, p, and a rule, f'_i, their AND, $p \wedge f'_i$ indicates a set of packets matched to the rule. If we have a sequence of rules that form a path of interest, $(f'_1, f'_2, \ldots)$, their AND, $\bigwedge_i f'_i$, is a set of packets traversing the path. This technique allows us to identify whether some packets can traverse a path violating a network policy. Reference [8] verified a network with hundreds of thousands of rules just in a few seconds. BDD yields the fastest record on the network configuration verification.

4. Conclusions

This paper describes applications of BDDs in the networking research. Readers interested in BDDs can try it with open-source implementations. CUDD[1] is a pure BDD implementation. Graphillion [19][2] provides an easy graph abstraction over BDDs.

Since BDD is a means to *exactly* represent a Boolean function, it has been often applied to applications that inhibit approximation like verification. Recently, a novel technique that constructs a *relaxed* BDD was proposed [20], which balances between exactness and scalability. We believe that this new approach will broaden the possibility for BDDs in the information networking community.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[2] R. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Compt.*, vol. C-35, no. 8, pp. 677–691, 1986.

[3] S. Minato, "Zero-suppressed BDDs for set manipulation in combinatorial problems," in *DAC*, pp. 272–277, 1993.

[4] G. Hardy, C. Lucet, and N. Limnios, "K-terminal network reliability measures with binary decision diagrams," *IEEE Trans. Rel.*, vol. 56, no. 3, pp. 506–515, 2007.

[5] S.-Y. Kuo, F.-M. Yeh, and H.-Y. Lin, "Efficient and exact reliability evaluation for networks with imperfect vertices," *IEEE Trans. Rel.*, vol. 56, no. 2, pp. 288–300, 2007.

[6] T. Inoue, K. Takano, T. Watanabe, J. Kawahara, R. Yoshinaka, A. Kishimoto, K. Tsuda, S. Minato, and Y. Hayashi, "Distribution loss minimization with guaranteed error bound," *IEEE Trans. Smart Grid*, vol. 5, no. 1, pp. 102–111, 2014.

[7] T. Inoue, T. Mano, K. Mizutani, S.-I. Minato, and O. Akashi, "Rethinking packet classification for global network view of software-defined networking," in *IEEE ICNP*, pp. 296–307, 2014.

[8] H. Yang and S. Lam, "Real-time verification of network properties using atomic predicates," in *IEEE ICNP*, pp. 1–11, 2013.

[9] L. Yuan, C.-N. Chuah, and P. Mohapatra, "ProgME: Towards programmable network measurement," *IEEE/ACM Trans. Netw.*, vol. 19, no. 1, pp. 115–128, 2011.

[10] Y.-W. E. Sung, C. Lund, M. Lyn, S. G. Rao, and S. Sen, "Modeling and understanding end-to-end class of service policies in operational networks," in *SIGCOMM'09*, pp. 219–230, 2009.

[11] G. Li, S. Hou, and H.-A. Jacobsen, "A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams," in *IEEE ICDCS*, pp. 447–457, 2005.

[12] S. Minato, "Techniques of BDD/ZDD: Brief history and recent activity," *IEICE Trans. Inf. and Syst.*, vol. 96, no. 7, pp. 1419–1429, 2013.

[13] J. Kawahara, T. Inoue, H. Iwashita, and S. ichi Minato, "Frontier-based search for enumerating all constrained subgraphs with compressed representation," tech. rep., Hokkaido University, TCS-TR-A-14-76, 2014.

[14] S. Hazelhurst, "Algorithms for analysing firewall and router access lists," *CoRR*, vol. cs.NI/0008006, 2000. http://arxiv.org/abs/cs.NI/0008006.

[15] R. Yoshinaka, J. Kawahara, S. Denzumi, H. Arimura, and S. Minato, "Counterexamples to the long-standing conjecture on the complexity of BDD binary operations," *Inf. Process. Lett.*, vol. 112, no. 16, pp. 636–640, 2012.

[16] D. E. Knuth, *The Art of Computer Programming: Combinatorial Algorithms Part 1*, vol. 4A. Addison-Wesley, USA, 2011.

[17] K. Sekine, H. Imai, and S. Tani, "Computing the Tutte polynomial of a graph of moderate size," in *Algorithms and Computations*, vol. 1004 of *Lecture Notes in Computer Science*, pp. 224–233, Springer, 1995.

[18] A. Srinivasan, T. Ham, S. Malik, and R. Brayton, "Algorithms for discrete function manipulation," in *IEEE ICCAD*, pp. 92–95, 1990.

[19] T. Inoue, H. Iwashita, J. Kawahara, and S.-i. Minato, "Graphillion: software library for very large sets of labeled graphs," *International Journal on Software Tools for Technology Transfer*, 2014.

[20] D. Bergman, A. A. Cire, W.-J. v. Hoeve, and J. N. Hooker, "Optimization bounds from binary decision diagrams," *INFORMS Journal on Computing*, vol. 26, no. 2, pp. 253–268, 2014.

[1] http://vlsi.colorado.edu/~fabio/CUDD/

[2] http://graphillion.org