IEICE TRANS. INF. & SYST., VOL.E100–D, NO.8 AUGUST 2017

INVITED PAPER *Special Section on Multiple-Valued Logic and VLSI Computing*

Power of Enumeration — Recent Topics on BDD/ZDD-Based Techniques for Discrete Structure Manipulation

Shin-ichi MINATO[†a)], ***Senior Member***

SUMMARY *Discrete structure manipulation* is a fundamental technique for many problems solved by computers. *BDDs/ZDDs* have attracted a great deal of attention for twenty years, because those data structures are useful to efficiently manipulate basic discrete structures such as logic functions and sets of combinations. Recently, one of the most interesting research topics related to BDDs/ZDDs is *Frontier-based search method,* a very efficient algorithm for enumerating and indexing the subsets of a graph to satisfy a given constraint. This work is important because many kinds of practical problems can be efficiently solved by some variations of this algorithm. In this article, we present recent research activity related to BDD and ZDD. We first briefly explain the basic techniques for BDD/ZDD manipulation, and then we present several examples of the state-of-the-art algorithms to show the *power of enumeration*.

key words: *BDD, ZDD, discrete structure, graph algorithm*

1. Introduction

Discrete structures are foundational materials for computer science and mathematics, which are related to set theory, symbolic logic, inductive proof, graph theory, combinatorics, probability theory, etc. Many problems are decomposed into discrete structures using simple primitive algebraic operations.

A Binary Decision Diagram (BDD) is a representation of a Boolean function, one of the most basic models of discrete structures. After the epoch-making paper [1] by Bryant in 1986, BDD-based methods have attracted a great deal of attention. The BDD was originally developed for the efficient Boolean function manipulation required in VLSI logic design, however, later they are also used for *sets of combinations* which represent many kinds of combinatorial patterns. A Zero-suppressed BDD (ZDD) [2] is a variant of the BDD, customized for representing a set of combinations. ZDDs have been successfully applied not only to VLSI design, but also for solving various combinatorial problems, such as constraint satisfaction, frequent pattern mining, and graph enumeration. Recently, ZDDs have become more widely known, since D. E. Knuth intensively discussed ZDD-based algorithms in the latest volume of his famous series of books [3].

Although a quarter of a century has passed since Bryant first put forth his idea, there are still many interesting and exciting research topics related to BDDs and ZDDs [4]. One of the most important topics would be that, Knuth presented an extremely fast algorithm "Simpath" [3] to construct a ZDD which enumerates all the paths connecting two points in a given graph structure. This work is important because many kinds of practical problems are efficiently solved by some variations of this algorithm. We generically call such ZDD construction methods "frontier-based methods."

The above techniques of data structures and algorithms have been implemented and published as an open software library, named "Graphillion" [5], [6]. Graphillion is a library for manipulating very large sets of graphs, based on ZDDs and frontier-based method. Graphillion is implemented as a Python extension in C++, to encourage easy development of its applications without introducing significant performance overhead.

In order to organize an integrated method of algebraic operations for manipulating various types of discrete structures, and to construct standard techniques for efficiently solving large-scale and practical problems in various fields, a governmental agency in Japan executed a nation-wide project: ERATO MINATO Discrete Structure Manipulation System Project [7] from 2009 to 2016. Many interesting research results were produced in this project, and some of topics are still attractive to be explored more.

This article presents recent research activity related to BDD and ZDD. We first briefly explain the basic techniques for BDD/ZDD manipulation, and then we present several examples of the state-of-the-art algorithms to show the *power of enumeration*.

2. BDDs and ZDDs

A Binary Decision Diagram (BDD) is a graph representation for a Boolean function, developed in VLSI design area. As illustrated in Fig. 1, it is derived by reducing a binary deci-

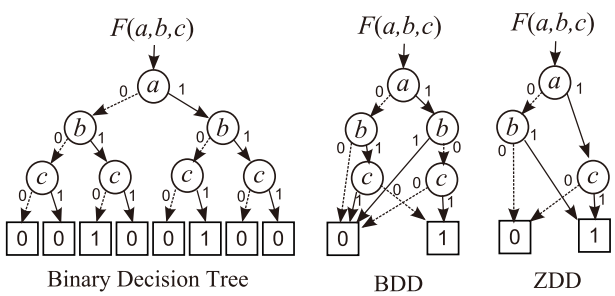


Fig. 1 Binary Decision Tree, BDDs and ZDDs.

Manuscript received October 21, 2016.
Manuscript publicized May 19, 2017.
[†]The author is with the Graduate School of Information Science and Technology, Hokkaido University, Sapporo-shi, 060–0814 Japan.
a) E-mail: minato@ist.hokudai.ac.jp
DOI: 10.1587/transinf.2016LOI0002

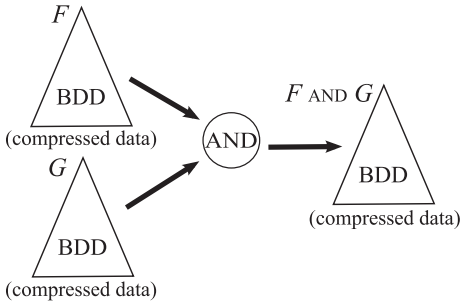


Fig. 2 Framework of BDD logic operation.

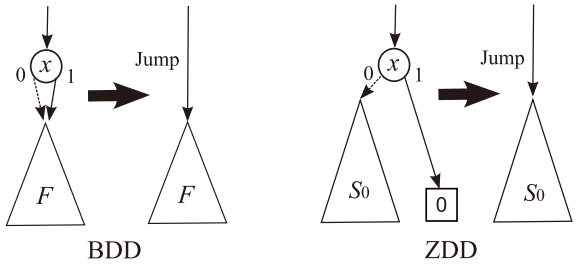


Fig. 3 ZDD reduction rule.

sion tree graph, which represents a decision making process by the input variables. If we fix the order of input variables, and apply the following two reduction rules, then we have a compact canonical form for a given Boolean function:

(1) Delete all redundant nodes whose two edges point to a same child node, and
(2) Share all equivalent nodes having a same pair of child nodes and a same variable.

The compression ratio of a BDD depends on the properties of Boolean function to be represented, but it can be 10 to 100 times more compact in some practical cases. Even though the BDD gives good compression, still we might need an exponential time and space when we start from a non-reduced binary decision tree. However, we can systematically construct a reduced BDD that is the result of a binary logic operation (i.e., AND or OR) for a given pair of BDDs, as shown in Fig. 2. This algorithm, proposed by Bryant [1], is based on a recursive procedure with hash table techniques, and it is much more efficient than generating binary decision trees when the BDDs have a good compression ratio. The computation time is bounded by the product of the BDD sizes of the two operands, and in many practical cases, it is linearly bounded by the sum of input and output BDD sizes.

BDD is based on the on-memory data processing techniques, and it enjoys the advantage of using random access memories. Recently, commodity PCs are equipped with Giga-Bytes of main memory, and we can solve larger-scale problems which used to be impossible due to memory shortage. Thus, especially after 2000s, the BDD application areas are spreading.

Zero-suppressed BDD (ZDD) is a variant of BDD, customized for manipulating sets of combinations. This data structure was first introduced by Minato [2]. ZDDs are based on the special reduction rules different from the ordinary one, as follows.

(1') Delete all nodes whose 1-edge directly points to the 0-terminal node, but do not delete the nodes which were deleted in ordinary BDDs. (Fig. 3)
(2) Share all equivalent nodes as well as ordinary BDDs.

An example of ZDD is also shown in Fig. 1. Here the three graphs represent a same instance of Boolean function.

This alternative reduction rule is extremely effective if we handle a set of sparse combinations. If the average appearance ratio of each item is 1%, ZDDs are possibly more compact than ordinary BDDs, up to 100 times. Such situations often appear in real-life problems, for example, in a supermarket, the number of items in a customer's basket is usually much less than all the items displayed there. Because of such an advantage, ZDD is now widely recognized as the most important variant of BDD. In 2009, D. Knuth presented a new fascicle of his famous book series [3], which has a section of ZDDs discussed minutely in 30 pages with 70 exercises.

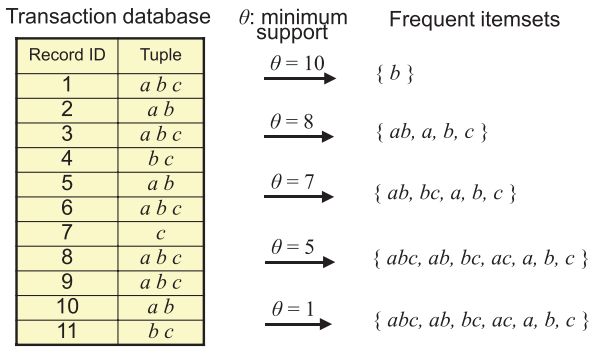
3. Modern Applications of BDDs/ZDDs

BDD/ZDD-based algorithm was originally invented, and have been developed mainly in VLSI logic design area since early 1990's. However, after 2000, BDDs/ZDDs are applied for not only VLSI design but also for more various purposes. Such modern applications of BDDs/ZDDs include data mining (frequent pattern mining) [8]–[10], probabilistic modeling [11]–[13], solving graph enumeration problems [3], [14], etc. Here we will present some of those applications.

3.1 Data Mining (Frequent Itemset Mining)

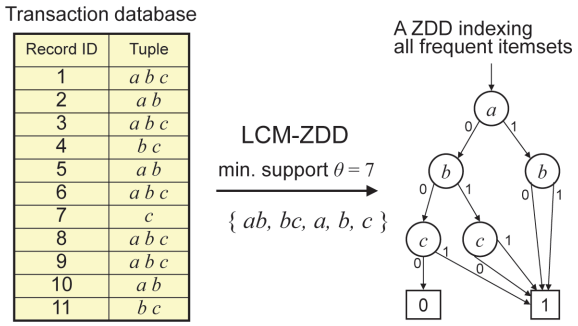
Discovering useful knowledge from large-scale databases has attracted considerable attention during the last decade. *Frequent itemset mining* is one of the most fundamental problems in knowledge discovery. The task is to find all frequent patterns each of which is a combinatorial items included in at least θ records of a given transaction database, where θ is called *minimum support*, the user specified threshold. An example is shown in Fig. 4. Since the pioneering work by Agrawal *et al.* [15], various algorithms have been proposed to solve the *frequent itemset mining problem* (cf., [16], [17]). In 2008, Minato et al. [10] proposed a fast algorithm *LCM over ZDDs (LCM-ZDD)* for generating very large-scale frequent itemsets using ZDDs. Their method is based on *LCM algorithm* [18], one of the most efficient state-of-the-art techniques for itemset mining, and directly generates compact output data structures on the main memory, to be efficiently post-processed by using ZDD-based algebraic operations.

LCM-ZDD does not touch the core algorithm of LCM,



Record ID	Tuple
1	a b c
2	a b
3	a b c
4	b c
5	a b
6	a b c
7	c
8	a b c
9	a b c
10	a b
11	b c

Fig. 4 Frequent itemset mining.



Record ID	Tuple
1	a b c
2	a b
3	a b c
4	b c
5	a b
6	a b c
7	c
8	a b c
9	a b c
10	a b
11	b c

Fig. 5 LCM over ZDDs method.

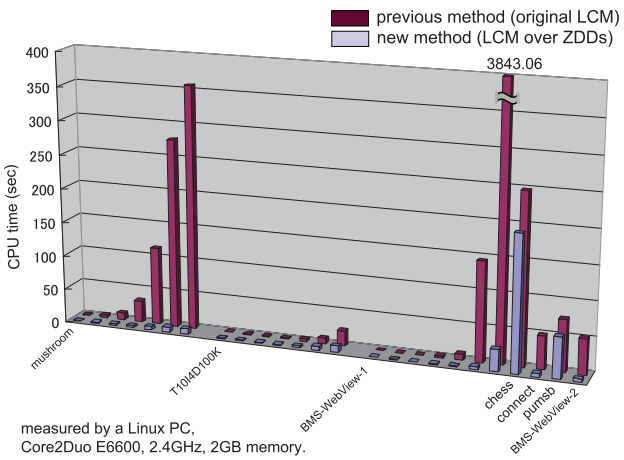


Fig. 6 Performance of LCM-ZDD.

and just generates a ZDD for the solutions obtained by LCM. It constructs a ZDD which is the union of all the itemsets found in the backtracking search, and finally returns a pointer to the root node of the ZDD. Figure 6 shows the experimental results [10] to compare the performances between LCM-ZDD and the original LCM. We can observe that LCM-ZDD is much efficient than the original one, especially when large numbers of frequent itemsets are output.

By utilizing LCM-ZDD, we can compare multiple sets of frequent itemsets generated under different conditions. As shown in Fig. 7, our ZDD-based method can store and index the two sets of frequent itemsets for the datasets of

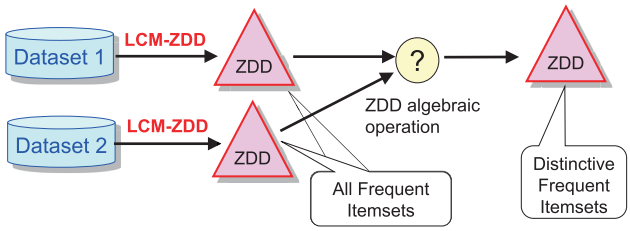


Fig. 7 Post-processing after LCM-ZDD.

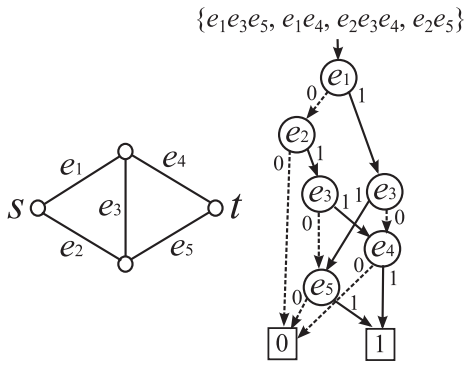


Fig. 8 ZDD representing the paths from s to t.

different time, and easily compute the intersection, union, and difference between the frequent itemsets. When many similar ZDDs are generated, their ZDD nodes are effectively shared within a monolithic multi-rooted graph, requiring much less memory than that required to store each ZDD separately.

3.2 Graph Enumeration Problems

Many real-life problems can be modeled by a kind of graph structures. ZDDs can be utilized for enumerating and indexing the solutions of a graph problem. When we assume a graph $G = (V, E)$ with the vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and the edge set $E = \{e_1, e_2, \ldots, e_m\}$, a graph enumeration problem is to compute a subset of the power set 2^E (or 2^V), each element of which satisfies a given property. In this model, we can consider that each solution is a combination of edges (or vertices), and a set of solutions can be represented by a ZDD. For example, Fig. 8 shows the ZDD representing the set of paths connecting the two vertices s and t of the graph. Each path can be represented as a combination of the edges, $\{e_1e_3e_5,\ e_1e_4,\ e_2e_3e_4,\ e_2e_5\}$. The ZDD also has four paths from the root node to the 1-terminal node, and each path corresponds to the solution of the problem, where $e_i = 1$ means to use the edge e_i, and $e_i = 0$ means not to use e_i.

There are a number of literature on ZDDs for solving graph problems since early 1990's, however, in 2009, D. E. Knuth has presented the surprisingly fast algorithm *Simpath* [3] to construct a ZDD which represents all the paths connecting two points s and t in a given graph (not necessarily the shortest ones but ones not passing through the same point twice). A part of the book page

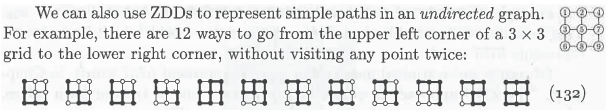


Fig. 9 Result of Simpath algorithm for a 2x2 grid (3x3 vertices) graph written in the Knuth's book [3] (Vol.4, Fascicle 1, p.121, or p.254 of Vol.4A).

Fig. 10 Sign and picture of the exhibition at Miraikan.

is shown in Fig. 9. This work is important because many kinds of practical problems can be efficiently solved by some variations of this algorithm. Knuth provides his own C source codes on his web page for public access, and the program is surprisingly fast. For example, in a 14×14 grid graph (420 edges in total), the number of self-avoiding paths between opposite corners is as great as 227449714676812739631826459327989863387613323440 ($\approx 2.27 \times 10^{47}$) ways. By applying the Simpath algorithm, the set of paths can be compressed into a ZDD with only 144759636 nodes, and the computation time is only a few minutes.

Here we will not describe the detailed mechanism of Simpath algorithm. One may refer to another survey paper [4] written by the author. Roughly speaking, the Simpath algorithm belongs to a dynamic programming method, based on the topological structure by scanning the graph from the start vertex to the goal vertex. Knuth calls the scanning line *frontier*. If the frontier grows larger in the computation process, more ZDD nodes are generated and more computation time is required. Thus, we had better keep the frontier small. The maximum size of the frontier is bounded by the *path width* of the given graph structures. Empirically, planar or narrow graphs are favorable.

Knuth described in his book [3] that the Simpath algorithm can easily be modified to generate not only *s-t* paths but also Hamilton paths, directed paths, some kinds of cycles, and many other problems by slightly changing the internal data structure. We generically call such ZDD construction method *Frontier-based search*. (or, em Frontier method in short.)

The Frontier method is different from the conventional ZDD construction, which repeats primitive set operations between two ZDDs. In general, the primitive set operations are efficiently implemented based on Bryant's BDD construction algorithm, but do not directly use the properties of the specific problem. Frontier method is sometimes much more efficient because it is a dynamic programming method based on the frontier, a kind of structural property of the given problem.

Graph enumeration problems are strongly related to many kinds of real-life problems. For example, path enumeration is of course important in geographic information systems, and is also used for dependency analysis of a process flow chart, fault analysis of industrial systems, etc. In order to utilize such state-of-the-art techniques, our research project developed an integrated software tool set, named "Graphillion" [5]. This tool provides an efficient means of ZDD construction by frontier-based method for a given graph problem, and also gives various ZDD operations in a simple user interface based on a graph library package written in Python. An interesting tutorial video and the open source codes are provided at the web page of "Graphillion.org.". We hope that many researchers and engineers will be interested in this tool and will utilize it for various kinds of problems.

4. Recent Topics on BDD/ZDD-Based Discrete Structure Manipulation

4.1 Exhibition on "Power of Enumeration"

In 2012, our research project collaborated with "Miraikan" (National Future Science Museum of Japan) to design an exhibition presenting the power of combinatorial explosion and the state-of-the-art techniques for solving such hard problems. As a work of the exhibition, we supervised a short educational animation video (Fig. 11). The video is mainly designed for junior high school to college students, so it does not use any difficult technical terms, and it is something like a funny science fiction story.

In this video, we used the simple path enumeration problem for $n \times n$ grid graphs. The story is that the teacher counts the total number of paths for children starting from $n = 1$, but she will be faced with a difficult situation since the number grows unbelievably fast. She would spend 250,000 years to count the paths for the 10×10 grid graph by using a super computer if she used a naive method. The story ends by telling that a state-of-the-art algorithm can finish the same problem in a few seconds.

The video is now shown in the official museum channel of YouTube [20] and already got 1.8 million views, which is an extraordinary in the case of scientific educational contents. In addition, it was our great pleasure to hear that Knuth also enjoyed this video and shared it to several his friends.

The video story tells the computation result up to $n = 11$, but it is interesting to know how large n we can compute for. We have worked for improving the algorithm for solving the large-scale problems as much as possible, and succeeded in counting the total number of self-avoiding *s-t* paths for the 26×26 grid graph. This is the current world record and is officially registered in the On-Line Encyclopedia of Integer Sequences [19] in Nov. 2013. The results of big numbers are listed in Table 1. The detailed techniques for solving larger problems are presented in the report by Iwashita et al. [21].

The power of enumeration is also demonstrated in a

Table 1 The world record of the self-avoiding path enumeration problems [19].

n	The number of paths
1	2
2	12
3	184
4	8512
5	1262816
6	575780564
7	789360053252
8	3266598486981642
9	41044208702632496804
10	156875803046475001321 4100
11	1824132915142480492414708852 36
12	6452803934327001896335718515848 2118
13	6945066476152136166427470154890735896 488
14	2274497146768127396318264593279898633876133234 40
15	2266745568862672746374567396713098934866324885408319028
16	68745445609149931587631563132489232824587945968099457285419306
17	6344814611237963971310297540795524400449443986866480693646369387855336
18	1782112840842065129893384946652325275167838065704767655931452474605826692782532
19	152334497170487999308074281031922969089945425533239455577602986673735506059287756925584 4
20	396289219982303756020729951713336250210633970573946377151523711337701068236403570670447206494039 8
21	313747510503710272042053813738221451310331219369872365306135199134643337938938579396557699224602131646386 8
22	7559702866673453396615191233152226193531037320724094811673914104795179257927436312349870388833176349872711714044399 2
23	554354293552374770099143184890614379306903799709643313325569586464840084073348855445663869240208757112420600854085134823933945720
24	123717122312070647583387448626735708323730419890129435396787270804849516955159304856413945507921530371918580282125122809266003045813867910 94
25	84029748578811334710070837454368091272960542937755385498247426239370284978982152569291785770839709601216256025060273165497184021064940499783756042474 08
26	17369931586279272931175440421236498900372229588288140604663703720910342413276134762789218193498006107082296223143380491348290026721931129627708738890853908108906396

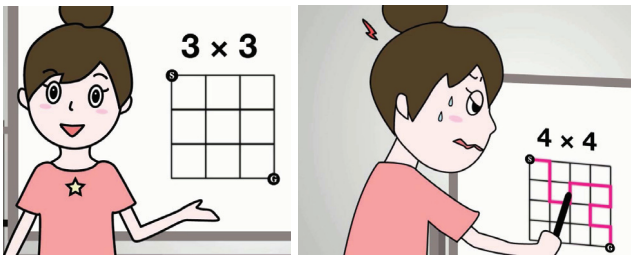


Fig. 11 Screenshots of the animation video [20].

problem of railway route search. Figure 12 shows one of our original tool *Ekillion* [22], to enumerate self-avoiding paths in a railway network of a large city area in Japan. For example, in Tokyo, Ochanomizu-station and Suidobashi-station are adjacent to each other, but we may find more than six million of self-avoiding path from and to the two stations in Tokyo metropolitan area. The longest self-avoiding paths of them includes as many as 343 stations. This tool is available at a web site for public.

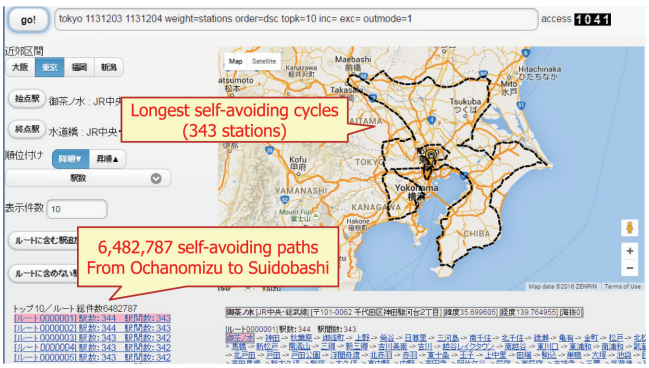


Fig. 12 Ekillion: path enumeration tool for railway networks.

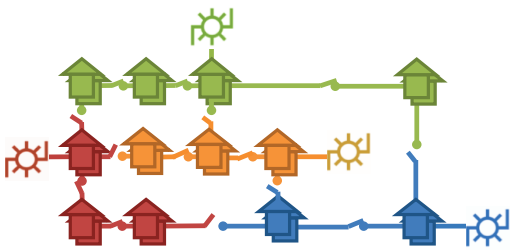


Fig. 13 An example of power distribution network.

4.2 Analysis of Power Distribution Networks

In 2011, a big earthquake was occurred in Japan. After the severe accident of the nuclear plants, the electric power supply networks receive more attentions from the society in Japan, since solar and wind power generators are not stable as the traditional systems. Thus, our research group accelerate our collaborative work with the researchers in power electric community.

As an output of our research project, Inoue et al. [23] presented the design method of electric power distribution systems using ZDDs. Figure 13 shows a trivial example of power distribution network. Here we have a number of switches to connect or disconnect the adjacent districts. Each district must be connected to a power substation, or black out otherwise. We have another constraint that any two power substations must not directly connected. We also need to consider that too much currency may burn a line, and that too long distance connection may cause voltage down. Now we have a combinatorial problem to find a good switching pattern. Even for this trivial example with 14 switches, we will find 210 feasible switching patterns out of 16384 combinations. A typical realistic benchmark network contains 468 switches and the search space becomes about 10^{140} combinations.

Most of civil engineering systems can be modeled by planar (or nearly planar) graphs, so the Frontier method is very effective in many cases. We succeeded in generating a ZDD to enumerate all the possible switching patterns for the realistic benchmark with 468 switches. The obtained ZDD represents as many as 10^{60} of valid switching patterns but the actual ZDD size is less than 100MByte, and computation time is around 30 minutes. After generating the ZDD,

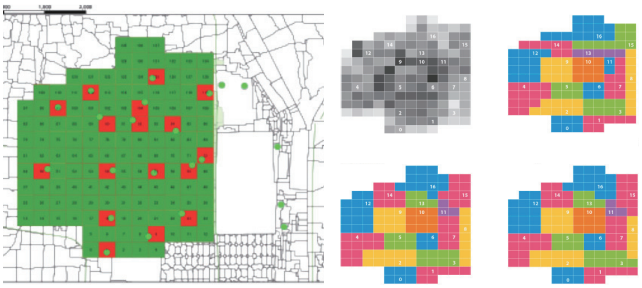


Fig. 14 Layout of evacuation regions in Kyoto city [24].

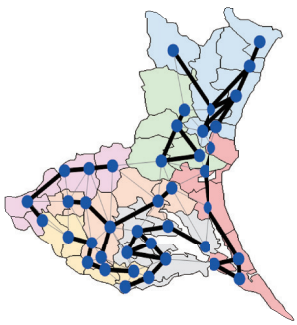


Fig. 15 Partitioning for electoral districts.

all valid switch patterns are compactly represented, and we can efficiently discover the switching patterns with maximum, minimum, and average cost. We can also apply other various additional constraints to the current solutions. Now our research project are collaborating with TEPCO (Tokyo Electric Power Company) to make experiments for evaluating energy loss on the real commercial network.

4.3 Evacuation Planning for Disasters

Evacuation planning in a city is an important problem in civil engineering. Figure 14 shows a layout problem of evacuation regions in North Ward of Kyoto city. Here we have a number of refuge spots in the city, and now we have a problem that which region may evacuate to which refuge spot. Every region must be assigned to a refuge spot. We also have constraints that too many refugees into one refuge spot may cause overflow, and that too long distance route takes too long time for evacuation. Thus, we can see that this is a very similar problem as the power distribution problem. ZDD construction using Frontier method is also effective for enumerating all possible solutions and analyzing the best solution of this problem. We presented our result at an international conference on operation research [24], collaborating with researchers in the civil engineering research community.

4.4 Partitioning Electoral Districts

Partitioning electorial districts is another important problem to support democratic social systems. Suppose that we would like to elect k representatives from a prefecture, which is composed of a number of cities. We divide the prefecture into k connected components to elect one representative for each component under the condition that any city must not be split and all the components are desired to be balanced. For this purpose, we represent the prefecture as a vertex-weighted graph, where its vertex corresponds to a city, two vertices are adjacent if and only if the corresponding cities have the common border, and the weight of a vertex means the population in the city. In this situation, we should reduce the disparity of weights (the ratio of maximum and minimum weights) as much as possible, under various geographical and social constraints. Kawahara, Hotta, et al. [25] developed a method for enumerating all the possible solutions of partitions satisfying a given disparity thresholds. For example, Ibaraki prefecture of Japan includes 41 city units to be partitioned into 7 districts. It can be represented in a graph including 41 vertices and 87 edges. They developed an algorithm based on Frontier method, and they enumerated 11,893,998,242,846 partitions of 7 connected components. After that they found 25,730,669 solutions which satisfies the condition of 1.4 or less disparity of voting weight. Computation time was about a half hour in a commodity PC. The disparity threshold can easily be changed as 1.3, 1.2, etc., and thus the politicians and governors may evaluate the feasibility of the current solution of electoral partition.

5. Conclusion

In this article, we presented recent research activities on BDD/ZDD-based discrete structure manipulation. Although many years have passed since BDDs/ZDDs were developed, there are still many interesting and exciting research topics related to them. Especially, the frontier-based search method is important because many kinds of practical problems are efficiently solved by some variations of this algorithm.

In order to construct standard techniques discrete structure manipulation for efficiently solving large-scale and practical problems in various fields, a nation-wide research project, JST ERATO [7], was executed from 2009 to 2016. The project was successfully finished, and a successor project, JSPS KAKENHI(S) [26], is now running until 2020. Many interesting research results were produced in those research projects, and some of topics are still attractive to be explored more. We hope to see many kinds of applications in real-life problems.

Acknowledgment

This work is partly supported by JSPS KAKENHI(S) 15H05711.

References

[1] R.E. Bryant, "Graph-based algorithms for Boolean function manipulation," IEEE Transactions on Computers, vol.C-35, no.8, pp.677–691, 1986.

[2] S. Minato, "Zero-suppressed BDDs for set manipulation in combinatorial problems," Proc. of 30th ACM/IEEE Design Automation Conference (DAC'93), pp.272–277, 1993.
[3] D.E. Knuth, The Art of Computer Programming: Bitwise Tricks & Techniques; Binary Decision Diagrams, Addison-Wesley, 2009.
[4] S. Minato, "Techniques of BDD/ZDD: Brief history and recent activity," IEICE Transactions on Information and Systems, vol.E96-D, no.7, pp.1419–1429, 2013.
[5] T. Inoue and et al., "Graphillion." http://graphillion.org/, 2013.
[6] T. Inoue, H. Iwashita, J. Kawahara, and S. Minato, "Graphillion: software library for very large sets of labeled graphs," International Journal on Software Tools for Technology Transfer (STTT), vol.18, no.1, pp.57–66, 2016.
[7] S. Minato, "Overview of ERATO Minato project: The art of discrete structure manipulation between science and engineering," New Generation Computing, vol.29, no.2, pp.223–228, 2011.
[8] E. Loekit and J. Bailey, "Fast mining of high dimensional expressive contrast patterns using zero-suppressed binary decision diagrams," Proc. The Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2006), pp.307–316, 2006.
[9] S. Minato, "VSOP (valued-sum-of-products) calculator for knowledge processing based on zero-suppressed BDDs," in Federation over the Web, ed. K. Jantke, A. Lunzer, N. Spyratos, and Y. Tanaka, Lecture Notes in Computer Science, vol.3847, pp.40–58, Springer Berlin Heidelberg, 2006.
[10] S. Minato, T. Uno, and H. Arimura, "LCM over ZBDDs: Fast generation of very large-scale frequent itemsets using a compact graph-based representation," in Advances in Knowledge Discovery and Data Mining, ed. T. Washio, E. Suzuki, K. Ting, and A. Inokuchi, Lecture Notes in Computer Science, vol.5012, pp.234–246, Springer Berlin Heidelberg, 2008.
[11] S. Minato, K. Satoh, and T. Sato, "Compiling bayesian networks by symbolic probability calculation based on zero-suppressed BDDs," Proc. of 20th International Joint Conference of Artificial Intelligence (IJCAI-2007), pp.2550–2555, 2007.
[12] M. Ishihata, Y. Kameya, T. Sato, and S. Minato, "Propositionalizing the EM algorithm by BDDs," In Proc. of 18th International Conference on Inductive Logic Programming (ILP 2008), pp.44–49, 2008.
[13] A. Darwiche, "SDD: A new canonical representation of propositional knowledge bases," Proc. of 22nd International Joint Conference of Artificial Intelligence (IJCAI-2011), pp.819–826, 2011.
[14] R. Yoshinaka, T. Saitoh, J. Kawahara, K. Tsuruma, H. Iwashita, and S. Minato, "Finding all solutions and instances of numberlink and slitherlink by ZDDs," Algorithms, vol.5, no.4, pp.176–213, 2012.
[15] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," ACM SIGMOD Record, vol.22, no.2, pp.207–216, 1993.
[16] B. Goethals, "Survey on frequent pattern mining," 2003. http://www.cs.helsinki.fi/u/goethals/publications/survey.ps.
[17] M.J. Zaki, "Scalable algorithms for association mining," IEEE Trans. Knowl. Data Eng., vol.12, no.3, pp.372–390, 2000.
[18] T. Uno, Y. Uchida, T. Asai, and H. Arimura, "LCM: an efficient algorithm for enumerating frequent closed item sets," Proc. Workshop on Frequent Itemset Mining Implementations (FIMI'03), 2003. http://fimi.cs.helsinki.fi/src/.
[19] "Number of nonintersecting (or self-avoiding) rook paths joining opposite corners of an $n \times n$ grid, the on-line encyclopedia of integer sequences," 2013. https://oeis.org/A007764.
[20] S. Doi and et al., "Time with class! let's count! (the art of 10^{64} – understanding vastness –)," 2012. YouTube video, http://www.youtube.com/watch?v=Q4gTV4r0zRs.
[21] H. Iwashita, Y. Nakazawa, J. Kawahara, T. Uno, and S. Minato, "Efficient computation of the number of paths in a grid graph with minimal perfect hash functions," Hokkaido University, Division of Computer Science, TCS Technical Reports, vol.TCS-TR-A-10-64, 2013.
[22] Y. Hamuro and et al., "Ekillion." http://www.nysol.jp/en/home/apps/ekillion/, 2014.
[23] 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 Transactions on Smart Grid, vol.5, no.1, pp.102–111, 2014.
[24] A. Takizawa, Y. Takechi, A. Ohta, N. Katoh, T. Inoue, T. Horiyama, J. Kawahara, and S. Minato, "Enumeration of region partitioning for evalcuation planning based on zdd," Proc. of International symposium on Operation Research & its Applications (ISORA2013), pp.64–71, 2014.
[25] J. Kawahara, T. Horiyama, K. Hotta, and S. Minato, "Generating all patterns of graph partitions within a disparity bound," Proc. of the 11th International Workshop of Algorithms and Computation (WALCOM2017), (LNCS 10167, Springer), vol.10167, pp.119–131, 2007.
[26] Japan Society for the Promotion of Science (JSPS), KAKENHI(S) 15H05711: Research on Core Algorithms for Discrete Structure Manipulation Systems, 6 2015. https://www.jsps.go.jp/english/e-grants/grants07_2015.html#sogo.

Shin-ichi Minato is a Professor at the Graduate School of Information Science and Technology, Hokkaido University. He received the B.E., M.E., and D.E. degrees in Information Science from Kyoto University in 1988, 1990, and 1995, respectively. He worked for NTT Laboratories from 1990 until 2004. He was a Visiting Scholar at the Computer Science Department of Stanford University in 1997. He joined Hokkaido University as an Associate Professor in 2004, and has been a Professor since October 2010. He also serves a Visiting Professor at National Institute of Informatics from 2015. His research interests include efficient representations and manipulation algorithms for large-scale discrete structures, such as Boolean functions, sets of combinations, sequences, permutations, etc. He served a Research Director of JST ERATO MINATO Discrete Structure Manipulation System Project from 2009 to 2016, and now he is leading JSPS KAKENHI(S) Project until 2020. He is a senior member of IEICE and IPSJ, and is a member of IEEE and JSAI.