INVITED PAPER *Special Section on Discrete Mathematics and Its Applications*

International Competition on Graph Counting Algorithms 2023

Takeru INOUE[†,††a)], ***Member***, **Norihito YASUDA**[†], ***Nonmember***, **Hidetomo NABESHIMA**[†††], ***Member***, **Masaaki NISHINO**[†], ***Nonmember***, **Shuhei DENZUMI**[†], ***Member***, ***and*** **Shin-ichi MINATO**[††††], ***Senior Member***

SUMMARY This paper reports on the details of the International Competition on Graph Counting Algorithms (ICGCA) held in 2023. The graph counting problem is to count the subgraphs satisfying specified constraints on a given graph. The problem belongs to #P-complete, a computationally tough class. Since many essential systems in modern society, e.g., infrastructure networks, are often represented as graphs, graph counting algorithms are a key technology to efficiently scan all the subgraphs representing the feasible states of the system. In the ICGCA, contestants were asked to count the *paths* on a graph under a length constraint. The benchmark set included 150 challenging instances, emphasizing graphs resembling infrastructure networks. Eleven solvers were submitted and ranked by the number of benchmarks correctly solved within a time limit. The winning solver, TLDC, was designed based on three fundamental approaches: backtracking search, dynamic programming, and model counting or #SAT (a counting version of Boolean satisfiability). Detailed analyses show that each approach has its own strengths, and one approach is unlikely to dominate the others. The codes and papers of the participating solvers are available: https://afsa.jp/icgca/.

key words: *graph counting algorithms, algorithm competitions, and ICGCA*

1. Introduction

Complex modern societal systems, composed of various components like infrastructure networks, are often represented as graphs to capture their underlying structures. These systems are usually operated or controlled to satisfy predefined constraints specific to each system, in order to function effectively. The constraints are often represented as a set of feasible *subgraphs* on the graph representation of the system; e.g., for an infrastructure network to function properly, operators represent the network topology as a graph and ensure the existence of a path, a subgraph connecting a supply source and a consumption point. Examining feasible system states, assessing their probabilities, and searching for the optimal state require comprehensive management of the set of feasible subgraphs. As a result, the problem of counting feasible subgraphs, known as the graph counting (GC) problem, emerges as the most fundamental form of algorithms that efficiently scan all the feasible subgraphs [1]. Recently, GC algorithms and their variants have found practical applications, especially in domains like infrastructure networks, which require examining every state without overlooking them [2]. Examples of applications include infrastructure networks such as power [3], [4], communications [5], and railroads [6]; floor planning and disaster evacuation [7], [8]; propagation phenomena in social networks [9] and epidemics [10]; internal networks of supercomputers [11]; and institutional design such as electoral districting [12]. To address the growing demand for GC algorithms and to further foster them, the International Competition on Graph Counting Algorithms (ICGCA) was held in 2023.

1.1 Literature Survey on Graph Counting Algorithms

The GC problem is known to be #P-complete, a computationally tough class [13]. Since the number of paths can be exponential with respect to the graph size, efficient algorithms have been enthusiastically studied for decades. This paper highlights the following three approaches to GC algorithm research.

- Backtracking (BT): Initial research on GC, which emerged in the 1960s and 1970s, relied on pure algorithmic approaches that visit and output feasible subgraphs one at a time, using search methods such as *backtracking* [14], [15]. This approach is relevant to the limited memory capacity available at the time and inherently requires time proportional to the number of feasible subgraphs. Thus, it does not scale well for instances with a large number of subgraphs.
- Dynamic programming (DP): Since the end of the 20th century, efficient counting methods using *dynamic programming* on decision diagrams have been intensively studied [16], [17]. These methods successfully handle an astronomical number of subgraphs in a graph with hundreds of edges. In the early stages of DP research, algorithms were developed for specific problems like network reliability evaluation [18] and path enumeration [19]. Common features among these algorithms were identified, and a versatile algorithmic framework called "frontier-based search" (FBS) [20] was developed, along with publicly released implementations

Manuscript received September 19, 2023.
Manuscript publicized January 15, 2024.
[†]NTT Communication Science Laboratories, NTT Corporation, Kyoto-fu, 619-0237 Japan.
[††]NTT Network Innovation Laboratories, NTT Corporation, Yokosuka-shi, 239-0847 Japan.
[†††]Graduate Faculty of Interdisciplinary Research, University of Yamanashi, Kofu-shi, 400-8511 Japan.
[††††]Graduate School of Informatics, Kyoto University, Kyoto-shi, 606-8501 Japan.
a) E-mail: takeru.inoue@ieee.org
DOI: 10.1587/transfun.2023DMP0006

such as TdZdd[†] and Graphillion[††] [21]. Thanks to their scalability and generality, these implementations have been utilized for many of the above applications.

- Model Counting (MC): In the field of artificial intelligence (AI), *model counting* is a fundamental problem encompassing graph counting [22]. MC is sometimes referred to as #SAT due to its counting version of the satisfiability (SAT) problem. In MC, systems are represented as Boolean formulae, and the goal is to count the feasible models that satisfy the given constraints. MC algorithms can be applied to GC by representing graphs and their constraints as Boolean formulae. Since interest has been rising in counting the number of models for practical problems, the MC competition[†††] [23] was launched in 2020. Note that since MC involves several techniques, we indicate which one is employed in each solver.

1.2 Related Algorithm Competitions and ICGCA Goals

Competitions serve various purposes, from specific problems for industry needs to fundamental advancements for academic research. In the former case, competitions sometimes provide large cash awards like the Netflix Prize [24]. On the other hand, mathematical problem solving like GC usually follows the latter purpose and often lacks specific monetary rewards. We follow a direction in the community of mathematical problem solving where many competitions and challenges have already been organized: SAT[††††] [25] (22 editions), SMT[†††††] [26] (19 editions), MaxSAT[†††††††] [27] (18 editions), CSP[*] (16 editions), QBF[**] [28] (13 editions), DIMACS[***] [29] (12 editions), PACE[****] [30] (8 editions), MC [23] (3 editions), and CoRe[*****] (2 editions).

Despite the success of MC competition in addressing more general problems than GC, we organized a GC competition for the following reasons. GC algorithms have features distinct from MC algorithms, enhancing computational efficiency greatly. This is implicitly supported by the fact that GC is used instead of MC in many application areas. On the other hand, GC and MC (and BT) share several technical features, and they could improve each other's performance through positive interactions. Unfortunately, these approaches have been pursued relatively independently and have had limited mutual influence.

[†] https://github.com/kunisura/TdZdd/
[††] https://github.com/takemaru/graphillion/
[†††] https://mccompetition.org/
[††††] http://www.satcompetition.org/
[†††††] https://smt-comp.github.io/2023/
[†††††††] https://maxsat-evaluations.github.io/2023/
[*] https://www.minizinc.org/challenge.html
[**] http://www.qbflib.org/index_eval.php
[***] http://dimacs.rutgers.edu/programs/challenge/
[****] https://pacechallenge.org/
[*****] https://core-challenge.github.io/2023/

ICGCA's primary goal is to bring the three approaches (BT, DP, and MC) onto the same platform, clarify their strengths and weaknesses, and foster the development of better algorithms through their interaction. The competition was carefully designed to prevent any one approach from gaining an undue advantage. The ICGCA also aims to identify new challenging benchmarks and promote novel approaches. We hope that active participation, collaboration, and long-term improvements will broaden the applicability of GC algorithms in practice and inspire many new applications.

The rest of this paper is organized as follows. Section 2 overviews the ICGCA and Sect. 3 describes the benchmark set. Section 4 presents the submitted solvers and their results, which are analyzed in detail in Sect. 5. The paper is concluded in Sect. 6.

2. ICGCA Overview

This section provides an overview of ICGCA, starting with the following timeline. The ICGCA website was opened in April 2023. Contestants received the benchmark set when they registered on the ICGCA website. They developed their solvers and submitted them to the ICGCA organizers by July 17, 2023. The solvers were evaluated by the organizers, and the results were announced on September 7, 2023.

Section 2.1 defines the problem. Section 2.2 describes the scoring method, and Sect. 2.3 shows the evaluation environments. Note that we have tried to keep the competition simple since this is the first competition on GC algorithms.

2.1 Problem

The ICGCA presented two variants of *path counting* problems. One specifies a pair of path terminals.

Problem 1 (PCS: Path Counting for a Single pair)**.** Given an undirected simple graph $G = (V, E)$, a pair of vertices $\{s, t\} \in V \times V$, and a maximum path length ℓ, count the exact number of simple paths between s and t whose length is at most ℓ.

Note that the path length is defined as the number of edges on the path.

The other variant specifies no vertex pairs and counts the paths between *all* the vertex pairs.

Problem 2 (PCA: Path Counting for All pairs)**.** Given an undirected simple graph $G = (V, E)$ and a maximum path length ℓ, count the exact number of simple paths between all $\{s, t\} \in V \times V$, where $s \neq t$ and the path length is at most ℓ.

Figure 1 shows an example of a PCS instance. A terminal pair (vertices 1 and 3) and a maximum path length of two are given. Since there are two paths between the terminal pair with at most two edges, the answer is *two* paths. For a PCA instance (if no terminal pair were specified), the

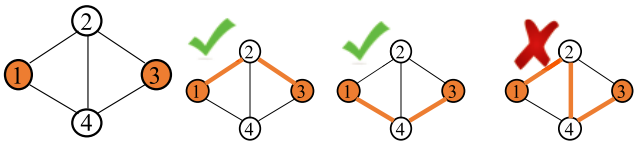

Fig. 1 Example of PCS instance: terminal pair is 1 and 3, and maximum path length is two edges.

```
p edge 4 5
e 1 2
e 1 4
e 2 3
e 2 4
e 3 4
l 2
t 1 3
```

Fig. 2 Input for Fig. 1 in extended DIMACS file format.

answer is 13 paths.

We focus on *undirected* graphs in this competition, because infrastructure networks, a primary application of GC algorithms, are often modeled as either undirected or bidirectional graphs. In addition, past work has mainly focused on undirected graphs, and competitions for directed graphs are considered premature.

We also concentrate on *paths* as a subgraph structure to be counted. Many other options are possible: cycles, trees, forests (sets of trees), matchings (independent vertex/edge sets), cliques, connected components, and partitions. However, paths are often examined in practical problems, especially for connecting supply sources and consumers in infrastructure networks. In addition, path counting remains an active research area [31], suggesting substantial room for improvement.

We impose a single constraint: *maximum path length*. Although we can devise various constraints, e.g., Hamiltonicity and way-pointing, too many constraints could overwhelm the contestants. On the other hand, without any constraints, the amount of feasible paths might also become overwhelming, disadvantaging the BT approach whose computational complexity strongly depends on path counts. In practical scenarios, it is often acceptable to ignore large detour paths, and so we chose the maximum path length as an essential constraint.

2.2 Scoring Methods

The winner is the solver that provides correct answers for the largest number of benchmarks. Note that each benchmark is allocated a time budget of 600 seconds in wallclock time. The benchmark set includes 150 instances: 100 for PCS and 50 for PCA.

#P problems, including GC, pose a severe challenge to scoring methods; while NP problems can be verified in polynomial time, for #P problems, an answer's correctness can only be verified by solving the instance itself. Although excluding hard instances from the benchmark set could resolve the verification issue, contestants would not be encouraged to develop a solver for hard instances. The MC competition, which faced the same issue, treated any answer for unknown benchmarks as "correct" [23]. The ICGCA employs a fairer scoring method. Since wrong answers for benchmarks with many paths are unlikely to match by chance, answers are labeled as "correct" if they match among multiple solvers. If the answers do not match (either the benchmark was solved by a single solver or the answers were split among solvers), they are labeled "tentatively correct." Then, the scores are presented with a range: ["# of correct answers," "# including tentatively correct answers"]. Therefore, there was a risk that the rankings might not be uniquely fixed, but the actual competition resulted in a uniquely fixed order. Since the answer verification is imperfect as described, solvers with incorrect answers were not penalized.

The ICGCA does not employ the PAR-2 system, a well-known scoring method used in the SAT competition [25], which reflects the time required for answering each benchmark in the rankings. This is for keeping the competition simple, but the PAR-2 system will be used to analyze competition results in Sect. 5.

2.3 Solver Requirements and Evaluation Environments

An instance is given in an extended DIMACS file format. Each line begins with a letter that defines the rest of the line. The allowed lines are as follows.

- `p`: The problem description must have form `p edge n m`, where `n` is the number of nodes (numbered 1..n) and `m` is the number of edges.
- `e`: An edge must have form `e v1 v2`, where `v1` and `v2` are the endpoints of the edge.
- `l`: The maximum path length must have form `l len`, where `len` is the maximum path length.
- `t`: The pair of path terminals must be of form `t v1 v2`, where `v1` and `v2` are the path terminals. This line is omitted for PCA instances.

Figure 2 shows the input for Fig. 1. Solvers are required to output just a number of paths to `stdout`. The solver codes are made public after the competition.

Solvers were evaluated on an ICGCA organizer computer with 12 CPU cores (Intel Core i7-1260P, up to 4.70 GHz) and 64 GB RAM, running Ubuntu Server 22.04 LTS. They were allowed to use multiple CPU cores, and portfolio solvers implementing multiple algorithms were also acceptable. Although the SAT competition had separate tracks for sequential, parallel, and portfolio solvers, the ICGCA had only a single track for parallel and portfolio solvers. This decision was made to avoid excessively restricting contestants' strategies without hosting multiple tracks.

To ensure correct scoring, we conducted the following evaluation. First, the pool of solvers was narrowed down to

the top three, which were then run through multiple rounds of evaluation on different computers to validate the consistency of their results.

3. Benchmarks

Since no benchmark set had been developed for GC competitions, we first established guidelines for preparing one.

- Focus on real-world graphs and their close counterparts, emphasizing infrastructure networks, i.e., graphs with at most a few thousand edges and sparse connectivity.
- Avoid biasing any particular approach by conducting preliminary experiments.
- Cover a spectrum of complexity from simple to challenging instances (those can not be easily solved by existing solvers, but can be solved with new innovations).

The preliminary experiments are described in Sect. 3.1. The selection of benchmarks and their statistics is presented in Sect. 3.2.

3.1 Preliminary Experiments

To measure the hardness of various instances, we conducted preliminary experiments. We prepared test solvers for the BT and DP approaches, although we found no MC solver that is easily applicable to GC instances. The BT test solver implements Yen's k-shortest path algorithm[†] [32], which visits paths in ascending order of length until the maximum path length ℓ is reached. The DP test solver employs Graphillion [21], which processes edges in the order of the path decomposition [33]. Note that these test solvers are easy to use with reasonable computational efficiency, although they are not state-of-the-art.

We prepared 1360 test instances. The graphs were chosen from typical structures: complete graphs, 2D grid graphs, path-like graphs, and tree-like graphs. A path-like graph connects small cliques with a few edges in a path-like manner. A tree-like graph is ditto. Path-/tree-like graphs mimic infrastructure networks, where regional subnetworks are connected by long-distance links. Note that although complete graphs are quite different from infrastructure networks, we included a few to understand the performance of the test solvers. The maximum path length ℓ was randomly chosen from the range of the graph diameter to the number of vertices (smaller values were more likely to be chosen). For PCS instances, the most distant vertex pair, $\{s, t\}$, was selected. The computer used for the preliminary experiments had 32 GB of memory, less than for real environments. The test solvers were run in a single thread.

Table 1 shows the results of the preliminary experiments. 392 instances were solved by both the BT and DP test solvers; 573 instances were not solved by either one. Overall, the BT test solver was impacted by the maximum path length, because a larger maximum path length yields more paths. The DP test solver was affected by *pathwidth*, which is a graph parameter that indicates the dissimilarity to a path: one for a path graph and $n - 1$ for a complete graph with $n = |V|$. The strong dependence of the DP approach on the pathwidth was known from the literature [33].

Table 1 Results of preliminary experiments.

	Solved by BT	Unsolved by BT
Solved by DP	392	283
Unsolved by DP	112	573

Table 2 Number of benchmarks with real and synthesized graphs for PCS and PCA.

	Real graphs	Synthesized graphs
PCS	15	85
PCA	30	20

The results were used to build a machine learning model, which was used to select instances with diverse hardness levels without favoring one approach over another. The feature space consists of the pathwidth, ℓ, n, and the graph's mean degree. The model was built using the support vector machine.

3.2 Benchmark Selection and Statistics

We selected benchmarks from real and synthetic graphs and show the number of selected benchmarks with real and synthetic graphs in Table 2.

First, we describe the real graphs included in the benchmark set. We collected 45 graphs, which are sparse with at most a few thousand edges, from communication networks (TopologyZoo[††] [34], RocketFuel[†††] [35], SNDlib[††††], and JPN48[†††††]) and software engineering domains (Rome[††††††]). These graphs have been used in recent studies of GC algorithms [36], [37]. The 45 graphs were associated with each benchmark, resulting in 45 benchmarks. No selection based on the hardness model was made for real graphs to include as many of them as possible in the benchmark set.

The remaining 105 benchmarks were created with synthetic graphs. To increase the diversity of those used in the preliminary experiments, we randomly modified them by removing three or fewer edges from the graph and rewiring 25 or fewer edges. 105 synthetic benchmarks were selected based on the hardness model. 59 benchmarks were predicted as unlikely to be solved by either the DP or BT test solvers, which implies they are quite challenging. 73 bench-

[†] https://github.com/yan-qi/k-shortest-paths-cpp-version/

[††] http://topology-zoo.org/

[†††] https://research.cs.washington.edu/networking/rocketfuel/

[††††] http://sndlib.zib.de/home.action

[†††††] https://www.ieice.org/cs/pn/jpn/jpnm.html

[††††††] http://www.graphdrawing.org/data.html

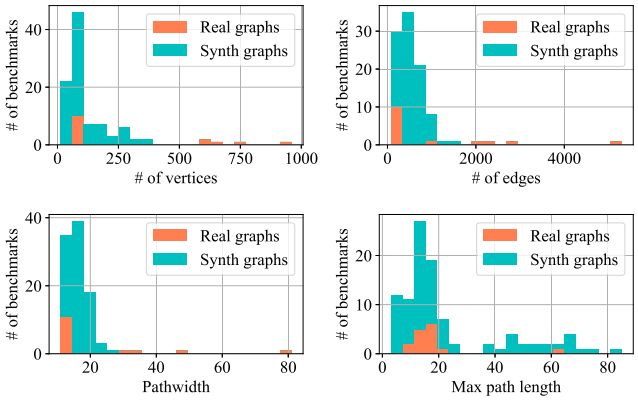

Fig. 3 Histograms for instance parameters in PCS benchmark set.

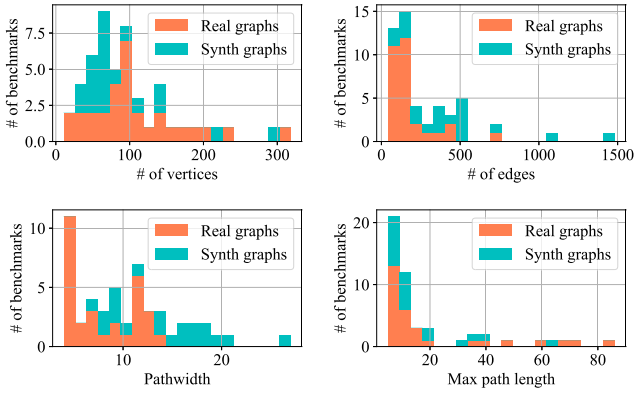

Fig. 4 Histograms for instance parameters in PCA benchmark set.

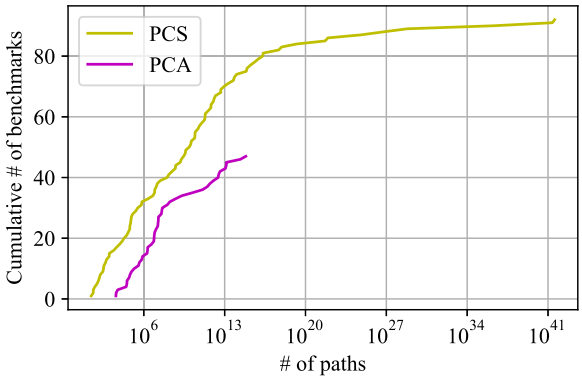

Fig. 5 Cumulative path counts.

Table 3 Solver scores.

	Overall	PCS	PCA
TLDC	[135, 136]	[90, 91]	45
diodrm	[131, 134]	[86, 87]	[45, 47]
TAG	126	83	43
Drifters	110	71	39
NaPS+GPMC	105	72	33
KUT_KMHT	99	61	38
asprune	65	40	25
castella	39	29	10
dimitri	[11, 12]	10	[1, 2]
Cypher	4	2	2
PCSSPC	0	0	0

marks were unlikely to be solved by the DP test solver, and 79 were unlikely to be solved by the BT test solver: roughly the same number.

Next we examine the statistics of the benchmark set. Figures 3 and 4 show the parameter distributions for the PCS and PCA benchmark sets. The PCA benchmarks, which count the paths for all the vertex pairs, have smaller parameters than the PCS benchmarks. Although all the parameters are mainly distributed around smaller regions, we found some very challenging benchmarks in the larger regions. In particular, some PCS benchmarks with real graphs have extremely large parameters, because the hardness is not controlled for real graphs. Figure 5 shows the cumulative path counts for the 139 benchmarks that were solved by at least one solver. The PCS benchmarks have many more paths, 10^{41} paths at most, due to the larger parameters. The PCA benchmarks have fewer paths, even though the paths for all the vertex pairs are counted.

4. Solvers and Results

Section 4.1 presents the ICGCA result, and Sect. 4.2 overviews the participating solvers.

4.1 Results

The ICGCA received 11 solvers from four countries: Austria, Germany, Japan, and USA. Table 3 ranks them. As noted in Sect. 2.2, the scores (the number of correctly solved benchmarks) are denoted in square brackets if they have a range. The best-performing solver overall on the combination of PCS and PCA benchmarks is `TLDC`, and the runner-up is `diodrm`; `diodrm` solved the most instances for PCA. Unfortunately, the bottom three solvers, `dimitri`, `Cypher`, and `PCSSPC`, did not perform as expected; they crashed during execution or yielded more wrong answers than correct ones. Therefore, we excluded them from the following analyses.

Figure 6 shows the cumulative solved benchmark plot together with the Virtual Best Solver (VBS). The VBS is a fictitious solver consisting of all the solvers that actually participated in the competition and an oracle which, given an input instance, invokes the solver that performed the best on that instance. The VBS performance highlights a certain upper bound on the performance achievable by the participating solvers. For the PCS benchmark set, `TLDC` maintained the lead from three seconds onward. For the PCA benchmark set, `diodrm` showed surprising performance comparable to the VBS, demonstrating its overwhelming strength in PCA. `TAG` showed great start-up and solved more instances in the sub-second range. Note that the steep increase for `diodrm` and `Drifters` at 1 and 100 seconds reflects the

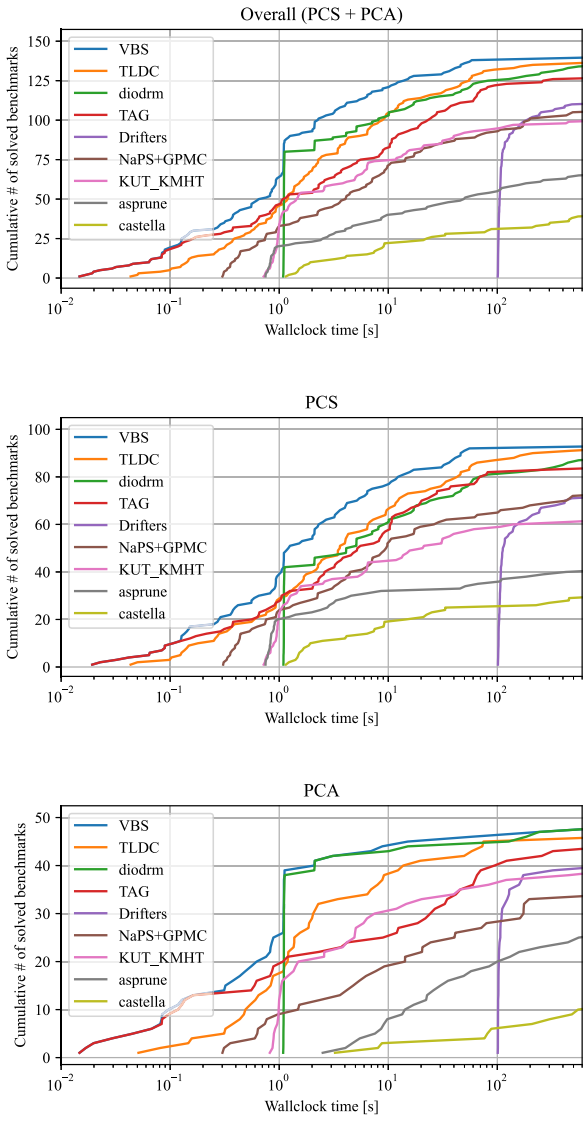

Fig. 6 Performance of top eight solvers and VBS.

fixed amount of preprocessing time spent on them.

4.2 Solvers

This subsection overviews the top eight solvers. The following description is based on short papers submitted by the contestants and our glances at their code. The codes and papers for all the solvers are available online: https://afsa.jp/icgca/. The approaches employed by each solver are shown in Table 4.

- `TLDC` (Too Long; Didn't Count) is a portfolio solver consisting of all three approaches: BT, DP, and MC. This solver performs BT in a depth-first search (DFS) manner, but it is much more efficient than the traditional naive implementation thanks to the techniques described later. It also employs FBS, a DP method [20], with an edge processing order based on the tree decomposition computed by htd[†] [38]. This solver has several ideas that make the solver more efficient, which are elegantly applied to both BT and DP methods, e.g., the search space is pruned based on the path length, and graph isomorphism tested by nauty[††] [39] reduces the cache entries. In addition, TLDC uses MC-derived ideas, e.g., it employs an ASP solver named Clingo[†††] to check whether the maximum length of a path can be achieved. `TLDC` is well-designed based on all three approaches and won the competition.

Table 4 Solver approaches.

	Portfolio	BT	DP	MC (via)	Other
TLDC	✓	✓	✓	✓(ASP)	
diodrm	✓	✓	✓		
TAG	✓		✓✓		
Drifters			✓		
NaPS+GPMC				✓(PB)	
KUT_KMHT	✓		✓✓		✓
asprune				✓(ASP)	
castella				✓(CSP)	
dimitri					✓
Cypher			✓		
PCSSPC		✓			

- `diodrm` is a portfolio solver following the BT and DP approaches. Like `TLDC`, this solver also employs DFS and FBS, but the FBS's edge order is determined by the community-based order. A large instance is partitioned by a method called *meet in the middle* to facilitate precomputation and parallelization. `diodrm` overwhelmed the other solvers for PCA by introducing the following innovative DFS techniques. A search is performed not for each vertex pair but only from each vertex, which also allows easy parallelization. The search is further accelerated by exploiting the fact that, for a graph with small neighborhood diversity, a path is still formed even if the vertices in an adjacent set are swapped.

- `TAG` is a portfolio solver consisting of two DP methods: FBS and HAMDP (Hamiltonian DP). FBS introduces a pruning strategy with path length and vertex degree, based on a traditional FBS design (implemented with TdZdd and the edge order is determined based on path decomposition). HAMDP is an interesting extension of a Hamiltonian-paths algorithm and outperforms FBS on some PCA benchmarks, according to their paper.

- `Drifters` implements FBS with TdZdd and utilizes Chokudai search to perform path decomposition to determine the edge processing order.

- `NaPS+GPMC` is an MC solver designed for path counting problems. First, it produces Pseudo-Boolean (PB) constraints of a path, which are converted into CNF form

[†] https://github.com/mabseher/htd/
[††] https://pallini.di.uniroma1.it/
[†††] https://potassco.org/clingo/

using NaPS[†] [40]. The number of models (paths) is counted with a variant of the projection model counting solver GPMC[††]. It was modified for the path counting by removing the time-consuming preprocessing, adopting a new variable choice strategy along with a path, and introducing hierarchical counting strategy.

- `KUT_KMHT` is a portfolio solver consisting of two DP methods and another type of counting method. The two DP methods are FBS with TdZDD and a bounding algorithm originally designed for network reliability evaluation [41]. The other method employs an exact cover algorithm[†††] modified for path counting.

- `asprune` is an MC solver. First, this solver converts a given instance to ASP facts with an ASP encoding for path counting. Then, the solver *enumerates* all the paths with Clingo. This indicates that even an explicit enumeration technique could achieve this level of results.

- `castella` is an MC solver. For a given instance, it generates a CSP, which is translated into a SAT instance using Sugar[††††] [42]. Then, the solver counts the path with GPMC, as does `NaPS+GPMC`. The gap between `castella` and `NaPS+GPMC` is due to the latter's several techniques specialized in path counting.

The top three solvers, `TLDC`, `diodrm`, and `TAG`, are all portfolio types. It is interesting that `Drifters` and `NaPS+GPMC` implement the different approaches but they performed equally well. In Sect. 5.6, we compare these two approaches by considering the two solvers as representatives of each approach. `KUT_KMHT` and `dimitri` implement approaches beyond our expectations, which are categorized as "others". For example, `dimitri` employ an algebraic approach based on graph isomorphism.

5. Analyses of Results

This section provides an additional analysis of the results by delving deeper beyond the rankings. Section 5.1 examines the results by graph type, and Sect. 5.2 discusses the PAR-2 scores. Section 5.3 analyzes the contributions of each solver to the VBS. Section 5.4 investigates the impact of instance parameters, and Sect. 5.5 discusses the similarity of the participating solvers. Section 5.6 compares the DP and MC approaches.

5.1 Scores Per Graph Types

This subsection examines the results for real and synthetic graphs separately (Fig. 7). The solvers are ordered on the

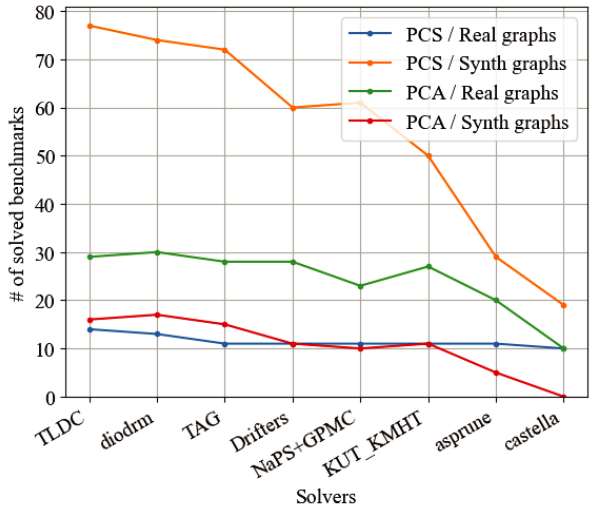

Fig. 7 Number of solved benchmarks per graph type.

x-axis by their overall ranking. For all four types, combinations of PCS/PCA and real/synthetic graphs, the lines decline from left to right; the rankings do not change significantly among the types. For the PCS benchmarks with real graphs, the differences among solvers are small, because many instances are fairly easy, except for a few (Fig. 3). For the PCA benchmarks with real graphs, `KUT_KMHT` performed well compared to the other types. Effective techniques might be found for such instances by exploring the unique innovations of `KUT_KMHT`.

5.2 PAR-2 Scores

This subsection ranks the solvers based on the PAR-2 scoring system [25]. The PAR-2 system assigns as many points as the amount of time (in seconds) required to answer the instance, while it assigns twice the time limit (1,200 points) if the instance was not solved. This means that the lower a solver's score is, the better its performance.

Figure 8 shows the PAR-2 scores accumulated for the 139 benchmarks solved in the competition. Compared to the rankings by the number of solved benchmarks, `Drifters`'s ranking worsened because it spent 100 seconds on preprocessing for each benchmark (note that `Drifters` would not have relied on a fixed-time preprocessing strategy if the ICGCA had employed the PAR-2 system). Surprisingly, `diodrm` was not penalized in the least by the PCA benchmarks, demonstrating its overwhelming strength for PCA. In the following analyses, the computation time for the unsolved benchmarks is considered to be 1,200 seconds according to the PAR-2 system.

5.3 Contributions to Virtual Best Solver

The VBS solves all the instances solved by at least one solver with the best time. By quantifying how much each participating solver contributes to the VBS performance, we can identify which solver is the most important in the state-

[†] https://www.trs.css.i.nagoya-u.ac.jp/projects/NaPS/
[††] https://git.trs.css.i.nagoya-u.ac.jp/k-hasimt/GPMC/
[†††] https://github.com/aaw/cover/
[††††] https://cspsat.gitlab.io/sugar/

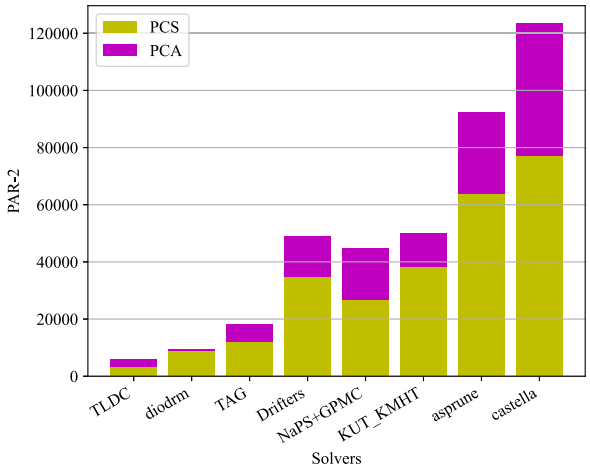

Fig. 8 PAR-2 scores (lower is better).

of-the-art for path counting. We examine the following three metrics derived from the notion of VBS.

- VBS-1 "The fastest takes it all": For each solver, we count the number of benchmarks in which it was the fastest to solve an instance.
- VBS-2 "Time aware, but proportional": A solver S solving an instance I in time T_I^S obtains T_I^{VBS}/T_I^S, where T_I^{VBS} is the computation time of VBS on I.
- VBS-3 "Split the point for solving": A solver S solving an instance I obtains $1/|\mathcal{S}_I|$, where $\mathcal{S}_I$ is the set of solvers solving I.

Figure 9 plots the three metrics. For VBS-1 and VBS-2, which both involve computation time, `diodrm` and `Drifters` with a fixed amount of preprocessing time have lower ranks (nevertheless, `diodrm` got the most points for PCA in VBS-1). On the other hand, `KUT_KMHT` improved its ranking because it showed comparable performance with the third solver (`TAG`), up to 10 seconds (Fig. 6). For VBS-3, `diodrm` slightly outperforms `TLDC` because there are three benchmarks solved only by `diodrm`, while just one benchmark was solved only by `TLDC`.

5.4 Impact of Instance Parameters

This subsection examines the impact of the instance parameters on the computation time. The parameters investigated include the number n of vertices, the number m of edges, the pathwidth, and the maximum path length ℓ. VBS's computation is used as a representative for all the solvers. Table 5 shows the Pearson correlation coefficients between the parameters and the computation time. For the PCS benchmark set, the maximum path length has a significant impact, followed by the pathwidth. This is because pruning strategies based on path length, which are employed by multiple solvers, would work effectively only for short paths. For the PCA benchmark set, pathwidth has the largest impact, while the maximum path length has a much weaker impact. This could be because paths between very close vertices are less constrained by the maximum path length.

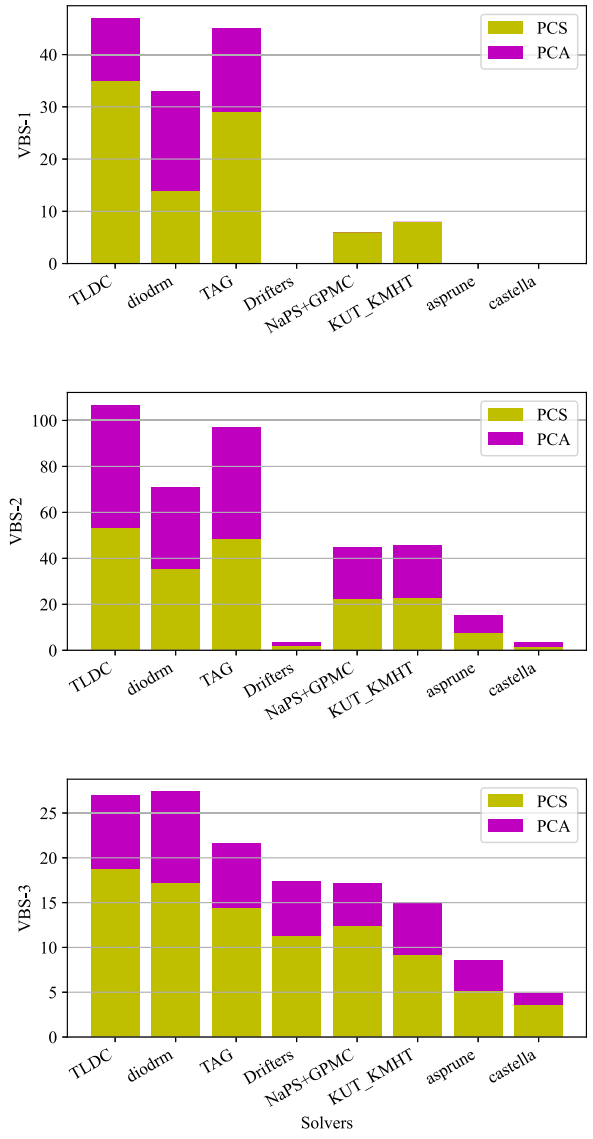

Fig. 9 VBS metrics.

Table 5 Correlation between instance parameters and VBS computation time.

	Overall	PCS	PCA
# n of vertices	0.22	0.23	0.16
# m of edges	0.28	0.28	0.35
Pathwidth	0.32	0.31	0.39
Max path length ℓ	0.47	0.54	0.29

5.5 Solver Similarity

To investigate the similarity among the solvers, we define a similarity metric based on the computation times. We first removed 11 benchmarks that were not solved by any solver. For the remaining 139 benchmarks, a PAR-2 score is as-

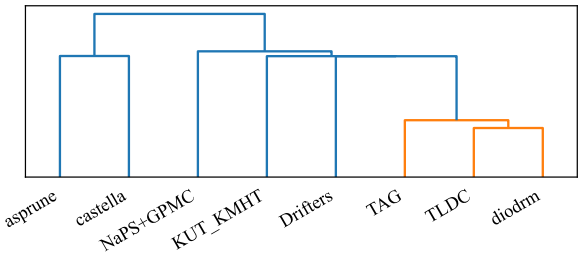

Fig. 10 Dendrogram based on computation-time similarity of solvers.

Table 6 Score improvement by portfolios between DP and MC solvers.

Portfolio: solver (approach)		# solved b'marks	Improvement
Drifters (DP)	NaPS+GPMC (MC)	122	+12
KUT_KMHT (DP)	NaPS+GPMC (MC)	121	+16
Drifters (DP)	KUT_KMHT (DP)	114	+4

signed to each solver-benchmark pair; i.e., each solver S is thus associated with the PAR-2 scores $S_1, \ldots, S_{139}$. The similarity of two solvers, S and S', normalized to the interval $[0, 1]$, is defined as,

$$\text{similarity}(S, S') = 1 - \frac{\sum |S_i - S'_i|}{139 \cdot 1200}. \quad (1)$$

The solvers are clustered based on their similarity and are illustrated as a dendrogram (Fig. 10). The height at which two solvers or clusters is joined reflects their similarity. For example, `TLDC` and `diodrm` solved most of the benchmarks, so they are joined low in the dendrogram. The top three solvers (`diodrm`, `TLDC`, and `TAG`) are clustered in the dendrogram. Interestingly, all the DP solvers (`diodrm`, `TLDC`, `TAG`, `Drifters`, and `KUT_KMHT`) are classified in the same subtree exclusively against the MC solvers, implying that each approach has its own characteristics.

5.6 Comparison between DP and MC Solvers

This subsection examines the dissimilarity between the DP and MC approaches by considering `Drifters` and `NaPS+GPMC` as representatives of both approaches. `KUT_KMHT`, another DP solver that solved roughly an equal number of benchmarks as both solvers, is also employed for comparison. The three solvers, `Drifters`, `NaPS+GPMC`, and `KUT_KMHT`, respectively solved 110, 105, and 99 benchmarks (Table 3).

We investigate how much the scores could be improved by a portfolio solver that employs two of the three solvers; the larger the score improvement, the more different characteristics the solvers have. As shown in Table 6, a portfolio of `Drifters` and `NaPS+GPMC` increases the score by 12 benchmarks, while a portfolio of DP solvers (`Drifters` and `KUT_KMHT`) increases it just by 4 benchmarks. Another portfolio of DP and MC (`NaPS+GPMC` and `KUT_KMHT`) greatly increases the score as well, by 16 benchmarks. These results show that the DP and MC approaches have different competencies, and that both are essential for the development of graph counting algorithms.

6. Conclusions

This paper minutely described the first GC competition, the ICGCA. The authors thank all the contestants for their enthusiasm and their strong contributions. We are pleased that so many solvers performed so well on the challenging benchmarks.

Since no competition on GC has ever been held before, we conducted the preliminary experiments to select the benchmark set of reasonable hardness levels. However, the winning solver solved more than 90% of the benchmarks, so it could have been more challenging.

For #P problems including GC, verifying the correctness of a given answer is not trivial, unlike NP problems. Thus, the scores could be given with a range, resulting in unfixed rankings; future competitions on #P problems must deal with this ambiguity.

In this competition, all the contestants submitted just a single version of their solvers, although in other competitions contestants often submitted multiple versions with different configurations. This is because the final benchmark set was made available online in ICGCA and the contestants could adjust their configurations before submission. The pros and cons of this decision are open to debate.

The main goal of the ICGCA was to provide a unified stage for the three independently developed approaches and to encourage their interaction. We believe that this goal was accomplished to some extent. Only the winning solver `TLDC` implements all three approaches, suggesting that each approach has its own potential. Our analysis showed that both the DP and MC approaches have different strengths, so we expect them to fuel the future development of GC algorithms. On the other hand, unfortunately, no detailed analysis of the BT approach was performed, because the `PCSSPC` solver, which is primarily based on the BT approach, failed to work properly. However, since experiments by the `TLDC` developers showed that the most contributing approach in `TLDC` is BT, we also look forward to future contributions from this approach.

As the first competition on GC algorithms, we kept the competition simple. In the future edition, it is worth considering to hold a main (sequential) track and a parallel track separately, possibly along with an approximate track. It is also worth employing the PAR-2 scoring system to reflect the computation time in the ranking. Anyone interested in the competition is welcome to contact us and join the discussion.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers 20H05961, JP20H05963, and 20H05964.

References

[1] S. Minato, "Power of enumeration — Recent topics on BDD/ZDD-based techniques for discrete structure manipulation," IEICE Trans.

Inf. & Syst., vol.E100-D, no.8, pp.1556–1562, Aug. 2017.
[2] L. Xing and S.V. Amari, Binary Decision Diagrams and Extensions for System Reliability Analysis, John Wiley & Sons, 2015.
[3] 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.
[4] L. Duenas-Osorio, K. Meel, R. Paredes, and M. Vardi, "Counting-based reliability estimation for power-transmission grids," Proc. AAAI Conference on Artificial Intelligence, 2017.
[5] K. Nakamura, T. Inoue, M. Nishino, N. Yasuda, and S. Minato, "A fast and exact evaluation algorithm for the expected number of connected nodes: An enhanced network reliability measure," Proc. IEEE INFOCOM, 2023.
[6] J. Kawahara, T. Saitoh, H. Suzuki, and R. Yoshinaka, "Solving the longest oneway-ticket problem and enumerating letter graphs by augmenting the two representative approaches with ZDDs," Computational Intelligence in Information Systems: Proceedings of the Computational Intelligence in Information Systems Conference (CIIS 2016), pp.294–305, Springer, 2017.
[7] A. Takizawa, Y. Miyata, and N. Katoh, "Enumeration of floor plans based on a zero-suppressed binary decision diagram," International Journal of Architectural Computing, vol.13, no.1, pp.25–44, 2015.
[8] A. Takizawa, Y. Takechi, A. Ohta, N. Katoh, T. Inoue, T. Horiyama, J. Kawahara, and S. Minato, "Enumeration of region partitioning for evacuation planning based on ZDD," 11th International Symposium on Operations Research and its Applications in Engineering, Technology and Management 2013 (ISORA 2013), pp.1–8, 2013.
[9] T. Maehara, H. Suzuki, and M. Ishihata, "Exact computation of influence spread by binary decision diagrams," Proc. 26th International Conference on World Wide Web, pp.947–956, 2017.
[10] F. Ishioka, J. Kawahara, M. Mizuta, S. Minato, and K. Kurihara, "Evaluation of hotspot cluster detection using spatial scan statistic based on exact counting," Jpn. J. Stat. Data. Sci., vol.2, pp.241–262, 2019.
[11] M. Koibuchi, I. Fujiwara, F. Chaix, and H. Casanova, "Towards ideal hop counts in interconnection networks with arbitrary size," 2016 Fourth International Symposium on Computing and Networking (CANDAR), pp.188–194, IEEE, 2016.
[12] J. Kawahara, T. Horiyama, K. Hotta, and S.i. Minato, "Generating all patterns of graph partitions within a disparity bound," WALCOM: Algorithms and Computation: 11th International Conference and Workshops, WALCOM 2017, Hsinchu, Taiwan, March 2017, Proceedings 11, pp.119–131, Springer, 2017.
[13] L.G. Valiant, "The complexity of enumeration and reliability problems," SIAM J. Comput., vol.8, no.3, pp.410–421, 1979.
[14] R. Tarjan, "Enumeration of the elementary circuits of a directed graph," SIAM J. Comput., vol.2, no.3, pp.211–216, 1973.
[15] R.C. Read and R.E. Tarjan, "Bounds on backtrack algorithms for listing cycles, paths, and spanning trees," Networks, vol.5, no.3, pp.237–252, 1975.
[16] K. Sekine, H. Imai, and S. Tani, "Computing the Tutte polynomial of a graph of moderate size," Proc. ISAAC, pp.224–233, 1995.
[17] O. Coudert, "Solving graph optimization problems with zbdds," Proc. European Design and Test Conference. ED & TC 97, pp.224–228, IEEE, 1997.
[18] 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.
[19] D.E. Knuth, The Art of Computer Programming, Addison-Wesley, 2011.
[20] J. Kawahara, T. Inoue, H. Iwashita, and S.i. Minato, "Frontier-based search for enumerating all constrained subgraphs with compressed representation," IEICE Trans. Fundamentals, vol.E100-A, no.9, pp.1773–1784, Sept. 2017.
[21] T. Inoue, H. Iwashita, J. Kawahara, and S. Minato, "Graphillion: Software library for very large sets of labeled graphs," Int. J. Softw. Tools Technol. Transf., vol.18, no.1, pp.57–66, 2016.
[22] C.P. Gomes, A. Sabharwal, and B. Selman, "Model counting," Handbook of Satisfiability, pp.993–1014, IOS Press, 2021.
[23] J.K. Fichte, M. Hecher, and F. Hamiti, "The model counting competition 2020," ACM J. Exp. Algorithmics, vol.26, pp.1–26, 2021.
[24] R.M. Bell and Y. Koren, "Lessons from the netflix prize challenge," ACM SIGKDD Explorations Newsletter, vol.9, no.2, pp.75–79, 2007.
[25] N. Froleyks, M. Heule, M. Iser, M. Järvisalo, and M. Suda, "SAT competition 2020," Artificial Intelligence, vol.301, p.103572, 2021.
[26] T. Weber, S. Conchon, D. Déharbe, M. Heizmann, A. Niemetz, and G. Reger, "The SMT competition 2015–2018," J. Satisf. Boolean Model. Comput., vol.11, no.1, pp.221–259, 2019.
[27] C.M. Li and F. Manya, "Maxsat, hard and soft constraints," Handbook of Satisfiability, pp.903–927, IOS Press, 2021.
[28] P. Marin, M. Narizzano, L. Pulina, A. Tacchella, and E. Giunchiglia, "Twelve years of QBF evaluations: QSAT is PSPACE-hard and it shows," Fundamenta Informaticae, vol.149, no.1-2, pp.133–158, 2016.
[29] C. Demetrescu, A.V. Goldberg, and D.S. Johnson, The Shortest Path Problem: Ninth DIMACS Implementation Challenge, American Mathematical Soc., 2009.
[30] E. Großmann, T. Heuer, C. Schulz, and D. Strash, "The PACE 2022 parameterized algorithms and computational experiments challenge: Directed feedback vertex set," 17th International Symposium on Parameterized and Exact Computation (IPEC 2022), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.
[31] G. Punzi, A. Conte, R. Grossi, and A. Marino, "An efficient algorithm for assessing the number of *st*-paths in large graphs," Proc. 2023 SIAM International Conference on Data Mining (SDM), pp.289–297, 2023.
[32] E.Q. Martins and M.M. Pascoal, "A new implementation of Yen's ranking loopless paths algorithm," Quarterly Journal of the Belgian, French and Italian Operations Research Societies, vol.1, no.2, pp.121–133, 2003.
[33] Y. Inoue and S. Minato, "Acceleration of ZDD construction for subgraph enumeration via pathwidth optimization," Technical Report, TCS-TR-A-16-80, Division of Computer Science, Hokkaido University, 2016.
[34] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," IEEE J. Sel. Areas Commun., vol.29, no.9, pp.1765–1775, 2011.
[35] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," ACM SIGCOMM Comput. Commun. Rev., vol.32, no.4, pp.133–145, 2002.
[36] H. Suzuki, M. Ishihata, and S. Minato, "Designing survivable networks with zero-suppressed binary decision diagrams," WALCOM: Algorithms and Computation, M.S. Rahman, K. Sadakane, and W.K. Sung, ed., Cham, pp.273–285, Springer International Publishing, 2020.
[37] K. Nakamura, T. Inoue, M. Nishino, and N. Yasuda, "Efficient network reliability evaluation for client-server model," IEEE GLOBECOM, pp.1–6, 2021.
[38] M. Abseher, N. Musliu, and S. Woltran, "HTD — A free, open-source framework for (customized) tree decompositions and beyond," Integration of AI and OR Techniques in Constraint Programming: 14th International Conference, CPAIOR 2017, Padua, Italy, June 2017, Proceedings 14, pp.376–386, Springer, 2017.
[39] B.D. McKay and A. Piperno, "Practical graph isomorphism, II," Journal of symbolic computation, vol.60, pp.94–112, 2014.
[40] M. Sakai and H. Nabeshima, "Construction of an ROBDD for a PB-constraint in band form and related techniques for PB-solvers," IEICE Trans. Inf. & Syst., vol.E98-D, no.6, pp.1121–1127, June 2015.
[41] Y.f. Niu and F.M. Shao, "A practical bounding algorithm for computing two-terminal reliability based on decomposition technique," Computers & Mathematics with Applications, vol.61, no.8,

pp.2241–2246, 2011.
[42] N. Tamura, A. Taga, S. Kitagawa, and M. Banbara, "Compiling finite linear CSP into SAT," Constraints, vol.14, pp.254–272, 2009.

Takeru Inoue is a Distinguished Researcher at Nippon Telegraph and Telephone Corporation (NTT) Laboratories, Japan. He received the B.E. and M.E. degrees in engineering science and the Ph.D. degree in information science from Kyoto University, Japan, in 1998, 2000, and 2006, respectively. In 2000, he joined NTT Laboratories. From 2011 to 2013, he was an ERATO Researcher with the Japan Science and Technology Agency, where his research focused on algorithms and data structures. His research interests widely cover algorithmic approaches in communication networks. Inoue was the recipient of several prestigious awards, including the Best Paper Award of the Asia-Pacific Conference on Communications in 2005, the Best Paper Award of IEEE International Conference on Communications in 2016, the Best Paper Award of IEEE Global Communications Conference in 2017, the Best Paper Award of IEEE Reliability Society Japan Joint Chapter in 2020, the IEEE Asia/Pacific Board Outstanding Paper Award in 2020, and the IEICE Paper of the Year in 2021. He serves as an Associate Editor of the IEEE Transactions on Network and Service Management. He is a member of IEEE.

Norihito Yasuda is a Senior Researcher, NTT Communication Science Laboratories. He received a B.A. in integrated human studies and an M.A. in human and environmental studies from Kyoto University in 1997 and 1999, and a D.Eng. in computational intelligence and system science from Tokyo Institute of Technology in 2011. He joined NTT in 1999. He also worked as a research associate professor with the Graduate School of Information Science and Technology, Hokkaido University, in 2015. His current research interests include discrete algorithms and natural language processing.

Hidetomo Nabeshima is an Associate Professor at Graduate Faculty of Interdisciplinary Research, University of Yamanashi, Japan. He received a B.E. and M.E. degrees from Toyohashi University of Technology, Japan, in 1996 and 1998, respectively, and received a Ph.D. in engineering from Kobe University, Japan in 2001. His research interests include Boolean satisfiability testing, constraint programming, and constraint solver development.

Masaaki Nishino is a Distinguished Researcher at Nippon Telegraph and Telephone Corporation (NTT) Laboratories, Japan. He received a B.E. degree in electrical and electronic engineering, an M.E., and a Ph.D. in informatics from Kyoto University, Japan, in 2006, 2008, and 2014. He joined NTT laboratories in 2008. His research interests include combinatorial algorithms, data structure, and machine learning.

Shuhei Denzumi is a Research Associate at Nippon Telegraph and Telephone Corporation (NTT) Laboratories, Japan. He received a bachelor's degree in engineering, a master's, and a Ph.D. in computer science from Hokkaido University, Japan, in 2010, 2012, and 2015. He was a Research Associate at the University of Tokyo from 2015 to 2022 and joined NTT in 2022. His research interests include data structure, data compression, and combinatorial algorithms.

Shin-ichi Minato is a Professor at Graduate School of Informatics, Kyoto University. He received the B.E., M.E., and D.E. degrees from Kyoto University in 1988, 1990, and 1995, respectively. He worked for NTT Laboratories from 1990 until 2004. He was a Visiting Scholar at Stanford University in 1997. He joined Hokkaido University as an Associate Professor in 2004, and has been a Professor since October 2010. From Apr. 2018, he is a Professor at Kyoto University (present position). His research interests include efficient representations and manipulation algorithms for large-scale discrete structures, such as Boolean functions, sets of combinations, sequences, permutations, etc. From 2020 to 2024, he serves a project leader of JSPS-KAKENHI: Algorithmic Foundations for Social Advancement (AFSA) Project. He is a senior member of IEICE, IPSJ and IEEE, and is a member of JSAI and JSCS.