Doctoral Dissertation

On the Utility of the Zero-Suppressed Binary Decision Diagram

Renzo Roel P. Tan

September 20, 2021

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of SCIENCE

Renzo Roel P. Tan

Thesis Committee:

Professor Kazushi Ikeda	(Supervisor)
Professor Shoji Kasahara	(Co-supervisor)
Associate Professor Jun-Ichiro Yoshimoto	(Co-supervisor)
Assistant Professor Makoto Fukushima	(Co-supervisor)
Assistant Professor Chie Hieida	(Co-supervisor)
Associate Professor Jun Kawahara	(Kyoto University)

On the Utility of the Zero-Suppressed Binary Decision Diagram*

Renzo Roel P. Tan

Abstract

Decision-diagram-based solutions for discrete optimization have been persistently studied. Among these is the use of the zero-suppressed binary decision diagram, a compact graph-based representation for a specified family of sets. Such a diagram may work out problems in combinatorics by efficient enumeration.

A wide range of combinatorial problems in operations research falls under arc routing problems, a domain which focuses on arc or edge features rather than node or vertex attributes. The generalized directed rural postman problem is a generic type of problem with the goal of finding the shortest path utilizing at least an edge from each category in a graph with labeled edges. Another is the undirected rural postman problem, a well-known problem in arc routing that seeks to determine a minimum cost walk that traverses a certain set of required edges on a given graph. The problems, arising in numerous real-world applications, are nondeterministic polynomial-time hard.

In brief, an extension to the frontier-based search approach for zero-suppressed binary decision diagram construction is proposed. The modification allows for the inclusion of a class-determined constraint in formulation. Variations of the generalized directed rural postman problem, proven to be nondeterministic polynomial-time hard, are solved on some rapid transit systems as illustration. Results are juxtaposed against standard integer programming in establishing the relative superiority of the new technique.

*Doctoral Dissertation, Graduate School of Information Science,
Nara Institute of Science and Technology, September 20, 2021.

A solution to the undirected rural postman problem based on the zero-suppressed binary decision diagram is also presented. Through an extension to the frontier-based search method of diagram construction, the approach solves the problem by efficient enumeration, producing all feasible routes in addition to the optimal route. Instances of the problem put forward in literature are then solved as benchmark for the decision-diagram-based solution. As reasonable time is consumed, the method also proves to be a practicable candidate in solving the problem.

Given the aforementioned routines, the study expands the utility of the zero-suppressed binary decision diagram through advancing an original graph measure – the relative isolation probability of a vertex – in seeking to interpret a consequential edge metric from a vertex-centric perspective. Concisely, the probability of relative isolation pertains to the likelihood of a vertex to be disconnected from all designated source vertices in a graph with probability-weighted edges. A two-step algorithm for efficient calculation is presented and evaluated. Contained within the procedure is a Monte Carlo simulation and the use of the zero-suppressed binary decision diagram, efficiently constructed through the frontier-based search. The novel measure is then computed for a diverse set of graphs, serving as benchmark for the proposed method. In closing, case studies on real-world networks are performed to ensure the consistency of the experimental with the actual.

Keywords:

combinatorial optimization, discrete algorithms, frontier-based search, subgraph enumeration, zero-suppressed binary decision diagram

Contents

List of Figures

1. Introduction

1.1. Background

The zero-suppressed binary decision diagram is a compressed data structure capable of storing families of sets [33]. Due to the recursive structure of representation, mathematical operations on families may be carried out comfortably [31]. The intersection and union, for instance, are swiftly calculated using the diagram. That being the case, feasible solutions to discrete optimization problems may be economically kept in a zero-suppressed binary decision diagram [26]. The number of feasible solutions, the optimal solutions, the mean and variance of solutions, and other statistics may be extracted without difficulty [31].

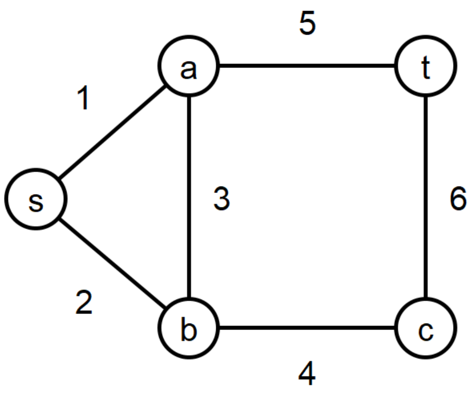
In combinatorial optimization over graphs, a zero-suppressed binary decision diagram may correspond to a collection of subgraphs [31]. The edge sets that make up each subgraph differentiate the elements in the collection. It is thus understood on this account that the nodes in the diagram are consistent with the edges and not the vertices of the graph. Should the items from the subsets in Figure 1.1a stand for the edges of the graph in Figure 1.1b, the elements of the zero-suppressed binary decision diagram in Figure 1.1c would be the paths shown in Figure 1.1d.

Regarding diagram construction, a fundamental approach is the frontier-based search [30]. The algorithm inputs the needed information to the nodes as they are being generated. As a consequence, requirements for the stored subgraphs are set in conformity with the given problem. The practicality of the zero-suppressed binary decision diagram has been growing. Domains such as architectural planning [43], disaster preparedness [44], grid power loss minimization [24], and route finding [45] present diverse applications.

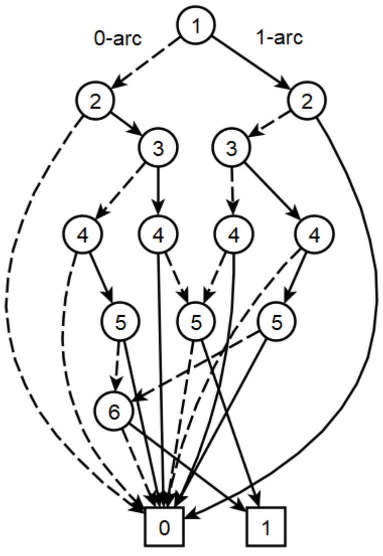
The study focuses on route finding settings. Various problems in operations

{1, 5} {1, 3, 4, 6}
{2, 3, 5} {2, 4, 6}

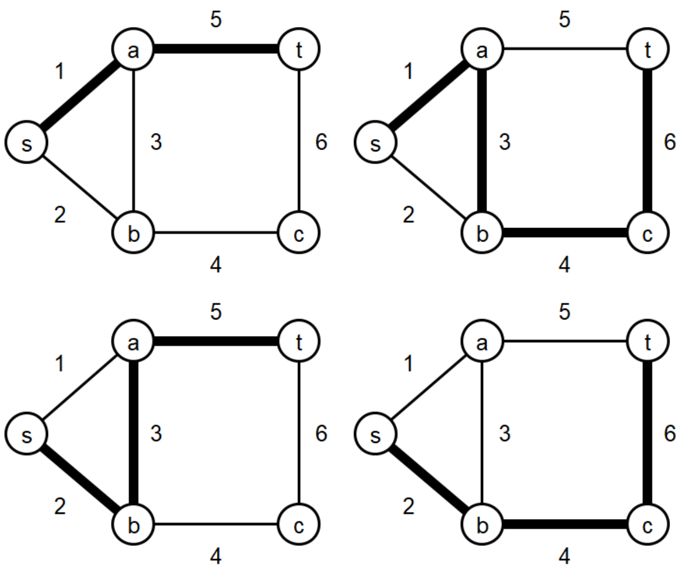
(a) A family of sets.



(b) A fixed graph.



(c) The diagram.



(d) Some paths.

Figure 1.1.: Representing a collection of subgraphs.

research may be posed as arc routing problems. See for example the following book [22]. Procedures such as mail delivery, garbage collection, and meter reading fall under the category. In general, arc routing consists of selecting an optimal route in a network, relying more on the edge properties than the vertex properties of the graph in the process.

The generalized directed rural postman is an arc routing problem that is defined on a graph with categorized edges. As illustration, consider for example a metro system where a station is a vertex and a traversal from a station to the next is an edge. Further, each edge is categorized; an edge is included in a line, which is represented by a color. In the context of the Osaka Metro, for example, one has the Midosuji line in red, the Tanimachi line in purple, the Yotsubashi line in blue, and so on. The goal of the problem is to find the shortest path in the graph that utilizes at least an edge from each category.

Another classic arc routing problem is the undirected rural postman problem [39], where the aim is to identify the route that covers a set of required edges with least cost – distance traveled, time consumed, *et cetera.* For instance, consider a postman who has to deliver the mail to a number of houses. The postman would have to traverse a subset of streets in the road network without missing any home that should receive mail. Such a circumstance may be formulated as an undirected rural postman problem where road intersections and road segments would be the vertices and edges of the graph, respectively. The required subgraph would then comprise sections of streets on which a house demands delivery.

1.2. Purpose

Decision-diagram-based solutions have been proposed in combinatorial optimization over graphs. More specifically, basic routing problems on mass rapid transit systems were solved by a compressed representation called the zero-suppressed binary decision diagram [45]. In line with the above, the utility of the aforementioned variant of the binary decision diagram in resolving other discrete problems in graph-theoretic contexts is explored.

Rooted in the insight, the research has the following objectives:

- Incorporate the use of the zero-suppressed binary decision diagram into discrete algorithms;
- Propose and evaluate enumerative techniques to solve complex problems in operations research; and

- Craft an original graph measure as motivated by the enumeration capabilities of the zero-suppressed binary decision diagram.

1.3. Overview

The succeeding chapters are organized as follows.

The second chapter examines preliminary concepts on the zero-suppressed binary decision diagram and the frontier-based search method of construction. The third looks into basic combinatorial applications that may help in diagram comprehension and interpretation. In the fourth chapter, one details the decision-diagram-based solutions for two prominent arc routing problems in operations research. Results on a novel metric motivated by the zero-suppressed binary decision diagram is discussed in the fifth chapter. To close, a synthesis and some recommendations for future research is contained in the sixth chapter.

2. Some Preliminaries

2.1. Zero-Suppressed Binary Decision Diagram

The zero-suppressed binary decision diagram is a graph-based data structure for the efficient storage and handling of families of sets [46]. More formally, consult the definition as follows [33].

Definition 1 (Zero-Suppressed Binary Decision Diagram)**.** Consider a universe U. For $x_k \in U$, $x_i < x_j$ if and only if $i < j$. A zero-suppressed binary decision diagram is a labeled directed acyclic graph satisfying the following properties.

1. There is only one node with indegree 0 called the root.
2. There are only two nodes with outdegree 0 called the 0-terminal and the 1-terminal, denoted by $\bot$ and $\top$, respectively.
3. A nonterminal node has exactly two outgoing arcs labeled by 0 and 1 called the 0-arc and 1-arc, respectively.
4. The destination of the 0-arc and the 1-arc of a nonterminal node is called the 0-child and 1-child, respectively.
5. A nonterminal node is labeled by an element of U.
6. The label of a nonterminal node is strictly smaller than those of its children.

There exists a unique reduced zero-suppressed binary decision diagram with the fewest nodes for a family of concern [33]. Reduction of a diagram in linear time apropos of the number of nodes is seen in [31].

Remark 1. A reduced zero-suppressed binary decision diagram adheres to the two points below.

- There is no node whose 1-child is the 0-terminal.
- There are no distinct nodes that have the same label, 0-child, and 1-child.

Based on the definition, a family of subsets $\mathcal{F}$ from U may be represented by a single zero-suppressed binary decision diagram $\mathcal{D}$. A path P from the root to the 1-terminal comprising 0-arcs and 1-arcs corresponds to a subset $U' \in \mathcal{F}$ if and only if for all $x \in U'$, there is a node labeled with x whose 1-arc is in P [34]. One proceeds to a theorem that hints at the inherent recursiveness of the diagram [33].

Theorem 1. Let diagram $\mathcal{D}$ correspond to family $\mathcal{F}$. The root e is either a terminal node or a nonterminal node.

1. If e is the 0-terminal then $\mathcal{F} = \emptyset$, the empty family.
2. If e is the 1-terminal then $\mathcal{F} = \{\emptyset\}$, the family containing only the empty set.
3. If e is nonterminal then it has two children. Let e_0 be the 0-child and e_1 be the 1-child of e. Denote the family with diagram rooted at e_i by $\mathcal{F}_i$. The family $\mathcal{F}$ may then be written as the union $\mathcal{F}_0 \cup \left(\bigcup_{x \in \mathcal{F}_1} x \cup \{e\}\right)$.

Expounding the theorem, the sets in $\mathcal{F}$ that do not contain e are connected to e through its 0-arc. On the other hand, the sets in $\mathcal{F}$ that do contain e are connected to e through its 1-arc. In notation, $\mathcal{F}_0 = \{x \mid x \in \mathcal{F}, e \notin x\}$ and $\mathcal{F}_1 = \{x \setminus \{e\} \mid x \in \mathcal{F}, e \in x\}$.

The recursive structure of the zero-suppressed binary decision diagram is revealed. For such a reason, the execution of family operations through the use of the diagram become straightforward [33]. Complexities for several important operations are explicitly stated [49]. Let two diagrams $\mathcal{D}_1$ and $\mathcal{D}_2$ correspond to families $\mathcal{F}_1$ and $\mathcal{F}_2$ with elements from universes U_1 and U_2. Computing for $|\mathcal{D}_1|$ or identically, the number of subsets in $\mathcal{F}_1$, is of time complexity $\mathcal{O}\left(n\left(\mathcal{D}_1\right)\right)$. The enumeration of elements represented by $\mathcal{D}_1$ is of $\mathcal{O}\left(|\mathcal{D}_1| \cdot |U_1|\right)$ complexity. Finally, the intersection $\mathcal{F}_1 \cap \mathcal{F}_2$ and the union $\mathcal{F}_1 \cup \mathcal{F}_2$ may each be computed in time complexity $\mathcal{O}\left(n\left(\mathcal{D}_1\right) \cdot n\left(\mathcal{D}_2\right)\right)$.

For problems under the domain of combinatorics, subset solutions may be represented by a zero-suppressed binary decision diagram [26]. The enumeration of solutions and accordingly, the optimal solution, the number of solutions, the mean and variance of solutions, and other data may be extracted with ease [31]. In a graph-theoretic setting, the edges of a graph may correspond to the items of a universe. Every node in the diagram is labeled with an edge in the graph and its 0-arc and 1-arc indicates the exclusion and inclusion, respectively, of the edge serving as label. As a path from the root node to the 1-terminal of the diagram is a subgraph of the given graph, the entirety of the diagram stands in for a collection of subgraphs. In the case of the paper, the zero-supressed binary decision diagram is utilized particularly for path enumeration. An illustration explaining how a diagram may represent a set of paths follows.

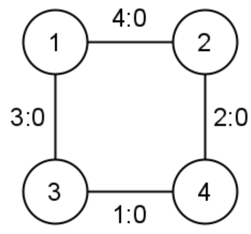
Consider the square grid of two vertices by two vertices shown in Figure 2.1a and the possible paths from the first vertex to the fourth vertex shown in Figures 2.1b and 2.1c. The zero-suppressed diagram representing all paths is Figure 2.1d.

All nodes are labeled by a variable number corresponding to an edge and a node identification number for reference in the diagram. A 1-arc is represented by a solid line and a 0-arc is represented by a dashed line. The former and latter correspond to the edge being present or not, respectively. At each level, one proceeds to the 1-child if the edge is included and to the 0-child if the edge is not. Any traversal from the root node to the 1-terminal with symbol $\top$ formed by 1-arcs and 0-arcs, then, is a path in the sample grid. In particular, taking the 0-arc of node e_1, the 1-arc of node e_2, and the 1-arc of node e_4 means taking the path from vertex 1 to vertex 4 through vertex 3; taking the 1-arc of node e_1 and the 1-arc of node e_3 means taking the path from vertex 1 to vertex 4 through vertex 2.

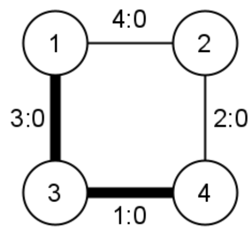
2.2. Frontier-Based Search

The frontier-based search is an approach for constructing a zero-suppressed binary decision diagram representing the set of subgraphs satisfying specified constraints [30]. Subgraphs of prescribed types such as paths, matchings, and trees, among others, may be stored in a diagram based on the context of the problem. An outline of the framework for representing a set of paths is provided.

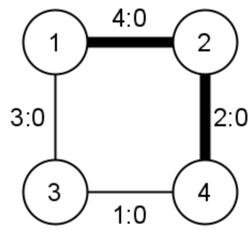
The construction of the zero-suppressed binary decision diagram representing



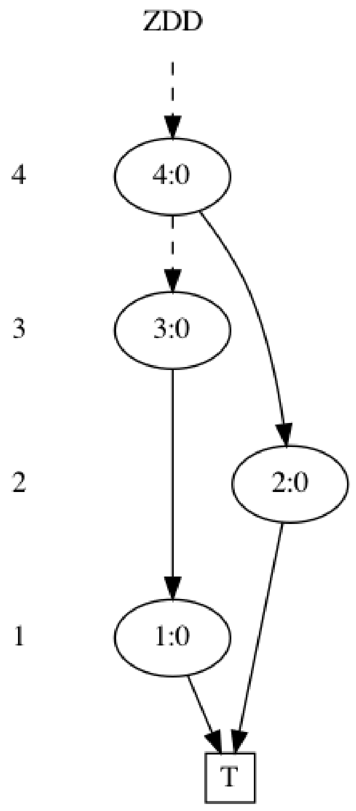
(a) The sample grid.



(b) The first path utilizing edges identified as 3 : 0 and 1 : 0 in the diagram.



(c) The second path utilizing edges identified as 4 : 0 and 2 : 0 in the diagram.



(d) A diagram representing all paths from the first vertex to the fourth vertex in a grid with four vertices.

Figure 2.1.: A zero-suppressed binary decision diagram representing paths from a grid with omission of the 0-terminal.

the set of all the s-t paths of a given graph is explained below as an example of the frontier-based search. This algorithm is similar to SimPath in [31]. Let $G = (V, E)$ be a weighted undirected graph that is simple and connected. There are to be no multi-edges in G; each element of E is uniquely defined by a 2-subset of V. Let s and t be vertices of V. A subgraph of G is denoted by $G' = (V', E')$, where $E' \subseteq E$ and $V' = \bigcup_{e \in E'} e$. Refining the notation, a union $\bigcup_{i=1}^{j} e_i$ is the set of vertices to which at least one of $e_1, e_2, e_3, \ldots, e_j$ is incident. No vertex of degree 0 may exist in any subgraph. One sets E as the universe for the zero-suppressed binary decision diagram. The elements of E are ordered, with $e_1 < e_2 < e_3 < \cdots < e_{|E|}$.

The diagram is constructed breadth-first, creating and labeling nodes starting from the element in the universe considered to be the smallest. To begin, the root node is labeled with e_1. For $i = 1, 2, 3, \ldots, |E| - 1$, a node labeled e_{i+1} is generated only after nodes labeled e_i are generated. Only the 1-terminal, the 0-terminal, or a node labeled e_{i+1} may serve as destination to both the 0-arc and the 1-arc of any node labeled e_i.

In motivating systematic construction, information on previous edge selection is maintained for vertices which are incident to both a processed edge and an unprocessed edge. Fittingly, the said set of vertices is called the frontier [30]. With $F_0 = F_{|E|} = \emptyset$, the jth frontier is defined in [30] as

$$F_j = \left(\bigcup_{i=1}^{j} e_i \right) \cap \left(\bigcup_{i=j+1}^{|E|} e_i \right)$$

for $j = 1, 2, 3, \ldots, |E| - 1$.

Concurrently, an array $n.\texttt{deg}$ that takes into account subgraph specifications is recorded on each node n. A specified subset of V is mapped to the set of natural numbers by the array. Moreover, to ensure the connectivity of an s-t path, the partition of frontier F_{j-1} is stored in n as $n.\texttt{comp}$. Vertex pairs that belong to the same connected component in G are to be in the same partition in $n.\texttt{comp}$.

Initially, for the root node r, $r.\texttt{deg}[v] = 0$ for all v. For $e = \{u, w\}$, if e is taken then $n.\texttt{deg}[u]$ and $n.\texttt{deg}[w]$ are incremented by one. For a vertex v, one designates the largest among the indices of incident edges as k. The node v is fixed once e_k has been processed. No further updating is done on $n.\texttt{deg}[v]$ as it is independent of $e_{k+1}, e_{k+2}, e_{k+3}, \ldots, e_{|E|}$. An s-t path is never completed if

$n.\texttt{deg}[s]$ or $n.\texttt{deg}[t]$ is not one or $n.\texttt{deg}[v]$ is not zero or two for $v \neq s, t$. In this case, the node becomes $\perp$. If there is no vertex on the frontier that has the same $n.\texttt{comp}$ value as $n.\texttt{comp}[v]$ then the node also becomes $\perp$ because this means that at least two connected components, one including v and another not including v, are generated and are never to be combined.

The node sharing strategy is then hired, merging two nodes n and n' if $n.\texttt{deg}$ is equal to $n'.\texttt{deg}$ and $n.\texttt{comp}$ is equal to $n'.\texttt{comp}$. If a vertex v is in F_{j-1}, $n.\texttt{deg}[v]$ is cached in e_j-labeled node n.

Through node sharing and pruning, the frontier-based search produces a zero-suppressed binary decision diagram. Node generation and information input are simultaneously done towards representing a collection of subgraphs. An unabridged discussion of the technique may be found in [30].

3. Literature Review

3.1. Solving Combinatorial Problems

Common combinatorial problems such as the combination and knapsack problems are used as setting for the utility of the zero-suppressed binary decision diagram [46]. These serve as preview to the investigation process hired in solving the succeeding problems.

3.1.1. The Combination Problem

Determining the ways in which k objects can be selected from n distinct objects irrespective of order is known as the combination problem. The family of k-element subsets from a universe with n elements may be represented using a single zero-suppressed binary decision diagram. The diagram in Figure 3.1 is for the problem of finding all ways to choose exactly 3 items from a cardinality 7 item set.

In the diagram, all nodes on a level labeled by a variable number from 7 to 1 and a node identification number represent an item in the universe. The solid 1-arc and the dashed 0-arc correspond to the item being present or not, respectively. A path formed by 1-arcs and 0-arcs from the root node to the 1-terminal with symbol $\top$ is a set of 3 items from the n given.

3.1.2. The Knapsack Problem

A familiar problem in combinatorics is the knapsack problem. Given a set of individually weighted items, the task is to resolve which subsets have total weights not exceeding a prescribed weight. These sets dictate which selection of objects may fill a fixed-size knapsack, from which the name derives. If one is to choose

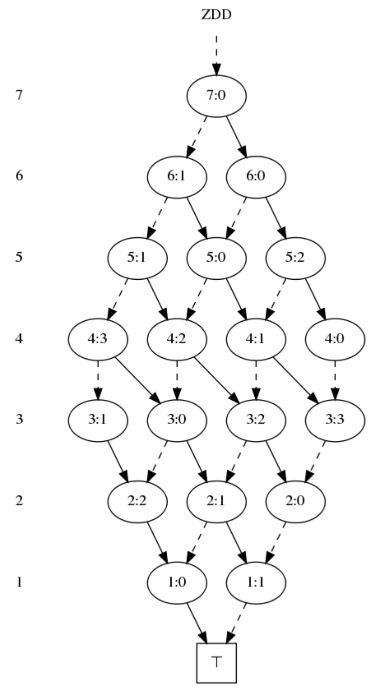


Figure 3.1.: The zero-suppressed binary decision diagram for the combination problem with $n = 7$ and $k = 3$.

from $n = 7$ different items with assigned weight tuple $w = (4, 3, 7, 5, 6, 8, 10)$ and weight constraint $W = 18$, the rotated diagram is in Figure 3.2.

3.1.3. Taking the Intersection

A variant of the knapsack problem above may be solved as well. Suppose 7 items with weights defined by the same sequence $(4, 3, 7, 5, 6, 8, 10)$ are given. To add, the weight capacity of the knapsack is 18 and the restriction that precisely 3 items may be carried is imposed. Noticeably, the formulation simply puts together the constraints for the combination and knapsack problems. Solving the problem is tantamount to taking the intersection of the two diagrams previously produced. Figure 3.3 shows the resulting zero-suppressed binary decision diagram rotated. Should a value tuple $(7, 2, 8, 3, 6, 9, 5)$ be incorporated into the problem, computing for the 3-set with maximum total value and with weight not exceeding the limit is straightforward. The set formed by taking the first, fifth, and sixth items has total weight 18 and yields a value of 22 as maximum.

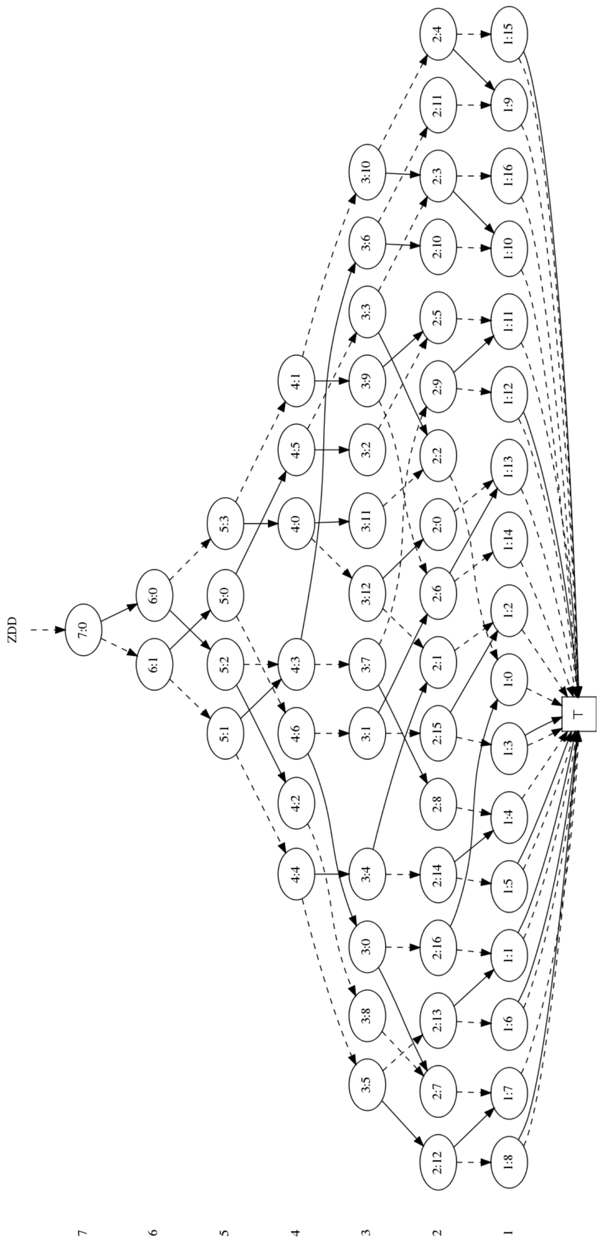


Figure 3.2.: The zero-suppressed binary decision diagram for the specified knapsack problem.

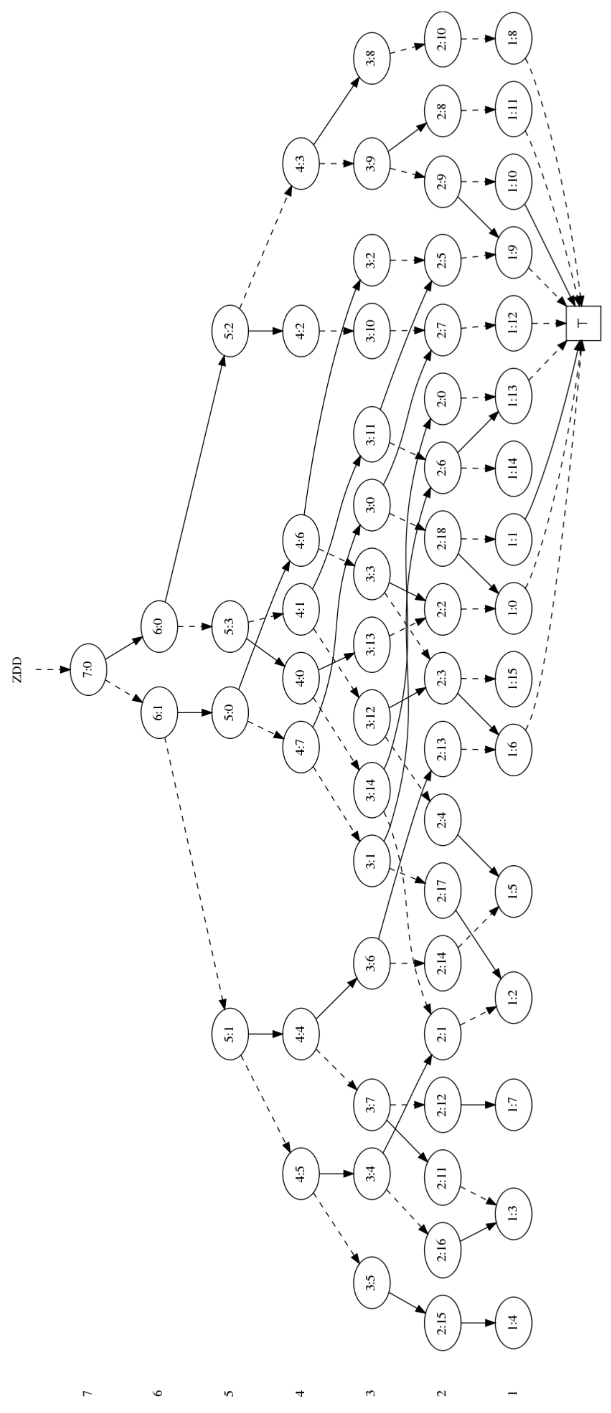


Figure 3.3.: The zero-suppressed binary decision diagram for the intersection of the combination and knapsack diagrams.

4. Enumeration Perspectives in Arc Routing

4.1. Generalized Directed Rural Postman Problem

Concisely, the section augments the frontier-based search for zero-suppressed binary decision diagram construction. The resulting algorithm accommodates additional constraints during computation. By the advanced routine, the generalized directed rural postman problem and its reverse are simultaneously solved for several transit networks. To justify the effectiveness of the method, numerical assessment is done in conclusion to the analysis.

An outline of the section is as follows. A summary of the generalized directed rural postman is first provided. The methodology for investigation is then laid out. Experiment results and reports on computational efficiency are presented next. The contribution of the section is emphasized in closing.

To begin, the definition of a path in graph theory is first delved into. It is essential in problem formulation.

Definition 2 (Path)**.** Given a graph, a path is a sequence of edges $y_1, y_2, \ldots, y_p$ where $y_i = \{v_{i-1}, v_i\}$ for $i = 1, 2, \ldots, p$ with vertex $v_i \neq v_j$ if $i \neq j$.

The question of finding the shortest journey in a metro network that uses each line at least once may be formulated as a known problem in operations research, the generalized directed rural postman [42]. The problem, along with the Chinese postman [14] and the rural postman [32] problems, falls under the field of arc routing, which focuses on arc or edge properties rather than node features [9]. A definition derived from [13] follows.

Problem 1 (Generalized Directed Rural Postman)**.** Given a set of weights W, a set of colors C, a graph $G = (V, E)$, an edge-weighting function $w : E \to W$, and a coloring function $c : E \to C$, find path E' such that $\sum_{e \in E'} w(e)$ is minimum and $\bigcup_{e \in E'} \{c(e)\} = C$.

The generalized directed rural postman problem is nondeterministic polynomial-time hard [9]. Accordingly, another nondeterministic polynomial-time hard problem is the reverse [19]. By seeking the maximum instead of the minimum weight, the problem is distinctively named the crazy generalized directed rural postman. Both the original and the reverse problems are hired by the study to further demonstrate the efficiency and effectiveness of the technique.

Problem 2 (Crazy Generalized Directed Rural Postman)**.** Given a set of weights W, a set of colors C, a graph $G = (V, E)$, an edge-weighting function $w : E \to W$, and a coloring function $c : E \to C$, find path E' such that $\sum_{e \in E'} w(e)$ is maximum and $\bigcup_{e \in E'} \{c(e)\} = C$.

As an aside, an exact algorithm for solving the original problem is introduced in [13]. Given in [2] are improvements through a branch-and-cut solution. A more recent study is [42], providing a clever workaround for finding the optimal solution in reasonable time.

The constraints for the generalized directed rural postman problem and the crazy generalized directed rural postman problem may be relaxed in cases where no solutions are found. In lieu of paths, one may search for trails.

Definition 3 (Trail)**.** Given a graph, a trail is a sequence of edges $y_1, y_2, \ldots, y_p$ where $y_i = \{v_{i-1}, v_i\}$ for $i = 1, 2, \ldots, p$ with edge $y_i \neq y_j$ if $i \neq j$.

In a weighted graph, a minimal path is a path of minimum weight; a path of maximum weight is called a maximal path. Consistently, a trail of minimum weight is a minimal trail and a trail of maximum weight is a maximal trail.

The proposed method is implemented in the C++ programming language aided by TdZdd[1], an existing library for diagram manipulation documented in [25] and

[1]`https://github.com/kunisura/TdZdd`

[46]. The ZDDLines[2] repository, utilized in [45], served as basis for the program. In addition, the entirety of the code is committed online to the GDRPDD[3] project.

Version 9.3.0 of g++ is used as compiler. Machine specifications include the Ubuntu 18.04.4 Long Term Support (Bionic Beaver) operating system, the Intel® Core™ i7-8565U processor at 1.80GHz, the NVIDIA® GeForce® MX250 graphics card, and a memory of 16GB.

A sample grid and chosen metro networks of increasing size are encoded. The first four rows of the text data for the sample network is shown as example.

```
1 2 100 1
2 3 100 1
3 4 100 1
4 5 100 1
```

Each row represents an edge and its properties. The first two numbers are the start and end vertices that define the edge. The weight of the edge, typically measured in units of distance or time, is given by the third number. The fourth number specifies the category of which the edge is part; in the case of a metro, this would be the line by which the edge is labeled or to which the two linked stations belong. Coordinate data for all station positions are also gathered for the purpose of visualization.

For each vertex pair (s, t) in the network, the following steps are done. First, the diagram for all paths from s to t is created. The task is accomplished through designating a degree constraint using the frontier-based search. For vertices s and t, a vertex degree of 1 is set. The remaining vertices are forced to be of degree 2 or 0 depending on the involvement in s-t path generation. This guarantees subgraphs included in the diagram to be paths. For trails, the limitation is eased to accept even degrees greater than 2.

The requirement of having to use each metro line at least once is then addressed. The `Lines` class summarizing the approach in generating the diagram that satisfies the said restriction is seen in Appendix A.1. Nodes in a level in the zero-suppressed binary decision diagram created by the `Lines` class collectively correspond to an edge in the graph. Stored in every node is a bitmask indicating

[2]`https://github.com/renzopereztan/ZDDLines`
[3]`https://github.com/renzopereztan/GDRPDD`

the line usage of the subgraph represented by a traversal from the root to the node. As the diagram is constructed from top to bottom, the states of the nodes are updated depending on the line information coming from the designated edge.

For a problem with color set $C = \{c_1, c_2, \ldots, c_{|C|}\}$, edge set $E = \{e_1, e_2, \ldots, e_m\}$, and coloring function $c : E \to C$, the bitmask is set to be of $|C|$ bits. Each bit ties in with the elements of C in order, implying that the kth bit indicates whether the color c_k is covered or not. The root node has a bitmask state with all bits being 0. For a node e_i, the 0-child retains state and the state of the 1-child is revised, flipping the bit designating $c(e_i)$ to 1 if it is 0. This happens routinely as one descends. A path from the root node ends in the 1-terminal if the resulting mask contains no bits with value 0, signifying all lines being utilized.

The intersection of the s-t path zero-suppressed binary decision diagram and the constraint diagram is taken afterward. The final step is to obtain the overall optimal paths for the graph. Attached as Appendices A.2 and A.3 are the `MinDist` and `MaxDist` classes based on [31] utilized to produce the paths with minimum and maximum weight.

The results are discussed in two parts. The outcome of the implementation based on the zero-suppressed binary decision diagram is in the first two subsections. This is further segmented into the sample grid setting and the metro context. The third subsection replicates a standing method in literature for comparison. An integer linear program for the generalized directed rural postman problem is examined.

4.1.1. Motivating Example

The simple graph consists of 22 vertices and 33 edges in a mesh-like pattern. All edge weights are set to be 100 units. The 9 lines are determined by the vertical and horizontal lines that cut through the grid.

The generalized directed rural postman problem and its reverse are solved in 56.96 seconds. The recorded time includes the enumeration of all paths in the network that satisfy the category-based constraint and the extraction of the paths that have the minimum and maximum cumulative weights. Note that through the approach, even the solutions with the second smallest weight, the third largest weight, and so on, are known and may easily be recalled.

There are 2 unique solutions of total weight 900 units found for the minimum. Figure 4.1 shows one of them. A total of 19 different paths exist for the maximum. A solution with the maximum weight of 2100 units is seen in Figure 4.2.

4.1.2. Decision-Diagram-Based Approach

Composed of 98 vertices and 108 edges is the Hong Kong Mass Transit Railway, referred to in the section as the Hong Kong Metro[4]. The network has 10 lines identified in Appendix B.1. For the purposes of the study, an edge is weighted one step. For every additional edge, the cost of the journey would increase by one step.

An out-turn of no solution was reached in 1010.67 seconds. No path that uses each line at least once exists for the metro. In response, relaxation is done. Trails of minimum and maximum weight that pass through all lines are sought. This problem-solving detour yields added complexity, hence the running time of 1172.33 seconds.

There are two minimal solutions. A trail with the minimum 35 steps is in Figure 4.3 and Table 4.1. Two solutions of 56 steps are maximal. Notice in Figure 4.4 that the algorithm forces itself to extend the trail in producing the maximum. Pairs of edges that are incident to the same vertices but are on different lines are likely to be taken as part of the trail.

The Osaka Metro[5] is chosen for the second experiment. Previous studies [45] have used the same network to find solutions to a constrained version of the simpler *s*-*t* path problem. The metro has 107 vertices and 125 edges, a larger network compared to the Hong Kong system. The edge weights are the distances between stations given in kilometers. The 9 lines are in Appendix B.2.

The time taken for the simultaneous solution of the two problems is 938.89 seconds. Interestingly, there is only one solution for the minimum. The path presented in Figure 4.5 and Table 4.2 with distance 16.2 kilometers is the optimal solution. For the crazy generalized directed rural postman problem, a maximum distance of 77.2 kilometers is registered. There are 30 ways to complete the journey as in Figure 4.6.

[4]`https://en.wikipedia.org/wiki/MTR`

[5]`https://en.wikipedia.org/wiki/Osaka_Metro`

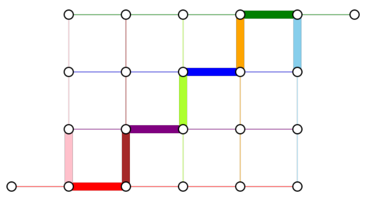


Figure 4.1.: A solution to the original problem for the sample grid.

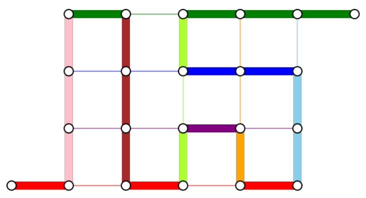


Figure 4.2.: A solution to the reverse problem for the sample grid.

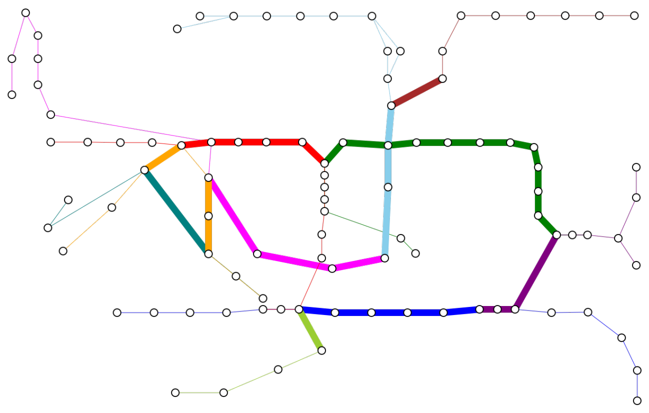


Figure 4.3.: A solution to the relaxed original problem for the Hong Kong Metro.

Table 4.1.: The Shortest Journey for the Hong Kong Metro

Move	**Initial Station**	**Terminal Station**	**Line**
1	Ocean Park	Admiralty	🟩
2	Admiralty	North Point	🟦
3	North Point	Yau Tong	🟪
4	Yau Tong	Prince Edward	🟩
5	Prince Edward	Lai King	🟥
6	Lai King	Tsing Yi	🟧
7	Tsing Yi	Kowloon	🟦
8	Kowloon	Nam Cheong	🟧
9	Nam Cheong	Hung Hom	🟪
10	Hung Hom	Tai Wai	🟦
11	Tai Wai	Che Kung Temple	🟫

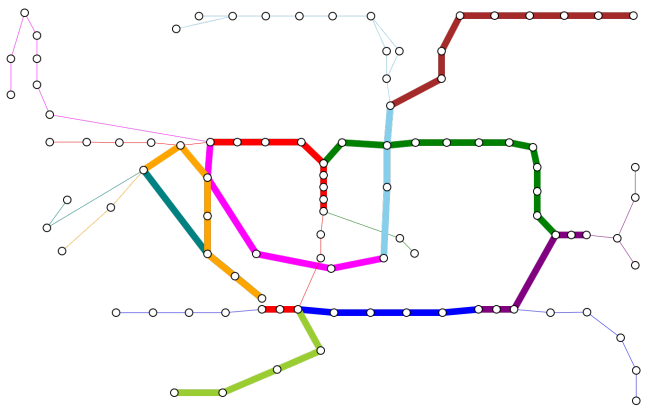


Figure 4.4.: A solution to the relaxed reverse problem for the Hong Kong Metro.

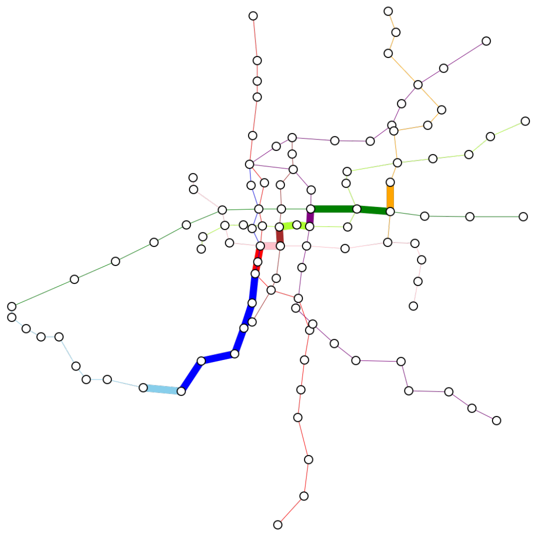


Figure 4.5.: The solution to the original problem for the Osaka Metro.

Table 4.2.: The Shortest Journey for the Osaka Metro

Move	Initial Station	Terminal Station	Line
1	Hirabayashi	Suminoekoen	Ⓟ
2	Suminoekoen	Daikokucho	Ⓨ
3	Daikokucho	Namba	Ⓜ
4	Namba	Nippombashi	Ⓢ
5	Nippombashi	Nagahoribashi	Ⓚ
6	Nagahoribashi	Tanimachi 6-Chome	Ⓝ
7	Tanimachi 6-Chome	Tanimachi 4-Chome	Ⓣ
8	Tanimachi 4-Chome	Midoribashi	Ⓒ
9	Midoribashi	Shigino	Ⓘ

The Taipei Metro[6], also known as the Taipei Mass Rapid Transit, comprises 136 vertices and 149 edges. One includes the Airport Line and considers 7 lines total listed in Appendix B.3 for the network. The edges are simply weighted in units of steps.

The solution on the metro results in a computation time of 2176.70 seconds. The minimum of 8 steps and maximum of 90 steps each share two solutions. For the original problem, a solution is seen in Figure 4.7 and Table 4.3; a solution for the reverse problem is in Figure 4.8.

4.1.3. Mathematical Programming

One pays particular attention to the generalized directed rural postman problem. Solving the crazy variant entails a similar strategy. The problem is posed as an integer program with linear constraints. A formulation motivated by [42] and [35] is employed.

Program 1 (Linear Program for the Generalized Directed Rural Postman Problem)**.** Minimize

$$\textstyle\sum_{(u,v,l)\in A} x_{u,v,l}$$

subject to

$$\begin{aligned}
&\textstyle\sum_{(u,v,l)\in A} x_{u,v,l},\\
&\textstyle\sum_{(u,v,l)\in A} x_{u,v,l} = \sum_{(v,w,l)\in A} x_{v,w,l} \ \forall v \in V \setminus \{s,t\},\\
&\textstyle\sum_{(s,v,l)\in A} x_{s,v,l} = \sum_{(u,t,l)\in A} x_{u,t,l} = 1,\\
&\textstyle\sum_{(u,v,l)\in A} x_{u,v,l} \geq 1 \ \forall l \in C,\\
&|V| \cdot x_{u,v,l} \geq f_{u,v,l} \ \forall\, (u,v,l) \in A,\\
&\textstyle\sum_{(u,v,l)\in A} f_{u,v,l} - \sum_{(v,w,l)\in A} f_{v,w,l} \geq y_0 \ \forall v \in V \setminus \{s\},\\
&\textstyle\sum_{(u,v,l)\in A} x_{u,v,l} - \sum_{(v,w,l)\in A} x_{v,w,l} \leq y_v \ \forall v \in V,\\
&x_{u,v,l} \in \{0,1\} \ \forall\, (u,v,l) \in A,\\
&f_{u,v,l} \in \mathbb{N} \ \forall\, (u,v,l) \in A, \text{ and}\\
&y_v \in \mathbb{N} \ \forall v \in V.
\end{aligned}$$

[6]https://en.wikipedia.org/wiki/Taipei_Metro

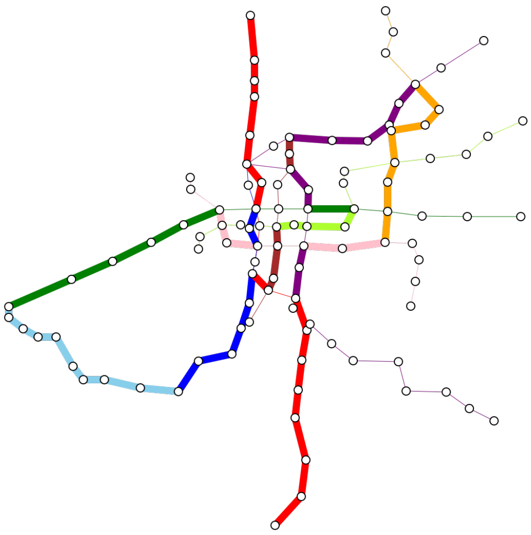


Figure 4.6.: A solution to the reverse problem for the Osaka Metro.

Table 4.3.: The Shortest Journey for the Taipei Metro

Move	**Initial Station**	**Terminal Station**	**Line**
1	Xing Fu	New Taipei Industrial Park	Y
2	New Taipei Industrial Park	Taipei Main	
3	Taipei Main	Zhongshan	R
4	Zhongshan	Songjiang Nanjing	G
5	Songjiang Nanjing	Zhongxiao Xinsheng	O
6	Zhongxiao Xinsheng	Zhongxiao Fuxing	BL
7	Zhongxiao Fuxing	Daan	BR

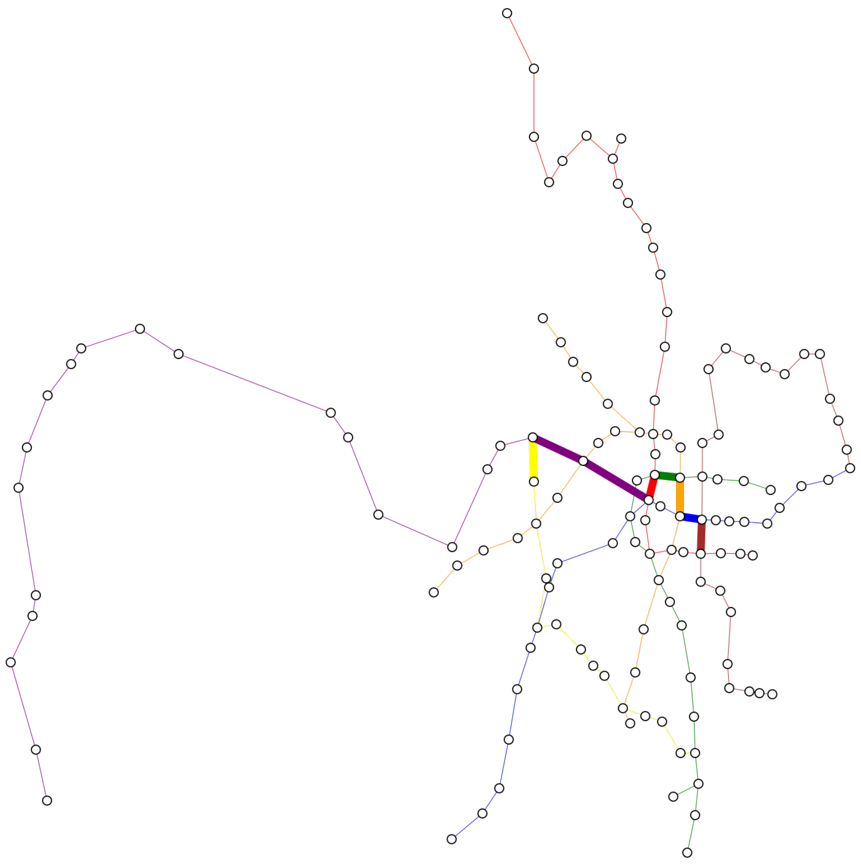


Figure 4.7.: A solution to the original problem for the Taipei Metro.

Table 4.4.: Some Experiment Statistics

Graph	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert C \rvert$	Time (s)
Sample	22	33	9	56.96
Hong Kong	98	108	10	1172.33[†]
Osaka	107	125	9	938.89
Taipei	136	149	7	2176.70

[†]Computation time for the relaxed problem.

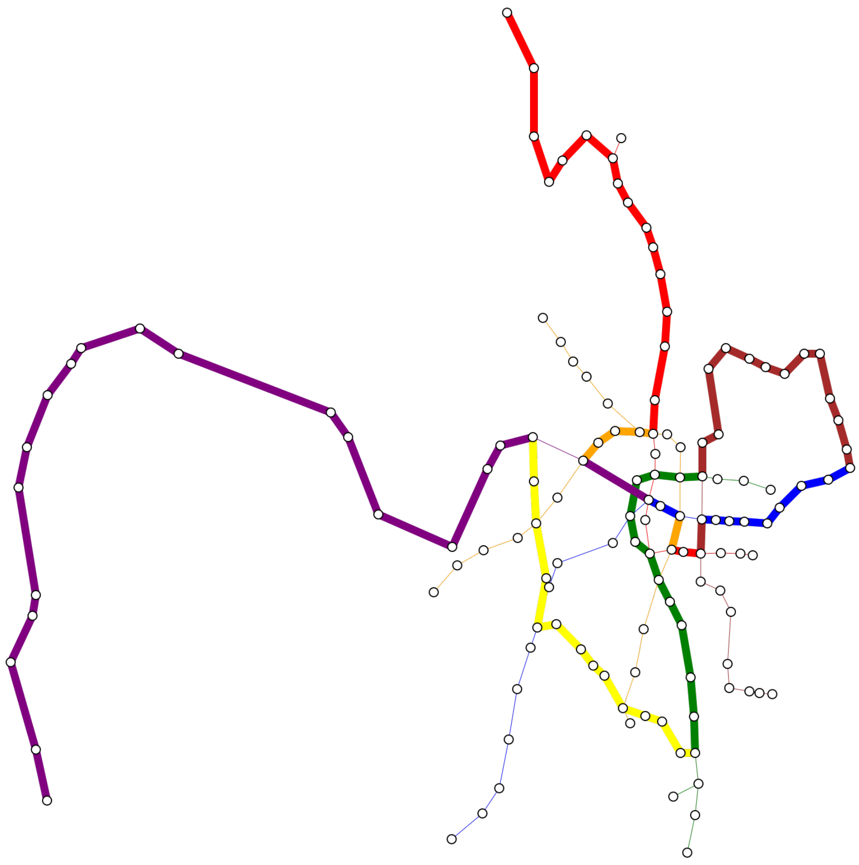


Figure 4.8.: A solution to the reverse problem for the Taipei Metro.

For each edge connecting vertices u and v on line l, a binary variable $x_{u,v,l}$ is assigned. Instinctively, the sum of the variables is to be minimized. A virtual source s and a virtual target t, both of which are connected to every vertex in the station, are added to overcome the crux of having to solve the program for each possible vertex pair. One may consult [42] and [6] for a thorough explanation of the constraints and some possible refinements.

Upon completing the integer programming experiment, the correctness of the solutions attained by the decision-diagram-based algorithm is confirmed. From the optimal solutions to the number of solutions, results from both methods match. A complete iteration for a network, however, took several hours. It is important to note that this is primarily because of the need for solving the model multiple times to get the number of solutions that give minimum.

4.2. Undirected Rural Postman Problem

The research augments the frontier-based search in zero-suppressed binary decision diagram construction to accommodate the class-based constrained brought about by the problem having required and nonrequired edges. One then chooses instances of the undirected rural postman problem used across several sources for tests on computational efficiency. In closing, some advantages of the proposed zero-suppressed binary decision diagram method are established.

The section is organized as follows. The preliminary concepts on the undirected rural postman problemis first surveyed. The methodology used in the study is then detailed. Following this, experiment results and the corresponding discussion are laid out. To close, a synthesis is contained.

The notion of a walk in graph theory is requisite for the formulation of the problem.

Definition 4 (Walk)**.** Given a graph, a walk is a sequence of edges $y_1, y_2, y_3, \ldots, y_p$ where y_k is the pair of vertices $\{w_k, w_{k+1}\}$ for $i = 1, 2, \ldots, p$.

In a walk, therefore, vertices and edges may be repeated in the sequence.

The undirected rural postman problem is first introduced in [39]. The problem, defined on a graph with required edges, involves designing a walk of least cost that traverses each edge in the required subgraph at least once. In contrast to the undirected Chinese postman problem [14] where the required subgraph is connected, the undirected rural postman would have no such guarantee [20]. The problem is proven to be nondeterministic polynomial-time hard [32]. The section hires an uncomplicated definition of the problem suitable for the method of solution. The definition is motivated by [15].

Problem 3 (Undirected Rural Postman Problem)**.** Given a graph $G = (V, E)$, a set of required edges $R \subset E$, a set of costs $C \subset \mathbb{R}^+$, and a cost function $c : E \to C$, find walk E' such that $\sum_{e \in E'} c(e)$ is minimum and $R \subseteq E'$.

A few applications from [15] are below.

- Street sweeping is a common example [3]. In a city, certain streets may require service more often than others.
- Snow plowing is also an application [21]. There are varying priority levels across categories of roads.
- Another example is school bus routing [1, 4, 12]. Students living in some street segments must board.

As there is prominence to the undirected rural postman problem, the different kinds of solutions have been frequently studied. Exact algorithms and heuristics exist in solving the undirected rural postman problem. Among the exact methods are a fundamental branch-and-bound scheme in [8], a cutting plane technique in [10], and a solution through an improved formulation in [17]. Heuristics that yield approximations are also used. The constructive approach by Frederickson [18] is most common; nevertheless, there are alternatives such as the Monte Carlo routine [11] and an improvement heuristics [23]. Recently, a solution based on genetic algorithms was also investigated [37].

Test grid configurations and chosen benchmark graphs of increasing size are converted into an edge list. The edge list would then be divided into two sublists – one for the required edges and another for the nonrequired edges. The first rows of the sublist containing required edges are shown as example.

```
3 4 1
11 12 1
13 14 1
```

A row represents an edge and its properties. Since an edge is defined by a set of two vertices, the first two numbers are the start and end vertices forming the edge. The cost of the edge, in most cases a distance or time measure, is specified by the third number. Visualization is done with the help of an online tool[7] for graphing.

The proposed decision-diagram-based approach in solving the undirected rural postman problem is outlined in Algorithm 1. The explanation follows.

The class `readGraph` begins `zddURPP` by inserting each edge into the graph twice. This workaround is imperative to allow `frontierBasedSearch` to use an edge at most two times. Every required edge is assigned an integer from 0 to $|R| - 1$ in `r`. The two copies of an edge is given the same identification number. Edges that are not required are assigned the number -1.

Following the steps supra, the maximum degree for a vertex in the graph is stored through `getMaxDegree`. This piece of information is required primarily by `degreeConstraint`, which prescribes the vertex degrees during the walk enumeration using `frontierBasedSearch`.

Whether or not each required edge labeled from 0 to $|R| - 1$ is used in the solution is kept track with `Required`. An important procedure in `Required` is `getChild`, seen in Algorithm 2. First, the required edges, including the nonrequired edges, are ensured to be sorted in order of requirement. Upon execution, the constraint sets a flag to true whenever an edge is used. The flag is checked whenever the current level being processed differs from the previous level based on the edge identification number. If the flag is true then it is reset to false and the the process continues; otherwise, the concerned branch of the zero-suppressed binary decision diagram is cut off. Edges with labels of -1 are skipped.

The intersection of the `degreeConstraint`, `frontierBasedSearch`, and `Required` zero-suppressed binary decision diagrams is taken afterward. Ultimately, the overall optimal walk for the graph is acquired following [31].

Implementation was done in the C++ language. The compiler used was g++

[7]`https://csacademy.com/app/graph_editor/`

Algorithm 1 ZDDURPP

```
1:  $G \Leftarrow readGraph()$
2:  $d \Leftarrow$ edge distance function
3:  $r \Leftarrow$ edge requirement assignment
4:
5:  $mvd \Leftarrow getMaxDegree(G)$    ▷ Store the maximum degree of a vertex in G
6:  $dc \Leftarrow degreeConstraint(G)$    ▷ Create a record of degree constraints for vertices in G
7:  $fbs \Leftarrow frontierBasedSearch(G)$    ▷ Create a diagram of all walks in G
8:  $req \Leftarrow Required(G)$    ▷ Create a diagram constraint to use all required edges
9:
10: for all $v \in G.V$ do
11:     $dc[v] = [0, 2, 4, \ldots, mvd]$
12: end for
13:
14: $zdd \Leftarrow dc \cap fbs \cap req$
15: $ans \Leftarrow zdd.evaluateMinResult()$
```

Algorithm 2 getChild for Required

```
procedure $getChild(flag, level, value)$

    if $r[level] \neq r[level-1]$ then
        if $flag = 0$ then
            return $0$
        end if
        $flag = 0$
    end if

    if $value = 1$ then
        $flag = 1$
    end if

    $level \Leftarrow level - 1$
    if $level = 0$ then
        return -1
    end if
    return $level$

end procedure
```

version 7.5.0. The TdZdd[8] library with documentation in [25] was employed for diagram manipulation. Moreover, the framework utilized is based on ZDDLines[9], the program used in [45]. Refer to the URPDD[10] repository for the complete code. Table 4.5 summarizes the machine specifications.

The experiments are discussed in three parts. Results from preliminary tests done on a sample grid is detailed in the first subsection. The second subsection presents the outcome from solving benchmark instances in literature [8]. The third contains some notes on the performance of the zero-suppressed binary decision diagram method.

4.2.1. Tests on a Grid Network

A test grid of 3 edges by 3 edges is used as setting. The total number of vertices is 16 and the total number of edges is 24. All edges are set to be of cost 1 unit. Varying configurations of the undirected rural postman problem were then generated, with $3 \leq |R| \leq 13$. The required edges were chosen arbitrarily for each problem.

The results from 8 problems are arranged by the number of required edges in Table 4.6. Every problem is given an identification number i for reference. To recapitulate, $|E|$ is the number of edges, $|R|$ is the number of required edges, and $|V|$ is the number of vertices. The number of nodes and elements in the resulting zero-suppressed binary decision diagram are denoted by $n(\mathcal{D})$ and $|\mathcal{D}|$, respectively. The time taken in seconds, the memory consumed in MB, and the cost z of the minimal solution is shown in the table.

As seen, the undirected rural postman problem is solved by the approach in reasonable time. Diagrams with tens of thousands of nodes representing billions of feasible solutions is constructed in less than a second on average. In addition, the minimum costs are attained exactly by the routine.

[8]`https://github.com/kunisura/TdZdd`

[9]`https://github.com/renzopereztan/ZDDLines`

[10]`https://github.com/renzopereztan/URPDD`

Table 4.5.: The specifications of the machine used in experimentation.

Operating System	Ubuntu 18.04.3 Long Term Support
Processor	Intel® Core™ i9-9920X at 3.50GHz
Graphics Card	NVIDIA® GeForce® RTX2080 Ti/PCIe/SSE2
Memory	128GB

Table 4.6.: Results of the experiment on the generated test grid networks.

i	$\|E\|$	$\|R\|$	$\|V\|$	Time (s)	Space (MB)	$n(\mathcal{D})$	$\|\mathcal{D}\|$	z
1	24	3	16	0.07	5	1.93E+04	2.99E+09	12
3	24	4	16	0.04	4	6.43E+03	2.25E+09	12
2	24	5	16	0.15	9	3.06E+04	1.72E+09	14
5	24	7	16	0.05	5	4.77E+03	1.08E+09	12
4	24	7	16	0.29	16	3.32E+04	1.00E+09	14
7	24	8	16	0.43	23	2.89E+04	8.52E+08	14
8	24	9	16	0.25	15	1.00E+04	6.35E+08	16
6	24	13	16	1.52	66	1.85E+04	2.69E+08	20

4.2.2. Instances from Literature

Benchmark instances from [8] are then chosen. A selection of 14 problems with $7 \leq |V| \leq 26$ and $10 \leq |E| \leq 47$ is solved. The costs of the edges differ per instance. The number of required edges in a problem ranges from 4 to 24. The required subgraph is determined in accord with the reference text.

Following the same format as the previous table is Table 4.7, in which the results are found. In the experiment, the generation of a diagram with hundreds of nodes representing thousands of solutions takes a fraction of a second whereas generating a diagram with millions of nodes representing tens of trillions of solutions would take around an hour. Given the difficulty of the task, such consumption of time is justified for decision-diagram-based enumeration [38, 45].

With regard to correctness, the algorithm gives the exact answers without fail. It outperforms the heuristics compiled in [37]. Against exact solutions, the enumerative solution demands more time in calculation. The advantages of the latter, however, are plenty. Apart from the obvious listing of all feasible solutions and drawing of the minimal walk, these include being able to produce the k solutions of lowest or highest cost, to find the mean and variance of all feasible solutions, to filter solutions based on some desired criterion, and others.

4.2.3. Notes on Performance

To gain deeper insight into the potentiality of the method, some observations are highlighted. One zeroes in on the results for two problem pairs – the fourth and fifth grid network configurations and the fourth and fifth problems from literature.

1. Given the same graph, more required edges means less decisions for the algorithm to make. There are three possibilities for an edge from a zero-suppressed binary decision diagram perspective. These are take the edge, take the edge twice, and do not take the edge. For a required edge, the option of not taking the edge is eliminated, substantially reducing processing.

2. The problem may be thought of as being the task of connecting parts of the required subgraph through choosing nonrequired edges that would serve as links. This means that the more connected the required subgraph is to begin

Table 4.7.: Results of the experiment on standard instances from literature.

i	$\|E\|$	$\|R\|$	$\|V\|$	Time (s)	Space (MB)	$n(\mathcal{D})$	$\|\mathcal{D}\|$	z
13	10	4	7	0.01	4	2.30E+02	3.53E+03	35
1	13	7	11	0.02	4	2.56E+02	5.62E+03	76
11	14	7	9	0.04	4	8.45E+02	1.39E+05	23
12	18	5	7	0.08	6	1.04E+04	5.96E+07	19
10	20	10	12	0.42	24	1.47E+04	2.44E+07	80
9	26	14	14	0.54	39	6.97E+03	3.75E+09	83
2	33	12	14	50.12	2230	4.91E+05	1.03E+12	152
5	35	16	20	31.01	1724	1.74E+05	1.13E+13	124
4	35	22	17	1297.53	71229	6.73E+05	3.89E+12	84
15	37	19	26	1714.08	106885	1.09E+06	1.47E+11	441
18	37	16	23	4532.78	126383	9.26E+06	1.40E+13	146
8	40	24	17	1552.16	114617	9.48E+05	1.71E+14	122
17	44	17	19	3481.13	126641	5.71E+06	4.06E+16	112
7	47	24	23	5289.39	126723	6.85E+05	6.62E+16	130

with, the more efficient the algorithm becomes. Figure 4.9 and Figure 4.10 provide an example, where the solution to the fifth grid problem is secured faster than the solution to the fourth possibly due to the required subgraph of the fifth being more connected in the first place.

3. Figure 4.11 and Figure 4.12 illustrate how the structure of the given graph affect execution drastically. The fifth benchmark instance is solved in 31.01 seconds; the fourth benchmark instance, having the same number of edges but a different structure, is solved in 1297.53 seconds. The sparsity of a graph is a huge factor in computation, especially in the case of a solution by enumeration using the zero-suppressed binary decision diagram.

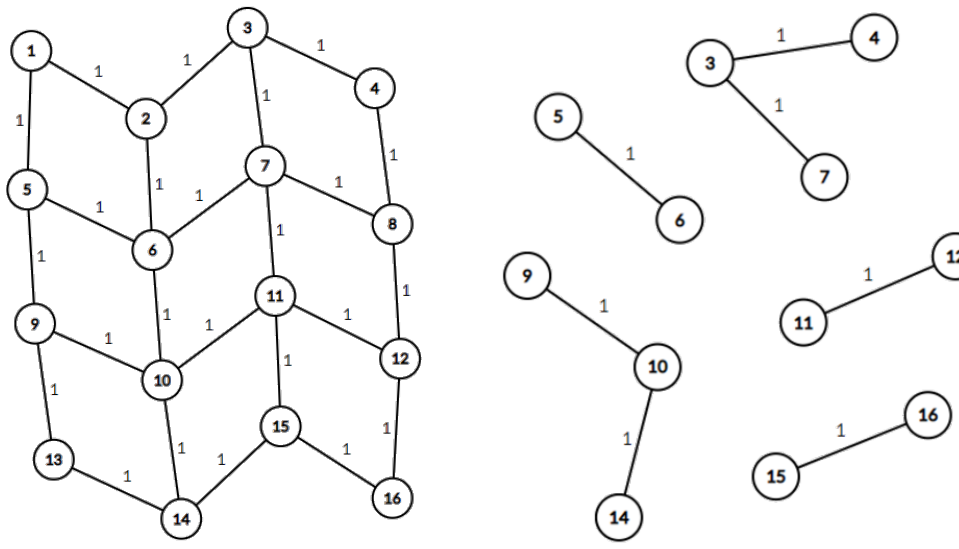


Figure 4.9.: The given graph and the required subgraph for the fourth sample grid.

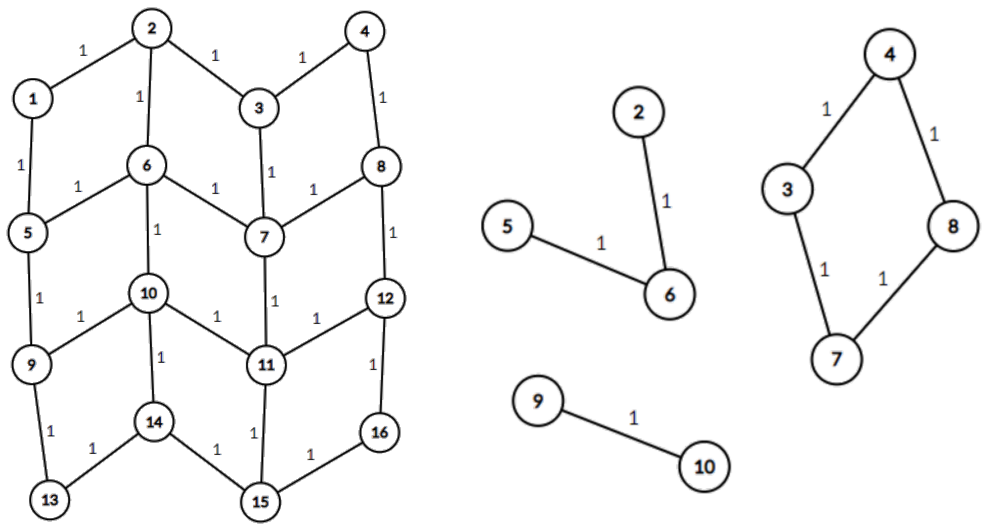


Figure 4.10.: The given graph and the required subgraph for the fifth sample grid.

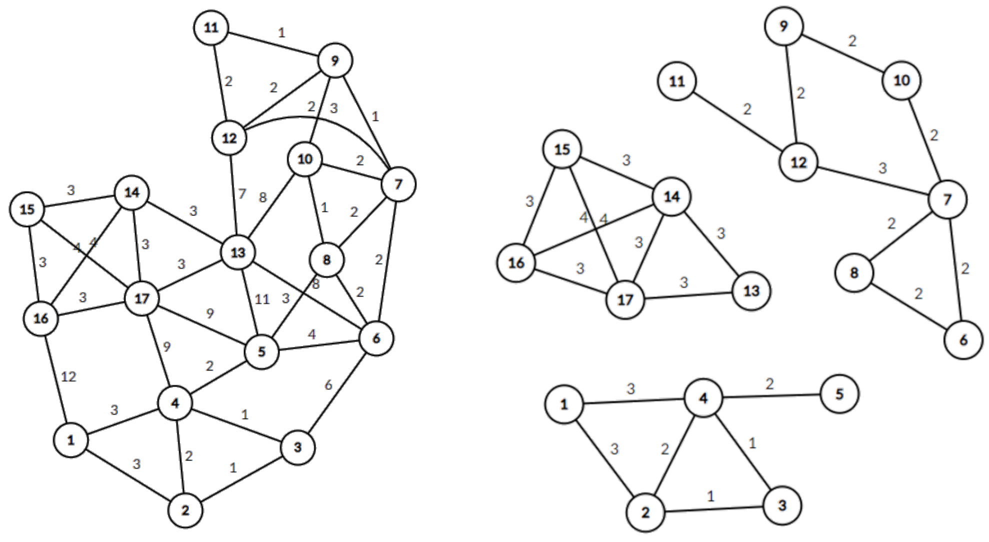


Figure 4.11.: The given graph and the required subgraph for the fourth benchmark problem.

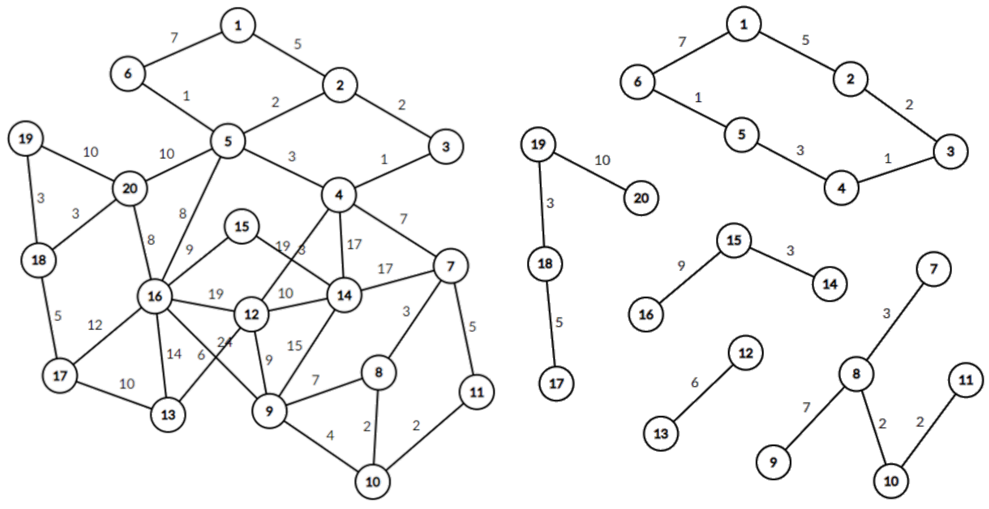


Figure 4.12.: The given graph and the required subgraph for the fifth benchmark problem.

5. Contemporary Measures for Network Resilience

The world operates through networks, thereby establishing the need to study the graphs through which they are represented. On that account, graph measures or graph metrics have long been a domain of interest. Classical metrics such as the connectivity, distance, betweenness, clustering, and reliability polynomial are commonly used to characterize graphs [16]. In addition, there are spectral measures that look into the matrices associated with the graphs [36].

Among the aforementioned are measures that aim to examine whether or not a network would remain to be functioning satisfactorily in the event of damage [40]. In the research, a new graph metric called the probability of relative isolation is introduced. The relative isolation probability reveals how likely it is that a vertex would be disconnected from sources given the failure of some edges. Lifelines such as road networks, water pipelines, power lines, communication systems, *et cetera* and their possible failure during stress are contexts to which the measure is especially applicable. Several advantages to its utility are being able to:

- Know which nodes may require urgent attention after a destructive episode;
- Have a reasonable basis for the sequence of links for immediate repair;
- Gauge if there are redundancies in the network design and construction;
- Discover prospective node or link groups for network reinforcement; and
- Estimate possible locations for the process of adding more network sources.

To compute for the vertex probabilities of relative isolation, a two-step procedure is devised. The first step employs the use of randomness and iteration through conducting a Monte Carlo experiment. The random numbers determine the survival or failure of the edges and consequently, the active components of the graph for each iteration. A check for any source connection is done on the vertices in the second step, having recourse to a compressed data representation known as the zero-suppressed binary decision diagram.

Benchmark configurations from literature and graphs for real networks are each translated into an edge list. Every row is an edge, represented by the two vertices through which it is defined. The source vertices and failure probabilities of the edges are kept in a separate file in order to conveniently create multiple scenarios. Visualization is done in the general-purpose diagramming software yEd[1] with version 3.20.1.

5.1. Probability of Relative Isolation

Requisite for the formulation of the metric is the notion of a path in graph theory.

Definition 5 (Path)**.** Given a graph, a path is a sequence $y_1, y_2, \ldots, y_p$ where $y_i = \{v_{i-1}, v_i\}$ is an edge of the graph for $i = 1, 2, \ldots, p$ with vertex $v_i \neq v_j$ of the graph if $i \neq j$. A path from vertex v_0 to vertex v_p is called a v_0-v_p path.

A path is therefore a sequence of edges that joins a sequence of vertices which are distinct. The formal definition of the relative isolation probability of a vertex in a given graph is below.

Definition 6 (Probability of Relative Isolation)**.** Consider the graph $G = (V, E)$ and the set of source vertices $S \subset V$. Let the probability of failure of an edge be $\pi(e_i)$ for $i = 1, 2, 3, \ldots, |E|$, with function $\pi : E \to [0, 1]$.

The number of iterations, which is essentially how many times the Monte Carlo experiment is executed, is set to be N. A random number $\rho_k(e_i)$ with function $\rho_k : E \to [0, 1]$ is assigned to each edge e_i on the kth iteration for $k = 1, 2, 3, \ldots, N$. For all $e_i \in E$, an indicator function μ_k is defined to be

$$\mu_k(e_i) = \begin{cases} 0 & \text{if } \rho_k(e_i) > \pi(e_i) \\ 1 & \text{otherwise} \end{cases},$$

[1]Link: `https://www.yworks.com/products/yed`.

assigning either a 0 or a 1 to each edge indicating its survival or failure, respectively, for iteration k.

Subsequently, a subgraph $G_k = (V_k, E_k)$ is produced, with $e_i \in E_k$ if and only if $\mu_k(e_i) = 0$ and $v_j \in V_k$ for $j = 1, 2, 3, \ldots, |V|$ if and only if there exists $e \in E_k$ such that $v_j \in e$ or if $v_j \in S$. More precisely, $E_k = \{e \in E \mid \mu_k(e) = 0\}$ and $V_k = \{v \in V \mid v \in e,\ e \in E_k\} \cup S$ in notation. For all $v_j \in V \setminus S$, a second indicator function is then defined as

$$\lambda_k(v_j) = \begin{cases} 0 & \text{if there exists a path in } G_k \text{ to some } s \in S \\ 1 & \text{otherwise} \end{cases},$$

specifying whether or not each vertex is reachable by a source for iteration k. If $v_j \in S$ then $\lambda_k(v_j) = 0$.

Given the above, the relative isolation probability of a vertex v_j within N instances, denoted by $\Pi_\iota^N(v_j)$, is

$$\Pi_\iota^N(v_j) = \frac{\sum_{l=1}^{N} \lambda_l(v_j)}{N}.$$

5.2. Monte Carlo Preliminary Processing

The preliminary processing step is first done through a Monte Carlo simulation, instituting the survival or failure of each edge and generating all G_k for $k = 1, 2, 3, \ldots, N$. Algorithm 3 shows the pseudocode, in which line 11 draws up the random numbers uniformly over the interval $(0, 1)$ for comparison to the relevant probabilities assigned to the edges. The product is a combined input file holding the resulting N instances of the graph.

5.3. Decision Diagram Computation Proper

The computation proper happens in the second step, outlined in Algorithm 4. For each simulation, a zero-suppressed binary decision diagram is constructed for each pair comprising a nonsource vertex v and a source vertex s to determine whether or not there is a path from s to v. The degree constraint dc prescribes v and s as the endpoints of a path, with both vertex degrees being one as designated

Algorithm 3 MCPrePro

```
1: $G \leftarrow$ readGraph()
2: $S \leftarrow$ readSources()
3: Assume $G = (V, E)$.
4:
5: print($|E|$, $N$)
6: printEOL
7:
8: for all $e \in E$ do
9:     print($e.u$, $e.v$)
10:    for $i = 1, \ldots, N$ do
11:        if (randUniform$(0, 1) \leq e.p$) then
12:            print(1)
13:        else
14:            print(0)
15:        end if
16:    end for
17:    printEOL
18: end for
19:
20: print(|S|)
21: printEOL
22:
23: for all $s \in S$ do
24:     print($s$)
25: end for
26:
27: printEOL
28: printEOF
```

Note that $e.p$ is the probability that the edge e is alive and $e.u$ and $e.v$ stand for the vertices of the edge e.

in Lines 11 and 14). The other vertices are either inner vertices of a path or are not included in the path, with vertex degrees being two or zero as designated in Lines 6, 20, and 25). Using dc, the zero-suppressed binary decision diagram denoted by *paths* in Line 15 that represents the set of all the paths from s to v is constructed by CONSTRUCTPATHZDD described in the previous section.

Thereafter, a path from s to v used for examining connectivity must only go through available edges. This constraint is represented by the zero-suppressed binary decision diagram *of* in Line 9. Let E_{avail} be the set of available edges in E. The function CONSTRUCTOMITFAILUREZDD constructs the diagram *of* representing the set of any subsets of E_{avail}, that is, the power set of E_{avail}. The diagram is constructed outside the for loops of v and s since it is independent from v and s.

The function ZDDINTERSECTION in Line 16 performs the intersection operation on the two zero-suppressed binary decision diagrams, constructing the diagram representing the intersection of the two families of sets represented by the two diagrams [5]. The intersection of *paths* and *of* generates the set of paths that use only the available edges as a zero-suppressed binary decision diagram. The existence of a path from the source to the vertex is then inspected by the comparison of *zdd* and $\perp$ in Line 17. If such a path does not exist then *zdd* becomes $\perp$, the diagram for the empty set. The check is carried out by the comparison of the two pointers directed to the roots of *zdd* and $\perp$ in constant time.

If a valid path from s to v is not found then the algorithm adds 1 to the isolation count of that vertex and moves on to the next vertex; otherwise, it simply moves on to the next vertex. In the end, isolation count of each vertex is divided by the number of simulations to get the probability of relative isolation.

As a comment, keep in mind that source vertices have a relative isolation probability of 0 since there always exists a path to itself. Sources are skipped when running the decision-diagram-based simulation part.

In summary, the algorithm calculates the relative isolation probabilities of the vertices by running a number of randomized simulations, checking if the vertex is disconnected from all sources in each simulation, and getting the percentage of simulations in which the vertex is relatively isolated.

The C++ programming language, coupled with version 9.3.0 of the g++ as

Algorithm 4 PIZDD

```
1: $G \leftarrow \textsc{readGraph}()$
2: $S \leftarrow \textsc{readSources}()$
3: Assume $G = (V, E)$.
4: Let $isolations$ and $dc$ be arrays whose indices are vertices.
5: $isolations[v] \leftarrow 0$ for all $v \in V$
6: $dc[v] \leftarrow \{0, 2\}$ for all $v \in V$
7:
8: for $sim = 1, \ldots, N$ do
9:     $of \leftarrow \textsc{constructOmitFailureZdd}(sim)$
10:    for $v \in V - S$ do
11:        $dc[v] \leftarrow \{1\}$
12:        $found \leftarrow$ false
13:        for $s \in S$ do
14:            $dc[s] \leftarrow \{1\}$
15:            $paths \leftarrow \textsc{constructPathZdd}(dc)$
16:            $zdd \leftarrow \textsc{zddIntersection}(paths, of)$
17:            if  $zdd \neq \perp$ then
18:                $found \leftarrow$ true
19:            end if
20:            $dc[s] \leftarrow \{0, 2\}$
21:            if $found$ then
22:                break
23:            end if
24:        end for
25:        $dc[v] \leftarrow \{0, 2\}$
26:        if not $found$ then
27:            $isolations[v] \leftarrow isolations[v] + 1$
28:        end if
29:    end for
30: end for
31:
32: for $v \in V$ do
33:     $\textsc{print}(v, isolations[v]/n)$
34: end for
```

compiler, is selected for implementation. Fundamental diagram manipulation is carried out through the TdZdd[2] library documented in [25]. The ZDDLines[3] program serves as foundation for the portion involving the enumeration of paths [45]. For the complete code, refer to the PIZDD[4] repository as published online.

Concerning machine specifications, the operating system is the Ubuntu 18.04.4 Long Term Support (Bionic Beaver). The processor is the Intel® Core™ i7-8565U running at 1.80GHz. A memory of 16GB and an NVIDIA® GeForce® MX250 graphics card are also made available for use.

5.4. Results on Varying Graphs

Two segments comprise the results. Outcomes on six sample networks are presented in the first part as benchmark. Graphs with varying structures, seen in Figures 5.1 to 5.6, are hand-picked from familiar published work [7, 48]. In the second part, the technique is evaluated on graphs representing real-world lifeline networks. One inspects the Bursa, Hanoi, and Kobe water supply systems [27, 28, 47].

Sample networks with $6 \leq |V| \leq 40$ and $15 \leq |E| \leq 58$ are first used as setting for the benchmarking experiment. In order, the six are a complete graph, a Petersen graph, two distinctive graphs, a square grid, and a rectangular mesh.

The number of simulations is initialized to 100, 1000, and 10000. To examine the algorithm correctness, the vertex relative isolation probabilities are computed for edge failure probabilities 0.50, 0.05, and 0.95. The routine is repeated nine times over per network.

Tables 5.1, 5.2, and 5.3 summarize the results. To reiterate, $|E|$ is the number of edges and $|V|$ is the number of vertices. The computation time for each of the two steps is presented both separately and as a sum. For example, $t_{\rho(e),1}$ and $t_{\rho(e),2}$ are costs in seconds for the Monte Carlo instance generator and the decision diagram connectivity test, respectively, and $t_{\rho(e)}$ is the overall time taken. The

[2]Link: `https://github.com/kunisura/TdZdd`.
[3]Link: `https://github.com/renzopereztan/ZDDLines`.
[4]Link: `https://github.com/renzopereztan/PIZDD`.

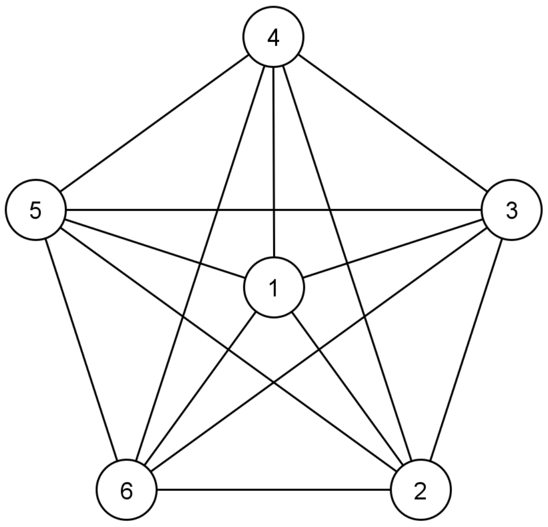


Figure 5.1.: Network 1.

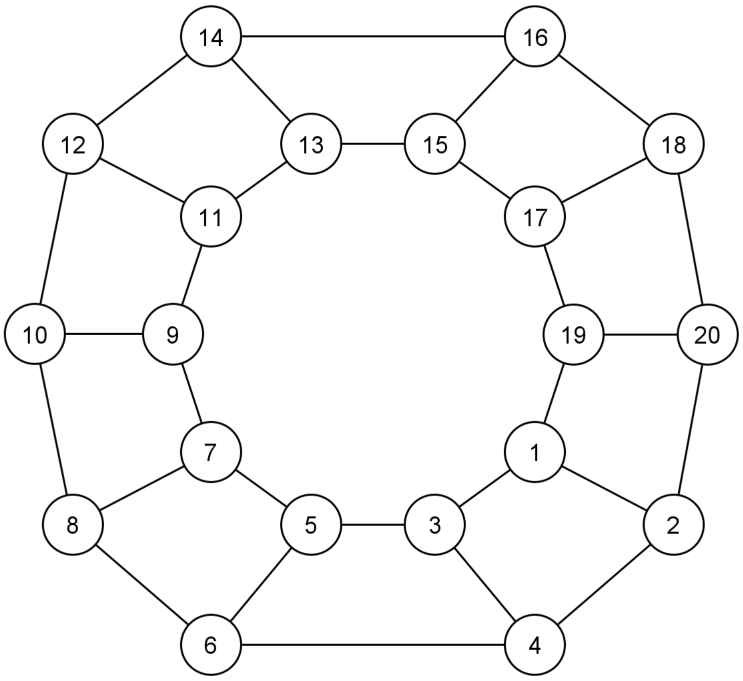


Figure 5.2.: Network 2.

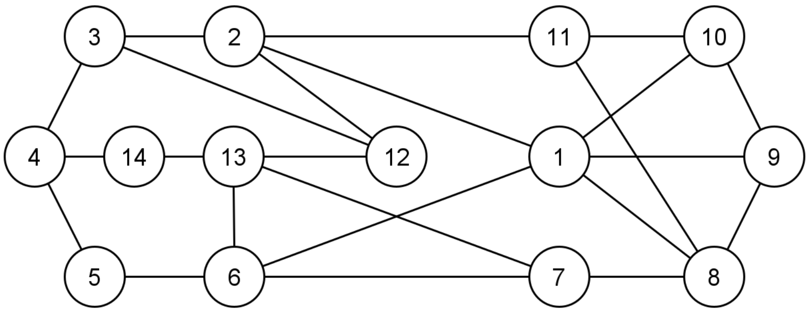


Figure 5.3.: Network 3.

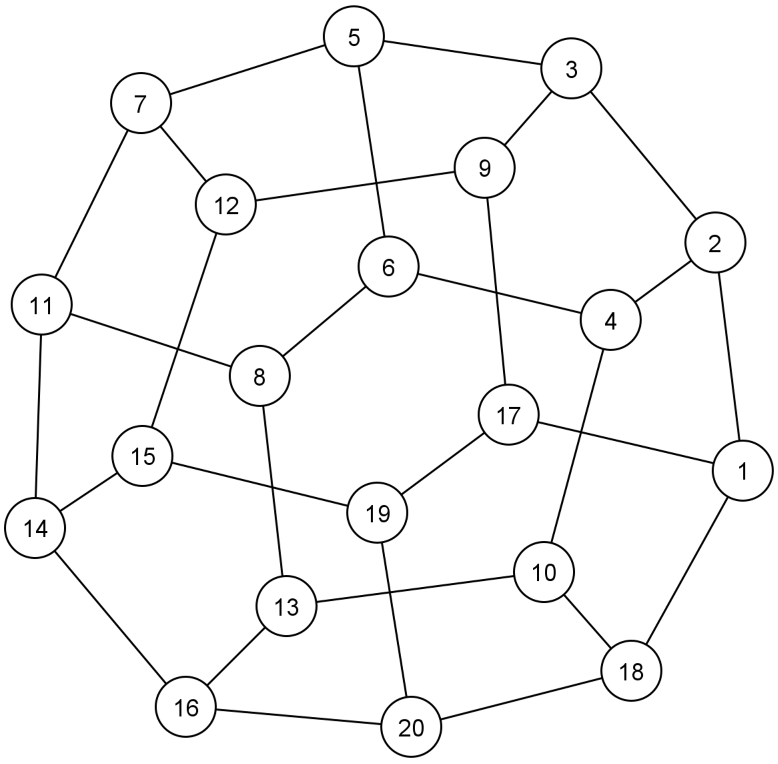


Figure 5.4.: Network 4.

average time consumed for all set-ups of equal N is $\bar{t}$. In symbols,

$$t_{\rho(e)} = t_{\rho(e),1} + t_{\rho(e),2}$$

and

$$\bar{t} = \frac{t_{\rho_1(e)} + t_{\rho_2(e)} + t_{\rho_3(e)}}{3}.$$

Considering the complexity of an enumerative decision-diagram-based approach, the consumption of time is fair [38,45]. The total cost ranges from several seconds for experiments with 100 iterations to several minutes for experiments with 10000 iterations.

The set of probabilities obtained for the graphs when all edges are set to fail half of the time are attached as Appendices C.1, C.2, C.3, C.4, C.5, and C.6. Setting the results from the different values for N side by side serve as preview to the convergence of the method.

The procedure is then applied to real-world systems. More specifically, the water distribution networks of three cities – Bursa in Turkey, Hanoi in Vietnam, and Kobe in Japan – are chosen for processing. For the three networks, $12 \leq |V| \leq 32$ and $17 \leq |E| \leq 34$.

The program is run thrice for each network, accommodating 100, 1000, and 10000 instances. The edge probabilities of failure differ per network. Information is sought from references that provide well-justified data. Details on the Bursa city network is found in [41]. Furthermore, a comprehensive analysis of the Hanoi city network may be seen in [47]. Lastly, the Kobe city network is treated in [29].

The results for the lifeline networks are in Tables 5.4, 5.5, and 5.6, following the same format as the previous tables. The vertex probabilities of relative isolation are consistently calculated in reasonable time. Less than two seconds is required for 100 iterations and less than two minutes is required for 10000 iterations. Likewise, the probabilities are noted as Appendices C.7, C.8, and C.9 for reference.

As an aside, accuracy is further ensured by running the entire pool of experiments multiple times. One remarks that the variability between the initializations is not substantial.

Table 5.1.: Summary figures of the experiment on baselines with $N = 100$.

Network	$\|E\|$	$\|V\|$	$t_{0.50,1}$	$t_{0.50,2}$	$t_{0.50}$	$t_{0.05,1}$	$t_{0.05,2}$	$t_{0.05}$	$t_{0.95,1}$	$t_{0.95,2}$	$t_{0.95}$	$\bar{t}$
1	15	6	0.0103	0.1736	0.1839	0.0024	0.1236	0.1261	0.0021	0.1919	0.1940	0.1680
2	30	20	0.0033	1.0636	1.0669	0.0162	0.5781	0.5943	0.0032	1.9426	1.9458	1.2023
3	23	14	0.0178	0.9262	0.9440	0.0028	0.8109	0.8137	0.0024	1.3442	1.3465	1.0347
4	30	20	0.0044	1.9757	1.9801	0.0038	1.2902	1.2940	0.0025	2.3726	2.3751	1.8831
5	40	25	0.0085	3.1381	3.1466	0.0036	2.4443	2.4478	0.0036	3.1186	3.1222	2.9056
6	58	40	0.0053	1.7131	1.7183	0.0033	1.0292	1.0325	0.0033	4.5098	4.5132	2.4213

Table 5.2.: Summary figures of the experiment on baselines with $N = 1000$.

Network	$\|E\|$	$\|V\|$	$t_{0.50,1}$	$t_{0.50,2}$	$t_{0.50}$	$t_{0.05,1}$	$t_{0.05,2}$	$t_{0.05}$	$t_{0.95,1}$	$t_{0.95,2}$	$t_{0.95}$	$\bar{t}$
1	15	6	0.0053	1.7947	1.8001	0.0051	1.2479	1.2530	0.0059	2.1511	2.1570	1.7367
2	30	20	0.0105	9.9300	9.9405	0.0083	5.8300	5.8383	0.0090	17.5446	17.5536	11.1108
3	23	14	0.0085	11.0732	11.0817	0.0149	7.9114	7.9262	0.0068	12.3743	12.3811	10.4630
4	30	20	0.0100	16.5517	16.5617	0.0094	12.7018	12.7112	0.0100	22.5323	22.5423	17.2718
5	40	25	0.0211	32.8138	32.8349	0.0108	24.4238	24.4346	0.0106	45.9786	45.9892	34.4196
6	58	40	0.0164	18.2109	18.2273	0.0174	12.0532	12.0706	0.0171	68.3464	68.3635	32.8871

Table 5.3.: Summary figures of the experiment on baselines with $N = 10000$.

Network	$\lvert E\rvert$	$\lvert V\rvert$	$t_{0.50,1}$	$t_{0.50,2}$	$t_{0.50}$	$t_{0.05,1}$	$t_{0.05,2}$	$t_{0.05}$	$t_{0.95,1}$	$t_{0.95,2}$	$t_{0.95}$	$\bar{t}$
1	15	6	0.0390	17.3316	17.3706	0.0376	12.3935	12.4311	0.0365	20.3875	20.4240	16.7419
2	30	20	0.0732	107.6360	107.7092	0.0689	55.6890	55.7579	0.0682	193.2180	193.2862	118.9178
3	23	14	0.0563	122.2990	122.3553	0.0636	84.4677	84.5313	0.0587	151.2630	151.3217	119.4028
4	30	20	0.0737	184.9180	184.9917	0.0744	140.5700	140.6444	0.0692	237.3070	237.3762	187.6708
5	40	25	0.0969	361.1590	361.2559	0.0929	252.3040	252.3969	0.0896	435.6780	435.7676	349.8068
6	58	40	0.1342	199.3880	199.5222	0.1306	116.1300	116.2606	0.1297	684.7590	684.8887	333.5572

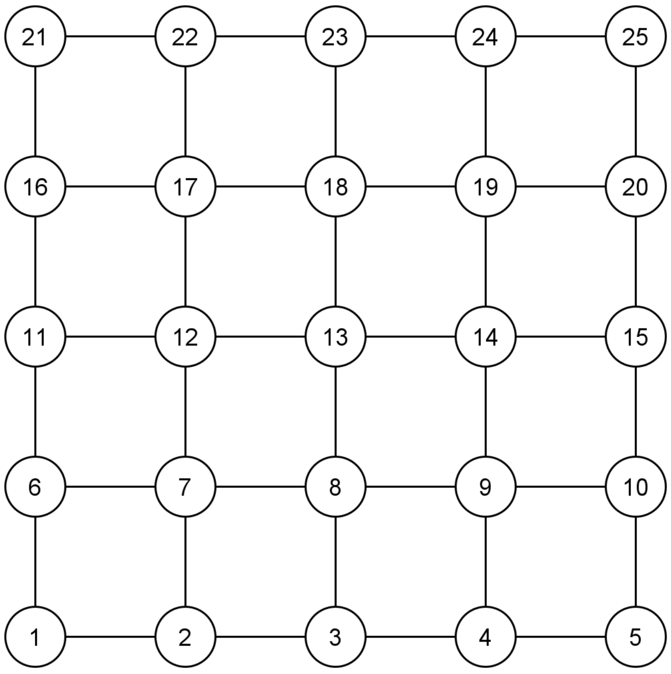


Figure 5.5.: Network 5.

Table 5.4.: Summary figures of the experiment on applications with $N = 100$.

Network	$\|E\|$	$\|V\|$	t_1	t_2	t
Bursa	17	12	0.0040	0.4289	0.4330
Hanoi	34	32	0.0033	1.4145	1.4178
Kobe	20	15	0.0025	0.5263	0.5288

Table 5.5.: Summary figures of the experiment on applications with $N = 1000$.

Graph	$\|E\|$	$\|V\|$	t_1	t_2	t
Bursa	17	12	0.006	4.2833	4.2898
Hanoi	34	32	0.0099	12.7125	12.7224
Kobe	20	15	0.0057	5.6930	5.6987

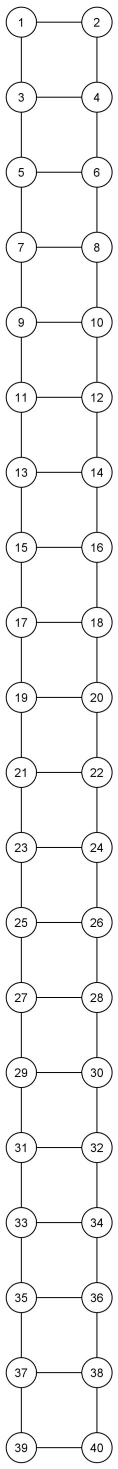


Figure 5.6.: Network 6.

Table 5.6.: Summary figures of the experiment on applications with $N = 10000$.

Network	$\lvert E\rvert$	$\lvert V\rvert$	t_1	t_2	t
Bursa	17	12	0.0448	41.478	41.5926
Hanoi	34	32	0.0985	114.5840	114.6825
Kobe	20	15	0.0505	57.1823	57.2328

6. Conclusion

In summary, a novel solution to the known generalized directed rural postman problem and the unconventional crazy generalized directed rural postman problem is put forth. The two nondeterministic polynomial-time hard problems were simultaneously solved by the decision-diagram-based procedure in minutes while a traditional integer program analogue consumed several hours on each problem. Moreover, the capacity for enumerating solutions and the flexibility in adjustment further the relative superiority of the use of the decision diagram.

Furthermore, the paper has advanced an enumerative technique for the solution of the undirected rural postman problem based on the zero-suppressed binary decision diagram. Exact solutions to a diverse set of problem instances were reached in time well-justified for methods of enumeration. Observations on performance were pointed out towards the improvement of the algorithm.

A novel measure for graphs consisting of edges that have known failure probabilities is also put forward. Its definition is accompanied by a demonstration of efficient calculation through a Monte Carlo method integrated with a decision-diagram-based technique. The new probabilistic metric is worked out across selected networks, proving reasonable computational cost.

For future research, preprocessing may be done to simplify the graphs before diagram construction. Another possible plan of action to reduce computation time is parallelization. As prospects for comparison are scarce, other classes of binary decision diagrams may also be considered so that the proposed approach may be set side by side with other decision-diagram-based solutions.

With respect to the proposed graph metric, an original consolidated measure for the whole graph based on the vertex probabilities of relative isolation is to be pursued. Regarding probability evaluation, crafting an algorithm that offers less computation time and an implementation on larger networks are suggested.

Appendices

A. Codes

A.1. Line Constraint

```
class Lines: public tdzdd::DdSpec<Lines, int, 2> {
public:
        Lines(){}
        int getRoot(int& state)
        const{
        state = 0;
        return n;}
        int getChild(int& state, int level, int value)
        const{
                if(value == 1) state |= (1<<l[n-level]);
                level--;

                if(level == 0){
                        if(state == ((1<<L)-1)<<1){
                                return -1;
                        }else{
                                return 0;
                        }
                }
        return level;
        }
};
```

A.2. Extracting the Minimum

```
class MinDist: public tdzdd::DdEval<MinDist,DistData> {
public:
        MinDist(){}
        void evalTerminal(DistData& data, bool one)
        const {
                data.val = one ? 0 : 100000;}
        void evalNode(DistData& data, int level,
        tdzdd::DdValues<DistData,2> const& values)
        const {
                const DistData& data0 = values.get(0);
                const DistData& data1 = values.get(1);
                if(data0.val <= data1.val + d[n-level]){
                        data.val = data0.val;
                        data.mask = data0.mask;
                }else{
                        data.val = data1.val + d[n-level];
                        data.mask = data1.mask;
                        data.mask[level-1] = 1;
                }
        }
};
```

A.3. Extracting the Maximum

```
class MaxDist: public tdzdd::DdEval<MaxDist,DistData> {
public:
        MaxDist(){}
        void evalTerminal(DistData& data, bool one)
        const {
                data.val = one ? 0 : INT_MIN;}
        void evalNode(DistData& data, int level,
        tdzdd::DdValues<DistData,2> const& values)
        const {
                const DistData& data0 = values.get(0);
                const DistData& data1 = values.get(1);
                if(data0.val >= data1.val + d[n-level]){
                        data.val = data0.val;
                        data.mask = data0.mask;
                }else{
                        data.val = data1.val + d[n-level];
                        data.mask = data1.mask;
                        data.mask[level-1] = 1;
                }
        }
};
```

B. Lines

B.1. Hong Kong Metro

Icon	**Name**
🟦	Airport Express Line
🟦	East Rail Line
🟦	Island Line
🟩	Kwun Tong Line
🟫	Tuen Ma Line
🟩	South Island Line
🟪	Tseung Kwan O Line Line
🟥	Tsuen Wan Line
🟧	Tung Chung Line
🟪	West Rail Line

B.2. Osaka Metro

Icon	Name
M	Midosuji Line
T	Tanimachi Line
Y	Yotsubashi Line
C	Chuo Line
S	Sennichimae Line
K	Sakaisuji Line
N	Nagahori Tsurumi-Ryokuchi Line
I	Imazatosuji Line
P	Nanko Port Town Line

B.3. Taipei Metro

Icon	Name
BR	Wenhu Line
R	Tamsui-Xinyi Line
G	Songshan-Xindian Line
O	Zhonghe-Xinlu Line
BL	Bannan Line
Y	Circular Line
✈	Airport Line

C. Probabilities

C.1. Network 1

j	$\Pi_\iota^{100}(v_j)$	$\Pi_\iota^{1000}(v_j)$	$\Pi_\iota^{10000}(v_j)$
$1^\dagger$	0.0000	0.0000	0.0000
2	0.0600	0.0400	0.0409
3	0.0400	0.0430	0.0435
4	0.0300	0.0480	0.0446
5	0.0500	0.0400	0.0463
$6^\dagger$	0.0000	0.0000	0.0000

†Indicates a source vertex.

C.2. Network 2

j	$\Pi_\iota^{100}(v_j)$	$\Pi_\iota^{1000}(v_j)$	$\Pi_\iota^{10000}(v_j)$
1	0.4500	0.3360	0.3234
$2^\dagger$	0.0000	0.0000	0.0000
3	0.5300	0.4580	0.4289
4	0.3900	0.3720	0.3533
5	0.5400	0.5070	0.5087
6	0.5200	0.5230	0.5005
7	0.5700	0.4910	0.5034
8	0.5500	0.5460	0.5164
9	0.4000	0.3810	0.3619
10	0.4100	0.4680	0.4423
$11^\dagger$	0.0000	0.0000	0.0000
12	0.3200	0.3330	0.3352
13	0.4100	0.3710	0.3590
14	0.4600	0.4320	0.4385
15	0.5200	0.5220	0.5016
16	0.5100	0.4920	0.5151
17	0.5100	0.5230	0.5118
18	0.4700	0.4950	0.4916
19	0.4400	0.4450	0.4391
20	0.3300	0.3280	0.3556

C.3. Network 3

j	$\Pi_{\iota}^{100}(v_j)$	$\Pi_{\iota}^{1000}(v_j)$	$\Pi_{\iota}^{10000}(v_j)$
1	0.1700	0.1520	0.1410
2	0.2400	0.3220	0.3113
3	0.3600	0.4350	0.4058
4	0.2900	0.3490	0.3411
$5^\dagger$	0.0000	0.0000	0.0000
6	0.2000	0.2140	0.2038
7	0.3200	0.3170	0.3067
8	0.1500	0.1910	0.1825
$9^\dagger$	0.0000	0.0000	0.0000
10	0.2300	0.2490	0.2257
11	0.3000	0.3180	0.3062
12	0.3900	0.4240	0.3964
13	0.2800	0.3500	0.3255
14	0.4900	0.4950	0.4922

C.4. Network 4

j	$\Pi_{\iota}^{100}(v_j)$	$\Pi_{\iota}^{1000}(v_j)$	$\Pi_{\iota}^{10000}(v_j)$
$1^\dagger$	0.0000	0.0000	0.0000
2	0.2600	0.3400	0.3321
3	0.4500	0.4000	0.4408
4	0.4400	0.4440	0.4424
5	0.4800	0.4240	0.4407
6	0.4900	0.4380	0.4460
7	0.3200	0.3490	0.3350
8	0.3800	0.3400	0.3455
9	0.4500	0.4320	0.4423
10	0.4200	0.4250	0.4394
$11^\dagger$	0.0000	0.0000	0.0000
12	0.4200	0.4390	0.4416
13	0.4700	0.4240	0.4493
14	0.3200	0.3390	0.3371
15	0.4500	0.4280	0.4450
16	0.4900	0.4360	0.4473
17	0.3800	0.3160	0.3301
18	0.2700	0.3240	0.3362
19	0.5000	0.4300	0.4494
20	0.4300	0.4420	0.4484

C.5. Network 5

j	$\Pi_{\iota}^{100}\left(v_j\right)$	$\Pi_{\iota}^{1000}\left(v_j\right)$	$\Pi_{\iota}^{10000}\left(v_j\right)$
$1^{\dagger}$	0.0000	0.0000	0.0000
2	0.3700	0.3370	0.3467
3	0.5000	0.4670	0.4983
4	0.5700	0.5610	0.5938
5	0.7100	0.6560	0.6805
6	0.3700	0.3320	0.3422
7	0.4400	0.3700	0.3828
8	0.4700	0.3990	0.4487
9	0.5300	0.4650	0.5078
10	0.6000	0.5530	0.5973
11	0.5100	0.5050	0.4902
12	0.5200	0.4480	0.4468
13	0.4800	0.4310	0.4448
14	0.5000	0.4170	0.4523
$15^{\dagger}$	0.5100	0.4620	0.5039
16	0.6800	0.5910	0.5756
17	0.5600	0.5040	0.5005
18	0.5000	0.4520	0.4510
19	0.3800	0.3590	0.3966
20	0.3700	0.3300	0.3556
21	0.7900	0.6900	0.6724
22	0.6600	0.5910	0.5832
23	0.6200	0.4820	0.4992
24	0.4000	0.3430	0.3565
25	0.0000	0.0000	0.0000

C.6. Network 6

j	$\Pi_\iota^{100}(v_j)$	$\Pi_\iota^{1000}(v_j)$	$\Pi_\iota^{10000}(v_j)$
$1^\dagger$	0.0000	0.0000	0.0000
2	0.4800	0.4520	0.4328
3	0.3800	0.4210	0.4275
4	0.5100	0.5430	0.5328
5	0.6200	0.6610	0.6453
6	0.6400	0.6850	0.6687
7	0.7200	0.7840	0.7698
8	0.7300	0.7820	0.7790
9	0.8000	0.8480	0.8518
10	0.7600	0.8550	0.8538
11	0.8800	0.9000	0.9020
12	0.8500	0.9050	0.9059
13	0.9200	0.9250	0.9373
14	0.9100	0.9320	0.9359
15	0.9400	0.9550	0.9569
16	0.9600	0.9500	0.9575
17	0.9600	0.9660	0.9667
18	0.9600	0.9690	0.9675
19	0.9900	0.9760	0.9720
20	0.9800	0.9780	0.9725
21	0.9900	0.9730	0.9726
22	0.9800	0.9820	0.9716
23	0.9700	0.9660	0.9689
24	0.9700	0.9720	0.9680
25	0.9500	0.9540	0.9569
26	0.9400	0.9540	0.9580
27	0.9300	0.9370	0.9364
28	0.9000	0.9380	0.9372
29	0.9000	0.9080	0.9026
30	0.8800	0.9160	0.9035
31	0.8300	0.8690	0.8538
32	0.8300	0.8670	0.8510
33	0.7700	0.7880	0.7799
34	0.7900	0.7790	0.7758
35	0.7100	0.6630	0.6733
36	0.6700	0.6320	0.6481
37	0.5100	0.5130	0.5304
38	0.4000	0.4190	0.4281
39	0.4800	0.4190	0.4299
$40^\dagger$	0.0000	0.0000	0.0000

C.7. Bursa Network

j	$\Pi_{\iota}^{100}(v_j)$	$\Pi_{\iota}^{1000}(v_j)$	$\Pi_{\iota}^{10000}(v_j)$
$1^\dagger$	0.0000	0.0000	0.0000
2	0.0200	0.0200	0.0250
3	0.0000	0.0140	0.0114
4	0.0000	0.0060	0.0037
5	0.0200	0.0080	0.0033
6	0.0100	0.0060	0.0044
$7^\dagger$	0.0000	0.0000	0.0000
8	0.0000	0.0000	0.0000
9	0.0000	0.0040	0.0038
10	0.0100	0.0320	0.0266
11	0.2100	0.1490	0.1439
12	0.0000	0.0000	0.0000

C.8. Hanoi Network

j	$\Pi_\iota^{100}(v_j)$	$\Pi_\iota^{1000}(v_j)$	$\Pi_\iota^{10000}(v_j)$
$1^\dagger$	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0000
4	0.0700	0.1090	0.1026
5	0.1500	0.2160	0.2147
6	0.2600	0.3460	0.3403
7	0.3400	0.4690	0.4586
8	0.4300	0.5460	0.5399
9	0.5200	0.5950	0.5848
10	0.5800	0.5970	0.5949
11	0.7300	0.6880	0.7011
12	0.8300	0.7600	0.7766
13	0.8900	0.8280	0.8317
14	0.6400	0.5830	0.5619
15	0.5400	0.5300	0.4906
16	0.4100	0.4050	0.3826
17	0.2800	0.3150	0.3109
18	0.2200	0.2090	0.2099
19	0.1600	0.1060	0.1084
20	0.0600	0.0550	0.0598
21	0.0600	0.0610	0.0729
$22^\dagger$	0.0000	0.0000	0.0000
23	0.2400	0.2310	0.2467
24	0.3600	0.3360	0.3452
25	0.3300	0.3770	0.3823
26	0.4500	0.4430	0.4322
27	0.4600	0.4510	0.4384
28	0.3600	0.3870	0.3756
29	0.4600	0.4500	0.4475
30	0.4900	0.4430	0.4503
31	0.3300	0.3770	0.3823
32	0.3300	0.3770	0.3823

C.9. Kobe Network

j	$\Pi_{\iota}^{100}(v_j)$	$\Pi_{\iota}^{1000}(v_j)$	$\Pi_{\iota}^{10000}(v_j)$
$1^{\dagger}$	0.0000	0.0000	0.0000
2	0.2300	0.2890	0.2552
3	0.2200	0.1600	0.1688
4	0.1100	0.0440	0.0418
5	0.1700	0.1460	0.1533
$6^{\dagger}$	0.0000	0.0000	0.0000
7	0.1700	0.1480	0.1490
8	0.1800	0.1640	0.1653
9	0.1200	0.1330	0.1400
10	0.0700	0.0440	0.0509
11	0.0600	0.0260	0.0268
12	0.0600	0.0270	0.0288
13	0.1100	0.0250	0.0338
14	0.0000	0.0000	0.0000
15	0.0000	0.0000	0.0000

Acknowledgements

For the people I love

References

[1] R. Angel, W. Caulde, R. Noonan, and A. Whinston. Computer-assisted school bus scheduling. *Management Science Vol. 18 No. 6*, pages 279–288, 1972.

[2] T. Avila, A. Corberan, I. Plana, and J. Sanchis. A new branch-and-cut algorithm for the generalized directed rural postman problem. *Transportation Science Vol. 50 No. 2*, pages 750–761, 2016.

[3] L. Bodin and S. Kursh. A computer-assisted system for the routing and scheduling of street sweepers. *Operations Research Vol. 26 No. 4*, pages 528–537, 1978.

[4] J. Braca, J. Bramel, B. Posner, and D. Simchi-Levi. A computerized approach to the new york city school bus routing project. *Columbia University Working Paper*, 1993.

[5] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers Vol. 35 No. 8*, pages 677–691, 1986.

[6] P. Cerny, T. Henzinger, L. Kovacs, A. Radhakrishna, and J. Zwirchmayr. Segment abstraction for worst-case execution time analysis. *Proceedings of Programming Languages and Systems: The 24th European Symposium of Programming*, pages 105–131, 2015.

[7] S. Chaterjee, V. Ramana, G. Vishwakarma, and A. Verma. An improved algorithm for k-terminal probabilistic network reliability analysis. *Journal of Reliability and Statistical Studies Vol. 10 Iss. 1*, pages 15–26, 2017.

[8] N. Christofides, V. Campos, A. Corberan, and E. Mota. An algorithm for the rural postman problem. *Imperial College London Technical Report IC.OR.81.5*, 1981.

[9] A. Corberan and G. Laporte. *Arc Routing: Problems, Methods, and Applications.* Society for Industrial and Applied Mathematics, 2015.

[10] A. Corberan and J. Sanchis. A polyhedral approach for the rural postman problem. *European Journal of Operations Research Vol. 79 Iss. 1*, pages 95–114, 1994.

[11] P. Fernandez de Cordoba, L. Garcia-Raffi, and J. Sanchis. A heuristic algorithm based on monte carlo methods for the rural postman problem. *Computers and Operations Research Vol. 25 Iss. 12*, pages 1097–1106, 1998.

[12] J. Desrosiers, J. Ferland, J. Rousseau, G. Lapalme, and L. Chapleau. TRANSCOL: A multi-period school bus routing and scheduling system. *Studies in the Management Sciences Vol. 22*, pages 47–71, 1986.

[13] M. Drexl. On the generalized directed rural postman problem. *Journal of the Operations Research Society Vol. 65 No. 8*, pages 1143–1154, 2014.

[14] J. Edmonds and E. Johnson. Matching, euler tours, and the chinese postman. *Mathematical Programming Vol. 5 No. 1*, pages 88–124, 1973.

[15] H. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part ii: The rural postman problem. *Operations Research Vol. 43 No. 3*, pages 399–414, 1995.

[16] W. Ellens and R. Kooij. Graph measures and network robustness. *arXiv e-print 1311.5064*, 2013.

[17] E. Fernandez, O. Meza, R. Garfinkel, and M. Ortega. On the undirected rural postman problem: Tight bounds based on a new formulation. *Operations Research Vol. 51 No. 2*, pages 281–291, 2003.

[18] G. Frederickson, M. Hecht, and C. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing Vol. 7 No. 2*, pages 178–193, 1978.

[19] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[20] G. Ghiani and G. Laporte. *Arc Routing: Problems, Methods, and Applications*, chapter The Undirected Rural Postman Problem, pages 85–99. Society for Industrial and Applied Mathematics, 2015.

[21] E. Haslam and J. Wright. Applications of routing technologies to rural snow and ice control. *Transportation Research Record No. 1304*, pages 202–211, 1991.

[22] A. Hertz. *Graph Theory, Combinatorics, and Algorithms*, chapter Recent Trends in Arc Routing, pages 215–236. Springer, 2006.

[23] A. Hertz, G. Laporte, and P. Hugo. Improvement procedures for the undirected rural postman problem. *INFORMS Journal on Computing Vol. 11 No. 1*, pages 53–62, 1999.

[24] 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*, pages 102–111, 2014.

[25] H. Iwashita and S. Minato. Efficient top-down ZDD construction techniques using recursive specifications. *Hokkaido University Division of Computer Science TCS Technical Report TCS-TR-A-13-69*, 2013.

[26] H. Iwashita, Y. Nakazawa, J. Kawahara, T. Uno, and S. Minato. ZDD-based computation of the number of paths in a graph. *Hokkaido University Division of Computer Science TCS Technical Report TCS-TR-A-12-60*, 2012.

[27] M. Javanbarg, J. Kiyono, and M. Ghazizadeh. Reliability analysis of lifeline networks using binary decision diagram. *In Proceedings of the 4th International Conference on Modern Research in Civil Engineering, and Architectural and Urban Development*, 2016.

[28] M. Javanbarg, C. Scawthorn, J. Kiyono, and Y. Ono. Reliability analysis of infrastructure and lifeline networks using obdd. *In Safety, Reliability, and*

Risk of Structures, Infrastructures, and Engineering Systems: Proceedings of the 10th International Conference on Structural Safety and Reliability, pages 3463–3470, 2010.

[29] M. Javanbarg and S. Takada. Redundancy model for water supply systems under earthquake environments. *Proceedings of the 5th International Conference on Seismology and Earthquake Engineering*, page 16094, 2007.

[30] J. Kawahara, T. Inoue, H. Iwashita, and S. Minato. Frontier-based search for enumerating all constrained subgraphs with compressed representation. *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences Vol. E100-A No. 9*, pages 1773–1784, 2017.

[31] D. Knuth. *The Art of Computer Programming Vol. 4 Fasc. 1.* Addison-Wesley, 2009.

[32] J. Lenstra and A. Rinnooy Kan. On general routing problems. *Networks Vol. 6 No. 3*, pages 273–280, 1976.

[33] S. Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. *Proceedings of the 30th International Design Automation Conference*, pages 272–277, 1993.

[34] S. Minato. Zero-suppressed BDDs and their applications. *International Journal on Software Tools for Technology Transfer Vol. 3 No. 2*, pages 156–170, 2001.

[35] R. Miyashiro, T. Kasai, and T. Matsui. Strictly solving the longest one-way ticket problem. *Proceedings of the 2000 Fall National Conference of the Operations Research Society of Japan*, pages 24–25, 2000.

[36] B. Mojar. The laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications: Proceedings of the 6th Quadrennial International Conference on the Theory and Applications of Graphs Vol. 2*, pages 871–898, 1991.

[37] M. Moreira and J. Ferreira. A genetic algorithm for the undirected rural postman problem. 2010.

[38] D. Morrison, E. Sewell, and S. Jacobson. Solving the pricing problem in a branch-and-price algorithm for graph coloring using zero-suppressed binary decision diagrams. *INFORMS Journal on Computing Vol. 28 No. 1*, pages 67–82, 2016.

[39] C. Orloff. A fundamental problem in vehicle routing. *Networks Vol. 4 Iss. 1*, pages 35–64, 1974.

[40] H. Perez-Roses. Sixty years of network reliability. *Mathematics in Computer Science Vol. 12*, pages 275–293, 2018.

[41] A. Selcuk and M. Yucemen. Reliability of lifeline networks with multiple sources under seismic hazard. *Natural Hazards Vol. 21*, pages 1–18, 2000.

[42] F. Sikora. The shortest way to visit all metro lines in a city. *arXiv Electronic Preprint arXiv:1709.05948*, 2018.

[43] A. Takizawa, Y. Miyata, and N. Katoh. Enumeration of floor plans based on a zero-suppressed binary decision diagram. *International Journal of Architectural Computing Iss. 1 Vol. 13*, pages 25–44, 2015.

[44] 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. *Proceedings of the 11th International Symposium on Operations Research and Its Applications in Engineering, Technology, and Management*, pages 1–8, 2013.

[45] R. Tan, J. Kawahara, A. Garciano, and I. Sin. A zero-suppressed binary decision diagram approach for constrained path enumeration. *Lecture Notes in Engineering and Computer Science: Proceedings of the World Congress on Engineering 2019*, pages 132–136, 2019.

[46] T. Toda, T. Saitoh, H. Iwashita, J. Kawahara, and S. Minato. ZDDs and enumeration problems: State-of-the-art techniques and programming tool. *Computer Software Vol. 34 No. 3*, pages 97–120, 2017.

[47] A. Vasan and S. Simonovic. Optimization of water distribution network design using differential evolution. *Journal of Water Resources Planning and Management Vol. 136 Iss. 2*, pages 279–287, 2010.

[48] F. Yeh and S. Kuo. Obdd-based network reliability calculation. *Electronics Letters Vol. 33 No. 9*, pages 759–760, 1997.

[49] R. Yoshinaka, J. Kawahara, S. Denzumi, H. Arimura, and S. Minato. Counterexamples to the long-standing conjecture on the complexity of BDD binary operations. *Information Processing Letters Vol. 112 No. 16*, pages 636–640, 2012.

Publication List

Journal Articles

[1] Renzo Roel P. Tan, Jun Kawahara, Kazushi Ikeda, Agnes Garciano, and Kyle Stephen S. See. Concerning a Decision-Diagram-Based Solution to the Generalized Directed Rural Postman Problem. International Journal of Computer Science 47 (2). 2020.

[2] Renzo Roel P. Tan, Kyle Stephen S. See, Jun Kawahara, Kazushi Ikeda, Richard M. de Jesus, Lessandro Estelito O. Garciano, and Agnes Garciano. The Relative Isolation Probability of a Vertex in a Multiple-Source Edge-Weighted Graph. Engineering Letters. Submitted (May 2021).

[3] Aldrich Ellis C. Asuncion, Renzo Roel P. Tan, Christian Paul O. Chan Shio, and Kazushi Ikeda. Recursive Linear Bounds for the Vertex Chromatic Number of the Pancake Graph. International Journal of Applied Mathematics. Submitted (July 2021).

Other Papers

[4] Renzo Roel P. Tan, Kazushi Ikeda, and Len Patrick Dominic M. Garces. On eigenvalue bounds for the finite-state birth-death process intensity matrix. Journal of Physics: Conference Series 1593. 2020.

[5] Renzo Roel P. Tan, Kazushi Ikeda, and John Paul C. Vergara. Hindsight-Combined and Hindsight-Prioritized Experience Replay. In Lecture Notes in Computer Science: Neural Information Processing. Springer Nature. 2020.

[6] Renzo Roel P. Tan, Florian Sikora, Kazushi Ikeda, and Kyle Stephen S. See. Arc Routing Based on the Zero-Suppressed Binary Decision Diagram. In Transactions on Engineering Technologies. Springer Nature. 2021.