*Algorithms* **2012**, *5*, 176-213; doi:10.3390/a5020176

OPEN ACCESS

*algorithms*

**ISSN 1999-4893**

www.mdpi.com/journal/algorithms

*Article*

# Finding All Solutions and Instances of Numberlink and Slitherlink by ZDDs

**Ryo Yoshinaka [1], Toshiki Saitoh [2,3,*], Jun Kawahara [2,3], Koji Tsuruma [2], Hiroaki Iwashita [2,3] and Shin-ichi Minato [2,3]**

[1] Graduate School of Informatics, Kyoto University, Kyoto, 606-8501, Japan;
E-Mail: ry@i.kyoto-u.ac.jp

[2] ERATO MINATO Project, Japan Science and Technology Agency, Kitaku North 14, West 9, Sapporo, 060-0814, Hokkaido, Japan; E-Mails: jkawahara@erato.ist.hokudai.ac.jp (J.K.); tsuruma@erato.ist.hokudai.ac.jp (K.T.); iwashita@erato.ist.hokudai.ac.jp (H.I.); minato@ist.hokudai.ac.jp (S.M.)

[3] Graduate School of Information Science and Technology, Hokkaido University, Kitaku North 14, West 9, Sapporo, 060-0814, Hokkaido, Japan

[*] Author to whom correspondence should be addressed; E-Mail: t-saitoh@erato.ist.hokudai.ac.jp;
Tel.: +81-11-728-8280.

*Received: 16 November 2011; in revised form: 29 February 2012 / Accepted: 19 March 2012 / Published: 5 April 2012*

**Abstract:** Link puzzles involve finding paths or a cycle in a grid that satisfy given local and global properties. This paper proposes algorithms that enumerate solutions and instances of two link puzzles, Slitherlink and Numberlink, by zero-suppressed binary decision diagrams (ZDDs). A ZDD is a compact data structure for a family of sets provided with a rich family of set operations, by which, for example, one can easily extract a subfamily satisfying a desired property. Thanks to the nature of ZDDs, our algorithms offer a tool to assist users to design instances of those link puzzles.

**Keywords:** link puzzles; Slitherlink; Numberlink; solvers; instance generations

## 1. Introduction

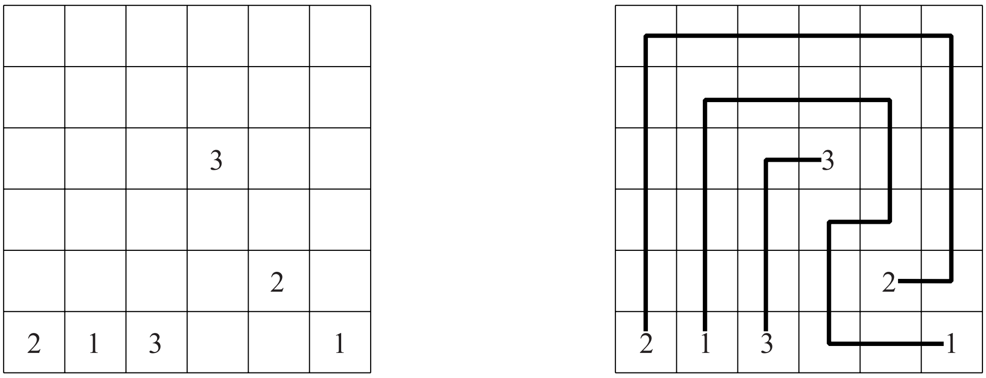
*Link puzzles*, for example Slitherlink, Numberlink, Masyu, *etc*., are logic puzzles that involve finding paths or a cycle in a grid that satisfy given local and global properties. This paper focuses on two simple link puzzles, *Numberlink* and *Slitherlink*, amongst others.

Numberlink is played on a grid where the rules are below [1].

1. Connect pairs of the same numbers with a continuous line.
2. Lines go through the center of the cells, horizontally, vertically, or changing direction, and never twice through the same cell.
3. Lines cannot cross, branch off, or go through the cells with numbers.

Figure 1 describes an instance of Numberlink. Numberlink is known to be NP-complete [2–4]. Numberlink is also studied as a model of VLSI layout design [5].

**Figure 1.** An instance of Numberlink and its solution.

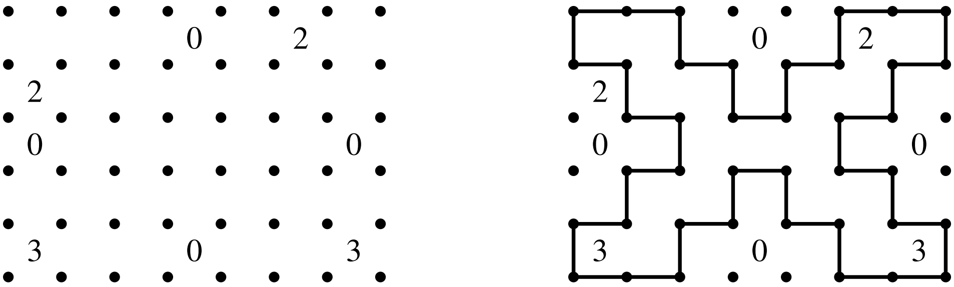


Slitherlink is played on a grid of lattice dots where we draw a loop satisfying below [1].

1. Connect adjacent dots with vertical or horizontal lines.
2. A single loop is formed with no crossing or branches.
3. The numbers indicate how many lines surround it, while empty cells may be surrounded by any number of lines.

We describe an instance of Slitherlink in Figure 2. Slitherlink is known to be NP-complete [6,7].

**Figure 2.** An instance of Slitherlink and its solution.



Link puzzles have been popularized by a Japanese publisher Nikoli and many puzzle designers are trying to produce good and interesting instances. The least common criterion on instances to be good is to admit exactly one solution, but what should be good and interesting is quite a subjective issue to respective designers. The complexity barrier suggests difficulties in the instance design.

On the other hand, as is usually the case for logic puzzles, computer scientists and programmers have been proposed several automatic solvers targeting those link puzzles. *Sugar* [8] is a SAT based constraint solver which is provided with formulation of Numberlink and Slitherlink. While it is often the case that a solution for an instance of Numberlink is found just by inspiration, many local solution methods for Slitherlink are known. Most Slitherlink solvers employ such methods [9–11].

Moreover, there are some attempts to automatically generate instances of Slitherlink. Shirai's algorithm [10] starts from a grid graph with no numbers, which admits many solutions, and repeatedly puts a random number into a random cell one by one until the obtained instance admits a unique solution. The opposite approached has been proposed by Shirai *et al*. [10] and Wan [11]. Their algorithm takes a cycle, which is a predetermined solution, as input and starts from a grid whose cells are all fulfilled by the numbers compatible with the solution. The algorithm then repeatedly removes the number in a random cell one by one until the obtained instance admits an unexpected solution. The algorithm outputs the instance obtained just before removing the last number.

Our contribution is located in this line of research. We propose solution and instance enumeration algorithms for Numberlink and Slitherlink. Unlike existing algorithms, ours output all the qualified solutions/instances at once in the form of a *zero-suppressed binary decision diagram (ZDD)*. A ZDD is a compressed data structure for representing and manipulating families of sets [12]. Recently, Knuth [13, exercise 225] has proposed an algorithm, called SIMPATH, that constructs a ZDD representing all $s$-$t$ paths on a given graph and two vertices $s$ and $t$, where a path is identified with the set of edges constituting the path. If an instance of Numberlink has only one pair of numbers in a grid, SIMPATH gives all solutions to it. Knuth has also proposed a modification of SIMPATH that enumerates all single cycles on an input graph in the form of a ZDD. Recall that to be a cycle is a necessary condition to be a solution for Slitherlink. Our solvers for the link puzzles construct a ZDD representing solutions in the manner of SIMPATH with taking the constraints from the problem instance into account. By the nature of enumeration, one can immediately decide whether an instance admits exactly one solution.

ZDDs can represent families of sets in a small space, yet an even more important and widely appreciated virtue of ZDDs is that one can quite efficiently perform fundamental mathematical operations on families of sets over ZDDs, like union, intersection, *etc*. Our algorithms for generating instances of the link puzzles apply the virtue of ZDDs. Our method based on ZDDs offers users flexible means to design puzzle instances. For example, puzzle designers can extract some specific kind of instances from the whole which they think "interesting" by those set operations over ZDDs.

This paper is organized as follows. Formal definitions of ZDDs, Numberlink and Slitherlink are given in Section 2. Section 3 introduces a variant of SIMPATH that enumerates all path matchings over a graph, on which other algorithms of ours are based. Our Numberlink solver is presented in Section 4 and Numberlink instance enumeration algorithm is proposed in Section 5. On the other hand, our Slitherlink solver and instance enumeration algorithm are presented in Sections 6 and 7, respectively. We also show experimental results on these algorithms in Section 8, where we compare the performance of known solving algorithms and ours. Moreover, we demonstrate how our method may help puzzle designers through presenting visually attractive instances of Slitherlink obtained by our algorithm. We then conclude the paper in Section 9.

## 2. Preliminaries

### *2.1. Zero-Suppressed Binary Decision Diagrams*

A *zero-suppressed binary decision diagram*, ZDD for short, represents a family of sets over a universal set $E$ whose elements are linearly ordered [12]. We index the elements of $E$ as $e_1, \dots, e_{|E|}$ where we write $e_i \leq e_j$ if and only if $i \leq j$. A ZDD is defined to be a labeled directed acyclic graph that satisfies the following properties.

- There is only one node with indegree 0, called the *root node*.
- There are just two nodes **0** and **1** with outdegree 0, called the *0-terminal* and *1-terminal*.
- Each node except terminals has just 2 outgoing arcs, which are labeled by $0$ and $1$ and called the *0-arc* and *1-arc*, respectively. We call the node pointed by the $j$-arc of a node $n$ the $j$-*child* of $n$.
- Each node $n$ except for the terminals is labeled by an element of $E$.
- The label of a non-terminal node is strictly smaller than those of its children.

If a node $n$ is the $j$-child of another node $n'$, we call $n'$ a $j$-*parent* of $n$ for $j \in \{0, 1\}$. We say that a path from the root node of a ZDD $Z$ is *valid* if it ends in a node which is not **0**. To each valid path $\pi$ we assign a set $Z(\pi) \subseteq E$ by

$$Z(\pi) = \{\, e_j \mid \text{the 1-arc of a node labeled with } e_j \text{ appears in } \pi \,\}$$

By representing a path from the root node of a ZDD by a sequence of $0$ and $1$, which we call a $0, 1$-*sequence*, we have an alternative inductive definition of the set:

$$\begin{aligned} Z(\pi \cdot 0) &= Z(\pi) \\ Z(\pi \cdot 1) &= Z(\pi) \cup \{e\} \end{aligned}$$

where $e$ is the label of the node in which $\pi$ ends. Each node $n \neq \mathbf{0}$ is assigned a set $Z(n) \subseteq 2^E$ defined by

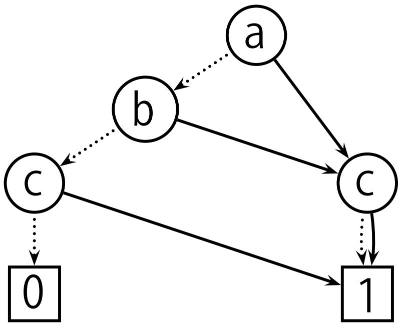
$$Z(n) = \{\, Z(\pi) \mid \text{a valid path } \pi \text{ ends in } n \,\}$$

An equivalent inductive definition is given by

$$\begin{aligned} Z(n) = {} & \{\, S \mid S \in Z(n') \text{ for some 0-parent } n' \text{ of } n \,\} \\ & \cup \{\, S \cup \{e\} \mid S \in Z(n') \text{ for some 1-parent } n' \text{ of } n \text{ such that } n' \text{ is labeled with } e \,\} \end{aligned}$$

The family of sets that a ZDD represents is defined to be $Z(\mathbf{1})$. Figure 3 shows an example of a ZDD.

**Figure 3.** The reduced ZDD representing $Z(\mathbf{1}) = \{\{\texttt{a}\}, \{\texttt{b}\}, \{\texttt{c}\}, \{\texttt{a}, \texttt{c}\}, \{\texttt{b}, \texttt{c}\}\}$. $E = \{\texttt{a}, \texttt{b}, \texttt{c}\}$ where $\texttt{a} < \texttt{b} < \texttt{c}$. Dotted lines represent 0-arcs and solid lines represent 1-arcs.



A ZDD is said to be *reduced* if

- there is no distinct nodes that have the same label, 0-child and 1-child,
- there is no node whose 1-child is $\mathbf{0}$.

It is known that every ZDD admits a unique reduced one that represents the same family of sets. Moreover the number of nodes of a ZDD is minimum amongst ZDDs representing the same family if and only if it is reduced. A linear-time algorithm that reduces a ZDD can be found in [13, pp. 84–85, algorithm R].

A (reduced) ZDD can represent a family of sets in a compact way by letting multiple sets in the family share some subgraphs of the ZDD. If sets in the family have some "regular" structures, the size of the ZDD can be logarithmically smaller than the sum of the cardinalities of the sets in it. An even more important and widely appreciated virtue of ZDDs is that one can efficiently perform fundamental mathematical operations on families of sets over ZDDs, like union, intersection, *etc*. The efficiency depends on the size of the ZDDs rather than the cardinality of the sets in the target families. Some packages of implementation of ZDDs are provided to users of different purposes, where for given two ZDDs representing families of sets $A$ and $B$, it can compute the ZDD for $A \cup B$, $A \cap B$, $A \setminus B$, $A \oplus B$ and so on. For $e \in E$, it can compute the ZDD for $\{ S \cup \{e\} \mid S \in A \}$, $\{ S \setminus \{e\} \mid S \in A \text{ and } e \in S \}$, $\{ S \mid S \in A \text{ and } e \notin S \}$ *etc*. Knuth [13] has discussed further operations for families of sets, which enable us to extract maximal elements $\{ S \in A \mid \neg\exists T \in A,\ S \subsetneq T \}$, minimal elements $\{ S \in A \mid \neg\exists T \in A,\ T \subsetneq S \}$ among others. For more details of ZDDs, readers are referred to [12,13], for example.

### 2.2. *Undirected Graphs*

The link puzzles concerned in this paper are played on undirected graphs. This paper uses ZDDs to represent subgraphs of graphs of puzzle instances. To avoid confusion among objects of those two kinds of graphs simultaneously in concern, we reserve words "nodes" and "arcs" for ZDDs, while we call the corresponding notions for graphs of puzzle instances "vertices" and "edges".

An *undirected graph* is a pair $G = (V, E)$ where $V$ is a set of *vertices* and $E \subseteq 2^V$ is a set of *edges*, which are sets of exactly two elements of $V$. An edge $e \in E$ is said to be *incident to* $v \in V$ if $v \in e$. The *degree* of a vertex $v$ in $G$ is defined by $|\{ e \in E \mid v \in e \}|$, that is, the number of edges incident to $v$. For an edge set $E' \subseteq E$, we also define *the degree of* $v$ *in* $E'$ by $|\{ e \in E' \mid v \in e \}|$. A vertex $v$ is *isolated* if its degree is $0$.

A nonempty edge set $P = \{e_1, \dots, e_{|P|}\} \subseteq E$ is called a *simple path* (or just a *path*) if there are pairwise distinct vertices $v_0, \dots, v_{|P|} \in V$ such that $e_i = \{v_{i-1}, v_i\}$ for $i = 1, \dots, |P|$. The path $P$ is also called a $v_0$-$v_{|P|}$ *path*.

A nonempty edge set $C = \{e_1, \dots, e_{|C|}\} \subseteq E$ is called a *(simple) cycle* if $|C| \geq 3$ and there are pairwise distinct vertices $v_1, \dots, v_{|C|} \in V$ such that $e_1 = \{v_1, v_{|C|}\}$ and $e_i = \{v_{i-1}, v_i\}$ for $i = 2, \dots, |C|$.

A possibly empty edge set $P \subseteq E$ is called a *path matching* if there are simple paths $P_1, \dots, P_k$ with $k \geq 0$ such that $P = \bigcup_{1 \leq i \leq k} P_i$ and $\bigcup P_i \cap \bigcup P_j = \emptyset$ for distinct $i, j \in \{1, \dots, k\}$ (For a family $P$ of sets over a universal set $V$, we define $\bigcup P = \bigcup_{e \in P} e = \{ v \in V \mid v \in e \text{ for some } e \in P \}$.). We remark that for a path matching $P$, a way of partitioning $P$ into paths satisfying this definition is unique. We say that $P$ *contains* a $u$-$v$ path if one of such paths is a $u$-$v$ path.

A family $T \subseteq 2^V$ of sets over $V$ is called a *pair matching* if $T$ consists of pairwise disjoint sets of size 2.

**Lemma 1.** *For a graph $G = (V, E)$, an edge set $P \subseteq E$ is not a path matching if and only if there is a vertex $v \in \bigcup P$ of degree more than 2 in $P$ or there is a subset $C \subseteq P$ that is a simple cycle.*

### *2.3. Link Puzzles*

**Definition 2** (Numberlink)**.** An *instance of the Numberlink problem* is a pair of a graph (Usually the graph $G$ in an instance is a rectangular grid as described in the introduction, but this paper targets the more general problem described here.) $G = (V, E)$ and a pair matching $h$ on $V$. A path matching on $G$ is said to be a *solution* of $(G, h)$ if and only if it contains $u$-$v$ paths for all $\{u, v\} \in h$ but no other.

An instance $(G, h)$ of the Numberlink problem is said to be *good* if $(G, h)$ admits exactly one solution $P$ and moreover the unique solution $P$ covers every vertex: *i.e.*, $\bigcup P = V$.

**Theorem 3.** *[2,3,5] For input $G$ and $h$, deciding whether $(G, h)$ admits a solution is NP-complete.*

**Definition 4** (Slitherlink)**.** An *instance of the Slitherlink problem* is a pair of a graph $G = (V, E)$ and a partial mapping $h : 2^E \rightharpoonup \mathbb{N}$ such that $h(E') \leq |E'|$ for all $E' \in \operatorname{dom}(h)$, where $\operatorname{dom}(h)$ denotes the domain of $h$ (Usually $G$ is a grid graph and $h(E')$ is defined only when $E'$ is a cycle consisting of just four edges.). We call $h$ a *hint assignment*. A *solution* of $(G, h)$ is a cycle $C$ over $G$ such that $h(E') = |C \cap E'|$ for all $E' \in \operatorname{dom}(h)$. An instance $(G, h)$ of the Slitherlink problem is said to be *good* if $(G, h)$ admits exactly one solution.

**Theorem 5.** *[14] For input $G$ and $h$, deciding whether $(G, h)$ admits a solution is NP-complete. For input $(G, h)$ and a solution $C$, deciding whether $(G, h)$ admits another solution $C' \neq C$ is NP-complete.*

In this paper, we assume that $G$ is connected on both puzzles.

## 3. ZDD for Path Matchings

Knuth has proposed an algorithm that constructs a ZDD representing all the $s$-$t$ paths on a given graph $G$ and two vertices $s$ and $t$ in $G$, as an answer to Exercise 225 in [13]. Based on Knuth's algorithm, this section presents an algorithm that constructs in a topdown manner a ZDD $Z_G$ representing all the path matchings over a graph $G = (V, E)$. This algorithm prepares for our puzzle solvers.

Let $G = (V, E)$ be an undirected graph. The edge set $E = \{e_1, \ldots, e_{|E|}\}$ is the universal set of $Z_G$, where we have $e_i \leq e_j$ if and only if $i \leq j$. We define $E^{\geq e_i} = \{\, e_j \in E \mid e_j \geq e_i \,\}$. The root node $n_{\mathrm{root}}$ of $Z_G$ is labeled with $e_1$ and if a node is labeled with $e_i$ then its children are labeled with $e_{i+1}$ unless they are terminal nodes, **0** or **1**. By $N_i$ we denote the set of nodes with the label $e_i$ for $i = 1, \ldots, |E|$. Hence $N_1$ is the singleton of the root node $n_{\mathrm{root}}$ and $N_{i+1}$ consists of the children of nodes in $N_i$. Every valid path ending in **1** has length $|E|$ in $Z_G$. For a $0, 1$-sequence $\pi$, we denote the length of $\pi$ by $|\pi|$. For $\pi$ with $|\pi| \leq |E|$, we define

$$E(\pi) = \{\, e_i \in E \mid \text{the } i\text{th number of } \pi \text{ is } 1 \,\}$$

We then have $Z_G(\pi) = E(\pi)$ if $\pi$ is a valid path in $Z_G$. For a valid path $\pi$, we let $n(\pi)$ denote the node in which $\pi$ ends. We have $Z_G(n) = \{\, E(\pi) \mid n(\pi) = n \,\}$. The ZDD $Z_G$ will be constructed so that

$$E(\pi) \subseteq E \text{ is a path matching if and only if } \pi \text{ is valid in } Z_G \tag{1}$$

Note that every subgraph of a path matching is a path matching.

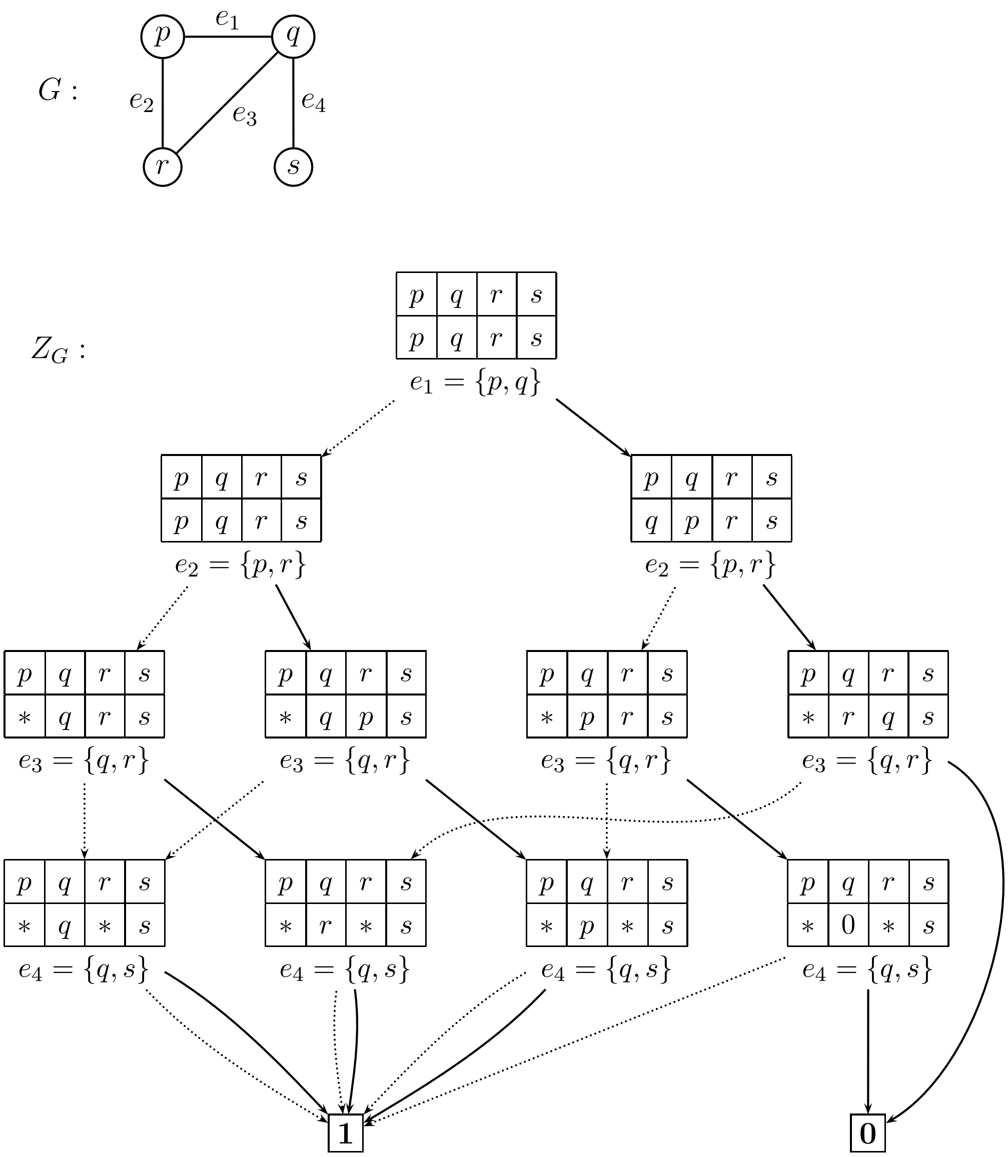
Our algorithm constructs $Z_G$ in a top-down manner. The construction consists of $|E|$ phases, where we create nodes in $N_{i+1}$ as children of nodes in $N_i$ in the $i$th phase. Figure 4 illustrates an example of $G$ and $Z_G$, which would help the reader understand our algorithm. We first initialize $N_i$ to be empty for all $i = 2, \ldots, |E|$ and $N_1$ to be the singleton of the root node $n_{\mathrm{root}}$. We create nodes in $N_i$ after the upper part of $Z_G$ consisting of the nodes in $\bigcup_{j<i} N_j$ and the arcs among them have been constructed for $i = 2, \ldots, |E|$.

The algorithm stores some topological information of the path matching $E(\pi)$ at the node $n = n(\pi)$ reached by a valid path $\pi$. Let $|\pi| = i - 1$, where we have $n \in N_i$ and $n$ is labeled by $e_i$ unless $n = \mathbf{1}$. We assign the node $n$ a function $\mathrm{mate}_n : V^{\geq e_i} \to V \cup \{0\}$, which we call a *mate function*, where $V^{\geq e_i} = \bigcup E^{\geq e_i}$ and $0 \notin V$. It is determined so that

$$\mathrm{mate}_n(v) = \begin{cases} v & \text{if } v \notin \bigcup E(\pi) \\ u & \text{if } E(\pi) \text{ contains a } u\text{-}v \text{ path} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

for each $v \in V^{\geq e_i}$. In other words, $\mathrm{mate}_n(v) = v$ if $v$ has degree 0 in $E(\pi)$, and $\mathrm{mate}_n(v) = 0$ if $v$ has degree 2 in $E(\pi)$. Particularly for the root node $n_{\mathrm{root}}$, we have $\mathrm{mate}_{n_{\mathrm{root}}}(v) = v$ for all $v \in V^{\geq e_1}$; the set $V^{\geq e_1}$ is identical to $V$, since $G$ has no isolated vertices. The domain of a mate function $\mathrm{mate}_n$ is restricted to $V^{\geq e_i}$, which is the set of vertices to which at least one edge in $E^{\geq e_i}$ is incident. We are not interested in the topological information on vertices that we will not visit any more. Note that $V^{\geq e_i} = V^{\geq e_{i+1}} \cup e_i$. Hereafter we stipulate that $V^{\geq e_{|E|+1}}$ denotes the empty set.

**Figure 4.** An example of $G$ and $Z_G$.



The table of each node $n$ represents $\mathrm{mate}_n$. For example, $\begin{array}{|c|c|c|c|}\hline p & q & r & s \\ \hline * & 0 & * & s \\ \hline \end{array}$ means that $\mathrm{mate}_n(q) = 0$, $\mathrm{mate}_n(s) = s$ and $p, r$ are not in the domain.

In the example of Figure 4, we have $V = V^{\geq e_1} = V^{\geq e_2} = \{p, q, r, s\}$, $V^{\geq e_3} = e_3 \cup e_4 = \{q, r, s\}$ and $V^{\geq e_4} = e_4 = \{q, s\}$. For $\pi = 1 \cdot 1$, $E(\pi) = \{e_1, e_2\}$ is a path matching consisting solely of a $q$-$r$ path. The domain of the mate function $\mathrm{mate}_{n(\pi)}$ is $V^{\geq e_3} = \{q, r, s\}$ and $\mathrm{mate}_{n(\pi)}(q) = r$, $\mathrm{mate}_{n(\pi)}(r) = q$ and $\mathrm{mate}_{n(\pi)}(s) = s$.

Let us consider the $i$th phase of our algorithm, where we already have constructed an incomplete ZDD until the nodes in $N_i$. We determine the children of a node $n(\pi) \in N_i$ for a valid path $\pi$. The arcs given in this phase correspond to the choice whether or not we pick $e_i$ in a path matching. Let $n = n(\pi)$.

Giving a 0-child to $n$ involves no difficulty. By assumption $E(\pi)$ is a path matching and by definition $E(\pi \cdot 0) = E(\pi)$. We create a node $n'$ in $N_{i+1}$ as the 0-child of $n$ and label $n'$ by $e_{i+1}$. That is, $n(\pi \cdot 0) = n'$. The mate function $\mathrm{mate}_{n'}$ of $n'$ should be identical to $\mathrm{mate}_n$ except that the domain

$\mathrm{dom}(\mathrm{mate}_{n'}) = V^{\geq e_{i+1}}$ may be smaller than $\mathrm{dom}(\mathrm{mate}_n) = V^{\geq e_i}$. In the case where $i = |E|$, the 0-child of $n$ is **1**.

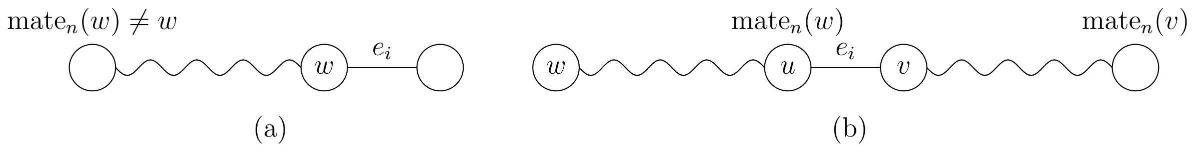
On the other hand, $E(\pi \cdot 1) \neq E(\pi)$ and it is not always the case that $E(\pi \cdot 1)$ is a path matching. If $E(\pi \cdot 1)$ is not a path matching, the 1-child of $n$ should be **0**. Otherwise, we give a node $n''$ with an appropriate mate function as the 1-child of $n$. We must decide whether $E(\pi) \cup \{e_i\}$ is still a path matching or not. Let $e_i = \{u, v\}$. We note that $u$ and $v$ are in the domain $V^{\geq e_i}$ of $\mathrm{mate}_n$. There exist several cases depending on the topology of $E(\pi)$. Recall that an edge set $P \subseteq E$ is a path matching if and only if the degree of every node in $P$ is at most 2 and no subset $P' \subseteq P$ is a cycle. In the case where $\mathrm{mate}_n(u) = 0$, the degree of $u$ in $E(\pi)$ is 2, so the degree of $u$ will be 3 in $E(\pi) \cup \{e_i\}$, which is therefore not a path matching. The same holds when $\mathrm{mate}_n(v) = 0$. Moreover, if $\mathrm{mate}_n(u) = v$ (equivalently $\mathrm{mate}_n(v) = u$), which means that $E(\pi)$ contains a $u$-$v$ path $P_{u,v} \subseteq E(\pi)$, then $E(\pi) \cup \{e_i\}$ contains a simple cycle $P_{u,v} \cup \{e_i\}$. We say that a mate function $m$ *rejects* an edge $\{u, v\}$ if $m(u) \in \{0, v\}$ or $m(v) \in \{0, u\}$. If $\mathrm{mate}_n$ rejects $e_i$, then $E(\pi \cdot 1)$ is not a path matching, hence the 1-child of $n$ should be **0**. On the other hand, if $\mathrm{mate}_n$ does not reject $e_i$, $E(\pi) \cup \{e_i\}$ is a path matching. So we need to give a node as the 1-child of $n$. Particularly in the case where $i = |E|$, the 1-child of $n$ will be **1**. If $i < |E|$, we create a node $n''$ in $N_{i+1}$ as the 1-child of $n$ (*i.e.*, $n'' = n(\pi \cdot 1)$) and assign a function $\mathrm{mate}_{n''}$ that is consistent with the requirement of Equation (2). Actually one can compute $\mathrm{mate}_{n''}$ from $\mathrm{mate}_n$ and $e_i = \{u, v\}$ by

$$\mathrm{mate}_{n''}(w) = \begin{cases} 0 & \text{if } w \in e_i = \{u, v\} \text{ and } \mathrm{mate}_n(w) \neq w \\ \mathrm{mate}_n(v) & \text{if } \mathrm{mate}_n(w) = u \\ \mathrm{mate}_n(u) & \text{if } \mathrm{mate}_n(w) = v \\ \mathrm{mate}_n(w) & \text{otherwise} \end{cases} \tag{3}$$

for $w \in \mathrm{dom}(\mathrm{mate}_{n''}) = V^{\geq e_{i+1}}$.

We explain why this is correct. The first case means that $w$ has degree 1 in $E(\pi)$. Hence, the degree of $w \in e_i$ will be 2 in $E(\pi) \cup \{e_i\}$ (Figure 5(a)). In the second case, $\mathrm{mate}_n(w) = u$ means either that $E(\pi)$ has a $w$-$u$ path or that $w = u$ and $w$ is isolated in $E(\pi)$. It is convenient for explanation to regard an isolated vertex $t$ as a $t$-$t$ path. According to this terminology, $E(\pi)$ contains a $w$-$u$ path and a $v$-$\mathrm{mate}_n(v)$ path. Then the edge $e_i = \{u, v\}$ will bridge those paths: $E(\pi) \cup \{e_i\}$ contains a $w$-$\mathrm{mate}_n(v)$ path (Figure 5(b)). The third case is similar to the second case. If none of those three cases apply, adding $e_i$ does not matter for the vertex $w$. We note that by the assumption, the first three cases are mutually exclusive.

**Figure 5.** updating mate.



Summarizing the above, by observing $\mathrm{mate}_n$, one can determine whether $E(\pi \cdot 1)$ is a path matching or not. Moreover one can compute both $\mathrm{mate}_{n'}$ and $\mathrm{mate}_{n''}$ for the 0-child $n'$ and 1-child $n''$ of $n$, respectively, unless they are **0** or **1**. Consequently, if two nodes $n_1$ and $n_2$ with the same label have the same mate function, their descendants will constitute subgraphs of exactly the same shape in $Z_G$. This means that such equivalent nodes $n_1$ and $n_2$ should be merged before generating children.

Our actual algorithm, shown as Algorithm 1, maintains each node set $N_i$ not to contain distinct nodes assigned the same mate function. This is realized by a procedure named GN (short for "getting a node") which takes two arguments: an edge index $i \in \{1, \dots, |E|\}$ and a mate function $m$. $\mathrm{GN}(i, m)$ returns a node labeled $e_i$ and assigned $m$. If such a node has already been created in $N_i$, that node is returned. Otherwise, we create one in $N_i$ and return it. It is conveniently assumed that $\mathrm{GN}(|E| + 1, m)$ gives the terminal node **1**. Note that this procedure may update $N_i$.

**Algorithm 1:** Enumeration of path matchings

```
Data: A graph $G = (V, E)$
begin
    create a root node and two terminal nodes 0 and 1;
    let $N_1 \leftarrow \{n_{\mathrm{root}}\}$, $N_i \leftarrow \emptyset$ for $i = 2, \dots, |E|$ and $\mathrm{mate}_{n_{\mathrm{root}}}$ be the identity on $V$;
    foreach $i = 1, \dots, |E|$ do
        foreach $n \in N_i$ do
            let the 0-child of $n$ be $\mathrm{GN}(i + 1, \mathrm{mate}_n|_{V^{\geq e_{i+1}}})$;
            if $\mathrm{mate}_n$ rejects $e_i$ then let the 1-child of $n$ be 0;
            else let the 1-child of $n$ be $\mathrm{GN}(i + 1, \mathrm{MU}(\mathrm{mate}_n, e_i)|_{V^{\geq e_{i+1}}})$;
        end
    end
    return the constructed diagram;
end
```

For a node labeled $e_i$ and assigned a mate function $m$ which does not reject $e_i$, the mate function of its 1-child is given by $\mathrm{MU}(m, e_i)|_{V^{\geq e_{i+1}}}$ in Algorithm 1, which is determined in the way described in Equation (3). That is, MU (short for "mate update") takes a mate function $m$ and an edge $e_i$ and returns a mate function defined by

$$\mathrm{MU}(m, e_i)(w) = \begin{cases} 0 & \text{if } w \in e_i \text{ and } m(w) \neq w \\ m(v) & \text{if } e_i = \{v, m(w)\} \\ m(w) & \text{otherwise} \end{cases}$$

for each $w \in \mathrm{dom}(m)$. The domain of $\mathrm{MU}(m, e_i)$ is the same as that of $m$. By $\mathrm{MU}(m, e_i)|_{V^{\geq e_{i+1}}}$ we denote the subfunction of $\mathrm{MU}(m, e_i)$ whose domain is restricted to $V^{\geq e_{i+1}}$.

We prove the correctness of Algorithm 1 in Appendix A.

## 4. Numberlink Solver

This section presents a Numberlink solver, which constructs a ZDD that represents all the solutions for a given instance $(G, h)$ of Numberlink. Recall that a solution for an instance $(G, h)$ of Numberlink is a path matching that satisfies a property represented by the pair matching $h$. It is in fact rather easy to modify Algorithm 1 to obtain a Numberlink solver.

Let us consider when we should connect an arc of a node to **0** in addition to the cases we have discussed in Section 3. Suppose that we have chosen edges from $\{e_1, \dots, e_i\}$ and by picking up other edges from $E^{\geq e_{i+1}}$, we would like to form a solution. Let $\pi$ be a $0, 1$-sequence of length $i$ that represents the choice. It is easy to see that if one of the following conditions holds, $E(\pi) \cup P$ cannot be a solution of $(G, h)$ for any edge set $P \subseteq E^{\geq e_{i+1}}$.

- $E(\pi)$ is not a path matching,
- there is $v \in \bigcup h$ of degree 2 in $E(\pi)$,
- there is $v \in \bigcup h \setminus V^{\geq e_{i+1}}$ of degree 0 in $E(\pi)$,
- there are $u, v \in \bigcup h$ such that $E(\pi)$ contains a $u$-$v$ path and $\{u, v\} \notin h$,
- there is a vertex $v \notin \bigcup h \cup V^{\geq e_{i+1}}$ of degree 1 in $E(\pi)$.

Our Numberlink solver, shown as Algorithm 2, constructs a ZDD so that if a path $\pi$ of length $i$ satisfies one of the above, then (a prefix of) $\pi$ leads to **0**. While Algorithm 1 connects no 0-arcs to **0**, our Numberlink solver sometimes connects 0-arcs to **0** according to the above conditions. This judgement is realized by observing the mate functions. For a mate function $m : V^{\geq e_i} \to V \cup \{0\}$ and an edge index $i$, we say that $(m, i, 0)$ is *incompatible with* $h$ if for some $v \in e_i \setminus V^{\geq e_{i+1}}$, one of the following conditions holds.

- $v \in \bigcup h$ and $m(v) = v$,
- $v \notin \bigcup h$ and $m(v) \notin \{0, v\}$.

When $(\mathrm{mate}_n, i, 0)$ is incompatible with $h$ for $n \in N_i$, the 0-arc of $n$ should be connected to **0**, since $P = E(\pi \cdot 0) \cup E'$ cannot be a solution for any $E' \subseteq E^{\geq e_{i+1}}$. In the first case, the degree of $v \in \bigcup h$ is 0 in $P$. In the second case, the degree of $v \notin \bigcup h$ is 1 in $P$.

**Algorithm 2:** Numberlink solver

```
Data: A graph $G = (V, E)$ and a pair matching $h$ over $V$
begin
    create a root node and two terminal nodes 0 and 1;
    let $N_1 \leftarrow \{n_{\mathrm{root}}\}$ and $N_i \leftarrow \emptyset$ for $i = 2, \dots, |E|$;
    foreach $i = 1, \dots, |E|$ do
        foreach $n \in N_i$ do
            if $(\mathrm{mate}_n, i, 0)$ is incompatible with $h$ then let the 0-child of $n$ be 0;
            else let the 0-child of $n$ be $\mathrm{GN}(i + 1, \mathrm{mate}_n|_{V^{\geq e_{i+1}}})$;
            if $(\mathrm{mate}_n, i, 1)$ is incompatible with $h$ then let the 1-child of $n$ be 0;
            else let the 1-child of $n$ be $\mathrm{GN}(i + 1, \mathrm{MU}(\mathrm{mate}_n, e_i)|_{V^{\geq e_{i+1}}})$;
        end
    end
    return the constructed diagram;
end
```

We prove the correctness of Algorithm 2 in Appendix B.

We say that $(m, i, 1)$ is *incompatible with* $h$ if for $e_i = \{u, v\}$, one of the following conditions holds.

- $m$ rejects $e_i$, *i.e.*, $m(v) \in \{0, u\}$,
- $v \in \bigcup h$ and $m(v) \neq v$,
- $m(u), m(v) \in \bigcup h \cup (V \setminus V^{\geq e_{i+1}})$ and $\{m(u), m(v)\} \notin h$.

When $(\mathrm{mate}_n, i, 1)$ is incompatible with $h$ for $n \in N_i$, the 1-arc of $n$ should be connected to **0**, since $P = E(\pi \cdot 1) \cup E'$ cannot be a solution for any $E' \subseteq E^{\geq e_{i+1}}$. In the first case, $P$ is not a path matching. In the second case, $v$ has degree more than 1 in $P$. In the third case, either $P$ contains a $\mathrm{mate}_n(u)$-$\mathrm{mate}_n(v)$ path or the degree of $\mathrm{mate}_n(u) \in \bigcup h$ or $\mathrm{mate}_n(v) \in \bigcup h$ is more than 1 in $P$.

**Remark 6.** It is obvious that no proper superset of a solution for $(G, h)$ can be another solution. One can modify Algorithm 2 so that a path $\pi$ directly leads to **1** if $E(\pi)$ is a solution. Replace GN by $\mathrm{GN}_h$ where

$$\mathrm{GN}_h(i, m) = \begin{cases} \mathbf{1} & \text{if for any } u \in \mathrm{dom}(m) \text{ and } v \in V, \text{ we have } (\{u, v\} \in h \text{ iff } m(u) = v \neq u) \\ \mathrm{GN}(i, m) & \text{otherwise} \end{cases}$$

**Remark 7.** An alternative definition of Numberlink requires a solution to cover all the vertices. This constraint drastically shrinks the search space. It is easy to modify our algorithm to obtain only such strict solutions. We say that $(m, i, 0)$ is *incompatible with* $h$ if it is incompatible in the above sense or there is $v \in V^{\geq e_i} \setminus V^{\geq e_{i+1}}$ such that $m(v) = v$, which means that $v$ is not covered.

## 5. Numberlink Instance Enumeration

This section presents an algorithm that enumerates all pair matchings $h$ that give a good instance $(G, h)$ of the Numberlink problem for a given graph $G$. Fixing $G$, we say that a pair matching $h$ is *good* if $(G, h)$ is a good instance. Let us first consider an easier task, which enumerates all the instances that admit at least one solution. A solution for an instance of the Numberlink problem is a path matching and conversely every path matching $P$ on $G$ consisting of $u_i$-$v_i$ simple paths for all $i = 1, \ldots, k$ is a solution for $(G, h_P)$ where $h_P$ consists of $\{u_i, v_i\}$ for all $i = 1, \ldots, k$. Our instance enumeration algorithm, shown as Algorithm 3, is again based on Algorithm 1, but now what we should enumerate is not edge sets but rather pair matchings. The ZDD $Z_G$ output by Algorithm 1 will be a part of the input of Algorithm 3.

For a valid path $\pi$ in $Z_G$, the corresponding path matching $E(\pi)$ consists of two kinds of simple paths: fixed paths and unfixed paths. *Fixed paths* are $u$-$v$ paths where $u, v \notin V^{\geq e_{|\pi|+1}} = \mathrm{dom}(\mathrm{mate}_{n(\pi)})$. In other words, every fixed path in $E(\pi)$ will be a constituent of any path matching represented as $E(\pi \cdot \rho)$ (We note that whether a simple path in a path matching $E(\pi)$ is fixed or unfixed depends on $\pi$ rather than $E(\pi)$ itself. Yet we prefer a concise phrasing relying on the reader's flexibility and assume that the vertex set in concern is understood.). We note that by definition $\mathrm{mate}_{n(\pi)}$ does not tell us whether $E(\pi)$ contains a fixed $u$-$v$ path. On the other hand, the presence of an *unfixed* $u$-$v$ *path* is observed in the mate function. If a $u$-$v$ path is unfixed, at least one of $u, v$ is in $V^{\geq e_{|\pi|+1}} = \mathrm{dom}(\mathrm{mate}_{n(\pi)})$. Without loss of generality, we assume $u \in \mathrm{dom}(\mathrm{mate}_{n(\pi)})$, where we have $\mathrm{mate}_{n(\pi)}(u) = v$. Since $u$ has an incident edge in $E^{\geq e_{|\pi|+1}}$, the path may be extended so that $u$ has degree 2.

**Algorithm 3:** Numberlink Generator

```
Data: A graph $G = (V, E)$
begin
    let $Z_G$ be the output by Algorithm 1 for $G$;
    let $S_{n(\epsilon)} \leftarrow \{\emptyset\}$ and $B_{n(\epsilon)} \leftarrow \emptyset$ for the root $n(\epsilon)$ of $Z_G$;
    let $N_{|E|+1} \leftarrow \{\mathbf{1}\}$;
    foreach $i = 2, \ldots, |E|, |E| + 1$ do
        foreach $n \in N_i$ do
            let $S_n \leftarrow \bigcup\{ S_{n',b} \mid n' \in N_{i-1}$ is a $b$-parent of $n \}$;
            let $B_n \leftarrow \bigcup\{ S_{n_1,b_1} \cap S_{n_2,b_2} \mid n_j \in N_{i-1}$ is a $b_j$-parent of $n$ for $j = 1, 2$ and $n_1 \neq n_2 \}$
                $\cup \bigcup\{ S_{n',0} \mid n'$ is a 0-parent of $n$ such that $\mathrm{mate}_{n'}(u) = u$ for some $u \in e_i \setminus V^{\geq e_{i+1}} \}$
                $\cup \bigcup\{ B_{n',b} \mid n'$ is a $b$-parent of $n \}$;
        end
    end
    return $S_{\mathbf{1}} \setminus B_{\mathbf{1}}$;
end
```

We prove the correctness of Algorithm 3 in Appendix C.

Let $T(\pi)$ denote the pair matching corresponding to the ends of fixed paths:

$$T(\pi) = \{ \{u, v\} \subseteq V \mid E(\pi) \text{ contains a } u\text{-}v \text{ path with } u, v \notin V^{\geq e_{|\pi|+1}} \}$$

Note that $T(\pi) \subseteq T(\pi \cdot \rho)$ for any $\pi$ and $\rho$ as long as $E(\pi \cdot \rho)$ is a path matching. We decorate each node $n$ of the ZDD $Z_G$ with the family

$$S_n = \{ T(\pi) \mid n(\pi) = n \}$$

When a path $\pi$ ends in $\mathbf{1}$ in $Z_G$, all the simple paths constituting $E(\pi)$ are fixed. By definition, $S_{\mathbf{1}}$ will be the family of pair matchings $h$ such that $(G, h)$ admits a solution. For different valid paths $\pi$ and $\rho$ ending in the same node $n = n(\pi) = n(\rho)$, $E(\pi)$ and $E(\rho)$ may have different fixed paths, but they share the same unfixed paths. We present an example of $T(\pi)$ and $S_n$ in Appendix F.

Let us think about how to compute the family $S_n$ of a node $n$ from the pair matching families $S_{n'}$ assigned to its parents $n'$. By definition, it is not hard to see that

$$T(\pi \cdot 0) \setminus T(\pi) = \{ \{u, v\} \subseteq V \mid E(\pi \cdot 0) \text{ contains a } u\text{-}v \text{ path with } u \in e_i \text{ and } u, v \notin V^{\geq e_{i+1}} \}$$
$$T(\pi \cdot 1) \setminus T(\pi) = \{ \{u, v\} \subseteq V \mid E(\pi \cdot 1) \text{ contains a } u\text{-}v \text{ path } P \text{ with } e_i \in P \text{ and } u, v \notin V^{\geq e_{i+1}} \}$$

where $i = |\pi| + 1$. Define

$$\mathrm{Fix}(m, i, 0) = \{ \{u, m(u)\} \mid u \in e_i,\ m(u) \notin \{u, 0\} \text{ and } u, m(u) \notin V^{\geq e_{i+1}} \}$$
$$\mathrm{Fix}(m, i, 1) = \{ \{m(u), m(v)\} \mid \{u, v\} = e_i \text{ and } m(u), m(v) \notin V^{\geq e_{i+1}} \}$$

We now have

$$T(\pi \cdot b) = T(\pi) \cup \mathrm{Fix}(\mathrm{mate}_{n(\pi)}, |\pi| + 1, b)$$

for both $b = 0, 1$. Therefore, for $n \in N_{i+1}$,

$$S_n = \bigcup \{\, S_{n',b} \mid n' \text{ is a } b\text{-parent of } n \text{ for } b \in \{0,1\} \,\}$$
$$\text{where } S_{n',b} = \{\, T \cup \mathrm{Fix}(\mathrm{mate}_{n'}, i, b) \mid T \in S_{n'} \,\}$$

In this way, one can recursively compute $S_n$ for all nodes except $\mathbf{0}$ of $Z_G$.

We would like to extract good ones from $S_{\mathbf{1}}$. In addition to $S_n$, we assign each node $n$ another family $B_n \subseteq S_n$ such that no pair matching $T \in B_n$ will be expanded to a good one. We construct $B_n$ so that

$$B_n = \{\, T(\pi_1) \in S_n \mid \text{there are distinct valid paths } \pi_1, \pi_2 \text{ ending in } n \text{ with } T(\pi_1) = T(\pi_2) \,\} \tag{4}$$
$$\cup \{\, T(\pi) \in S_n \mid \text{there is a valid path } \pi \text{ ending in } n \text{ with } \bigcup E(\pi) \cup V^{\geq e_{|\pi|+1}} \neq V \,\} \tag{5}$$

This definition is explained as follows. For the first line Equation (4), suppose that there are distinct valid paths $\pi_1$ and $\pi_2$ ending in the same node $n = n(\pi_1) = n(\pi_2)$ such that $T(\pi_1) = T(\pi_2)$. Then, $E(\pi_1 \cdot \rho)$ is a path matching if and only if so is $E(\pi_2 \cdot \rho)$ for any $\rho$. Moreover, $E(\pi_1 \cdot \rho)$ consists of $u_i$-$v_i$ paths for $i = 1, \dots, k$ for some $k$ if and only if so does $E(\pi_2 \cdot \rho)$. That is, the instance $(G, \{\, \{u_i, v_i\} \mid i = 1, \dots, k \,\})$ has two distinct solutions $E(\pi_1 \cdot \rho)$ and $E(\pi_2 \cdot \rho)$. The second part Equation (5) of the definition of $B_n$ is easier. If $E(\pi) \cup V^{\geq e_{|\pi|+1}} \neq V$, then there is a vertex $v \notin V \setminus V^{\geq e_{i+1}}$ which has degree 0 in $E(\pi)$ and thus in $E(\pi \cdot \rho)$ for any $\rho$. If $E(\pi \cdot \rho)$ is a path matching consisting of $u_i$-$v_i$ paths for $i = 1, \dots, k$ for some $k$, the instance $(G, \{\, \{u_i, v_i\} \mid i = 1, \dots, k \,\})$ has a solution that does not cover all the vertices.

We can compute the family $B_n$ of a node $n$ from the pair matching families assigned to its parents according to the definition of $B_n$:

$$B_n = \bigcup \{\, S_{n_1,b_1} \cap S_{n_2,b_2} \mid n_j \text{ is a } b_j\text{-parent of } n \text{ for } j = 1, 2 \text{ and } n_1 \neq n_2 \,\} \tag{6}$$
$$\cup \bigcup \{\, S_{n',0} \mid n' \text{ is a } 0\text{-parent of } n \text{ such that } \mathrm{mate}_{n'}(u) = u \text{ for some } u \in e_i \setminus V^{\geq e_{i+1}} \,\} \tag{7}$$
$$\cup \bigcup \{\, B_{n',b} \mid n' \text{ is a } b\text{-parent of } n \,\}, \tag{8}$$

where $B_{n,b} = \{\, T \cup \mathrm{Fix}(\mathrm{mate}_n, i, b) \mid T \in B_n \,\}$ for $b \in \{0, 1\}$. The lines Equations (6) and (7) come from Equations (4) and (5), respectively. Suppose that $T(\pi) \in B_{n(\pi)}$. Then regardless of whether it is due to Equation (4) or Equation (5), we have $T(\pi \cdot \rho) \in B_{n(\pi \cdot \rho)}$ for any $\rho$ as long as $\pi \cdot \rho$ is valid. Those pair matchings are captured by Equation (8).

## 6. Slitherlink Solver

The goal of this section is to give a Slitherlink solver, which constructs a ZDD that represents all the solutions for a given instance $(G, h)$ of Slitherlink. Recall that a solution is a simple cycle that satisfies a property represented by the hint assignment $h : 2^E \rightharpoonup \mathbb{N}$. As a preparation, we present an algorithm that constructs a ZDD representing all the simple cycles on a given graph $G$. Actually this algorithm is an answer to Exercise 226 of [13].

Based on the fact that any proper subgraph of a cycle is actually a path matching, it is easy to modify Algorithm 1 so that simple cycles shall be enumerated. We allow vertices to have degree 1 temporarily during the construction, but when we have determined which of the edges incident to a vertex $v$ shall be used, the degree of $v$ must be $0$ or $2$. In addition, we allow to add an edge $e = \{u, v\}$ to a path matching $E(\pi)$ if $E(\pi)$ consists solely of a single $u$-$v$ path: it completes a cycle.

Both conditions are judged by observing the mate functions. We say that a mate function $m$ and an edge $e_i$ *form a cycle* if for any $v$ in the domain of $m$,

$$m(v) = \begin{cases} u & \text{if } e_i = \{u, v\} \\ v \text{ or } 0 & \text{if } v \notin e_i \end{cases}$$

If $\mathrm{mate}_n$ and $e_i$ form a cycle for $n \in N_i$, the 1-child of $n$ should be $\mathbf{1}$.

We say that $(m, i)$ *has a fixed end* if

$$m(v) \notin \{0, v\} \text{ for some } v \in e_i \setminus V^{\geq e_{i+1}}$$

For a valid path $\pi$ in $Z_G$, if $(\mathrm{mate}_{n(\pi)}, |\pi|)$ has a fixed end, $E(\pi) \cup E'$ has a vertex of degree $1$ for any $E' \subseteq E^{\geq e_{i+1}}$. We say that $m$ *declines* $e_i$ if either $m$ rejects $e_i$ or $(\mathrm{MU}(m, e_i), i)$ has a fixed end. If $\mathrm{mate}_n$ declines $e_i$ for $n \in N_i$, the $1$-child of $n$ should be $\mathbf{0}$.

In addition, since the empty edge set is not a cycle, the corresponding path in our ZDD should end in $\mathbf{0}$.

Algorithm 4 constructs a ZDD for representing all the simple cycles on a graph $G$. We prove the correctness of Algorithm 4 in Appendix D.

**Algorithm 4:** Enumeration of simple cycles

```
Data: A graph $G = (V, E)$
begin
    create a root node and two terminal nodes $\mathbf{0}$ and $\mathbf{1}$;
    let $N_1 \leftarrow \{n_{\mathrm{root}}\}$ and $N_i \leftarrow \emptyset$ for $i = 2, \ldots, |E|$;
    foreach $i = 1, \ldots, |E|$ do
        foreach $n \in N_i$ do
            if $(\mathrm{mate}_n, i)$ has a fixed end or $i = |E|$ then let the $0$-child of $n$ be $\mathbf{0}$;
            else let the $0$-child of $n$ be $\mathrm{GN}(i+1, \mathrm{mate}_n|_{V^{\geq e_{i+1}}})$;
            if $\mathrm{mate}_n$ and $e_i$ form a cycle then let the $1$-child of $n$ be $\mathbf{1}$;
            else if $\mathrm{mate}_n$ declines $e_i$ then let the $1$-child of $n$ be $\mathbf{0}$;
            else let the $1$-child of $n$ be $\mathrm{GN}(i+1, \mathrm{MU}(\mathrm{mate}_n, e_i)|_{V^{\geq e_{i+1}}})$;
        end
    end
    return the constructed diagram;
end
```

We are now ready to give our Slitherlink solver, shown as Algorithm 5, based on Algorithm 4.

**Algorithm 5:** Slitherlink solver

```
Data: A graph $G = (V, E)$ and a partial map $h : 2^E \rightharpoonup \mathbb{N}$
begin
    create a root node and two terminal nodes 0 and 1;
    let $N_1 \leftarrow \{n_{\text{root}}\}$ and $N_i \leftarrow \emptyset$ for $i = 2, \ldots, |E|$;
    foreach $i = 1, \ldots, |E|$ do
        foreach $n \in N_i$ do
            if $(\text{mate}_n, i)$ has a fixed end or $(\text{count}_n, i)$ is incompatible with $h$ then
                let the 0-child of $n$ be 0;
            else
                let the 0-child of $n$ be $\text{GN}(i + 1, \text{mate}_n|_{V^{\geq e_{i+1}}}, \text{count}_n)$;
            end
            if $\text{mate}_n$ and $e_i$ form a cycle and $\text{CU}(\text{count}_n, e_i)$ matches $h$ then
                let the 1-child of $n$ be 1;
            else if $\text{mate}_n$ declines $e_i$ or $(\text{CU}(\text{count}_n, e_i), i)$ is incompatible with $h$ then
                let the 1-child of $n$ be 0;
            else
                let the 1-child of $n$ be $\text{GN}(i + 1, \text{MU}(\text{mate}_n, e_i)|_{V^{\geq e_{i+1}}}, \text{CU}(\text{count}_n, e_i))$;
            end
        end
    end
    return the constructed diagram;
end
```

Let $(G, h)$ be a Slitherlink instance. To ensure that the computed cycle is compatible with the hint assignment $h$, we assign another function $\text{count}_n : \text{dom}(h) \to \mathbb{N}$ to each node $n$ in addition to $\text{mate}_n$. The function counts the number of picked edges in $D$ in the domain of $h$. That is, for a valid path $\pi$, we shall have $\text{count}_{n(\pi)}(D) = |E(\pi) \cap D|$ for each $D \in \text{dom}(h)$. Especially $\text{count}_{n_{\text{root}}}(D) = 0$ for each $D \in \text{dom}(h)$. Updating the counter is very easy. For a mapping $c : \text{dom}(h) \to \mathbb{N}$ and an edge $e \in E$, we define $\text{CU}(c, e) : \text{dom}(h) \to \mathbb{N}$ (it stands for "counter update") by

$$\text{CU}(c, e)(D) = \begin{cases} c(D) & \text{if } e \notin D \\ c(D) + 1 & \text{if } e \in D \end{cases}$$

for every $D \in \text{dom}(h)$.

For $\pi$ of length $i$, when it is clear that there is no $P' \subseteq E^{\geq e_{i+1}}$ such that $E(\pi) \cup P'$ is compatible with $h$, the path $\pi$ should end in $\mathbf{0}$. We say that $(c, i)$ is *incompatible with* $h$ if there is $D \in \text{dom}(h)$ such that either

- $c(D) > h(D)$ or
- $c(D) + |D \cap E^{\geq e_{i+1}}| < h(D)$.

We say that $c$ *matches* $h$ if $c(D) = h(D)$ for all $D \in \text{dom}(h)$.

The function GN is modified so that it handles a counter function in addition to a mate function. It takes an edge index $i \in \{1, \dots, |E|\}$, a mate function $m$ and a counter function $c$. $\mathrm{GN}(i, m, c)$ returns a node labeled $e_i$ and assigned $m$ and $c$. If such a node has already been created in $N_i$, that node is returned. Otherwise, we create one in $N_i$ and return it. Note that this procedure may update $N_i$.

We give a formal proof for the correctness of Algorithm 5 in Appendix E.

## 7. Slitherlink Instance Enumeration

In this section, we discuss instance enumeration for the Slitherlink problem, assuming that a graph $G = (V, E)$, a solution cycle $C$ over $G$, and the maximum domain $\mathcal{E} \subseteq 2^E$ of hint assignments are given. Let $h : \mathcal{E} \to \mathbb{N}$ be a hint assignment such that $h(E') = |C \cap E'|$ for all $E' \in \mathcal{E}$. The objective is to enumerate every $\mathcal{E}' \subseteq \mathcal{E}$ that leads to a good instance $(G, h')$ of the Slitherlink problem such that $\mathrm{dom}(h') = \mathcal{E}'$ and $h'(E') = h(E')$ for all $E' \in \mathcal{E}'$. A good instance $(G, h')$ must not have any compatible cycles other than $C$. Therefore, the instance enumeration problem is solved by computing

$$H = \left\{ \mathcal{E}' \subseteq \mathcal{E} \;\middle|\; \bigvee_{E' \in \mathcal{E}} (E' \in \mathcal{E}' \;\wedge\; h(E') \neq |C' \cap E'|) \;\; \text{for all} \;\; C' \in \mathcal{C} \setminus \{C\} \right\} \tag{9}$$

where $\mathcal{C}$ is the set of all cycles on $G$. We use Algorithm 4 to construct a ZDD for $\mathcal{C}$. For this sake, we invoke the dynamic virtue of ZDDs that allows us to perform mathematical set operations quite easily.

A ZDD for the family $\mathcal{C}$ can be regarded as a function $\mathcal{C}(\boldsymbol{x}) : \{0,1\}^{|E|} \to \{0,1\}$ such that

$$\mathcal{C}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \{e \mid x_e = 1\} \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases}$$

where $\boldsymbol{x} = (x_{e_1}, \dots, x_{e_{|E|}})$ is a vector of binary variables representing if each edge is taken. A ZDD for $\{C\}$ can also be regarded as a function $C(\boldsymbol{x}) : \{0,1\}^{|E|} \to \{0,1\}$ such that

$$C(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \{e \mid x_e = 1\} = C \\ 0 & \text{otherwise} \end{cases}$$

Let $\boldsymbol{y} = (y_{E'_1}, \dots, y_{E'_{|\mathcal{E}|}})$ be a vector of binary variables representing if the $i$th element $E'_i$ of $\mathcal{E}$ is included in $\mathcal{E}'$. Now we can rewrite Equation (9) in terms of $\boldsymbol{x}$ and $\boldsymbol{y}$ as follows, which can be computed by conventional operations on ZDDs:

$$H(\boldsymbol{y}) = \forall \boldsymbol{x}. \left( \neg \mathcal{C}(\boldsymbol{x}) \vee C(\boldsymbol{x}) \vee \bigvee_{E' \in \mathcal{E}} \left( y_{E'} \wedge h(E') \neq \sum_{e \in E'} x_e \right) \right)$$

Once $H(\boldsymbol{y})$ is computed, ZDD operations enable us to restrict it easily to some interesting subset, such as instances with the minimum or minimal hints, and to output all instances explicitly.

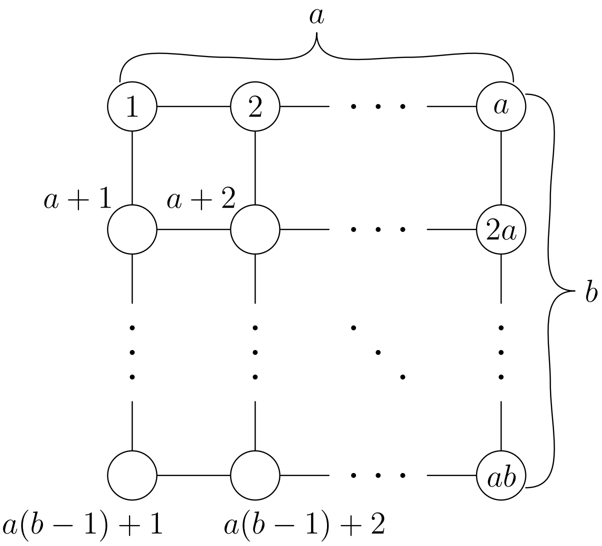
## 8. Experiments

The following experiments for the Numberlink solvers, the Slitherlink solvers and the Slitherlink generators are performed on AMD Opteron Processor$^{\mathrm{TM}}$ 8393, 3.09 GHz with 512 GB memory running SuSE 10, and those for the Numberlink generators are performed on AMD Opteron Processor$^{\mathrm{TM}}$ 6136, 2.40 GHz with 256 GB memory running SuSE 10.

### *8.1. Solvers for Numberlink and Slitherlink*

In this subsection, we discuss our experimental results on the comparison of our algorithm for solving Numberlink and Slitherlink with existing methods which are based on CPLEX 12.3 and Sugar. We implemented Algorithm 2 for Numberlink solver and Algorithm 5 for Slitherlink solver in C++ and compiled them by G++ 4.6 with `-O3` optimization option. In our implementation, a variable order (an edge order) is lexicographical on sets of two vertices where vertices are labeled such as Figure 6.

**Figure 6.** Grid graph $G_{b,a}$ and its vertex labels.



We prepared the benchmark programs. We formulated solving Numberlink [15] and Slitherlink [16] as integer programs and solve them by CPLEX, which is an integer programming solver. Sugar is a SAT based constraint solver [8]. Sugar provides the formulation of Numberlink [17] and Slitherlink [18]. We also compared them with a heuristic solver *slink* [9] specialized for Slitherlink, which is published as a C source code.

Table 1 shows the running times of these methods for solving Numberlink. In the table, BN$x$ denotes the $x$th instance in [19]. C88 is the Vol. 88 instance in Puzzle Championships 2010 [20]. Let $G_{\alpha,\beta}$ denote the $\alpha \times \beta$ grid graph. In Table 2, we show the running times of these methods with the restriction that requires solutions to cover all vertices (see Remark 7).

**Table 1.** Running time in seconds for solving Numberlink. Our algorithm could not solve C88 due to insufficient memory.

| **Instance** | **Graph** | **Time CPLEX** | **Time Sugar** | **Time Algorithm 2** | **Memory Algorithm 2 (MB)** |
|---|---|---|---|---|---|
| BN1 | $G_{8,8}$ | 0.02 | 1.179 | 0.008 | 2 |
| BN15 | $G_{8,8}$ | 13.95 | 1.336 | 0.004 | 2 |
| BN30 | $G_{10,10}$ | 88.32 | 1.896 | 0.016 | 5 |
| BN43 | $G_{10,10}$ | 6,410.67 | 1.924 | 0.072 | 18 |

**Table 1.** *Cont.*

| Instance | Graph | Time CPLEX | Time Sugar | Time Algorithm 2 | Memory Algorithm 2 (MB) |
|---|---|---|---|---|---|
| BN52 | $G_{10,10}$ | 0.45 | 1.520 | 0.012 | 3 |
| BN64 | $G_{10,10}$ | 108.07 | 1.704 | 0.136 | 11 |
| BN72 | $G_{10,10}$ | 276.07 | 1.388 | 0.012 | 4 |
| BN79 | $G_{10,10}$ | $15,117.47$ | 1.496 | 0.220 | 37 |
| BN85 | $G_{20,15}$ | $>100,000$ | $7,213.079$ | 862.518 | $43,846$ |
| BN99 | $G_{20,15}$ | $>100,000$ | 14.097 | $1,658.772$ | $67,094$ |
| C88 | $G_{20,36}$ | $>100,000$ | $>100,000$ | – | – |

**Table 2.** Running time in seconds for solving Numberlink with restriction that requires solutions to cover all vertices.

| Instance | Graph | Time CPLEX | Time Sugar | Time Algorithm 2 | Memory Algorithm 2 (MB) |
|---|---|---|---|---|---|
| BN1 | $G_{8,8}$ | 0.01 | 1.148 | 0.004 | 2 |
| BN15 | $G_{8,8}$ | 0.40 | 0.964 | 0.000 | 2 |
| BN30 | $G_{10,10}$ | 0.82 | 1.576 | 0.024 | 3 |
| BN43 | $G_{10,10}$ | 21.97 | 1.552 | 0.012 | 5 |
| BN52 | $G_{10,10}$ | 0.10 | 1.296 | 0.004 | 2 |
| BN64 | $G_{10,10}$ | 18.40 | 1.368 | 0.012 | 5 |
| BN72 | $G_{10,10}$ | 1.50 | 1.648 | 0.012 | 2 |
| BN79 | $G_{10,10}$ | 26.74 | 1.452 | 0.048 | 10 |
| BN85 | $G_{20,15}$ | $>100,000$ | $5,431.991$ | 185.264 | $8,761$ |
| BN99 | $G_{20,15}$ | $>100,000$ | 12.585 | 175.147 | $9,727$ |

Table 3 describes the running time of these methods as Slitherlink solvers. BS$x$ denotes the $x$th instance in [21]. S10 is a large sample problem on [22]. C95 is the Vol. 95 instance in Puzzle Championships 2010 [20].

**Table 3.** Running time in seconds for solving Slitherlink. Our algorithm could not solve C95 due to insufficient memory.

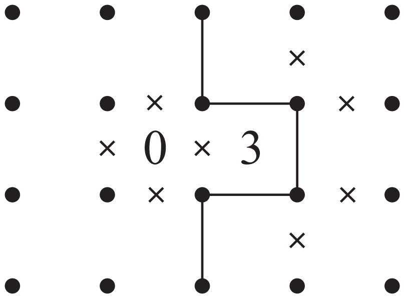
| Instance | Graph | Time CPLEX | Time Sugar | Time slink | Time Algorithm 5 | Memory Algorithm 5 (MB) |
|---|---|---|---|---|---|---|
| BS1 | $G_{11,11}$ | 0.07 | 5.460 | 0.001 | 0.116 | 34 |
| BS12 | $G_{11,11}$ | 0.28 | 4.696 | 0.004 | 0.384 | 41 |
| BS25 | $G_{11,11}$ | 0.12 | 6.496 | 0.001 | 0.060 | 34 |
| BS37 | $G_{11,11}$ | 0.14 | 8.773 | 0.004 | 0.044 | 34 |

**Table 3.** *Cont.*

| Instance | Graph | Time CPLEX | Time Sugar | Time slink | Time<br>Algorithm 5 | Memory<br>Algorithm 5 (MB) |
|---|---|---|---|---|---|---|
| BS43 | $G_{19,11}$ | 0.17 | 8.349 | 0.001 | 0.060 | 36 |
| BS54 | $G_{19,11}$ | 0.40 | 7.380 | 0.008 | 0.304 | 37 |
| BS68 | $G_{25,15}$ | 0.39 | 12.433 | 0.004 | 0.384 | 39 |
| BS77 | $G_{25,15}$ | 0.90 | 13.281 | 0.004 | 0.376 | 42 |
| BS89 | $G_{37,21}$ | 1.56 | 48.035 | 0.016 | 9.329 | 407 |
| BS96 | $G_{37,21}$ | 3.07 | 186.612 | 0.104 | 26.286 | 904 |
| S10 | $G_{21,37}$ | 41.43 | $1,411.944$ | 0.036 | $4,569.330$ | $138,053$ |
| C95 | $G_{32,46}$ | 95.93 | $1,076.839$ | 0.088 | – | – |

This table shows that slink is by far the fastest. A reason is that slink is a heuristic solver for Slitherlink which uses information specialized for Slitherlink. For example, when two adjacent cells are assigned "0" and "3" respectively, 12 of the surrounding edges are immediately determined to be or not to be in any solution (see Figure 7).

**Figure 7.** An example of information used by a heuristic Slitherlink solver.



## 8.2. *Numberlink Generator*

In this subsection, we show experimental results for enumerating all Numberlink instances for grid graphs by a variant of Algorithm 3. Our actual implementation is purely top-down, which is different from Algorithm 3. By interweaving the idea of Algorithm 3 into Algorithm 1, we construct $Z_G$ assigning nodes mate functions as well as pair matching families representing the ends of fixed paths simultaneously. In our algorithm, $S_n$ and $B_n$ are calculated by set operations such as union and intersection. We implemented these sets by ZDDs whose variables are unordered pairs $\{v, v'\}$ of vertices $v, v' \in V$. For efficiency, our implementation maintains $\langle W_n, B_n \rangle$ instead of $\langle S_n, B_n \rangle$ for each node $n$ in $Z_G$, where $W_n = S_n \setminus B_n$.

Tables 4, 5 and 6 show the running time of our algorithm, the number of good instances and that of nodes of $S_1 \setminus B_1$ for $G_{\alpha,\beta}$.

**Table 4.** Running time of Numberlink generator for $G_{\alpha,\beta}$.

| $\alpha\backslash\beta$ | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 2 | <0.01 | <0.01 | <0.01 | <0.01 | 0.01 | 0.02 |
| 3 | – | <0.01 | 0.01 | 0.08 | 0.48 | 2.67 |
| 4 | – | – | 0.21 | 3.56 | 44.13 | 488.12 |
| 5 | – | – | – | 107.99 | 3,553.29 | >100,000 |
| 6 | – | – | – | – | >100,000 | >100,000 |
| 7 | – | – | – | – | – | >100,000 |

**Table 5.** The number of good instances of Numberlink for $G_{\alpha,\beta}$.

| $\alpha\backslash\beta$ | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 2 | 2 | 10 | 36 | 126 | 454 | 1,632 |
| 3 | – | 86 | 807 | 6,690 | 58,422 | 499,733 |
| 4 | – | – | 16,410 | 338,460 | 6,901,105 | 141,123,690 |
| 5 | – | – | – | 16,027,290 | 784,030,205 | Time Out |
| 6 | – | – | – | – | Time Out | Time Out |
| 7 | – | – | – | – | – | Time Out |

**Table 6.** The number of ZDD nodes of $W_1$ in Numberlink generator for $G_{\alpha,\beta}$.

| $\alpha\backslash\beta$ | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 2 | 4 | 18 | 50 | 117 | 235 | 419 |
| 3 | – | 99 | 491 | 1,686 | 4,918 | 12,255 |
| 4 | – | – | 3,499 | 22,599 | 108,954 | 439,506 |
| 5 | – | – | – | 226,800 | 1,977,335 | Time Out |
| 6 | – | – | – | – | Time Out | Time Out |
| 7 | – | – | – | – | – | Time Out |

For $G_{6,6}$, the computation did not finish within one week. We tried to compute the number of instances with restricting the number of pairs to at most $\ell$. Table 7 shows the result of this computation for $G_{6,6}$ and $G_{7,7}$. Let $W_{n,i} = \{T \in W_n \mid |T| = i\}$ and $B_{n,i} = \{T \in B_n \mid |T| = i\}$. We maintain $\langle W_{n,0}, \ldots, W_{n,\ell}, B_{n,0}, \ldots, B_{n,\ell}\rangle$ instead of $\langle W_n, B_n\rangle$ in each node $n$. We omit the method of computing $W_{n,i}$ and $B_{n,i}$. Then, we obtain that the number of minimum pairs with which $G_{6,6}$ gives a good instance is 3 from this result. We give the complete list of those instances on the graph $G_{6,6}$ with 3 pairs of vertices in Appendix G.

**Table 7.** Running time, the number of good instances with at most $\ell$ pairs of numbers, and that of ZDD nodes of $W_1 = S_1 \setminus B_1$.

| **Graph** | $\boldsymbol{\ell}$ | **Time** | **Good instances** | $\boldsymbol{\lvert W_{n,\ell} \rvert}$ |
|---|---|---|---|---|
| $G_{6,6}$ | 0 | 39.63 | 0 | 1 |
| $G_{6,6}$ | 1 | 141.13 | 0 | 1 |
| $G_{6,6}$ | 2 | 1,329.35 | 0 | 1 |
| $G_{6,6}$ | 3 | 12,161.5 | 304 | 472 |
| $G_{6,6}$ | 4 | >1,000,000 | – | – |
| $G_{7,7}$ | 0 | 2,732.86 | 0 | 1 |
| $G_{7,7}$ | 1 | 14,226 | 0 | 1 |
| $G_{7,7}$ | 2 | >1,000,000 | – | – |

### *8.3. Slitherlink Generator*

For the Slitherlink generator, we tried to generate some interesting examples of Slitherlink instances in order to confirm the usefulness of our tool. Our tool generates Slitherlink instances based on three kinds of input: a graph, a solution cycle, and the maximum domain of hint assignments. In usual cases where instances are made with grid graphs, a domain of hint assignments is represented by a set of rectangular cells.

Figure 8 shows an example of Slitherlink generation. We drew a solution cycle on a $5 \times 7$ grid of cells and marked all 35 cells as the maximum set of hint cells. When they were given as the input, our tool successfully constructed a ZDD for a family of sets of hint cells in 8 s, from which we can obtain all the 2,912,556,380 good Slitherlink instances that have the given solution cycle. The tool can optionally restrict generated instances to minimal ones (instances such that any hint cannot be removed without making the solution not unique). It took 0.1 s for this example and the number of instances was reduced to 32,639. The output example shown in Figure 8 is the one selected automatically by our tool as the most difficult instance among them in our criteria, in which the difficulty is evaluated by smallness of the number of cells with hint values "4", "0", "3", "1", and "2" in this order of precedence. Another output example was shown already in Figure 2, which is one of the 81 good instances that consist of minimum hints.

**Figure 8.** Input and output of Slitherlink generation; example #1.

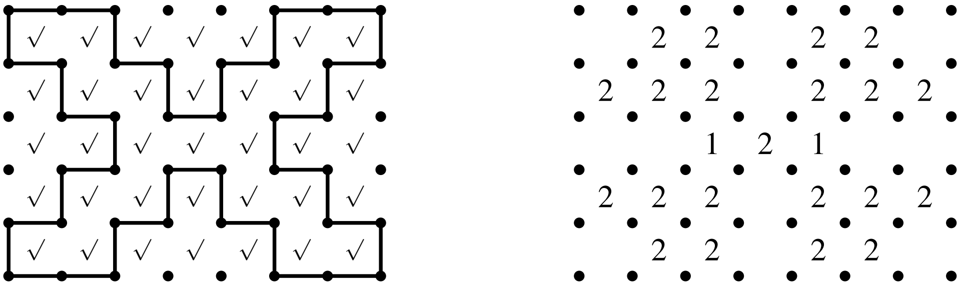


Figure 9 shows another input example, in which the solution cycle is drawn on a 8 × 8 grid of cells. Unfortunately, the computation could not be finished even in a day when all 64 cells are marked as candidates of hint locations. Our tool, however, was still able to assist making instances of the Slitherlink problem according to the designer's intent. We reduced the candidate locations to 36 cells (the right picture of Figure 9) and got 1,669,424 instances, 1,850 minimal instances, and 4 minimum instances (Figure 10) in 2 s. It is also interesting that the same mechanism can be used to modify existing instances. We found that hand-made instances often include many redundant hints (e.g., 12 out of 40) and our tool can be used to make them more difficult to solve.

**Figure 9.** Input of Slitherlink generation; example #2.

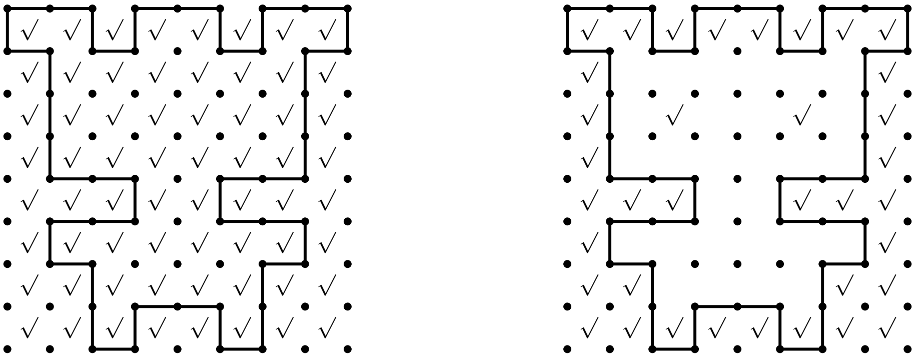
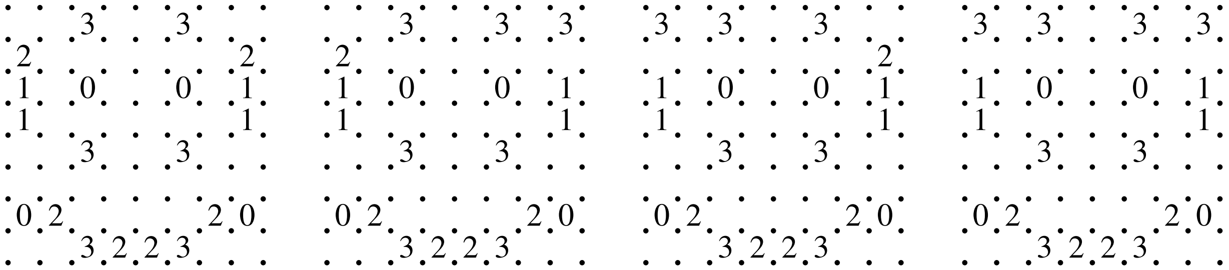
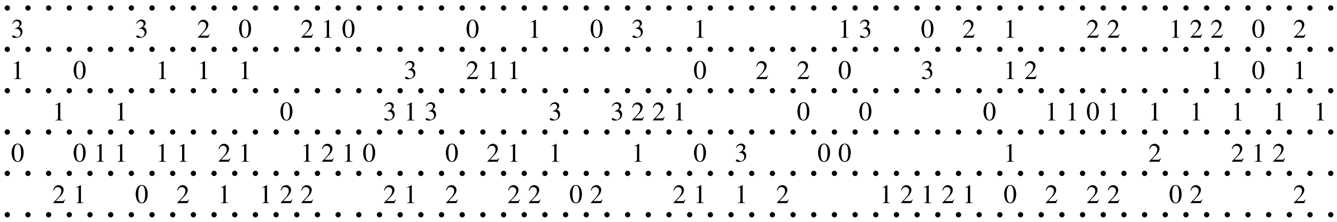


**Figure 10.** Output of Slitherlink generation; example #2.



Finally, we introduce an enjoyable application of our tool, which can make a puzzle containing a secret message. Unlike the original Slitherlink, we allow multiple cycles as its solution in order to embed multiple letters into the puzzle. The algorithm of Slitherlink instance generation is modified by replacing computation of "all cycles on $G$" with "all combinations of disjoint cycles on $G$", which is actually not difficult to be computed by using conditions on degree of the graph vertices. Figure 11 is our message to readers.

**Figure 11.** An example of message puzzle instance.



## 9. Conclusions

This paper has proposed algorithms that enumerate solutions and instances for two link puzzles, Numberlink and Slitherlink, based on Knuth's path enumeration algorithm SIMPATH.

Our Numberlink solver (Algorithm 2) is faster than Sugar and CPLEX. In Slitherlink, our solution enumeration algorithm (Algorithm 5) shows better performance than Sugar. CPLEX is sometimes faster than our algorithm. Slink runs much faster than our algorithm and CPLEX to find a solution for Slitherlink instances. The result looks rather reasonable, since Sugar and CPLEX are designed for quite general purposes and find only one solution, while Slink is specially designed to solve Slitherlink problems. Our algorithms are located in between. The core of our algorithms is specialized to enumerate path matchings on a general graph, to which we have plugged mechanisms that decide whether the currently obtained path matchings can be expanded to a solution. It is not difficult to accelerate our algorithms by employing known local solution methods for the link puzzles on grid graphs. Yet we have largely ignored such detailed improvements. Rather we emphasize the generality of our approach that should be valid to design solution enumeration algorithms for various link puzzles including Masyu, Yajirin and others. To build a ZDD representing solutions, what we need to do is to assign each node an appropriate configuration like $\mathrm{mate}$ and $\mathrm{count}$ which can be locally updated and which tells whether the currently obtained path matchings may be grown up to solutions.

Another point we would like to emphasize is that our algorithms give all the solutions at once as a ZDD unlike existing solvers. Our instance enumeration algorithms rely much on this feature of our approach. We benefit from the virtue of ZDDs as a set manipulation system. As we have demonstrated in Section 8.3, we have flexible means on ZDDs to extract instances with several properties from the whole set of good instances, like non-redundant ones, the hardest ones, ones that use specific cells, and so on. The authors believe manipulating puzzle instances on a ZDD is quite beneficial to puzzle designers. It is future work to develop an assistant tool for puzzle designers that equips several convenient functions with a friendly interface.

## Acknowledgements

The authors are very much grateful to the anonymous reviewers for their valuable comments and suggestions that considerably improved the quality of our paper.

## References

1. Nikoli: Web Nikoli. Available online: http://www.nikoli.co.jp/ (accessed on 21 March 2012)
2. Kotsuma, K.; Takenaga, Y. *NP-Completeness and Enumeration of Number Link Puzzle [in Japanese]*; IEICE Technical Report, COMP2009-49; The Institute of Electronics, Information and Communication Engineering: Tokyo, Japan, 2010; pp. 1–7.
3. Kramer, M.R.; van Leeuwen, J. *Wire-Routing is NP-Complete*; Report No. RUU-CS-82-4; Department of Computer Science, University of Utrecht: Utrecht, The Netherlands, 1982.
4. Takahashi, J.; Suzuki, H.; Nishizeki, T. Shortest noncrossing paths in plane graphs. *Algorithmica* **1996**, *16*, 339–357.
5. Richards, D. Complexity of single-layer routing. *IEEE Trans. Comput.* **1984**, *33*, 286–288.

6. Yato, T. *On the NP-Completeness of the Slither Link Puzzle [in Japanese]*; IPSJ SIG Notes AL-74; Information Processing Society of Japan: Tokyo, Japan, 2000; pp. 25–32.
7. Hearn, R.A.; Demaine, E.D. *Games, Puzzles, and Computation*; A K Peters, Ltd.: Boca Raton, FL, USA, 2009.
8. Tamura, N. Sugar: A SAT-Based Constraint Solver. Available online: http://bach.istc.kobe-u.ac.jp/sugar/ (accessed on 21 March 2012)
9. Itogawa, A. Slither Link Kaito Program (Solving Slither Link Program: In Japanese). Available online: http://www2.ttcn.ne.jp/~itogawa/product/slitherlink.html (accessed on 21 March 2012)
10. Shirai, H.; Igarashi, C.; Tajima, Y.; Kotani, Y. Solving and Making Problems of Slither Link [in Japanese]. In *Proceedings of the 11th Game Programming Workshop in Japan 2006*, Kanagawa, Japan, 10–12 November 2006; pp. 32–39.
11. Wan, J. Logical Slither Link; Bsc These, University of Manchester, Manchester, UK, 2009.
12. Minato, S. Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems. In *Proceedings of the 30th International Design Automation Conference (DAC '93)*, Dallas, TX, USA, 14-18 June 1993; pp. 272–277.
13. Knuth, D. *The Art of Computer Programming, Volume 4, Fascicle 1*; Addison-Wesley Professional: Boston, MA, USA, 2009.
14. Yato, T.; Seta, T. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2003**, *E86-A*, 1052–1060.
15. GLPK deno Pencil Puzzle Koryakuho (Solving Pencil Puzzle Using GLPK : In Japanese). Available online: http://www21.tok2.com/home/kainaga11/glpk/glpk.htm (accessed on 21 March 2012)
16. Sugimura, Y. Seisu Keikakuho niyoru Slither Link no Kaiho (Slither Link Algorithm Using Integer Programming: In Japanese). In *Proceedings of the 1st Games and Puzzles Mini Workshop*, Kyoto, Japan, 12 September, 2005.
17. Tamura, N. Solving Number Link Puzzles with Sugar Constraint Solver [in Japanese]. Available online: http://bach.istc.kobe-u.ac.jp/sugar/puzzles/numberlink.html (accessed on 21 March 2012)
18. Tamura, N. Solving Slither Link Puzzles with Sugar Constraint Solver [in Japanese]. Available online: http://bach.istc.kobe-u.ac.jp/sugar/puzzles/slitherlink.html (accessed on 21 March 2012)
19. Nikoli. *Numberlink 1*; Nikoli: Tokyo, Japan, 1989.
20. Nikoli: Nikoli.com Puzzle Championship. Available online: http://www.nikoli.com/en/event/puzzle_hayatoki.html (accessed on 21 March 2012)
21. Nikoli. *Slitherlink 1*; Nikoli: Tokyo, Japan, 1992.
22. Nikoli: Sample Problems of Slitherlink Puzzle. Available online: http://www.nikoli.com/en/puzzles/slitherlink/ (accessed on 21 March 2012)

## A. Correctness of Algorithm 1 (Path Matching Enumeration)

This section proves that Algorithm 1 constructs a ZDD representing the family of path matchings over $G$. Hereafter $\mathrm{mate}_n$ refers to the function assigned to the node $n$ by Algorithm 1 rather than the one given in Equation (2). Actually those two coincide as we will prove below.

For a $0, 1$-sequence $\pi$ of length $i$ such that $E(\pi)$ is a path matching, we define a mate function $\mathrm{mate}_\pi : V^{\geq e_{i+1}} \to V \cup \{0\}$ by

$$\mathrm{mate}_\pi(v) = \begin{cases} v & \text{if } v \notin \bigcup E(\pi) \\ u & \text{if } E(\pi) \text{ contains a } u\text{-}v \text{ path} \\ 0 & \text{otherwise} \end{cases}$$

In the sequel, we assume that $\mathrm{mate}_\mathbf{1}$ represents the special mate function whose domain is empty.

**Lemma 8.** *Let $\pi$ be a $0, 1$-sequence of length $i \leq |E|$. If $E(\pi)$ is a path matching, then $\pi$ is a valid path in $Z_G$ and* $\mathrm{mate}_{n(\pi)} = \mathrm{mate}_\pi$.

*Proof.* We prove the lemma by induction on $i$. For $i = 0$, $E(\pi)$ is the empty set, which is a special case of a path matching, and $\mathrm{mate}_\pi$ is the identity function on $V^{\geq e_1} = V$ by definition. The empty path $\pi$ ends in the root node $n_\mathrm{root}$ where we have $\mathrm{mate}_{n_\mathrm{root}} = \mathrm{mate}_\pi$.

Suppose that the lemma holds for $\pi$ of length $i - 1$ where $0 < i \leq |E|$. If $E(\pi \cdot 0)$ is a path matching, $\pi$ ends in a node $n \notin \{\mathbf{0}, \mathbf{1}\}$ in $Z_G$ such that $\mathrm{mate}_n = \mathrm{mate}_\pi$. By the algorithm, $\pi \cdot 0$ ends in a node $n' \neq \mathbf{0}$ such that

$$\mathrm{mate}_{n'} = \mathrm{mate}_n|_{V^{\geq e_{i+1}}} = \mathrm{mate}_\pi|_{V^{\geq e_{i+1}}} = \mathrm{mate}_{\pi \cdot 0}$$

Suppose that $E(\pi \cdot 1)$ is a path matching. Let $E(\pi \cdot 1)$ consist of pairwise disjoint simple paths $P_1, \dots, P_k$. Without loss of generality, we may assume that $e_i \in P_1$. We note that $e_i \cap \bigcup(P_2 \cup \dots \cup P_k) = \emptyset$. Suppose that there are $m$ distinct vertices $v_1, \dots, v_m \in V$ such that $P_1 = \{\, \{v_j, v_{j+1}\} \mid 1 \leq j < m \,\}$. For each $v \in V^{\geq e_{i+1}}$, we have

$$\mathrm{mate}_{\pi \cdot 1}(v) = \begin{cases} v_m & \text{if } v = v_1 \\ v_1 & \text{if } v = v_m \\ 0 & \text{if } v \in \{v_2, \dots, v_{m-1}\} \\ \mathrm{mate}_\pi(v) & \text{otherwise} \end{cases}$$

(Case 1) $P_1 = \{e_i\} = \{\{v_1, v_2\}\}$. Then $E(\pi)$ consists of simple paths $P_2, \dots, P_k$. By the assumption we have

$$\mathrm{mate}_n(v_1) = \mathrm{mate}_\pi(v_1) = v_1$$
$$\mathrm{mate}_n(v_2) = \mathrm{mate}_\pi(v_2) = v_2$$

Hence $\mathrm{mate}_n$ does not reject $e_i$. Let $n''$ be the 1-child of $n$. We have $\mathrm{mate}_{n''} = \mathrm{MU}(\mathrm{mate}_n, e_i)|_{V^{\geq e_{i+1}}}$. Recall the definition of MU:

$$\mathrm{MU}(\mathrm{mate}_n, e_i)(u) = \begin{cases} 0 & \text{if } u \in e_i \text{ and } \mathrm{mate}_n(u) \neq u \\ \mathrm{mate}_n(v) & \text{if } e_i = \{v, \mathrm{mate}_n(u)\} \\ \mathrm{mate}_n(u) & \text{otherwise} \end{cases}$$

for all $u$ in the domain of $\mathrm{mate}_n$. Since $\mathrm{mate}_n(v_1) = v_1$ and $\mathrm{mate}_n(v_2) = v_2$, the first case of the definition does not apply. The only vertices $u$ such that $\mathrm{mate}_n(u) \neq \mathrm{MU}(\mathrm{mate}_n, e_i)(u)$ are $v_1$ and $v_2$. By the second case of the definition, we have $\mathrm{MU}(\mathrm{mate}_n, e_i)(v_1) = v_2$ and $\mathrm{MU}(\mathrm{mate}_n, e_i)(v_2) = v_1$. Hence we obtain $\mathrm{mate}_{\pi\cdot 1} = \mathrm{mate}_{n''}$.

(Case 2) $P_1 \setminus \{e_i\}$ is a simple path. That is, either $e_i = \{v_1, v_2\}$ or $e_i = \{v_{m-1}, v_m\}$. Without loss of generality, we assume $e_i = \{v_1, v_2\}$. Since $P_1 \setminus \{e_i\}$ is not empty, $m > 2$. Then $E(\pi)$ consists of simple paths $P_1 \setminus \{e_i\}, P_2, \ldots, P_k$. By the assumption we have

$$\mathrm{mate}_n(v_1) = \mathrm{mate}_\pi(v_1) = v_1$$
$$\mathrm{mate}_n(v_2) = \mathrm{mate}_\pi(v_2) = v_m$$

Hence $\mathrm{mate}_n$ does not reject $e_i$. Also note that

$$\mathrm{mate}_n(v_m) = \mathrm{mate}_\pi(v_m) = v_2$$

if $v_m \in V^{\geq e_i}$. Let $n''$ be the 1-child of $n$, for which we have $\mathrm{mate}_{n''} = \mathrm{MU}(\mathrm{mate}_n, e_i)|_{V^{\geq e_{i+1}}}$. By the definition of MU, we have

$$\begin{aligned} \mathrm{MU}(\mathrm{mate}_n, e_i)(v_1) &= \mathrm{mate}_n(v_2) = v_m \\ \mathrm{MU}(\mathrm{mate}_n, e_i)(v_2) &= 0 \\ \mathrm{MU}(\mathrm{mate}_n, e_i)(v_m) &= \mathrm{mate}_n(v_1) = v_1 \quad \text{ if } v_m \in V^{\geq e_i} \end{aligned}$$

and for other vertices $u \in V^{\geq e_{i+1}} \setminus \{v_1, v_2, v_m\}$ we have $\mathrm{MU}(\mathrm{mate}_n, e_i)(u) = \mathrm{mate}_n(u)$. The induction hypothesis $\mathrm{mate}_n = \mathrm{mate}_\pi$ implies $\mathrm{mate}_{n''} = \mathrm{mate}_{\pi\cdot 1}$.

(Case 3) $P_1 \setminus \{e_i\}$ is a path matching consisting of two simple paths $P_1'$ and $P_1''$. That is, $e_i = \{v_j, v_{j+1}\}$ for some $j \in \{2, \ldots, m-2\}$. Then $E(\pi)$ consists of simple paths $P_1', P_1'', P_2, \ldots, P_k$. By the assumption we have

$$\mathrm{mate}_n(v_j) = \mathrm{mate}_\pi(v_j) = v_1$$
$$\mathrm{mate}_n(v_{j+1}) = \mathrm{mate}_\pi(v_{j+1}) = v_m$$

Hence $\mathrm{mate}_n$ does not reject $e_i$. Let $n''$ be the 1-child of $n$, for which we have $\mathrm{mate}_{n''} = \mathrm{MU}(\mathrm{mate}_n, e_i)|_{V^{\geq e_{i+1}}}$. By the definition of MU, we have

$$\begin{aligned} \mathrm{MU}(\mathrm{mate}_n, e_i)(v_1) &= \mathrm{mate}_n(v_{j+1}) = v_m && \text{if } v_1 \in V^{\geq e_i} \\ \mathrm{MU}(\mathrm{mate}_n, e_i)(v_j) &= 0\,, && \text{where } v_j \in V^{\geq e_i} \\ \mathrm{MU}(\mathrm{mate}_n, e_i)(v_{j+1}) &= 0\,, && \text{where } v_{j+1} \in V^{\geq e_i} \\ \mathrm{MU}(\mathrm{mate}_n, e_i)(v_m) &= \mathrm{mate}_n(v_j) = v_1 && \text{if } v_m \in V^{\geq e_i} \end{aligned}$$

and for other vertices $u \in V^{\geq e_{i+1}} \setminus \{v_1, v_j, v_{j+1}, v_m\}$, we have $\mathrm{mate}_n(u) = \mathrm{MU}(\mathrm{mate}_n, e_i)(u)$. The induction hypothesis $\mathrm{mate}_n = \mathrm{mate}_\pi$ implies $\mathrm{mate}_{n''} = \mathrm{mate}_{\pi\cdot 1}$. □

**Lemma 9.** *Let $\pi$ be a $0, 1$-sequence of length $i \leq |E|$. If $\pi$ ends in a node $n \neq \mathbf{0}$ in $Z_G$, then $E(\pi)$ is a path matching and* $\mathrm{mate}_n = \mathrm{mate}_\pi$.

*Proof.* We show that if $E(\pi)$ is not a path matching for a $0,1$-sequence $\pi$ of length $|E|$, then there is a prefix $\pi' \cdot 1$ of $\pi$ such that $\pi' \cdot 1$ ends in $\mathbf{0}$ in $Z_G$.

Let $\pi'$ be the longest prefix of $\pi$ such that $E(\pi')$ is a path matching. Indeed such $\pi'$ exists, since $E(\epsilon)$ is a path matching for the empty sequence $\epsilon$. Since $E(\pi')$ is a path matching and so is $E(\pi' \cdot 0)$, hence $\pi' \cdot 1$ is a prefix of $\pi$ such that $E(\pi' \cdot 1)$ is not a path matching. By Lemma 8, $\pi'$ is a valid path ending in a node $n$ with $\mathrm{mate}_n = \mathrm{mate}_{\pi'}$. Let $E(\pi' \cdot 1) = E(\pi') \cup \{e_i\}$.

(Case 1) There exists a vertex $v$ of degree more than 2 in $E(\pi' \cdot 1)$. Since $E(\pi')$ is a path matching, $v$ has degree 2 in $E(\pi')$ and $v \in e_i$. By Lemma 8, we have $\mathrm{mate}_n(v) = 0$. Then $\mathrm{mate}_n$ rejects $e_i$ and $\pi' \cdot 1$ ends in $\mathbf{0}$.

(Case 2) There exists no vertex of degree more than 2 in $E(\pi' \cdot 1)$ and there exists a simple cycle $C \subseteq E(\pi' \cdot 1)$. Since $E(\pi')$ is a path matching, we have $C \not\subseteq E(\pi')$ and thus $e_i \in C$. There are distinct vertices $v_1, \dots, v_m \in V$ for some $m \geq 3$ such that $e_i = \{v_1, v_m\}$ and $C \setminus \{e_i\} = \{\, \{v_j, v_{j+1}\} \mid 1 \leq j < m \,\}$. Let $C' = C \setminus \{e_i\}$, which is a simple path. Since $E(\pi' \cdot 1)$ has no vertex of degree more than 2, the degrees of $v_1$ and $v_m$ are 1 in $C'$. By $v_1, v_m \in V^{\geq e_i}$ and Lemma 8, we have $\mathrm{mate}_n(v_1) = v_m$ and $\mathrm{mate}_n(v_m) = v_1$. Thus $\mathrm{mate}_n$ rejects $e_i$ and $\pi' \cdot 1$ ends in $\mathbf{0}$. □

**Theorem 10.** *Algorithm 1 constructs a ZDD representing all the path matchings on $G$.*

*Proof.* By Lemmas 8 and 9. □

## B. Correctness of Algorithm 2 (Numberlink Solver)

Let $Z_{G,h}$ denote the ZDD constructed by Algorithm 2 for input $(G, h)$.

**Lemma 11.** *If a $0,1$-sequence $\pi$ ends in a node $n \notin \{\mathbf{0}, \mathbf{1}\}$ of $Z_{G,h}$, then $E(\pi)$ is a path matching and $\mathrm{mate}_n = \mathrm{mate}_\pi$.*

*Proof.* Obviously if $\pi$ is a valid path in $Z_{G,h}$, then it is also valid in $Z_G$. Moreover, Algorithms 1 and 2 assign the same mate function to the nodes reached by the same valid path $\pi$. Thus the lemma follows from Lemma 8. □

**Lemma 12.** *Let $\pi$ be a $0,1$-sequence of length $i-1$ such that $E(\pi)$ is a path matching. If $(\mathrm{mate}_\pi, i, 0)$ is incompatible with $h$, then there is no $E' \subseteq E^{\geq e_{i+1}}$ such that $E(\pi \cdot 0) \cup E'$ is a solution for $(G, h)$.*

*Proof.* Suppose that $(\mathrm{mate}_\pi, i, 0)$ is incompatible with $h$. Recall the definition that $(m, i, 0)$ is incompatible with $h$ if for some $v \in e_i \setminus V^{\geq e_{i+1}}$, either

- $v \in \bigcup h$ and $m(v) = v$, or
- $v \notin \bigcup h$ and $m(v) \notin \{0, v\}$.

Accordingly we have two cases.

(Case 1) Suppose that there is $u \in (e_i \setminus V^{\geq e_{i+1}}) \cap \bigcup h$ such that $\mathrm{mate}_\pi(u) = u$. No edges incident to $u$ belong to $E(\pi)$ since $\mathrm{mate}_\pi(u) = u$. No edges incident to $u$ belong to $E^{\geq e_{i+1}}$ since $u \notin V^{\geq e_{i+1}}$. The vertex $u$ has degree 0 in $E(\pi) \cup E'$ for any $E' \subseteq E^{\geq e_{i+1}}$. Thus $E(\pi) \cup E'$ is not a solution, since $u \in \bigcup h$.

(Case 2) Suppose that there is $u \in (e_i \setminus V^{\geq e_{i+1}}) \setminus \bigcup h$ such that $\mathrm{mate}_\pi(u) \notin \{0, u\}$. The vertex has degree 1 in $E(\pi)$. No edges incident to $u$ belong to $E^{\geq e_{i+1}}$ by $u \notin V^{\geq e_{i+1}}$. The vertex $u \notin \bigcup h$ has degree 1 in $E(\pi) \cup E'$ for any $E' \subseteq E^{\geq e_{i+1}}$. Thus $E(\pi) \cup E'$ is not a solution, since $u \notin \bigcup h$. □

**Lemma 13.** *Let $\pi$ be a $0,1$-sequence of length $i-1$ such that $E(\pi)$ is a path matching. If $(\mathrm{mate}_\pi, i, 1)$ is incompatible with $h$, then there is no $E' \subseteq E^{\geq e_{i+1}}$ such that $E(\pi \cdot 1) \cup E'$ is a solution for $(G, h)$.*

*Proof.* Suppose that $(\mathrm{mate}_\pi, i, 1)$ is incompatible with $h$. Recall the definition that for $e_i = \{u, v\}$, $(\mathrm{mate}_\pi, i, 1)$ is incompatible with $h$ if one of the following conditions holds.

- $\mathrm{mate}_\pi(v) \in \{0, u\}$,
- $v \in \bigcup h$ and $\mathrm{mate}_\pi(v) \neq v$,
- $\mathrm{mate}_\pi(u), \mathrm{mate}_\pi(v) \in \bigcup h \cup (V \setminus V^{\geq e_{i+1}})$ and $\{\mathrm{mate}_\pi(u), \mathrm{mate}_\pi(v)\} \notin h$.

Let $m = \mathrm{mate}_\pi$. Accordingly there are three cases.

(Case 1) Suppose that $m(v) \in \{0, u\}$. In this case, it is easy to see that $E(\pi \cdot 1)$ is not a path matching. Thus $E(\pi \cdot 1) \cup E'$ is not a solution for any $E' \subseteq E^{\geq e_{i+1}}$.

(Case 2) Suppose that $v \in \bigcup h \setminus \{m(v)\}$. If $m(v) = 0$, then Case 1 applies. Otherwise, $E(\pi)$ contains a $v$-$m(v)$ path. Then $v$ has degree 2 in $E(\pi \cdot 1) = E(\pi) \cup \{\{u, v\}\}$. Thus $E(\pi \cdot 1) \cup E'$ is not a solution for any $E' \subseteq E^{\geq e_{i+1}}$, since $v \in \bigcup h$.

(Case 3) Suppose that $m(v) \in \bigcup h \cup (V \setminus V^{\geq e_{i+1}})$, $m(u) \notin V^{\geq e_{i+1}}$ and $\{m(u), m(v)\} \notin h$. It is easy to see that $E(\pi \cdot 1)$ has a $m(u)$-$m(v)$ path. Let $E' \subseteq E^{\geq e_{i+1}}$ be such that $E(\pi \cdot 1) \cup E'$ is a path matching.

(Case 3.1) Suppose that $m(v) \in \bigcup h$. If $m(v) \in \bigcup E'$, then $m(v)$ has degree 2 in $E(\pi \cdot 1) \cup E'$, which is not a solution. If $m(v) \notin \bigcup E'$, then $E(\pi \cdot 1) \cup E'$ contains a $m(u)$-$m(v)$ path. Since $\{m(u), m(v)\} \notin h$, $E(\pi \cdot 1) \cup E'$ is not a solution.

(Case 3.2) Suppose that $m(u), m(v) \notin V^{\geq e_{i+1}}$. Then $E(\pi \cdot 1) \cup E'$ contains a $m(u)$-$m(v)$ path. Since $\{m(u), m(v)\} \notin h$, $E(\pi \cdot 1) \cup E'$ is not a solution. □

**Corollary 14.** *If $E(\pi)$ is a solution for $(G, h)$ and $|\pi| = |E|$, then $\pi$ ends in* **1** *in $Z_{G,h}$.*

*Proof.* For any prefix $\pi' \cdot b$ of length $i$ of $\pi$, $(\mathrm{mate}_{\pi'}, i, b)$ is never incompatible with $h$ by Lemmas 12 and 13. Together with Lemma 11, $\pi' \cdot b$ is a valid path in $Z_{G,h}$. □

**Lemma 15.** *Suppose that $E(\pi)$ is not a solution for $(G, h)$ for a $0,1$-sequence $\pi$ of length $|E|$. Then there are $0,1$-sequences $\pi'$ and $\pi''$ such that $\pi = \pi' \cdot b \cdot \pi''$ for some $b \in \{0, 1\}$ and $(\mathrm{mate}_{\pi'}, |\pi'| + 1, b)$ is incompatible with $h$.*

*Proof.* Suppose that $E(\pi)$ is not a solution for $(G, h)$. By the proofs of Lemmas 9 and 11, it is clear that if $E(\pi)$ is not a path matching, then the lemma holds. We assume that $E(\pi)$ is a path matching. We have the following cases.

1. $E(\pi)$ contains a $u$-$v$ path but $\{u, v\} \notin h$, where

    1.1. $u, v \in \bigcup h$,

    1.2. $u \in \bigcup h$ and $v \notin \bigcup h$,

    1.3. $u, v \notin \bigcup h$.

2. $E(\pi)$ does not contain a $u$-$v$ path for some $\{u,v\} \in h$, where
   - 2.1. $u$ has degree $0$,
   - 2.2. $u$ has degree $2$,
   - 2.3. $u$ has degree $1$.

Case 2.3 implies that $E(\pi)$ contains a $u$-$t$ path for some $t \neq v$, which is covered by Cases 1.1 or 1.2. We discuss Cases 1.1, 1.2, 1.3, 2.1 and 2.2.

(Case 1.1) $u, v \in \bigcup h$. Let $P$ be the $u$-$v$ path contained in $E(\pi)$ and $e_i = \max P$. Let $\pi'$ be the prefix of length $i-1$ of $\pi$ and $m = \mathrm{mate}_{\pi'}$. Clearly $\pi' \cdot 1$ is a prefix of $\pi$. Let $u', v' \in e_i$ be such that $m(u') = u$ and $m(v') = v$. We then have $m(u'), m(v') \in \bigcup h$ and $\{m(u'), m(v')\} \notin h$. Thus $(m, i, 1)$ is incompatible with $h$.

(Case 1.2) $u \in \bigcup h$ and $v \notin \bigcup h$. Let $P$ be the $u$-$v$ path contained in $E(\pi)$ and $e_i = \max(P \cup E_v)$ where $E_v = \{\, \{v, v'\} \in E \mid v' \in V \,\}$ is the set of edges incident to $v$. We have $v \notin V^{\geq e_{i+1}}$ by the choice of $e_i$. Let $\pi'$ be the prefix of length $i-1$ of $\pi$ and $m = \mathrm{mate}_{\pi'}$.

Suppose that $e_i = \max P$. Then $\pi' \cdot 1$ is a prefix of $\pi$. Let $u', v' \in e_i$ be such that $m(u') = u$ and $m(v') = v$. Thus $m(u') \in \bigcup h$, $m(v') \notin V^{\geq e_{i+1}}$ and $\{m(u'), m(v')\} \notin h$, which means that $(m, i, 1)$ is incompatible with $h$.

Suppose that $e_i \neq \max P$. Then $\pi' \cdot 0$ is a prefix of $\pi$ and $E(\pi)$ contains the $u$-$v$ path. We have $v \notin \bigcup h$ and $m(v) = u \notin \{0, v\}$. That is, $(m, i, 0)$ is incompatible with $h$.

(Case 1.3) $u, v \notin \bigcup h$. Let $e_i = \min\{\max E_u,\ \max E_v\}$. We assume without loss of generality that $e_i = \{u, u'\}$ for some $u' \in V$. We have $u \notin V^{\geq e_{i+1}}$ by the choice of $e_i$. Let $\pi'$ be the prefix of length $i-1$ of $\pi$ and $m = \mathrm{mate}_{\pi'}$.

Suppose that $e_i \notin E(\pi)$. For the edge $e_j$ in the $u$-$v$ path incident to $u$, we have $e_j < e_i$. Thus we have $m(u) \notin \{0, u\}$ and $u \notin V^{\geq e_{i+1}}$, which means that $(m, i, 0)$ is incompatible with $h$, where $\pi' \cdot 0$ is a prefix of $\pi$.

Suppose that $e_i \in E(\pi)$. Then $m(u) = u$ and $\pi' \cdot 1$ is a prefix of $\pi$. We have $u \notin V^{\geq e_{i+1}}$ and $\{m(u), m(u')\} \notin \bigcup h$. Hence $(m, i, 1)$ is incompatible with $h$.

(Case 2.1) Suppose that $u \in \bigcup h$ has degree $0$ in $E(\pi)$. Let $e_i = \max E_u$ and $\pi'$ be the prefix of length $i-1$ of $\pi$, where $\pi' \cdot 0$ is a prefix of $\pi$. Then $u \in \bigcup h \setminus V^{\geq e_{i+1}}$ and $\mathrm{mate}_{\pi'}(u) = u$, which means that $(\mathrm{mate}_{\pi'}, i, 0)$ is incompatible with $h$,

(Case 2.2) Suppose that $u \in \bigcup h$ has degree $2$ in $E(\pi)$. Let $e_i = \max\{\, \{u, u'\} \in E(\pi) \mid u' \in V \,\}$, $\pi'$ be the prefix of length $i-1$ of $\pi$, where $\pi' \cdot 1$ is a prefix of $\pi$. Then $u \in \bigcup h$ and $\mathrm{mate}_{\pi'}(u) \neq u$, which means that $(\mathrm{mate}_{\pi'}, i, 1)$ is incompatible with $h$. □

**Corollary 16.** *If $E(\pi)$ is not a solution for $(G, h)$ and $|\pi| = |E|$, then $\pi$ has a prefix $\pi'$ which ends in* **0** *in $Z_{G,h}$.*

*Proof.* By Lemmas 11 and 15. □

## C. Correctness of Algorithm 3 (Numberlink Instance Enumeration)

In this section, for a node $n$ of $Z_G$, by $S_n$ and $B_n$ we denote the pair matching families assigned to $n$ by Algorithm 3. For each $0,1$-sequence $\pi$ such that $E(\pi)$ is a path matching, let us define

$$T(\pi) = \{\, \{u, v\} \subseteq V \mid E(\pi) \text{ contains a } u\text{-}v \text{ path with } u, v \notin V^{\geq e_{|\pi|+1}} \,\}$$

**Lemma 17.** *Suppose that $E(\pi)$ is a path matching and let $i = |\pi| + 1$ and $m = \mathrm{mate}_\pi$. We have*

$$T(\pi \cdot 0) = T(\pi) \cup \{\, \{u, m(u)\} \subseteq V \mid u \in e_i,\ m(u) \notin \{u, 0\} \text{ and } u, m(u) \notin V^{\geq e_{i+1}} \,\}$$

*If $E(\pi \cdot 1)$ is a path matching, then*

$$T(\pi \cdot 1) = T(\pi) \cup \{\, \{m(u), m(v)\} \subseteq V \mid \{u, v\} = e_i \text{ and } m(u), m(v) \notin V^{\geq e_{i+1}} \,\}$$

*Proof.* By definition,

$$\begin{aligned} T(\pi \cdot 0) &= \{\, \{u, v\} \subseteq V \mid E(\pi \cdot 0) \text{ contains a } u\text{-}v \text{ path with } u, v \notin V^{\geq e_{i+1}} \,\} \\ &= \{\, \{u, v\} \subseteq V \mid E(\pi) \text{ contains a } u\text{-}v \text{ path with } u, v \notin V^{\geq e_i} \,\} \\ &\quad \cup \{\, \{u, v\} \subseteq V \mid E(\pi) \text{ contains a } u\text{-}v \text{ path with } u \in e_i \setminus V^{\geq e_{i+1}} \text{ and } v \notin V^{\geq e_{i+1}} \,\} \\ &= T(\pi) \cup \{\, \{u, m(u)\} \subseteq V \mid u \in e_i \setminus V^{\geq e_{i+1}},\ m(u) \notin V^{\geq e_{i+1}} \text{ and } m(u) \notin \{u, 0\} \,\} \end{aligned}$$

and

$$\begin{aligned} T(\pi \cdot 1) &= \{\, \{u, v\} \subseteq V \mid E(\pi \cdot 1) \text{ contains a } u\text{-}v \text{ path with } u, v \notin V^{\geq e_{i+1}} \,\} \\ &= \{\, \{u, v\} \subseteq V \mid E(\pi) \text{ contains a } u\text{-}v \text{ path with } u, v \notin V^{\geq e_i} \,\} \\ &\quad \cup \{\, \{u, v\} \subseteq V \mid E(\pi \cdot 1) \text{ contains a } u\text{-}v \text{ path } P \text{ with } e_i \in P \text{ and } u, v \notin V^{\geq e_{i+1}} \,\} \\ &= T(\pi) \cup \{\, \{u, v\} \subseteq V \mid \{u', v'\} = e_i,\ u = m(u') \notin V^{\geq e_{i+1}} \text{ and } v = m(v') \notin V^{\geq e_{i+1}} \,\} \quad \square \end{aligned}$$

**Lemma 18.** *For each node $n$ of $Z_G$, the algorithm assigns $S_n = \{\, T(\pi) \mid n(\pi) = n \,\}$. Moreover, $S_{n,b} = \{\, T(\pi \cdot b) \mid n(\pi) = n \,\}$ for each $b \in \{0, 1\}$.*

*Proof.* We show the lemma by induction on the length of $\pi$. If $\pi$ is empty, then $T(\pi) = \emptyset$. Indeed Algorithm 3 sets $S_{n_{\mathrm{root}}} = \{\emptyset\}$.

Let $\pi = \pi' \cdot b$ for some $\pi'$ and $b \in \{0, 1\}$. By the induction hypothesis, we have $T(\pi') \in S_{n(\pi')}$. By Lemma 17, we have

$$T(\pi' \cdot b) = T(\pi') \cup \mathrm{Fix}(\mathrm{mate}_{n(\pi')}, |\pi'| + 1, b) \in S_{n(\pi'),b} \subseteq S_{n(\pi' \cdot b)}$$

We have proven that $\{\, T(\pi \cdot b) \mid n(\pi) = n \,\} \subseteq S_{n,b}$. and $\{\, T(\pi) \mid n(\pi) = n \,\} \subseteq S_n$.

Conversely, we show by induction on $i$ that if $T \in S_n$ for $n \in N_i$, then there is $\pi$ such that $n(\pi) = n$ and $T(\pi) = T$. For $i = 1$, $S_n = \{\emptyset\}$ and the empty path $\epsilon$ satisfies the claim. Suppose that $T \in S_n$ for some $n \in N_{i+1}$ with $i \geq 1$. Then by definition there is a $b$-parent $n' \in N_i$ of $n$ for some $b \in \{0, 1\}$ such that $T \in S_{n',b}$. By definition, there is $T' \in S_{n'}$ such that

$$T = T' \cup \mathrm{Fix}(\mathrm{mate}_{n'}, i, b)$$

By the induction hypothesis, there is $\pi'$ such that $n(\pi') = n'$ and $T' = T(\pi')$. Lemma 17 ensures that $T(\pi' \cdot b) = T$. This proves that $S_{n,b} \subseteq \{\, T(\pi \cdot b) \mid n(\pi) = n \,\}$ and $S_n \subseteq \{\, T(\pi) \mid n(\pi) = n \,\}$. □

**Corollary 19.** *$(G, h)$ admits a solution if and only if $h \in S_1$.*

*Proof.* By Lemma 18. □

**Lemma 20.** *If $T(\pi) \in B_n$ for a $0, 1$-sequence $\pi$, then $T(\pi \cdot b) \in B_{n,b}$ for each $b \in \{0, 1\}$.*

*Proof.* Similarly to the proof of Lemma 18. □

**Lemma 21.** *If there are distinct valid paths $\pi_0$ and $\pi_1$ such that $n(\pi_0) = n(\pi_1)$ and $T(\pi_0) = T(\pi_1)$, then $T(\pi_0) \in B_{n(\pi_0)}$.*

*Proof.* We prove the lemma by induction on $i = |\pi_0| = |\pi_1|$. For $i = 0$, the claim holds since only the empty path ends in the root node.

Let $\pi_0 = \pi'_0 \cdot b_0$ and $\pi_1 = \pi'_1 \cdot b_1$ for some $b_0, b_1 \in \{0, 1\}$.

(Case 1) $n(\pi'_0) = n(\pi'_1)$. In this case, we have $\mathrm{mate}_{\pi'_0} = \mathrm{mate}_{\pi'_1}$ and $\mathrm{mate}_{\pi'_0 \cdot b_0} = \mathrm{mate}_{\pi'_1 \cdot b_1}$. Let $m' = \mathrm{mate}_{\pi'_0}$ and $m = \mathrm{mate}_{\pi_0}$. Clearly each of $u, v \in e_i$ has the same degree in $E(\pi'_0)$ and in $E(\pi'_1)$, since $u$ and $v$ are in the domain $V^{\geq e_i}$ of $\mathrm{mate}_{\pi'_0} = \mathrm{mate}_{\pi'_1}$. To derive a contradiction, suppose that $b_0 \neq b_1$, where we assume without loss of generality that $b_0 = 0$ and $b_1 = 1$. Then $u$ has different degrees in $E(\pi_0)$ and in $E(\pi_1)$. In this case, the fact that $\mathrm{mate}_{\pi'_0 \cdot b_0} = \mathrm{mate}_{\pi'_1 \cdot b_1}$ implies that $u$ is not in the domain $V^{\geq e_{i+1}}$ of those mate functions. The same argument applies to $v$. Hence, $\{m'(u), m'(v)\} \in T(\pi_1)$ by definition. The fact that $E(\pi_1)$ is a path matching implies $m'(u) \neq v$, which means that $E(\pi'_0)$ does not contain a $u$-$v$ path. Hence $E(\pi_0)$ does not have a $m'(u)$-$m'(v)$ path and $\{m'(u), m'(v)\} \notin T(\pi_0)$. This contradicts to the assumption that $T(\pi_0) = T(\pi_1)$. Hence we conclude that $b_0 = b_1$. By Lemma 17, we have

$$T(\pi_0) \setminus T(\pi'_0) = T(\pi_1) \setminus T(\pi'_1)$$

Together with $T(\pi'_0) \subseteq T(\pi_0)$, $T(\pi'_1) \subseteq T(\pi_1)$ and $T(\pi_0) = T(\pi_1)$, we conclude that $T(\pi'_0) = T(\pi'_1)$. By $\pi_0 = \pi'_0 \cdot b_0 \neq \pi'_1 \cdot b_0 = \pi_1$, we have $\pi'_0 \neq \pi'_1$. By the induction hypothesis, $T(\pi'_0) = T(\pi'_1) \in B_{n'}$. Thus by Lemma 20, $T(\pi_0) = T(\pi'_0 \cdot b_0) \in B_{n',b_0} \subseteq B_n$.

(Case 2) $n(\pi'_0) \neq n(\pi'_1)$. By Lemma 18, we have $T(\pi'_0 \cdot b_0) \in S_{n(\pi'_0),b_0}$ and $T(\pi'_1 \cdot b_1) \in S_{n(\pi'_1),b_1}$. Hence $T(\pi_0) = T(\pi'_0 \cdot b_0) = T(\pi'_1 \cdot b_1) \in B_n$. □

**Lemma 22.** *For any valid path $\pi$ such that $\bigcup E(\pi) \cup V^{\geq e_{|\pi|+1}} \subsetneq V$, it holds that $T(\pi) \in B_{n(\pi)}$.*

*Proof.* We prove the lemma by induction on $i = |\pi|$. For $i = 0$, the claim holds trivially by $V^{\geq e_1} = V$. Let $\pi = \pi' \cdot b$ for some $b \in \{0, 1\}$.

(Case 1) $e_i \subseteq \bigcup E(\pi) \cup V^{\geq e_{i+1}}$. We have

$$\bigcup E(\pi') \cup V^{\geq e_i} \subseteq \bigcup E(\pi) \cup V^{\geq e_i} = \bigcup E(\pi) \cup V^{\geq e_{i+1}} \cup \{e_i\} = \bigcup E(\pi) \cup V^{\geq e_{i+1}} \subsetneq V$$

By the induction hypothesis, we have $T(\pi') \in B_{n(\pi')}$. Hence $T(\pi) = T(\pi' \cdot b) \in B_{n(\pi'),b} \subseteq B_{n(\pi)}$.

(Case 2) $e_i \nsubseteq \bigcup E(\pi) \cup V^{\geq e_{i+1}}$. The fact $e_i \notin E(\pi)$ implies that $b = 0$. Let $u \in e_i \setminus (\bigcup E(\pi) \cup V^{\geq e_{|\pi|+1}})$. We have $\mathrm{mate}_{\pi'}(u) = u$. By Lemma 18 and the fact $u \notin V^{\geq e_{|\pi|+1}}$, we have

$$T(\pi) = T(\pi' \cdot 0) \in S_{n(\pi'),0} \subseteq B_{n(\pi)}$$ □

**Lemma 23.** *If $T \in B_n$ for some node in $Z_G$, then one of the following holds:*

- *there are distinct* $\pi_1$ *and* $\pi_2$ *such that* $n = n(\pi_1) = n(\pi_2)$ *and* $T = T(\pi_1) = T(\pi_2)$*,*
- *there is* $\pi$ *such that* $n = n(\pi)$*,* $T = T(\pi)$ *and* $\bigcup E(\pi) \cup V^{\geq e_{|\pi|+1}} \subsetneq V$*.*

*Proof.* We prove the lemma by induction on $i$ such that $n \in N_i$. For $i = 0$, the claim holds trivially by $B_{n_{\text{root}}} = \emptyset$.

For $i \geq 1$, there are three cases according to the definition of $B_n$ in the algorithm.

(Case 1) $T \in S_{n_1,b_1} \cap S_{n_2,b_2}$ for some $b_j$-parents $n_j$ of $n$ for $j = 1, 2$ such that $n_1 \neq n_2$. By Lemma 18, there are $\pi_j$ such that $T = T(\pi_j \cdot b_j) \in S_{n_j,b_j}$ and $T(\pi_j) \in S_{n_j}$ for $j = 1, 2$. The fact $n_1 \neq n_2$ implies that $\pi_1 \cdot b_1 \neq \pi_2 \cdot b_2$. We have $n = n(\pi_1 \cdot b_1) = n(\pi_2 \cdot b_2)$ and $T = T(\pi_1) = T(\pi_2)$.

(Case 2) $T \in S_{n',0}$ for some 0-parent $n'$ of $n$ such that $\text{mate}_{n'}(u) = u$ for some $u \in e_i \setminus V^{\geq e_{i+1}}$. By Lemma 18, there is $\pi'$ such that $T = T(\pi' \cdot 0) \in S_{n',0}$ and $T(\pi') \in S_{n'}$. The fact that $\text{mate}_{n'}(u) = u$ implies that the degree of $u$ is 0 in $E(\pi') = E(\pi' \cdot 0)$. Let $\pi = \pi' \cdot 0$. Then, we have $n = n(\pi)$, $T = T(\pi)$ and $u \notin \bigcup E(\pi) \cup V^{\geq e_{|\pi|+1}}$.

(Case 3) $T \in B_{n',b}$ for some $b$-parent $n'$ of $n$. There is $T' \in B_{n'}$ such that $T = T' \cup \text{Fix}(\text{mate}_{n'}, i, b)$. By the induction hypothesis on the fact $T' \in B_{n'}$, one of the following holds:

- there are distinct $\pi'_1$ and $\pi'_2$ such that $n' = n(\pi'_1) = n(\pi'_2)$ and $T' = T(\pi'_1) = T(\pi'_2)$,
- there is $\pi'$ such that $n' = n(\pi')$, $T' = T(\pi')$ and $\bigcup E(\pi') \cup V^{\geq e_{|\pi|}} \subsetneq V$.

In the former case, $\pi_j = \pi'_j \cdot b$ for $j = 1, 2$ satisfies the lemma. In the latter case, let $u \in V \setminus (\bigcup E(\pi') \cup V^{\geq e_{|\pi|}})$. By $e_i \subseteq V^{\geq e_{|\pi|}}$, we have $u \notin E(\pi' \cdot b)$. Thus we have $u \notin E(\pi) \cup V^{\geq e_{|\pi|+1}}$ for $\pi = \pi' \cdot b$. □

## D. Correctness of Algorithm 4 (Cycle Enumeration)

Let $Y_G$ be the output by Algorithm 4 for $G$. The following lemma, which we will use without explicit reference, is implied by the proof of Lemma 8.

**Lemma 24.** *If* $\pi$ *is a valid path in* $Y_G$ *and* $n(\pi) \neq \mathbf{1}$*, then* $E(\pi)$ *is a path matching and* $\text{mate}_{n(\pi)} = \text{mate}_\pi$.

For $Y_G$, we have the following stronger lemma.

**Lemma 25.** *Let* $\pi$ *be a* $0, 1$*-sequence of length at most* $|E|$*. If* $E(\pi)$ *is a path matching such that every vertex in* $V \setminus V^{\geq e_{|\pi|+1}}$ *has degree* $0$ *or* $2$ *in* $E(\pi)$*, then* $\pi$ *is a valid path ending in a node* $n \neq \mathbf{0}, \mathbf{1}$ *in the constructed ZDD and we have* $\text{mate}_n = \text{mate}_\pi$.

*Proof.* We prove the lemma by induction on the length of $\pi$. Clearly the lemma holds if $\pi$ is the empty sequence. Suppose that $\pi$ of length $i - 1$ satisfies the condition of the lemma. Let $n = n(\pi)$. It is enough to show the following by Lemma 24.

- If $(\text{mate}_n, e_i)$ has a fixed end, then there is a vertex in $V \setminus V^{\geq e_{i+1}}$ of degree 1 in $E(\pi \cdot 0)$.
- If $E(\pi \cdot 1)$ is a path matching and $(\text{MU}(\text{mate}_n, e_i), e_i)$ has a fixed end, then there is a vertex in $V \setminus V^{\geq e_{i+1}}$ of degree 1 in $E(\pi \cdot 1)$.

Suppose that $(\text{mate}_n, e_i)$ has a fixed end. By definition, there is $v \in e_i \setminus V^{\geq e_{i+1}}$ with $\text{mate}_n(v) \notin \{0, v\}$, which means that $v$ has degree 1 in $E(\pi \cdot 0)$.

Suppose that $E(\pi \cdot 1)$ is a path matching and $(\mathrm{MU}(\mathrm{mate}_n, e_i), e_i)$ has a fixed end. By definition, there is $v \in e_i \setminus V^{\geq e_{i+1}}$ with $\mathrm{MU}(\mathrm{mate}_n(v), e_i)(v) \notin \{0, v\}$, which means that $v$ has degree $1$ in $E(\pi \cdot 1)$ by the arguments on the function MU in the proof of Lemma 8. □

**Corollary 26.** *Let $\pi$ be a $0,1$-sequence of length $i \leq |E|$. If $E(\pi \cdot 1)$ is a simple cycle, then $\pi \cdot 1$ ends in $\mathbf{1}$.*

*Proof.* Suppose that $E(\pi \cdot 1)$ is a cycle. Let $i = |\pi| + 1$ and $e_i = \{u, v\}$. That is, $E(\pi \cdot 1) = E(\pi) \cup \{\{u, v\}\}$. Since $E(\pi)$ is a $u$-$v$ simple path, the only vertices of degree 1 in $E(\pi)$ are $u$ and $v$. Since $u, v \in V^{\geq e_i}$, Lemma 25 implies that $\pi$ ends in a node $n \neq \mathbf{0}, \mathbf{1}$. We have $\mathrm{mate}_n(u) = v$, $\mathrm{mate}_n(v) = u$ and $\mathrm{mate}_n(w) \in \{0, w\}$ for other $w \in V^{\geq e_i}$. That is, $\mathrm{mate}_n$ and $e_i$ form a cycle. The algorithm connects $n$ and $\mathbf{1}$ by a 1-arc. □

**Lemma 27.** *Let $\pi$ be a $0,1$-sequence of length $i \leq |E|$. If $\pi$ ends in a node $n \notin \{\mathbf{0}, \mathbf{1}\}$, then $E(\pi)$ is a path matching such that $\mathrm{mate}_\pi = \mathrm{mate}_n$ and every vertex in $\bigcup E(\pi) \setminus V^{\geq e_{i+1}}$ has degree $0$ or $2$ in $E(\pi)$.*

*Proof.* We show the lemma by induction on $i$. A proof almost identical to that for Lemma 9 applies to the claim that if $\pi$ ends in a node $n \notin \{\mathbf{0}, \mathbf{1}\}$, then $E(\pi)$ is a path matching such that $\mathrm{mate}_\pi = \mathrm{mate}_n$. We here show that in that case every vertex in $V \setminus V^{\geq e_{i+1}}$ has degree $0$ or $2$ in $E(\pi)$.

If $\pi$ is empty, then $V \setminus V^{\geq e_{i+1}}$ is the empty set.

Suppose that $\pi = \pi' \cdot 0$ for some $\pi'$. We have $E(\pi) = E(\pi')$. By the induction hypothesis, every vertex in $V \setminus V^{\geq e_i}$ has degree $0$ or $2$ in $E(\pi')$ where $i = |\pi|$. If $V^{\geq e_{i+1}} = V^{\geq e_i}$, there is nothing to prove. If there is $u \in V^{\geq e_i} \setminus V^{\geq e_{i+1}}$, then $u \in e_i$. By definition, $(\mathrm{mate}_{n(\pi')}, i)$ does not have a fixed end, *i.e.*, $\mathrm{mate}_{\pi'}(u) = \mathrm{mate}_{n(\pi')}(u) \in \{0, u\}$. That is, $u$ has degree $0$ or $2$.

The case where $\pi = \pi' \cdot 1$ for some $\pi'$ can be proven by combining the discussions for the case where $\pi = \pi' \cdot 0$ and the correctness of the mate updating function MU, established in the proof of Lemma 8. □

**Corollary 28.** *If a $0,1$-sequence $\pi$ ends in $\mathbf{1}$, then $E(\pi)$ is a simple cycle.*

*Proof.* By the algorithm, $\pi$ must have the form $\pi' \cdot 1$ and $\mathrm{mate}_{n(\pi')}$ and $e_i$ form a cycle for $i = |\pi|$. That is, we have

$$\mathrm{mate}_{n(\pi')}(v) = \begin{cases} u & \text{if } e_i = \{u, v\} \\ v \text{ or } 0 & \text{if } v \notin e_i \end{cases}$$

Applying Lemma 27 to $\pi'$, we conclude that $E(\pi')$ is a path matching consisting solely of a $u$-$v$ path for $e_i = \{u, v\}$. Clearly $E(\pi) = E(\pi') \cup \{e_i\}$ is a simple cycle. □

**Theorem 29.** *Algorithm 4 constructs a ZDD that represents all the simple cycles on $G$.*

*Proof.* By Corollaries 26 and 28. □

## E. Correctness of Algorithm 5 (Slitherlink Solver)

Let $Y_{G,h}$ be the output by Algorithm 5 for $(G, h)$.

**Lemma 30.** *If a $0,1$-sequence $\pi$ ends in $\mathbf{1}$ in $Y_{G,h}$, then $E(\pi)$ is a solution for $(G, h)$.*

*Proof.* By Lemma 27, If $\pi$ ends in a node $n \notin \{\mathbf{0}, \mathbf{1}\}$, then $E(\pi)$ is a path matching such that every vertex in $V \setminus V^{\geq e_{i+1}}$ has degree 0 or 2 in $E(\pi)$ and $\mathrm{mate}_\pi = \mathrm{mate}_n$. Moreover, it is easy to show by induction on $i$ that $|E(\pi) \cap D| = \mathrm{count}_n(D)$ for all $D \in \mathrm{dom}(h)$.

If $\pi$ ends in $\mathbf{1}$, by definition of GN, $E(\pi)$ is a cycle such that $|E(\pi) \cap D| = h(D)$ for all $D \in \mathrm{dom}(h)$. That is, $E(\pi)$ is a solution for $(G, h)$. □

**Lemma 31.** *If a* $0, 1$*-sequence* $\pi$ *of length* $i$ *ends in* $\mathbf{0}$ *in* $Y_{G,h}$*, then* $E(\pi) \cup E'$ *is not a solution for* $(G, h)$ *for any* $E' \subseteq E^{\geq e_{i+1}}$.

*Proof.* Suppose that $\pi = \pi' \cdot b$ ends in $\mathbf{0}$ where $b \in \{0, 1\}$. Let $n = n(\pi')$, $i = |\pi|$ and

$$c(D) = \begin{cases} \mathrm{count}_n(D) & \text{if } b = 0 \\ \mathrm{CU}(\mathrm{count}_n, e_i)(D) & \text{if } b = 1 \end{cases}$$

for all $D \in \mathrm{dom}(h)$. The proof of Lemma 25 shows that if $(\mathrm{mate}_n, i)$ has a fixed end or $\mathrm{mate}_n$ declines $e_i$, then the lemma holds. It is enough to discuss the case where $(c, i)$ is incompatible with $h$. In this case, either $c(D) > h(D)$ or $c(D) + |D \cap E^{\geq e_{i+1}}| < h(D)$ for some $D \in \mathrm{dom}(h)$. In the former case, $|E(\pi) \cap D| = c(D) > h(D)$ implies that $|(E(\pi) \cup E') \cap D| > h(D)$ for any $E' \subseteq E^{\geq e_{i+1}}$. That is, $E(\pi) \cup E'$ cannot be a solution. In the latter case,

$$|E(\pi) \cap D| + |D \cap E^{\geq e_{i+1}}| = c(D) + |D \cap E^{\geq e_{i+1}}| < h(D)$$

implies that $|(E(\pi) \cup E') \cap D| < h(D)$ for any $E' \subseteq E^{\geq e_{i+1}}$. That is, $E(\pi) \cup E'$ cannot be a solution. □

**Corollary 32.** $Y_{G,h}$ *represents the set of all solutions for* $(G, h)$.

*Proof.* By Lemmas 30 and 31. □

## F. Computation Example for Numberlink Instance Enumeration

We explain an example of $T(\pi)$ and $S_n$ for some valid path $\pi$ and some node $n$ defined in Section 5 (see Figure 12).

**Figure 12.** An example of computing $T(\pi)$ and $S_n$.

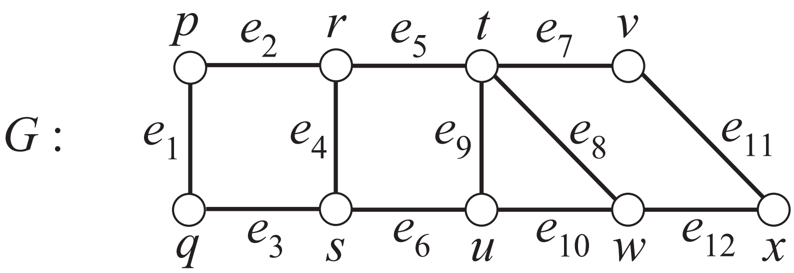
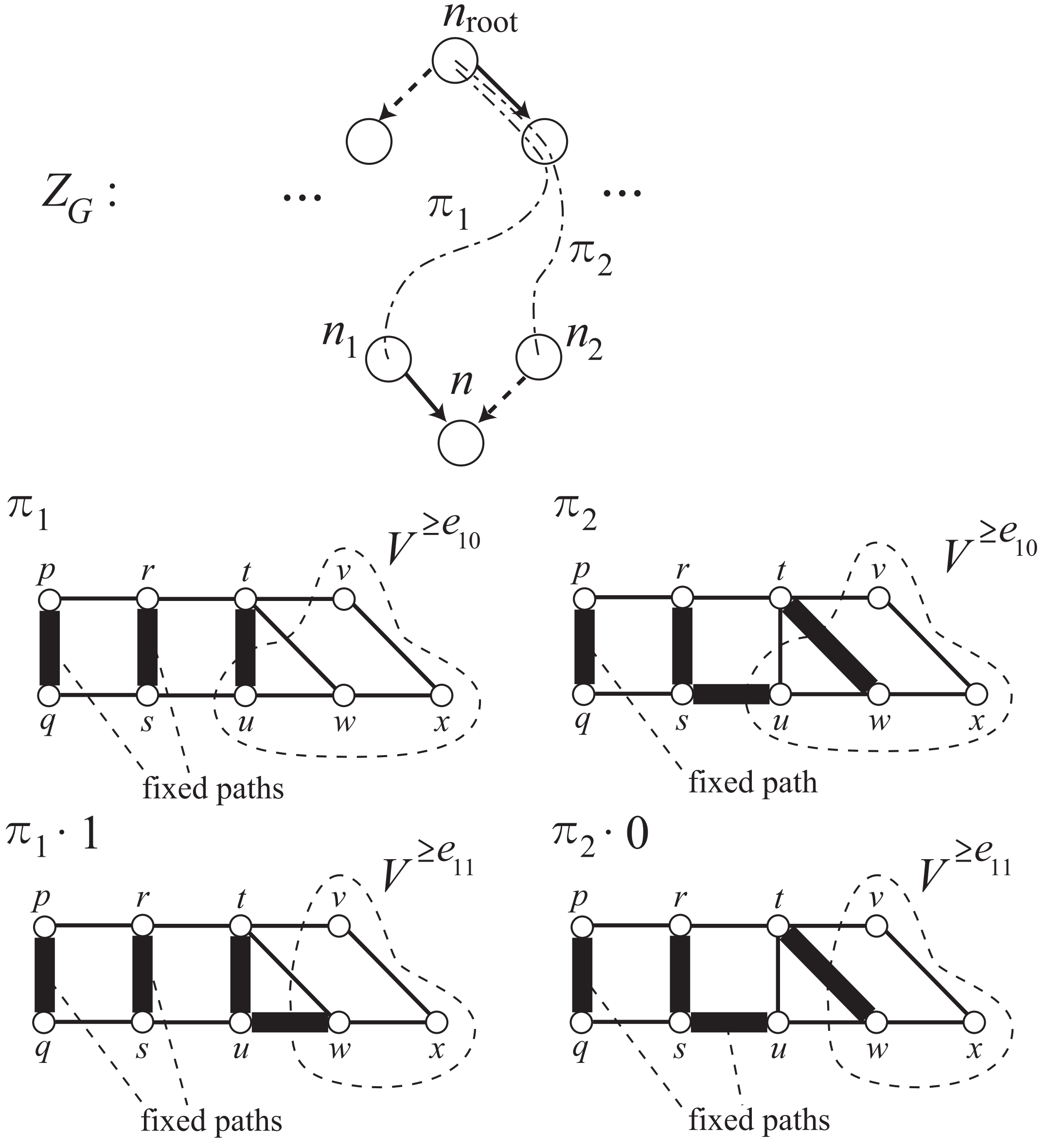


**Figure 12.** *Cont.*



A given graph $G = (V, E)$ and $Z_G$ are shown in the top half of the figure. The dashed lines represent valid paths $\pi_1 = 100100001$ and $\pi_2 = 100101010$. The four graphs in the bottom half of the figure are subgraphs whose vertex sets are $V$ and whose edge sets are $E(\pi_1)$, $E(\pi_2)$, $E(\pi_1 \cdot 1)$ and $E(\pi_2 \cdot 0)$, which are drawn as bold lines. Since $\mathrm{mate}_{n(\pi_1\cdot 1)}(v) = \mathrm{mate}_{n(\pi_2\cdot 0)}(v) = v$, $\mathrm{mate}_{n(\pi_1\cdot 1)}(w) = \mathrm{mate}_{n(\pi_2\cdot 0)}(w) = t$ and $\mathrm{mate}_{n(\pi_1\cdot 1)}(x) = \mathrm{mate}_{n(\pi_2\cdot 0)}(x) = x$ and for any other vertices $u$, both $\mathrm{mate}_{n(\pi_1\cdot 1)}(u)$ and $\mathrm{mate}_{n(\pi_2\cdot 0)}(u)$ are undefined, $\mathrm{mate}_{n(\pi_1\cdot 1)} = \mathrm{mate}_{n(\pi_2\cdot 0)}$ holds. Therefore, $n(\pi_1 \cdot 1) = n(\pi_2 \cdot 0)$. Let $n = n(\pi_1 \cdot 1) = n(\pi_2 \cdot 0)$. Let $P_1 = \{\{p, q\}\}$, $P_2 = \{\{r, s\}\}$ and $P_3 = \{\{r, s\}, \{s, u\}\}$. $E(\pi_1)$, $E(\pi_2)$, $E(\pi_1 \cdot 1)$ and $E(\pi_2 \cdot 0)$ include two fixed paths $P_1$ and $P_2$, one fixed path $P_1$, two fixed paths $P_1$ and $P_2$, and two fixed paths $P_1$ and $P_3$ respectively. It is easy to verify that $T(\pi_1) = \{\{p, q\}, \{r, s\}\}$, $T(\pi_2) = \{\{p, q\}\}$, and $S_n = \{T(\pi_1 \cdot 1), T(\pi_2 \cdot 0)\} = \{\{\{p, q\}, \{r, s\}\}, \{\{p, q\}, \{r, u\}\}\}$ hold.

## G. Minimum Good Numberlink Instances over $G_{6,6}$

Figures 13 and 14 complete all of the 38 good instances of Numberlink $(G_{6,6}, h)$ with $|h| = 3$ modulo symmetry.

**Figure 13.** Good instances on $G_{6,6}$ (1).

| | | | | | |
|---|---|---|---|---|---|
| 1 | | | | | |
| | | | | | |
| | | | | | |
| | | 1 | 2 | | |
| | 2 | | 3 | | |
| | 3 | | | | |

| | | | | | |
|---|---|---|---|---|---|
| 1 | | | | | |
| | | | | | |
| | | | | | |
| | | 1 | | | |
| | | | 3 | 2 | |
| 2 | 3 | | | | |

| | | | | | |
|---|---|---|---|---|---|
| 1 | | | | | |
| | | | | | |
| | | | | | |
| | | 2 | | | |
| | 1 | | | 3 | |
| | 3 | 2 | | | |

| | | | | | |
|---|---|---|---|---|---|
| 1 | | | | | |
| | | | | | |
| | | | 2 | | |
| | | | | | |
| | | | 1 | | |
| 3 | 2 | | | | 3 |

| | | | | | |
|---|---|---|---|---|---|
| 1 | | | | | |
| | | | | | |
| | | | 3 | | |
| | 2 | | | | |
| | | | | | 2 |
| 3 | | | | | 1 |

| | | | | | |
|---|---|---|---|---|---|
| 1 | | | | | |
| | | | | | |
| | | | 2 | | |
| | | | | | |
| | 3 | | | | 3 |
| | | | 2 | | 1 |

| | | | | | |
|---|---|---|---|---|---|
| 1 | | | | | |
| | | | | | |
| | | | 2 | | |
| | | | | | 3 |
| | 3 | | | | |
| | | | | 2 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| 1 | | | | | |
| | | | | | |
| | | | | 3 | |
| | | 2 | | | |
| | | | | | 2 |
| 3 | | | | | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 2 |
| | 1 | | | | |
| | | | | | |
| | | | 3 | | |
| | 2 | | | | |
| 3 | 1 | | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 2 |
| | 1 | | | | |
| | | | | | |
| | | | 3 | | |
| | 2 | | | | |
| | 3 | 1 | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 2 |
| | 1 | | | | |
| | | | | | |
| | | | 3 | | |
| | | | | | |
| 3 | 2 | 1 | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | 1 | | | | |
| | | | | | |
| | | 2 | | | |
| | | | | | |
| 2 | 3 | 1 | | | 3 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 2 |
| | | | | | |
| | 1 | | | | |
| | | | 3 | | |
| | | | | | |
| 3 | 2 | | | | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 2 |
| | | | 1 | | |
| | | | | | |
| | | | 2 | | |
| | | | | | |
| 3 | | | | 1 | 3 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 2 |
| | | | 1 | | |
| | | | | | |
| | | | 2 | | |
| | | | | | 3 |
| 3 | | | | | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 2 |
| | | | | | |
| | | 1 | | | |
| | | | | | |
| | 2 | | | 3 | |
| 3 | 1 | | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 2 |
| | | | | | |
| | | 1 | | | |
| | | | | | |
| | 2 | | | 3 | |
| | 3 | 1 | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 2 |
| | | | | | |
| | | 1 | | | |
| | | | | | |
| | | | | 3 | |
| 3 | 2 | 1 | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | 1 | | | |
| | | | | | |
| | 2 | | | | 2 |
| 3 | | | 1 | | 3 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | 1 | | | |
| | | | | | 2 |
| | 2 | | | 3 | 1 |
| 3 | | | | | |

**Figure 14.** Good instances on $G_{6,6}$ (2).

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | 1 | | | |
| | | | | | 2 |
| | 2 | | | | |
| 3 | | | | 1 | 3 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 2 |
| | | | | | |
| | | 1 | | | |
| | | | | | |
| | | 2 | | | 3 |
| 3 | | | | | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 2 |
| | | | | | |
| | | 1 | | | |
| | | | | 3 | |
| | | | | | |
| 3 | 2 | | | | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | 1 | | | |
| | | | | | 2 |
| | 2 | | | | 3 |
| 3 | | | | | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | 1 | | | |
| | | | | | |
| | 2 | | | 3 | 2 |
| 3 | | | | | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 1 |
| | | | | | |
| | | | | | |
| | | | 1 | | |
| | 2 | 3 | | | 2 |
| | | | | | 3 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 1 |
| | | | | | |
| | | | | | |
| | | | 3 | | |
| | 2 | | | 1 | |
| | | | | 3 | 2 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 1 |
| | | | | | |
| | | | | | |
| | | | 3 | | |
| | 2 | | | | |
| | | | 3 | 1 | 2 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 1 |
| | | | | | |
| | | | | | |
| | | | 3 | | 2 |
| | 2 | | | | |
| | | | | 3 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 1 |
| | | | | | |
| | | | | | |
| | | | 3 | | |
| | 2 | | | | 2 |
| | | | 3 | | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | 1 | |
| | | | 2 | 3 | |
| | | | 1 | | |
| | | | | 2 | |
| 3 | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | 1 | |
| | | | | | |
| | | | 3 | | |
| | | | | 2 | |
| 2 | | | 1 | 3 | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | 1 | |
| | | | | | |
| | | | 3 | | |
| | | | | 2 | |
| 2 | | | | 1 | 3 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | 1 | | |
| | | | | | |
| | 2 | | | 3 | |
| 3 | 1 | | | | 2 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | 1 | | |
| | | | | | |
| | 2 | | | 3 | |
| | 3 | 1 | | | 2 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | 1 | | |
| | | | | | |
| | | | | 2 | |
| 2 | 3 | 1 | | | 3 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | 1 | | |
| | | | 3 | 2 | |
| | 2 | | | 1 | |
| | | | | | 3 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | 1 | | |
| | | | 3 | | |
| | | | | 2 | |
| 2 | | 3 | | | 1 |