Abstract

This paper addresses the problem of finding a class of representative itemsets up to subitemset isomorphism. An efficient algorithm is of practical importance in the domain of optimal sorting networks. Although only exponential algorithms for solving the problem exist in the literature, the complexity classification of the problem has never been addressed. In this paper, we present a complexity classification of the itemset isomorphism and subitemset isomorphism problems. We prove that the problem of checking whether two itemsets are isomorphic to each other is GI-Hard; the Graph Isomorphism (GI) problem is known to be in NP and LWPP, but widely believed to be neither P nor NP-Complete. As an immediate consequence, we prove that finding a class representative itemsets up to subitemset isomorphism is GI-Hard — at least as hard as the graph isomorphism problem.

1 Introduction

A sorting network is a mathematical object consisting of \(n\) wires and comparators. Sorting networks are oblivious to the input data and always perform the same set of pre-determined operations to produce a sorted list of numbers. The problem of finding optimal sorting networks was first proposed [1] by Bose and Nelson more than 50 years ago. There are two common measures for the optimality of a sorting network — number of levels (depth) and number of comparators.

The problem studied in this paper is central [2] [3] [4] [5] to the optimal sorting network search problem. A comparator network \(C\) can be represented as an itemset \(S_C\) — also referred to as the outputs of a comparator network in the literature [2]. When searching for sorting networks of optimal depth, we need not consider the comparator network \(B\) if there exists a comparator network \(A\) of the same depth as \(B\) such that \(S_A \preceq B\); the subitemset isomorphism relation \(\preceq\) studied in this paper is formally described in Definition 1.3. The same property holds for networks consisting of same number of comparators, rather than same number of levels (depth). Hence, the problem of finding a class representative itemsets up to \(\preceq\) has a very important practical application in the sorting networks domain — we use it to reduce the search space in searching for optimal sorting networks.

In other words, if we consider the set \(G^n_3\) [2] [3] [5] of all \(n\)-input comparator networks consisting of exactly three levels then it is enough to consider only
the class representative $R^n_0$ of $G^n_3$ up to $\preceq$ in the optimal depth sorting network problem. Special cases of this problem have received attention in research. Parberry [7] shows that it is enough to consider exactly one prefix for a sorting network for the optimal depth problem; i.e. he proves $|R^n_0| = 1$. Codish et al. [5] analyse the second layer candidates of an optimal depth sorting network by providing a regular expression for the set $R^n_2$. Bundala and Zavodny [2] [3] [4] give an exponential algorithm of worst case complexity $O((|F|^2 \times n! \times 2^n))$ for finding a class representative up to $\preceq$ of a dataset of $F$ over a domain of $n$ elements.

For the minimal comparator sorting network problem, Codish et al. [6] present an exponential algorithm of worst case complexity $O((|F|^2 \times n! \times 2^n))$, although not stated explicitly, for finding a class representative up to $\preceq$ within a dataset $F$ over a domain of $n$ elements. Codish et al. present various search space pruning techniques to speedup the execution time. They apply the algorithm to present a computer-assisted proof for the minimal number of comparators of a nine and ten-input sorting networks.

There is no analysis on the complexity classification of the problem of finding a class representative up to $\preceq$ in the work of Bundala and Zavodny and Codish et al., although they all present exponential algorithms of worst case complexity $O((|F|^2 \times n! \times 2^n))$ for deterministically solving the problem. In this paper, we give a formal proof that the problem of finding class representative up to $\preceq$ is at least as hard as the Graph Isomorphism (GI) problem.

The graph isomorphism problem is one of two listed [8] by Garey and Johnson but yet to be classified as P or NP-Complete. Over the years, there is substantial research on the GI problem: fast practical algorithms with or without domain restrictions [9] [10] [11], complexity analysis [12] [13] [14], GI-Complete problems [15], etc. More importantly, it is commonly believed that GI-Complete problems form a uniquely defined complexity class that sits between P and NP-Complete, but this is yet to be proved.

The complexity classification proof presented in Section 2 is rather technical. Hence, foremost we need to rigorously define the mathematical objects and operations used throughout the proof.

1.1 The Problem

1.1.1 Objects

This section defines all of the mathematical objects that are used throughout this paper. Visual examples of all object types are presented in Figure [1]. Unless otherwise stated, we assume that we are working in the domain $D = \{d_1, d_2, \ldots, d_n\}$ of $n$ distinct elements.

An item over the domain $D$ is a set of elements. We represent an item $I$ as a binary string of length $n$ where the $i$-th bit is equal to 1 iff the element $d_i \in I$ for all $1 \leq i \leq n$; i.e. $I \in \{0, 1\}^n$. See Figure [1(a)] for examples of items.

An itemset over the domain $D$ is a set of items. We represent an itemset $S$ as a matrix with $|S|$ rows and $n$ columns over the field $\{0, 1\}$. See Figure [1(b)] for examples of itemsets.

A dataset over the domain $D$ is an ordered set of itemsets by cardinality in ascending order. See Figure [1(c)] for examples of datasets.
1.1.2 Operations

So far we have defined all of the objects in Section 1.1.1. Now we define the respective operations that we investigate in this paper.

**Definition 1.1.** Let $S$ and $T$ be itemsets over the domains $D_S$ and $D_T$, respectively. We say that $S$ is isomorphic to $T$ iff there exists a bijection $J : D_S \rightarrow D_T$ such that $J(S) = T$, also written as $S \cong T$; where $J(S) = \{\{J(d) \mid d \in I\} \mid I \in S\}$. If $D_S = D_T$ then we refer to $J$ as an automorphism.

**Remark 1.2.** Let $S$ and $T$ be itemsets over the domains $D_S$ and $D_T$ such that $S$ is isomorphic to $T$ given by $J(S) = T$. Since $S$ and $T$ are sets there exists a bijection $\sigma : S \rightarrow T$ which maps the items from $S$ to the items in $T$ such that $\forall s \in S$ we have $J(s) = \sigma(s)$.

**Definition 1.3.** Let $S$ and $T$ be itemsets over the same domain $D$. We say that $S$ is subset of $T$ up to isomorphism iff there exists a bijective $J : D \leftrightarrow D$ such that $J(S) \subseteq T$, also written as $S \preceq T$.

1.1.3 Problem Statement

Codish et al. [6] (Section 3) note that the relation $\preceq$ is an equivalence relation. Therefore, given a dataset $F$, the relation $\preceq$ partitions the set $F$ into equivalence classes. The real-world problem found in the sorting networks domain [2] [3] [4] [5] [6] is that of finding a class representative of $F$ up to $\preceq$. In this paper, we focus on the complexity classification of the problem of finding a class of representative itemsets of a given dataset $F$ up to $\preceq$.

1.1.4 Terminology

We have chosen the labels of the objects to match that of itemset mining algorithms [15] [17] [18] [19] because the extremal sets identification problem is a sub-problem of the main task. Codish et al [5] [6] describe a subset of our problem as ‘words up to permutations’ instead of the generalization of ‘subitemsets up to isomorphism’. We consider the choice of naming objects to be personal preference, because all that is important is the mathematical structure of the object that we work with, not the labels used. Hence, we are as rigorous as possible in this section, when it comes to defining the objects.

As the core problem is defined on itemsets over the same domain $D$ we refer to automorphism (same domains) existence between itemsets rather than isomorphism (different domains). When working with itemsets over the same domain $D = \{d_1, d_2, \ldots, d_n\}$ then an automorphism can be represented as a permutation of $n$ elements. Hence, when working in the same domain we interchange the terminology of automorphism and permutation; i.e. $A \preceq B$ can read as ‘there exists an automorphism $I : D \rightarrow D$ s.t. $I(A) \subseteq B$’ or ‘there exists a permutation $\pi \in \Pi_n$ s.t. $\pi(A) \subseteq B$’ or ‘$A$ is subset of $B$ up to permutation’.

1.2 Contributions

The main contributions of this work can be summarized as follows.
• **Itemset Isomorphism: GI-Hard** — In Section 2 we present a proof that the problem of itemset isomorphism (equality up to bijection of itemsets) is GI-Hard.

• **Subitemset Isomorphism: GI-Hard** — As an immediate consequence, the problem of finding a class representative itemsets up to subitemset isomorphism within a dataset is GI-Hard, that is at least as hard as GI. This problem has been encountered before in recent research [2] [3] [4] [6] [5] in the domain of sorting networks, but its complexity has never been classified.

## 2 Complexity Analysis

The problem of finding class representative up to \( \preceq \) is actively studied in recent research [2] [3] [4] [6] [5] but has never been classified into a complexity class. The core contribution of this paper, is to classify the problem of itemset isomorphism as GI-Hard. As an immediate corollary, the problem of finding class representative up to subitemset isomorphism within a dataset is GI-Hard. Having a proof of this classification is a major step in the domain of optimal sorting networks — the practical applications of the problem. Before we prove our main results — Theorem 2.6 and Corollary 2.7 — we must formally define the Graph Isomorphism (GI) decision problem [8] [15], the Itemset Isomorphism (II) decision problem and the Subitemset Isomorphism (SI) decision problem.

**Definition 2.1.** Graph Isomorphism (GI) decision problem:

**Input:** Two undirected graphs \( G = \langle V_G, E_G \rangle \) and \( H = \langle V_H, E_H \rangle \).

**Question:** Is there a bijection \( I : V_G \to V_H \) s.t. \((v, w) \in E_G \iff (I(v), I(w)) \in E_H\)?

**Definition 2.2.** Itemset Isomorphism (II) decision problem:

**Input:** Two itemsets \( S \) and \( T \) over the domains \( D_S \) and \( D_T \), respectively.

**Question:** Is there a bijection \( J : D_S \to D_T \) s.t. \( J(S) = T \)?

**Definition 2.3.** Subitemset Isomorphism (SI) decision problem:

**Input:** Two itemsets \( S \) and \( T \) over the domain \( D \).

**Question:** Is there a bijection \( J : D \to D \) s.t. \( J(S) \subseteq T \)?

Before presenting a rather technical proof that the II decision problem is GI-Hard, we give a brief discussion on how the GI and SI problems “differ”. Intuitively, the two problems are very similar as the inputs to both problems can be represented as zero-one matrices — see Figures 2 and 3; however, there are two fundamental differences.

The first fundamental difference — in the GI problem a swap of vertices is represented as a swap of two rows and two columns of the zero-one adjacency matrix (Figure 2), whereas in the II problem a swap of two domain elements is represented as a swap of two columns (Figure 3). The second fundamental difference — checking a solution to the GI problem, the two zero-one matrices must match exactly, whereas, in the II problem any reordering of the rows is permitted (recall Remark 1.2).

**Lemma 2.4.** \( GI \leq_P II \).
The proof of Lemma 2.4 is a rather technical one. However, the proof is constructive, and we present examples in Figures 2.5 and 2.6 for the essential steps of the proof. A detailed explanation of the examples following the steps of the proof of Lemma 2.4 is presented in Section 3.

Proof. Define the function \( f : \langle G, H \rangle \mapsto \langle S, T \rangle \) where \( \langle G, H \rangle \) is input to GI and \( \langle S, T \rangle \) is an input to II. The itemset \( S = \{ S_g \mid g \in V_G \} \) where the items \( S_g = \{(g, v) \in E_G \mid v \in V_G \} \). Similarly, the itemset \( T = \{ T_h \mid h \in V_H \} \) where the items \( T_h = \{(h, w) \in E_H \mid w \in V_H \} \). We now show that the function \( f \) is a poly-time reduction of Graph-Isomorphism to Itemset-Isomorphism.

First, we need to show that the function \( f \) is a polynomial time one. It is obvious, that this is the case, because \( f \) does no computation and simply, re-structures the input. Hence, the reduction function \( f \) is polynomial time.

To prove that the presented poly-time reduction is correct, we need to show that a Graph-Isomorphism instance is satisfiable (yes instance), if and only if the created Itemset-Isomorphism instance is satisfiable.

Suppose that the Graph-Isomorphism instance is satisfiable: there exists a bijection \( I : V_G \mapsto V_H \) s.t. \( (v, w) \in E_G \) iff \( (I(v), I(w)) \in E_H \). We claim that \( J : (v, w) \mapsto (I(v), I(w)) \) satisfies \( J(S) = T \). To see this, consider any item \( S_g = \{(g, x) \in E_G \mid x \in V_G \} \) and apply the bijection \( J \) to it. Then clearly we have \( J(S_g) = \{(I(g), I(x)) \in V_H \mid x \in V_G \} = \{(I(g), y) \in V_H \mid y \in V_H \} = T_{I(g)} \). Hence, we have shown that, if any Graph-Isomorphism instance \( \langle G, H \rangle \) is satisfiable then the created Itemset-Isomorphism instance \( f(\langle G, H \rangle) \) is satisfiable.

Now suppose that the created Itemset-Isomorphism instance \( f(\langle G, H \rangle) \) = \( \langle S, T \rangle \) is satisfiable: there is a bijection \( J : E_G \mapsto E_H \) s.t. \( J(S) = T \). By Definition 2.1 of itemset isomorphism and Remark 2.2, we know there exists a bijection \( \sigma \) that maps the items in \( J(S) \) to the items in \( T \). Hence, \( \sigma : V \mapsto H \) is such that for any \( S_g \in S \) we have \( T_{\sigma(g)} \in T \), and vice versa. We claim that \( \sigma \) gives a graph isomorphism from \( G \) to \( H \). To see this, notice that for all \( (v, w) \in E_G \) we have \( \sigma(v), \sigma(w) = J((v, w)) \) (we are working with undirected graphs). But, from the assumption we know that \( J((v, w)) \in E_H \); to go from \( E_H \) to \( E_G \) is the same because \( \sigma \) is bijective, hence \( \sigma^{-1} \) exists. Therefore, we have shown that if the created Itemset-Isomorphism instance \( f(\langle G, H \rangle) \) = \( \langle S, T \rangle \) is satisfiable then the original Graph-Isomorphism instance \( \langle G, H \rangle \) is satisfiable. \( \square \)

Lemma 2.5. II ∈ \( NP \).

Proof. We need to show that a polynomial time verifier of the Itemset-Isomorphism problem exists to conclude that II is in \( NP \). It is easy to see that, given a bijection \( J \) the verifier needs to check if \( J(S) = T \). Clearly the application of the bijection \( J \) to \( S \) can be done in polynomial time. The equality checking can be done in polynomial time because \( J(S) \) and \( T \) are sets of a polynomial number of elements each. \( \square \)

Theorem 2.6. II is GI-Hard.

Proof. Follows immediately by applying Lemmas 2.4 and 2.5. \( \square \)

Corollary 2.7. SI is GI-Hard.
Proof. An immediate consequence to Theorem 2.6 because obviously \( II \leq_P SI \) and \( SI \in NP \).

Furthermore, finding a class representative up to subitemset isomorphism is also GI-Hard — clearly poly-time reducible to the SI problem.

3 Examples

The examples presented in Figures 4 and 5 demonstrate how to apply the poly-time transformation function \( f \) (defined in the proof of Lemma 2.4) to an instance \( \langle G, H \rangle \) of the GI problem to produce an instance \( \langle S, T \rangle \) of the II problem. From the examples, it is clear that \( \langle G, H \rangle \) is satisfiable if and only if \( \langle S, T \rangle \) is satisfiable.

Following the proof of Lemma 2.4 and the two Figures 4 and 5, we see exactly how to construct \( J \) using \( I \), and vice versa. Note, that Lemma 2.4 works only for undirected graphs; a more technical proof is required for the case of directed graphs but is not necessary for the complexity classification of the itemset isomorphism problem.

3.1 Figure 4

The example presented in Figure 4 shows two isomorphic graphs and their \( f \)-corresponding isomorphic itemsets; recall \( f \) from proof of Lemma 2.4. It is clear that, the two graphs \( G \) and \( H \) are uniquely isomorphic — there exists a unique bijection \( I : V_G \rightarrow V_H \) that satisfies \( (v, w) \in E_G \leftrightarrow (I(v), I(w)) \in E_H \).

Hence, given the satisfiable instance \( \langle G, H \rangle \) and the bijection \( I : V_G \rightarrow V_H \), in the proof of Lemma 2.4 we claim that \( J : (v, w) \rightarrow (I(v), I(w)) \) satisfies \( J(S) = T \). One can easily check the graphs and itemsets in Figure 4 to verify the correctness of this claim.

Now, suppose we are given a bijection \( J : E_G \rightarrow E_H \) s.t. \( J(S) = T \). Clearly for the example in Figure 4, we have a unique \( \sigma = I \) that maps the items in \( J(S) \) to the items in \( T \).

3.2 Figure 5

The example presented in Figure 5 shows two isomorphic graphs and their \( f \)-corresponding isomorphic itemsets. This is example is more explanatory than the one presented in Figure 5 because the isomorphisms between the graphs \( G \) and \( H \) are not unique. Using the constructive proof of Lemma 2.4 we see that the graph isomorphism \( I_1 \) corresponds to the itemset isomorphism \( J_1 \) and similarly, \( I_2 \) corresponds to \( J_2 \).

4 Conclusion and Future Work

Fast algorithms for the Subitemset Isomorphism (SI) problem are of practical importance in searching for optimal sorting networks. The SI problem is encountered in recent [2] [4] [3] [6] research but its computation complexity classification has not been discussed. This paper proves the Itemset Isomorphism problem is GI-Hard. As a corollary, the Subitemset Isomorphism (SI) problem is
shown to be GI-Hard. The complexity analysis presented here, is of importance to research aimed at fast practical algorithms [6] [20] for the SI problem, as well as, extending the list of GI-Hard problems which are of practical importance too.

For future work, we aim to classify the SI problem more precisely, rather than the lower complexity bound given here. We suspect, that the problem of Subitemset Isomorphism is NP-Complete. The reason, there is an intuitive relation between the pair of Graph-Isomorphism (GI-Complete) and the Subgraph-Isomorphism (NP-Complete) and the pair of Itemset-Isomorphism (GI-Hard) and Subitemset-Isomorphism (GI-Hard).

5 Acknowledgements

Work supported by the Irish Research Council (IRC) and Science Foundation Ireland grant 12/IA/1381.

References

[1] R. C. Bose, R. J. Nelson, A sorting problem. J. ACM 9 (2) (1962) 282–296. doi:10.1145/321119.321126.
URL http://doi.acm.org/10.1145/321119.321126

[2] D. Bundala, M. Codish, L. Cruz-Filipe, P. Schneider-Kamp, J. Závodný, Optimal-depth sorting networks. CoRR abs/1412.5302.
URL http://arxiv.org/abs/1412.5302

[3] D. Bundala, J. Zavodny, Optimal sorting networks. CoRR abs/1310.6271.
URL http://arxiv.org/abs/1310.6271

[4] D. Bundala, J. Zavodny, Optimal sorting networks, in: A. H. Dediu, C. Martín-Vide, J. L. Sierra-Rodríguez, B. Truthe (Eds.), Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings, Vol. 8370 of Lecture Notes in Computer Science, Springer, 2014, pp. 236–247.

[5] M. Codish, L. Cruz-Filipe, P. Schneider-Kamp, The quest for optimal sorting networks: Efficient generation of two-layer prefixes. CoRR abs/1404.0948.
URL http://arxiv.org/abs/1404.0948

[6] M. Codish, L. Cruz-Filipe, M. Frank, P. Schneider-Kamp, Twenty-five comparators is optimal when sorting nine inputs (and twenty-nine for ten). CoRR abs/1405.5754.
URL http://arxiv.org/abs/1405.5754

[7] I. Parberry, A computer assisted optimal depth lower bound for sorting networks with nine inputs, in: F. R. Bailey (Ed.), SC, IEEE Computer Society / ACM, 1989, pp. 152–161.

[8] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.
[9] H. L. Bodlaender, Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees, J. Algorithms 11 (4) (1990) 631–643. doi:10.1016/0196-6774(90)90013-5. URL http://dx.doi.org/10.1016/0196-6774(90)90013-5

[10] H. Gazit, J. H. Reif, A randomized parallel algorithm for planar graph isomorphism, in: SPAA, 1990, pp. 210–219. doi:10.1145/97444.97687. URL http://doi.acm.org/10.1145/97444.97687

[11] F. Wagner, S. Datta, N. Limaye, P. Nimbhorkar, T. Thierauf, Planar graph isomorphism is in log-space Electronic Colloquium on Computational Complexity (ECCC) 16 (2009) 52. URL http://eccc.hpi-web.de/report/2009/052

[12] J. Köbler, U. Schöning, J. Torán, Graph isomorphism is low for PP, Computational Complexity 2 (1992) 301–330. doi:10.1007/BF01200427. URL http://dx.doi.org/10.1007/BF01200427

[13] S. Toda, Graph isomorphism: Its complexity and algorithms (abstract), in: C. P. Rangan, V. Raman, R. Ramanujam (Eds.), Foundations of Software Technology and Theoretical Computer Science, 19th Conference, Chennai, India, December 13-15, 1999, Proceedings, Vol. 1738 of Lecture Notes in Computer Science, Springer, 1999, p. 341.

[14] D. S. Johnson, The np-completeness column: an ongoing guide, Journal of Algorithms 6 (1985) 434–451.

[15] K. S. Booth, C. J. Colbourn, Problems polynomially equivalent to graph isomorphism, Computer Science Department, Univ., 1979.

[16] P. Pritchard, An old sub-quadratic algorithm for finding extremal sets, Inf. Process. Lett. 62 (6) (1997) 329–334.

[17] R. J. Bayardo, B. Panda, Fast algorithms for finding extremal sets, in: SDM, SIAM / Omnipress, 2011, pp. 25–34.

[18] M. Fort, J. A. Sellarès, N. Valladares, Finding extremal sets on the GPU J. Parallel Distrib. Comput. 74 (1) (2014) 1891–1899. doi:10.1016/j.jpdc.2013.07.004. URL http://dx.doi.org/10.1016/j.jpdc.2013.07.004

[19] M. Marinov, N. Nash, D. Gregg, A practical algorithm for finding extremal sets, TCD-CS-2015-03, Technical Report, Department of Computer Science, Trinity College Dublin.

[20] M. Marinov, D. Gregg, A practical algorithm for finding extremal sets up to permutation, TCD-CS-2015-04, Technical Report, Department of Computer Science, Trinity College Dublin.
Item — a set of elements over the domain $D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$. The items $a$, $b$, $c$ and $d$ are presented. We can always represent a set over a domain $D$ as a binary string of length $|D|$ where the $i$-th bit equals 1 iff the element $d_i$ is contained in the set.

Itemset — set of items over the domain $D$. The two itemsets $S = \{b, d\}$ and $T = \{a, c, d\}$ over the domain $D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$ are presented. Remember that there are no duplicate items within an itemset.

Dataset — ordered set of itemsets over the domain $D$. The dataset $F = \langle S, T \rangle$ over the domain $D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$ is presented. Remember that the itemsets within a dataset are ordered increasingly by cardinality.

Figure 1: Graphical representation of the mathematical objects that are used throughout the paper — item, itemset and dataset. For all of the examples in this figure, we use a dataset $D = \{d_1, d_2, \ldots, d_7\}$ of seven elements. For a formal definition, please refer to Section 1.1.1.
Figure 2: This figure presents a graph $G$ and its adjacency matrix. We also present a swap of the vertices 1 and 4 of $G$ to obtain $H$. To construct the adjacency matrix of $H$ from the adjacency matrix of $G$, we first swap the rows 1 and 4 of $G$ and then swap the columns 1 and 4. Since, every permutation (of the vertices of $G$) can be written as a sequence of swaps, this figure shows the methodology of applying a permutation to a graph.
Figure 3: This figure presents an itemset $S$ over the domain $D = \{d_1, d_2, d_3, d_4\}$ and its matrix representation (as described in Section 1.1.1 and Figure 1(b)). We also present a swap of the domain elements $d_1$ and $d_4$ of $S$ to obtain $T$. To construct the matrix representation of $T$ from the matrix of $S$, we need to swap the two columns $d_1$ and $d_4$. Since, every permutation (of the domain $D$) can be written as a sequence of swaps, this figure shows the methodology of applying a permutation to an itemset.
Figure 4: An example of two isomorphic graphs $G$ and $H$ together with the corresponding isomorphic itemsets $S$ and $T$ generated by the poly-time reduc-
tion function $f : \langle G, H \rangle = \langle S, T \rangle$, as described in the proof of Lemma 2.4. This figure serves as a detailed example of the constructive proof to Lemma 2.4. In the figure we see that, there is a unique isomorphism between $G$ and $H$, given by $I$; and a unique isomorphism $J$ between $S$ and $T$. For detailed explanation of this figure refer to Section 3.1.
Figure 5: An example of two isomorphic graphs $G$ and $H$ together with the corresponding isomorphic itemsets $S$ and $T$ generated by the poly-time reduction function $f : \langle G, H \rangle = \langle S, T \rangle$, as described in the proof of Lemma 2.4. This figure serves as a detailed example of the constructive proof to Lemma 2.4. In the figure we see that, there are exactly two isomorphisms between $G$ and $H$, given by $I_1$ and $I_2$; and exactly two isomorphisms $J_1$ and $J_2$ between $S$ and $T$, where $I_1$ corresponds to $J_1$ and $I_2$ corresponds to $J_2$. For an in-depth explanation of this figure refer to Section 3.2.