NP-hardness problems for target controllability of complex networks

WU Kai Chiu

Åbo Akademi

A thesis submitted for the degree of

Master of Computer Science

2016

To my parents and all of my friends,
for their endless support and encouragement.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Eugen Czeizler and Ion Petre for their motivation and support for my research and study. I would also like to thank Helsa Chan, Soarer Siu and Angela Wu for their comments.

Finally and without hesitation, I would like to thank Coco Yiu for serving as a reader and her comments and suggestions.

Abstract

Network controllability is used to model complex networks. Controlling the entire system is of particular interest in various fields. The optimization version for the minimal controllability problem (MCP) is known to be **NP-HARD** by Olshevsky [26]. Structural controllability allows one to analyse the system when the exact strength of interactions is difficult to measure. The optimization variant for structural controllability is shown to be solvable by a polynomial-time bipartite maximal matching algorithm [17].

In this thesis, we study three polynomial-time algorithms for structural controllability and structural output controllability [9, 17, 34] and analyse their correctness and optimality. We discover that the algorithms in Gao *et al.* [9], Wu *et al.* [34] are not correct. We present counterexamples to both correctness and optimality for the algorithm by Gao *et al.* [9]. We also construct a counterexample to the optimality for the algorithm by Wu *et al.* [34]. The subtle flaws in their proofs are analyzed. Furthermore, we show that two variants of the MCP problem regarding structural output controllability are **NP-HARD**: edge-bounded variant and outdegree bounded variant. The NP-hardness is shown by an explicit construction of a logspace reduction from the 3SAT problem. The limitations of the construction and equivalence of other variants are also discussed.

Contents

List of Figures

Chapter 1

Introduction and Motivation

Control theory is a mathematical model for the dynamics of complex systems. The study of complex network systems and control theory is applied widely, for example, in electrical grids, transport networks, social communication networks and biological systems [24, 25, 31, 35]. Its significance to biological networks is particularly worth noting. The brain, cancer, and several biological systems are modelled as complex network systems [11, 20]. Its application in medicine also includes drug target identification [2, 34].

It is known that controllability of a system can be verified by computation of the rank of the controllability matrix corresponding to the system [17]. By applying graph theory to control theory, Lin [15] obtained a characterization of structural controllability in terms of graph theoretic structure of cactus covering. Lin's result was later generalized by Shields and Pearson [30] to multi-input systems. In the 1990s, Poljak [29] subsequently gave four equivalent formulations for structural controllability. The results were generalized to output systems by Murota and Poljak [23].

Computational complexity classifies problems by difficulties. The problem class **P** represents problems that can be solved efficiently. The class **NP** intuitively represents problems that can be easily verified. Any problem that is as difficult as all the problems in **NP** is said to be **NP-HARD**.

Various optimization problems for different complex systems were formulated [13, 25, 27, 31]. Usually, the computational complexity class of a problem allows one to understand the nature of the problem. We are interested in optimization problems regarding the structural output controllability of systems (precise definition to be given in Chapter 4). Olshevsky [26] showed that the minimal controllability problem (MCP) for controllability is **NP-HARD**. Liu *et al.* [17] showed that the MCP problem for structural controllability for an entire system has a polynomial-time algorithm and hence the corresponding problem is **P**. The MCP problem was extended to structural controllability for a preselected part of a system, which is known as structural output controllability, with two proposed algorithms (see Gao *et al.* [9], Wu *et al.* [34]).

In this thesis, however, counterexamples to the two proposed algorithms for structural output controllability are given. The hidden flaws in the corresponding proofs are revealed. A code written in Sage, a computer algebra system, is used to verify the correctness of the computation of rank of the structural output controllability matrix. A few concrete computational counterexamples are shown. Two variants of the MCP problem concerning structural output controllability are shown to be **NP-HARD**. This gives strong evidence that no polynomial time algorithm can solve the MCP problem for structural output controllability unless **P**=**NP**.

1.1 Linear systems and controllability

A classical example of a linear system is the control model of a robot. One would like to control the movement of the robot by an external controller that adjusts the parameters of some particular parts (such as joints) of the robot. Controllability is the measure of one's ability to drive the system from an initial state to an arbitrary state. On the other hand, structural controllability studies a family of systems instead of a single system. They are formulated in Section 2.2.

1.2 Variants of minimal controllability problem (MCP)

The basic problem is briefly stated as (see Chapter 4 for mathematical formulation):

> Given a linear system (without inputs), find the minimum number of internal states needed to be affected by an input so that the resulting system is controllable.

Olshevsky [26] showed that any approximation to the answer within a factor of $c \log n$ for $c > 0$ is **NP-HARD**.

We call *full controllability problem* (FCP) the problem for structural controllability involving the entire system. Similarly, the *target controllability problem* (TCP) is considered as the variant of MCP for structural controllability of a preselected part of the given system. Two versions of the partial controllability problem are further formulated: *degree bounded version* and *edge bounded version.* The degree bounded version restricts the number of affected states of the system for each external input; while the edge bounded version restricts the total number of connections from external inputs to the states. Both versions are proved to be **NP-HARD** in this thesis.

1.3 Outline of the remaining chapters

Chapter 2 covers the preliminaries and rigorous definitions:

The mathematical background required is linear algebra, control theory and graph theory. Section 2.1 reviews a few notions in linear algebra and the definition of rank for control theory. The intuition behind control theory and its classical results are discussed in Section 2.2. Graph theory and the results beyond classical control theory are mentioned in Sections 2.3 and 2.4. Our proofs highly rely on reductions and the well known **NP-HARD** problem 3SAT. The intuitive ideas, notions and the definitions of the 3SAT problem are given in Section 2.5.

Chapter 3 covers the work by Barabási *et al.* [2], Gao *et al.* [9] and Wu *et al.* [34].

In Section 3.1, the original algorithm for structural controllability by Liu *et al.* [17] is stated. The analysis of its correctness is mentioned. In the remaining sections, both algorithms proposed by Gao *et al.* [9] and Wu *et al.* [34] are stated. We construct counterexamples to the two algorithms and reveal the flaws in their proof.

Chapter 4 covers our major results about the NP-hardness of the variants. Their decision versions are shown to be **NP-HARD**. Mathematical formulation of the problems is stated. Reductions to the decision problems from 3SAT are shown.

Chapter 5 summarizes the results obtained and suggests further direction for the research of practical algorithms.

Chapter 2

Preliminaries

In this chapter, we will review some concepts and known results from linear algebra, control theory for discrete time-invariant linear system, graph theory and computational complexity theory. Connections between control theory and graph theory will also be discussed.

2.1 Linear algebra

We will review the notions of vector space, subspace, dimension, and rank of a matrix. One may refer to FRIEDBERG *et al.* [8] and VALENZA [32] for the abstract and structural views of linear algebra. A computational and elementary approach can be found in LIPSCHUTZ [16] and MATTHEWS [19].

A *vector space V (over a ground field K)* is a set of objects equipped with two binary operations (addition) $+ : V \times V \to V$ and (scalar multiplication) $\cdot : K \times V \to V$ together satisfying the vector space axioms. The objects in V are called *vectors* and the elements in K are called *scalars*. The vector space axioms are stated as follows:

Definition (Vector space)**.** Any vector space V over a field K satisfies the following:

1. $(V, +)$ is an additive group:

   (a) (associativity) $(u + v) + w = u + (v + w)$ for any $u, v, w \in V$.

(b) (commutative) $u + v = v + u$ for any $u, v \in V$.

(c) (existence of identity) There exists an unique vector z, called the *zero vector*, such that $z + v = v + z = v$ for all $v \in V$.

(d) (existence of inverse) For all $v \in V$, there exists $w \in V$ such that $v + w = z$.

2. $(\lambda\mu)\, v = \lambda\,(\mu v)$ for all $\lambda, \mu \in K$ and $v \in V$.

3. $(\lambda + \mu)\, v = \lambda v + \mu v$ for all $\lambda, \mu \in K$ and $v \in V$.

4. $\lambda\,(u + v) = \lambda u + \lambda v$ for all $\lambda \in K$ and $u, v \in V$.

5. $1v = v$ for all $v \in V$, where 1 is the multiplicative identity of K.

We denote by $\mathbf{0}$ the zero vector. In our particular application, only vector spaces over the ground field $\mathbb{R}$ will be considered, i.e. $K = \mathbb{R}$.

A nonempty subset $W \subseteq V$ is said to be a *vector subspace* of V if the subset W itself together with the addition and scalar multiplication defined on V is also a vector space, or equivalently, W is closed under addition and scalar multiplication. The *span* of a family of vectors $\{v_1, v_2, \ldots, v_n\}$ is given by

$$\operatorname{span}(v_1, v_2, \ldots, v_n) = \left\{ \sum_{k=1}^{n} \lambda_k v_k \mid \lambda_k \in K \right\}.$$

The span of a set of vectors S, which is also called the *subspace generated (or spanned) by* S, is the smallest vector subspace of V that contains S. Note that every subspace contains the zero vector. A family of vectors $\{v_1, v_2, \ldots, v_n\} \subseteq V$ is said to be *linearly independent* if the zero vector can be represented uniquely in $\operatorname{span}(v_1, v_2, \ldots, v_n)$, i.e. $\sum \lambda_k v_k = \mathbf{0} \implies \lambda_1 = \lambda_2 = \cdots = \lambda_n = 0$. The family $\{v_1, v_2, \ldots, v_n\}$ *spans the vector space* V if $\operatorname{span}(v_1, v_2, \ldots, v_n) = V$.

A *basis* for a vector space V is a family of linearly independent vectors which spans V. It is well known that every vector space has a basis and there exists a bijection

between any two of its bases. Consequently, the *dimension* of V is well-defined as the cardinality of its any basis. The dimension of a vector subspace is defined similarly.

Let $M_{m\times n}(\mathbb{R})$ be the set of $m \times n$ matrices over the real numbers. We denote A_{ij} the entry in the i^{th} row and j^{th} column. We can treat a matrix A as a collection of m row vectors or n column vectors. The *row space* (the *column space*) of a matrix is the span of the family of row vectors (column vectors). It is known that both the row space and the column space of a matrix have equal dimension, which we define as the *rank*, $\operatorname{rank}(A)$, of the matrix A. Numerically, given a matrix, one can apply the Gaussian Elimination algorithm[1] to find the number of linearly independent column vectors, which is exactly the rank.

For a finite collection of column vectors (of same size), $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$, we denote $[\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n]$ the corresponding matrix that consists of the column vectors of disjoint union $\bigsqcup \mathcal{C}_k$.

2.2 Control Theory

This section is devoted to discrete time-invariant linear dynamic system and structural controllablity. We follow the terminologies as in Murota and Poljak [23].

A *discrete time-invariant linear output system* having n states, m inputs and ℓ outputs can be modelled by

$$\begin{cases} \boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t \\ \boldsymbol{y}_t = C\boldsymbol{x}_t \end{cases}$$

where A, B, C are matrices of size $n\times n$, $n\times m$ and $\ell\times n$ respectively, $\boldsymbol{x}_t \in \mathbb{R}^n$, $\boldsymbol{u}_t \in \mathbb{R}^m$ and $\boldsymbol{y}_t \in \mathbb{R}^\ell$ are the *state vectors*, *input vectors* and *output vectors* with discrete time variable $t \in \mathbb{Z}_{\geq 0}$. The matrices describe all interactions within the system and hence all information completely. Strictly speaking, A, B and C describe the state dynamics,

[1] Detail and worked example can be found in [16, Ch.5, pp. 151-155, 175-179] and [19, Ch. 3.5].

the input dynamics and the output dynamics of a system; n, m and ℓ are the number of state, input and output variables of the system respectively. We denote (A, B, C) the system with matrices A, B and C; and (A, B) (omitting C) when C is the identity matrix I_n. Without loss of generality, we assume $\operatorname{rank} C = \ell$, for if not, one could have simply taken the maximum linearly independent set of row vectors and computed the remaining linearly dependent row vectors. For our application, we restrict every row of C having exactly one non-zero entry. Nevertheless, all the definitions and results still hold for general matrix C.

An output value $\boldsymbol{y} \in \mathbb{R}^\ell$ is *reachable from an initial state* $\boldsymbol{x} = \boldsymbol{x}_0 \in \mathbb{R}^n$, denoted $\boldsymbol{x}_0 \rightsquigarrow \boldsymbol{y}$ if there exists a finite sequence of inputs $\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_t \in \mathbb{R}^m$ such that $\boldsymbol{y}_t = \boldsymbol{y}$. For a system (A, B, C), its *output controllable subspace* is the vector subspace $\left\{\boldsymbol{y} \in \mathbb{R}^\ell \mid \mathbf{0} \rightsquigarrow \boldsymbol{y}\right\}$ which consists of all values reachable from the initial state $\mathbf{0} \in \mathbb{R}^n$. We denote by $d(A, B, C)$ the dimension of the output controllable subspace of (A, B, C).

A system (A, B, C) is said to be *output controllable* if $d(A, B, C) = \ell = \operatorname{rank} C$; and simply *controllable* when the condition holds for $C = I_n$.

The *output controllability matrix* of a system (A, B, C) is given by

$$\mathcal{OC}(A, B, C) := \left[CB, CAB, CA^2B, \ldots, CA^{n-1}B\right].$$

The following result concerning the output controllable subspace and its output controllability matrix is known as the Kalman's rank criteria. Its proof is given in the appendix for completeness.

Theorem 2.2.1 (Kalman)**.** *Let (A, B, C) be a system, then*

$$d(A, B, C) = \operatorname{rank} \mathcal{OC}(A, B, C).$$

Hence, the system (A, B, C) is output controllable iff $\operatorname{rank} \mathcal{OC}(A, B, C) = \operatorname{rank} C$.

A related concept can be extended to a family of systems, where two systems in the same family share the same set of non-zero interactions. In view of this, we can discuss structural equivalence of two systems by disregarding the strength of the interactions and only focusing on their existences.

Two matrices A and B of the same size are said to be *structurally equivalent* and denoted $A \sim B$ if they have zero on the same positions, i.e.

$$A_{ij} = 0 \text{ if and only if } B_{ij} = 0 \text{ for all entries } A_{ij}, B_{ij}.$$

Two systems (A, B, C) and (A', B', C') are *structurally equivalent* if $A \sim A'$, $B \sim B'$ and $C \sim C'$.

It is an exercise to verify that structural equivalence is an equivalence relation. Therefore, we can discuss the generic properties shared among the equivalent class $[A, B, C]$ of a given system (A, B, C). Conventionally, whenever the word generic or structural is used, it is understood that the corresponding property is considered among the equivalent class of a given system. The *generic dimension* $\operatorname{gd}(A, B, C)$ of a system (A, B, C) is defined as the maximum dimension among all output controllable subspaces, i.e.

$$\operatorname{gd}(A, B, C) = \max \{ d(A', B', C') \mid (A', B', C') \in [A, B, C] \}.$$

Structural output controllability and *structural controllability* are defined by replacing $d(A, B, C)$ in the definition of output controllability and controllability with $\operatorname{gd}(A, B, C)$. According to [23] and [33], the set $\{(A', B', C')\}$ of system not satisfying the maximality condition forms an algebraic variety and its complement is measure zero and is open and dense with respect to operator norm.

Note. We assume the initial state of the system is always zero. It is a purely technical reason for the well-definedness of a vector subspace, which must contain the zero vector. Practically, one can start with an arbitrary state $\boldsymbol{x}_0$ and consider the affine

linear transformation $\boldsymbol{x} \mapsto \boldsymbol{x} - \boldsymbol{x}_0$ to obtain a vector subspace.

Example 2.2.2. Consider the system (A, B) given by

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ e_0 & 0 & 0 & 0 \\ e_1 & 0 & 0 & 0 \\ 0 & 0 & e_2 & 0 \end{pmatrix}, B = \begin{pmatrix} e_3 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

where $n = 4$, $m = 1$ and $C = I$.

It is not structurally controllable as we can see

$$\mathcal{OC}(A, B) = \begin{pmatrix} e_3 & 0 & 0 & 0 \\ 0 & e_0e_3 & 0 & 0 \\ 0 & e_1e_3 & 0 & 0 \\ 0 & 0 & e_1e_2e_3 & 0 \end{pmatrix}$$

has rank $3 < n$.

2.3 Graph Theory

We will review necessary notions and terminologies in undirected graph and directed graph. There are various formulations and terminologies for graph theory. In general, we follow and refer to the treatise from Bollobás [3], Bondy and Murty [4] and Diestel [6]. In particular, the terms *walks* and *paths* are chosen instead of *paths* and *simple paths*. Fournier [7] and Lovász [18] contain interesting results about the theory.

The concepts of directed matching, cactus covering and linking in directed graph play an important role in structural controllability [17], [23], [29]. Their connections will be discussed in the next section.

2.3.1 Undirected Graph

Definition 2.3.1. An *undirected graph* $G = (V, E)$ is a collection of vertices V and edges E, where $E \subseteq V \times V$ and is symmetric, i.e. $(u, v) \in E \implies (v, u) \in E$ for any $u, v \in V$.

We shall identify the pair (u, v) and (v, u), and call it the edge uv or vu.

Two distinct edges are said to be

- *adjacent* if they have a common vertex;
- *(vertex) disjoint* otherwise.

Two vertices $u \neq v \in V$ are *adjacent* if $uv \in E$.

A graph $G' = (V', E')$ is *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$.

The *degree* $d_G(v)$ of a vertex v is the number of its adjacent vertices.

When it is important to indicate the associated graph, we use subscript notation to denote the objects, like V_G, E_G.

Note. From the definition, a graph may contain self-loops but not parallel edges.

Definition 2.3.2. Let $G = (V, E)$ be an undirected graph. Consider a vertex subset $A \subseteq V$ and an edge subset $F \subseteq E$.

The *induced graph* $G[A]$ is the graph (A, E'), where $E' = \{uv \in E \mid u, v \in A\}$.

The unique spanning subgraph containing all vertices V and edges F is denoted by $G[F] = (V, F)$.

Definition 2.3.3. A *walk* is a sequence of (not necessarily distinct) adjacent edges $e_1, e_2, \ldots, e_n$, where $e_k = u_k u_{k+1}$. It is also called a walk from u_1 to u_{n+1} or simply $u_1 u_{n+1}$-walk and denoted $W : u_1 \overset{*}{\to} u_{n+1}$.

The vertices u_1 and u_{n+1} are called the *initial* vertex and *terminal* vertex respectively and the vertices $u_2, u_3, \ldots, u_n$ are the *internal* vertices.

The *length* of a walk is the number of edges n and the *order* of the walk is the number of distinct vertices it contains.

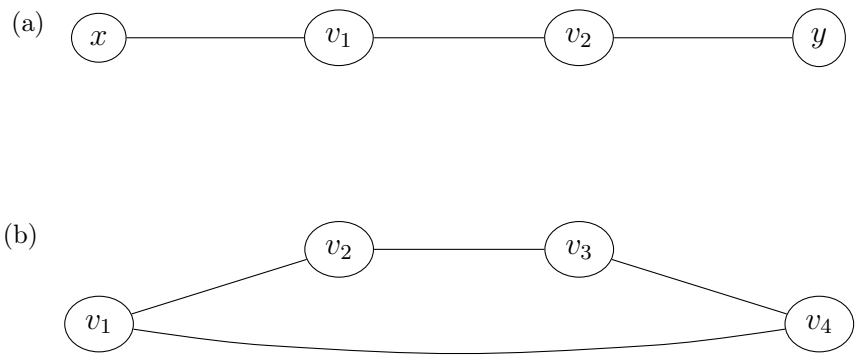

Figure 2.1: Undirected connected graphs. (a) P_4: A path of 4 vertices (hence of length 3). (b) C_4: A cycle of 4 vertices (and of length 4).

A *path* is a walk in which every vertex appears at most once. We denote by P_n a path of n vertices.

A *cycle* is a uv-path with an additional edge vu added at the end of the path P_n, where $n \geq 3$. We denote C_n ($n \geq 3$) a cycle of n distinct vertices.

Two walks are said to be

- *(vertex) disjoint* if they share no common vertices;
- *independent* or *edge-disjoint* if they share no common edges.

Definition 2.3.4. The *distance* $d_G(u, v)$ between two distinct vertices $u, v \in V$ of G is defined as the minimum length of any path from u to v or ∞ if no such path exists; and zero for $d_G(u, u)$.

The distance between two sets of vertices $A, B \subseteq V$, $d_G(A, B)$, is defined as the minimum length among every vertex pair $(a, b) \in A \times B$, i.e.

$$d_G(A, B) = \inf \{d(a, b) \mid a \in A, b \in B\}.$$

The *neighbourhood* $N_G(u)$ of $u \in V$ is the collection of adjacent vertices of u, or equivalently,

$$N_G(u) = \{v \mid d(u, v) = 1\}.$$

Figure 2.2: Bipartite graphs. (a) $|X| = |Y| = 3$. (b) Star.

Similarly, we define the *neighbourhood* $N_G(U)$ of a vertex subset $U \subseteq V$ to be

$$N_G(U) = \{v \notin U \mid d(U, v) = 1\}.$$

The *connected component* $N_G^*(u)$ of $u \in V$ is the collection of vertices of finite length from u

$$N_G^*(u) = \{v \mid d(u, v) < \infty\}.$$

One can easily check that the relation $u \sim v$ defined by $N_G^*(u) = N_G^*(v)$ is an equivalence relation and induces a partition of the vertices.

Definition 2.3.5. Two vertices u, v are connected if $d_G(u, v) < \infty$.

A graph G is *connected* if there is one and only one connected component in G.

Definition 2.3.6. A vertex subset $A \subseteq V$ is said to be *independent* (or *stable*) if it has no adjacent pair of vertices.

Definition 2.3.7. A *bipartite graph* is a graph $G = (V, E)$ with partition of vertices $V = X \cup Y$ such that both X and Y are independent.

Definition 2.3.8. A *matching* M of an undirected graph is a collection of vertex-disjoint edges with no self-loop.

A vertex v is matched (or saturated) by M if $e = uv \in M$ for some $u \in V$.

A matching M is said to be

- *maximum* if its cardinality is maximum among all matchings;

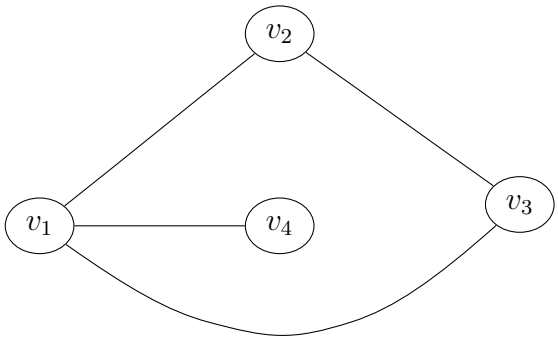

Figure 2.3: A maximal (but not maximum) matching is given by $\{v_1v_2\}$. The matching $\{v_1v_4, v_2v_3\}$ is both perfect and maximum.

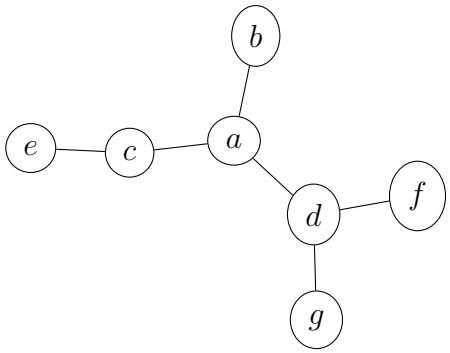

Figure 2.4: Tree

- *maximal* if $M \cup e$ is not a matching for all $e \in E$;
- *perfect* if every vertex is saturated by M.

Definition 2.3.9. A graph G is *acyclic* if it contains no cycle C_n for $n \geq 3$.

A *tree* is an acyclic connected graph.

It is well known that for a graph G the following are equivalent:

1. G is a tree;
2. G is connected and has exactly n vertices and $n - 1$ edges;
3. There is an unique path from u to v for any $u, v \in G$.

2.3.2 Directed Graph

We studied the concept of undirected paths, cycles and matchings.We will discuss the notions of their directed analogue. Cacti covering and linking structure, which were studied by Murota and Poljak in [23] and [29], will also be discussed in this section. Directed matching was introduced in Liu *et al.* [17]. An equivalent form of their formulation will be stated and other concepts like induced directed graph are generalized naturally.

Definition 2.3.10. A *directed graph (or digraph)* $D = (V, E)$ is a collection of vertices V and edges E, where $E \subseteq V \times V$.

We denote uv the edge (u, v), the vertices u and v are called the *initial (or head)* and the *terminal (or tail)* of the edge.

Two edges e_1, e_2 are *adjacent* if the initial of one edge is the same as the terminal of the other.

The *indegree* $d_D^-(v)$ of a vertex v is the number of edges with tail v, i.e.

$$d_D^-(v) = \#\{u \mid uv \in E\}.$$

Similarly, the *outdegree* $d_D^+(v)$ is defined by

$$d_D^+(v) = \#\{u \mid vu \in E\}.$$

Definition 2.3.11. The *underlying graph* $\mathcal{U}(D)$ of a directed graph D is the undirected graph constructed by adding necessary edges such that E is symmetric.

Definition 2.3.12. A (di)-walk is a sequence of adjacent edges. Directed paths and cycles are defined similarly. We denote P_n and C_n $(n \geq 2)$ their directed counterparts whenever a directed graph is discussed.

Definition 2.3.13. Suppose u and v are two distinct vertices in a digraph D. The distance from u to v, $d_D(u, v)$, is defined by the minimum length of any di-path from

u to v, if it exists; and otherwise ∞. We also define $d_D(u,u)$ to be zero.

The *outgoing neighbourhood* of v is given by

$$N_D^1(v) = \{u \mid d_D(v,u) = 1\}.$$

The *incoming neighbourhood of* v is given by

$$N_D^{-1}(v) = \{u \mid d_D(u,v) = 1\}.$$

Similarly, we have the *outgoing connected component* of v

$$N_D^+(v) = \{u \mid d_D(v,u) < \infty\}$$

and its *incoming connected component*

$$N_D^-(v) = \{u \mid d_D(u,v) < \infty\}.$$

The above definitions can be generalized for a vertex subset naturally.

The connectedness for digraph is further categorized into strong connectedness (bi-directional) and weak connectedness (underlying graph).

Definition 2.3.14. Two vertices u, v in a digraph D are *weakly connected* if they are connected in the underlying graph $\mathcal{U}(D)$; and are *strongly connected* if $d(u,v) < \infty$ and $d(v,u) < \infty$.

Both weak connectedness and strong connectedness are equivalence relations for the vertex set. Their corresponding partitions raise the concept of connected components.

Definition 2.3.15. Let $v \in V_D$ be a vertex. The *weakly connected component* (WCC) of v in D is its connected component in $\mathcal{U}(D)$. The *strongly connected component*

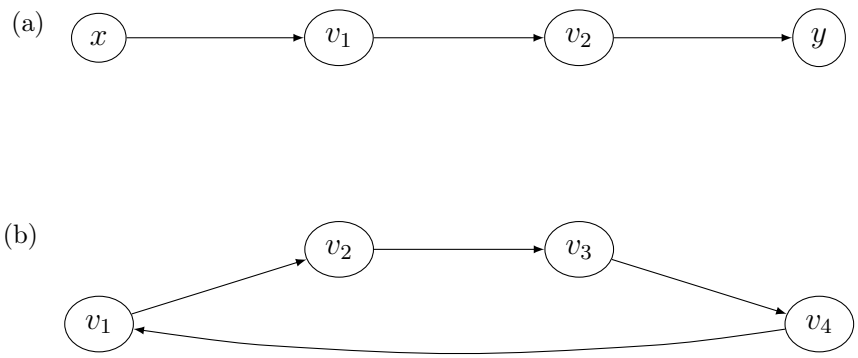

Figure 2.5: Connectedness for digraphs. (a) P_4 has 4 strongly connected components and is weakly connected. (b) C_4 is strongly connected.

(SCC) of v is the equivalent class induced by strong connectedness, i.e.

$$\{u \in V_D \mid u, v \text{ are strongly connected in } D\}.$$

A digraph is *weakly connected* if its underlying graph is connected; and is *strongly connected* if it has a unique strongly connected component.

From the definition, we can see that a directed path is weakly connected but not strongly connected; and a directed cycle is strongly connected.

Definition 2.3.16. The *undirected bipartite representation* $B = \mathcal{B}(D)$ of a digraph D is formulated as follows:

1. Construct vertices v^- (incoming) and v^+ (outgoing) for each $v \in V_D$

$$V_B = \left\{v^+ \mid v \in V_D\right\} \cup \left\{v^- \mid v \in V_D\right\}.$$

2. Construct edges u^+v^- for each edge $uv \in E_D$

$$E_B = \left\{u^+v^- \mid uv \in E_D\right\}.$$

The corresponding undirected graph $B = (V_B, E_B)$ is defined as the undirected bi-

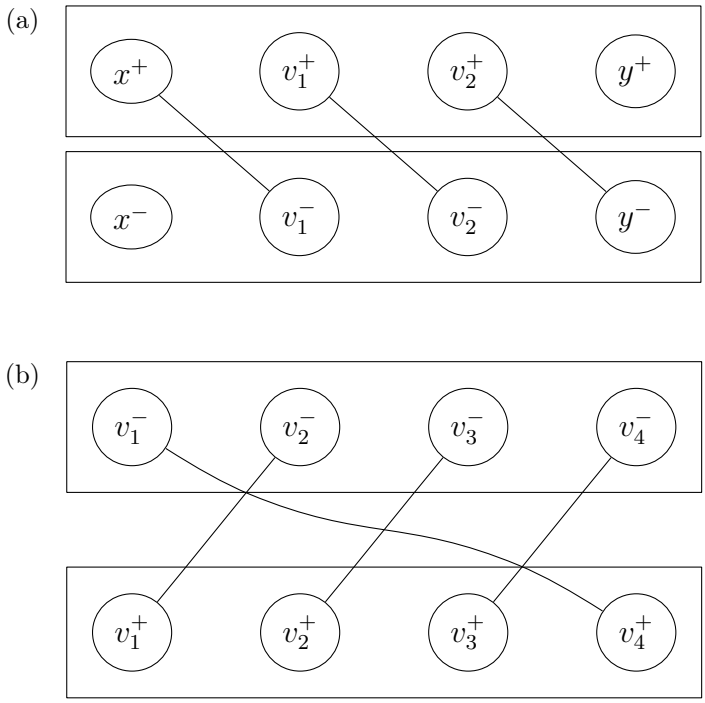

Figure 2.6: Bipartite representations. (a) $\mathcal{B}(P_4)$. (b) $\mathcal{B}(C_4)$.

partite representation of D.

Clearly, the set of incoming vertices and outgoing vertices is a partition. With this notion, we can extend our definition of undirected matching to directed matching.

Definition 2.3.17. A (di-)matching $\mathcal{M}$ of a digraph D is a matching for its undirected bipartite representation $\mathcal{B}(D)$. A vertex $v \in V_D$ is said to be *matched (or saturated) by* $\mathcal{M}$ if v^- is saturated by $\mathcal{M}$ in $\mathcal{B}(D)$.

An equivalent formulation of di-matching was given by Liu *et al.* [17] as follows:

An edge subset $\mathcal{M} \subseteq E_D$ is a matching if no two edges share a common initial or terminal; and a vertex $v \in V_D$ is *matched (or saturated) by* $\mathcal{M}$ if $uv \in \mathcal{M}$ for some $u \in V_D$.

It is clear from the bipartite representation that P_4 has a di-matching of size 3 and C_4 has a perfect di-matching.

Next, we will discuss some important structures in a digraph. Cacti covering,

maximum linkings and trees are graph structures that were studied in relation to structural controllability.

Definition 2.3.18. Let $U \subseteq V$ be a vertex subset. A digraph D is said to have a *U-rooted cacti configuration* if it satisfies the following:

1. V_D is covered by vertex disjoint di-paths with initial vertices in U and di-cycles;
2. the covering induces a vertex partition for V_D, i.e. any two of the above di-walks are pairwise vertex disjoint;
3. $N_D^+(U) = V_D$, i.e. there exists a di-path from U to v for each $v \in V_D$.

Fig. 2.7 shows an example of u-rooted cacti configuration, which consists of the di-path $uefg$ and a di-cycle $C_4 : a \to b \to c \to d \to a$. None of Fig. 2.7b and Fig.2.7c possess a u-rooted cacti configuration. In Fig. 2.7b, we can choose only one of the di-paths among $\{u \to e \to f, u \to g\}$ for disjointness. In Fig. 2.7c, $N^+(u)$ does not cover all vertices. Indeed, the third condition implies weakly connectedness.

Definition 2.3.19. A *linking* $\mathcal{L}$ is a collection of pairwisely vertex disjoint di-walks in D. The size of a linking $\mathcal{L}$ is its cardinality $|\mathcal{L}|$.

Let $A, B \subseteq V$ be two disjoint vertex subsets, a (A, B)-linking is a linking in which every walk has initial in A and terminal in B.

A collection $\overline{\mathcal{L}}$ of (A, B)-diwalks is said to be *congestion-free linking* if it satisfies the condition that the distances from the intersecting vertex to the corresponding terminals are distinct for any intersecting di-walks, i.e.

$$d_{D[W_1]}(v, t_1) \neq d_{D[W_2]}(v, t_2) \text{ for any di-walks } W_1, W_2 \text{ having a common vertex } v,$$

where t_1 and t_2 are the terminals of W_1 and W_2 respectively.

Congestion-free linking of a digraph is equivalent to linking in dynamic graph, which was described as agents by Murota and Poljak [23]. Intuitively, we (and our

friends) begin our driving journey on our assigned di-walks at a given starting time (one person for each di-walk). All of us follow our own di-walk and drive to the next vertex every second. Congestion-free means that it is feasible to arrange the starting time so that all of us arrive at the destination at the same moment and none of us visit the same vertex at any particular instance.

In Fig. 2.8a, a maximum linking is given by $\{a \to b, c \to d\}$. If we take $A = \{a\}$ and $B = \{b, c, d\}$, then $\{a \to b, a \to b \to c, a \to b \to c \to d\}$ is a (A, B)-congestion-free linking. Indeed, it is more useful to consider maximum (A, B)-linking or (A, B)-congestion-free linking for a given vertex subset A and B. In Fig. 2.8b, if $A = \{a\}$, and B is the remaining, the maximum (A, B)-congestion-free linking has size 3.

Unlike undirected tree, which has various equivalent formulation, an acyclic digraph needs not have path uniqueness. Even if the underlying graph of a digraph is a tree, it is not sufficient to ensure the existence of a vertex v such that $N^+(v)$ could cover the remaining vertices. One can observe this by reversing any one of the edges of P_4. On the other hand, strong connectedness implies the existence of dicycles. Here, we reserve the term tree for path non-ambiguity.

Definition 2.3.20. A digraph is *acyclic* if it has no dicycles, and it is said to be a *directed acyclic graph* (DAG).

Definition 2.3.21. A DAG is a *(di-)tree* if for every pair $u, v \in V_D$, there is at most one path from u to v.

We can observe that the digraph in Fig. 2.9a is a DAG, which has no dicycles. However, it is not a di-tree as there are two paths $a \overset{*}{\to} d$. Namely, $a \to b \to f \to d$ and $a \to d$. Fig. 2.9b gives a simple appearance of a di-tree. Note that there is no path between a and g.

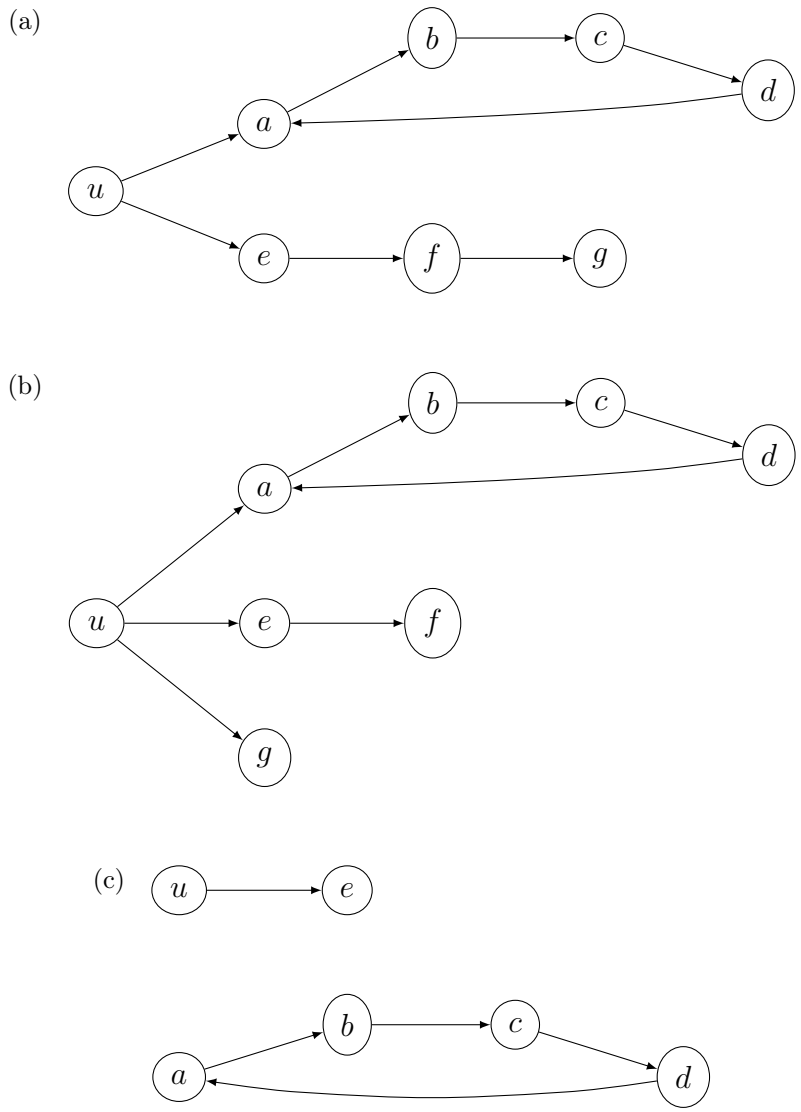

Figure 2.7: Cacti configuration. (a) u-rooted cacti configuration given by $uefg$ and C_4. (b) and (c) do not have u-rooted cacti configuration.

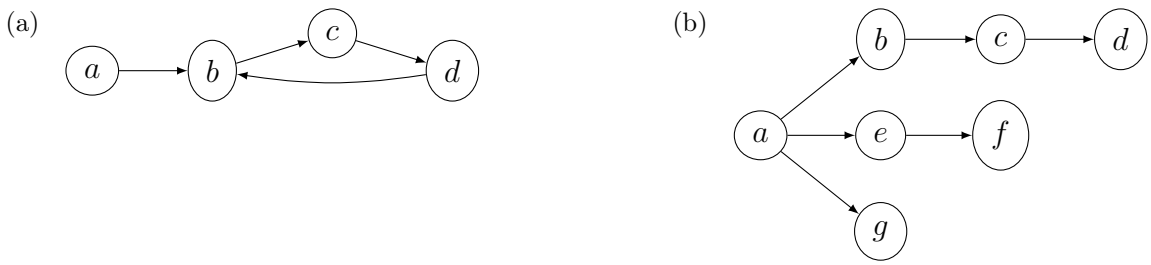

Figure 2.8: Linkings

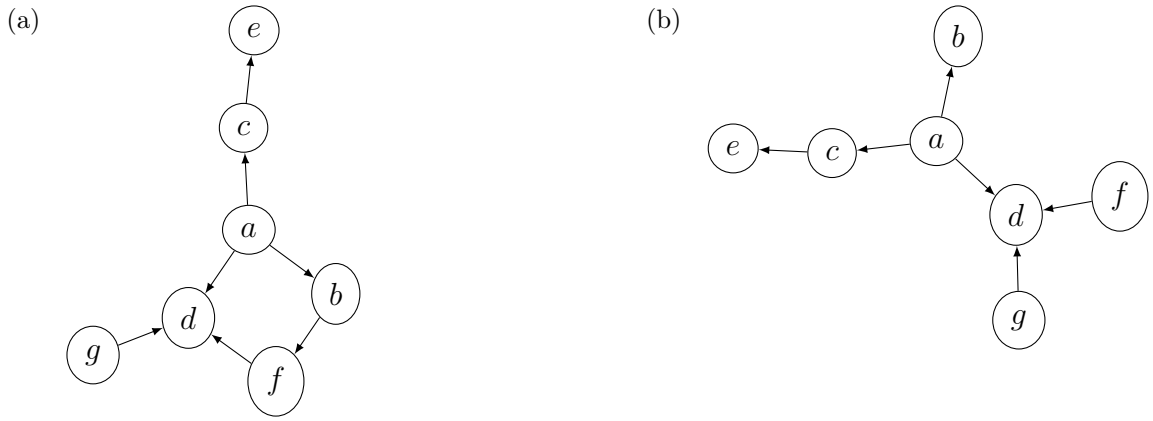

Figure 2.9: (a) DAG. (b) Di-tree.

2.4 Graph theoretic approach to control theory

Given a system (A, B, C), we can visualize its structure (equivalent class) as a digraph with vertices which represent the states, inputs and outputs of the system and edges representing the non-zero interactions. It turns out there exists a criteria to the structural controllability (i.e. $C = I_n$) of the system in its digraph constructed. This approach was initiated by Lin [14] for single-input systems and further extended to multi-input systems by Shields and Pearson [30]. Further results concerning (A, B, C) were shown by Poljak [29]. There exist weaker criteria for the case where C is non-singular as shown by Murota and Poljak [23].

In general, a vertex in a digraph can be split into its incoming and outgoing counterpart as we have seen in the bipartite representation of a digraph. The row vectors and column vectors of a matrix play a similar role. It is well known that every matrix represents a linear transformation from its column space to its row space (induced by left multiplication of the matrix to a column vector). We identify the vectors of a matrix as vertices in a graph. More precisely, the row vectors and column vectors of a matrix correspond to the incoming vertices and outgoing vertices.

For a given system (A, B, C), recall that n is the number of state variables, m is the number of input variables and ℓ is the number of output variables. Assuming the standard basis is used, we construct a vertex for each variable of the system and label

them as follows:

$$\mathcal{V} = \{v_1, v_2, \ldots, v_n\} \text{ (state vertices)},$$
$$\mathcal{U} = \{u_1, u_2, \ldots, u_m\} \text{ (input vertices)},$$
$$\mathcal{C} = \{c_1, c_2, \ldots, c_\ell\} \text{ (output vertics)}.$$

The matrix A represents a transformation within $\mathcal{V}$; matrix B gives the interactions from $\mathcal{U}$ to $\mathcal{V}$; and similarly matrix C is a transformation from $\mathcal{V}$ to $\mathcal{C}$. Hence, we can construct the directed edges to represent all the non-zero interactions as follows:

$$E_A = \{v_j v_i \mid A_{ij} \neq 0\} \text{ (structural state edges)},$$
$$E_B = \{u_j v_i \mid B_{ij} \neq 0\} \text{ (structural input edges)},$$
$$E_C = \{v_j c_i \mid C_{ij} \neq 0\} \text{ (structural output edges)}.$$

In our application, we assume that every row of C has exactly one non-zero entry, which gives a bijection between $\mathcal{C}$ and a subset $\mathcal{C}_\mathcal{V}$ (called the *target set*) of $\mathcal{V}$.

Definition 2.4.1. The (structural) digraph $G(A, B, C)$ of a system consists of the above constructed vertices $\mathcal{U} \cup \mathcal{V} \cup \mathcal{C}$ and edges $E_A \cup E_B \cup E_C$. Occasionally, we also consider the digraph $G(A, B)$ without output, i.e. $(\mathcal{U} \cup \mathcal{V}, E_A \cup E_B)$. Similarly, $G(A)$ for the digraph $(\mathcal{V}, E_A)$.

We have the first result regarding the digraph $G(A, B)$ and structural controllability of (A, B) in terms of cacti covering.

Definition 2.4.2. Let (A, B) be a system. The *cacti covering number* $\mu(A, B)$ of the system is the maximum cardinality of state vertices subset that can be covered by a $\mathcal{U}$-rooted cacti configuration in $G(A, B)$, i.e.

$$\mu(A, B) = \max\{|\mathcal{S}| : \mathcal{S} \subseteq \mathcal{V} \text{ and } \mathcal{S} \text{ is covered by a cacti configuration in } G(A, B)\}.$$

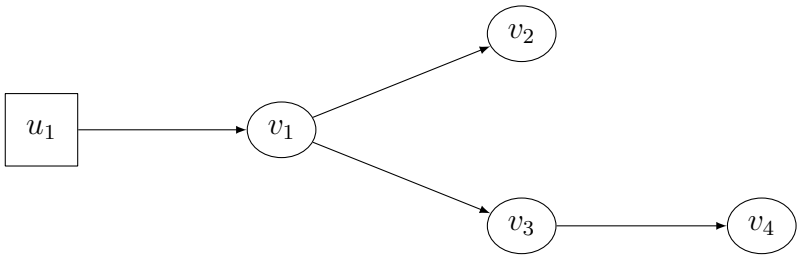

Figure 2.10: Digraph of the uncontrollable system in Example 2.2.2.

Theorem 2.4.3 ([12]). *The system (A, B) is structurally controllable iff it has a $\mathcal{U}$-rooted cacti configuration in $G(A, B)$. Furthermore, we have the equality*

$$\mu(A, B) = \mathrm{gd}(A, B).$$

The second result concerning congestion-free linkings in $G(A, B)$ and $\mathrm{gd}(A, B)$ was shown by Poljak [29]. His work was formulated in terms of dynamic graph. Dynamic graph provides a graph interpretation of the determinant of any subsquare matrix of $\mathcal{OC}(A, B, C)$.

Definition 2.4.4. Let (A, B) be a system. Its *congestion-free linking number* $\lambda(A, B)$ is defined as the maximum size of $(\mathcal{U}, \mathcal{V})$-congestion-free linkings (of length at most n) in $G(A, B)$.

Theorem 2.4.5 ([29]). *The system (A, B) has equal congestion-free linking number and cacti covering number,*

$$\lambda(A, B) = \mu(A, B).$$

Hence, $\lambda(A, B) = \mathrm{gd}(A, B)$.

In Fig. 2.10, which we saw in previous section that the corresponding system has generic dimension 3, the maximum u_1-rooted cacti covered set $\{v_1, v_3, v_4\}$ is covered by the dipath $u_1v_1v_3v_4$. On the other hand, $\{u_1v_1, u_1v_1v_3, u_1v_1v_3v_4\}$ forms a $(u_1, \mathcal{V})$-congestion-free linking of size 3.

Further result about structural controllability was obtained for flexible interac-

tions: whether the structural controllability will be affected if the strength of the existing interactions of a system (A, B) changes over time. Mathematically, we are looking for the rank of

$$[B_{n-1}, A_{n-1}B_{n-2}, \ldots, A_{n-1}A_{n-2}\cdots A_1B_0],$$

where $B_k \sim B$ and $A_k \sim A$ for all $k = 0, 1, \ldots, n-1$. It turns out that the above relaxed controllability matrix has the same rank as $\mathcal{OC}(A, B)$[29].

We summarize the results for structural controllability discussed as the following:

Theorem 2.4.6 (Four number equality[29]). *For a system (A, B), the following quantities are equal:*

1. *(cacti covering number)*

$$\mu(A, B),$$

2. *(generic dimension)*

$$\mathrm{gd}(A, B),$$

3. *(relaxed generic dimension)*

$$\mathrm{rank}\,[B_{n-1}, A_{n-1}B_{n-2}, \ldots, A_{n-1}A_{n-2}\cdots A_1B_0],$$

   where $B_k \sim B$ and $A_k \sim A$ for all $k = 0, 1, \ldots, n-1$,

4. *(congestion-free linking number)*

$$\lambda(A, B).$$

Unfortunately, there exists no such equality for output system when C is non-singular. A counterexample was shown by Murota and Poljak [23].

We extend our definitions from (A, B) to (A, B, C) by replacing every state vertices set $\mathcal{V}$ with the selected target vertices set $\mathcal{C}_{\mathcal{V}}$ and still consider the digraph $G(A, B)$.

Definition 2.4.7. Let (A, B, C) be a system. The *cacti covering number* $\mu(A, B, C)$ of the system is the maximum cardinality of target subsets that can be covered by a $\mathcal{U}$-rooted cacti configuration in $G(A, B)$, i.e.

$$\mu(A, B) = \max\{|\mathcal{S}| : \mathcal{S} \subseteq \mathcal{C}_{\mathcal{V}} \text{ and } \mathcal{S} \text{ is covered by a cacti configuration in } G(A, B)\}.$$

Definition 2.4.8. Let (A, B, C) be a system. Its *congestion-free linking number* $\lambda(A, B, C)$ is defined as the maximum size of $(\mathcal{U}, \mathcal{C}_{\mathcal{V}})$-congestion-free linking (of length at most $n + 1$) in $G(A, B)$.

Instead of a beautiful equality, we have an inequality for the quantities.

Theorem 2.4.9 ([23]). *For a system (A, B, C), we have the inequality*

$$\mu(A, B, C) \leq \mathrm{gd}(A, B, C) \leq \lambda(A, B, C).$$

We conclude this section by a few examples showing cases for strict inequalities.

Example 2.4.10. Consider the uncontrollable system (A, B, C)

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ e_0 & 0 & 0 & 0 \\ e_1 & 0 & 0 & 0 \\ 0 & 0 & e_2 & 0 \end{pmatrix}, B = \begin{pmatrix} e_3 \\ 0 \\ 0 \\ 0 \end{pmatrix}, C = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The matrix C indicates our target vertices set is $\{v_2, v_4\}$. Its output controllability matrix is

$$\mathcal{OC}(A, B, C) = \begin{pmatrix} 0 & e_0e_3 & 0 & 0 \\ 0 & 0 & e_1e_2e_3 & 0 \end{pmatrix}$$

which has rank 2. Fig. 2.11 shows its digraph and clearly $\mu(A, B, C) = 1$ as it is a u_1-rooted di-tree.

Murota and Poljak [23] presented the following example in which all the inequalities are strict.

Example 2.4.11 ([23])**.** Consider the system (A, B, C)

$$A = \begin{pmatrix} 0 & e_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e_4 & e_6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e_7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_8 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ e_9 \\ e_{10} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

and

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The target set is $\{v_1, v_2, v_3, v_7, v_9\}$. Its output controllability matrix has rank 4. A code written in Sage for computing the rank is in the appendix. Fig. 2.12 shows that $\mu(A, B, C) = 3$ by choosing the di-path $u_1v_4v_3v_1$. One can easily construct a $(u_1, \mathcal{C}_\mathcal{V})$-congestion-free linking of size 5 by exploiting the di-cycle. A non-computational proof using dynamic graph for the generic dimension can be found in [23].

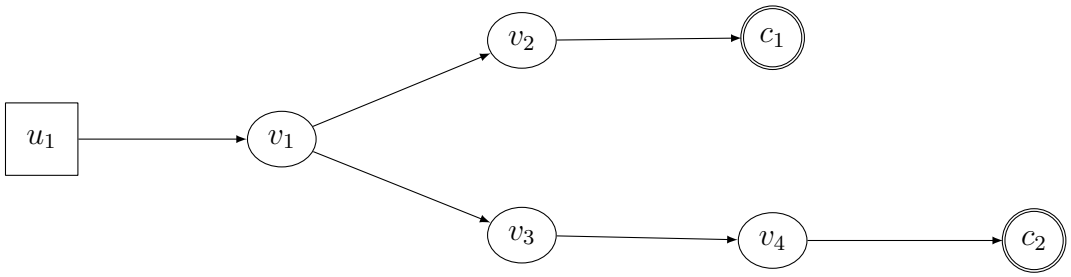

Figure 2.11: System with $\mu(A, B, C) = 1 < \mathrm{gd}(A, B, C) = 2$.

2.5 Computational complexity theory

In this section, we will merely focus on *computational efficiency* rather than *computability*. Roughly speaking, we shall only focus on problems solvable by a computer. In particular, we are interested in the comparison between two quantities, the "time" or "number of steps" spent on solving a problem instance; and the input size of a problem instance. Among all solvable problems, we will look deeply on the class **NP-HARD**. Without being too technical, only the intuitions and ideas behind the terminologies and the class of **NP-HARD** problems will be mentioned. For our purpose, it is sufficient to study a few properties of the 3SAT problem. The reduction discussed in this section will be the polynomial-time reduction (the reduction used in our proof in Ch. 4 is a stronger logspace reduction. This does not affect the NP-hardness obtained for our results). A rigorous treatise of computability can be found in BRIDGES [5]. ARORA AND BARAK [1] and PAPADIMITRIOU [28] contain various complexity classes and their hierarchies. GAREY AND JOHNSON [10] contains a few **NP-HARD** problems and their NP-hardness proofs.

The computational model chosen is the *multi tape Turing machine* (TM) under *binary encoding scheme*. The *running time* of a "problem", with respect to a given Turing machine, is a function of the input size of the task with output measured as the "number of steps" taken by the Turing machine to halt (and return desired results). A "problem" is said to be *tractable* if there is a Turing machine such that the running time to "solve" any problem instance is bounded above by a polynomial

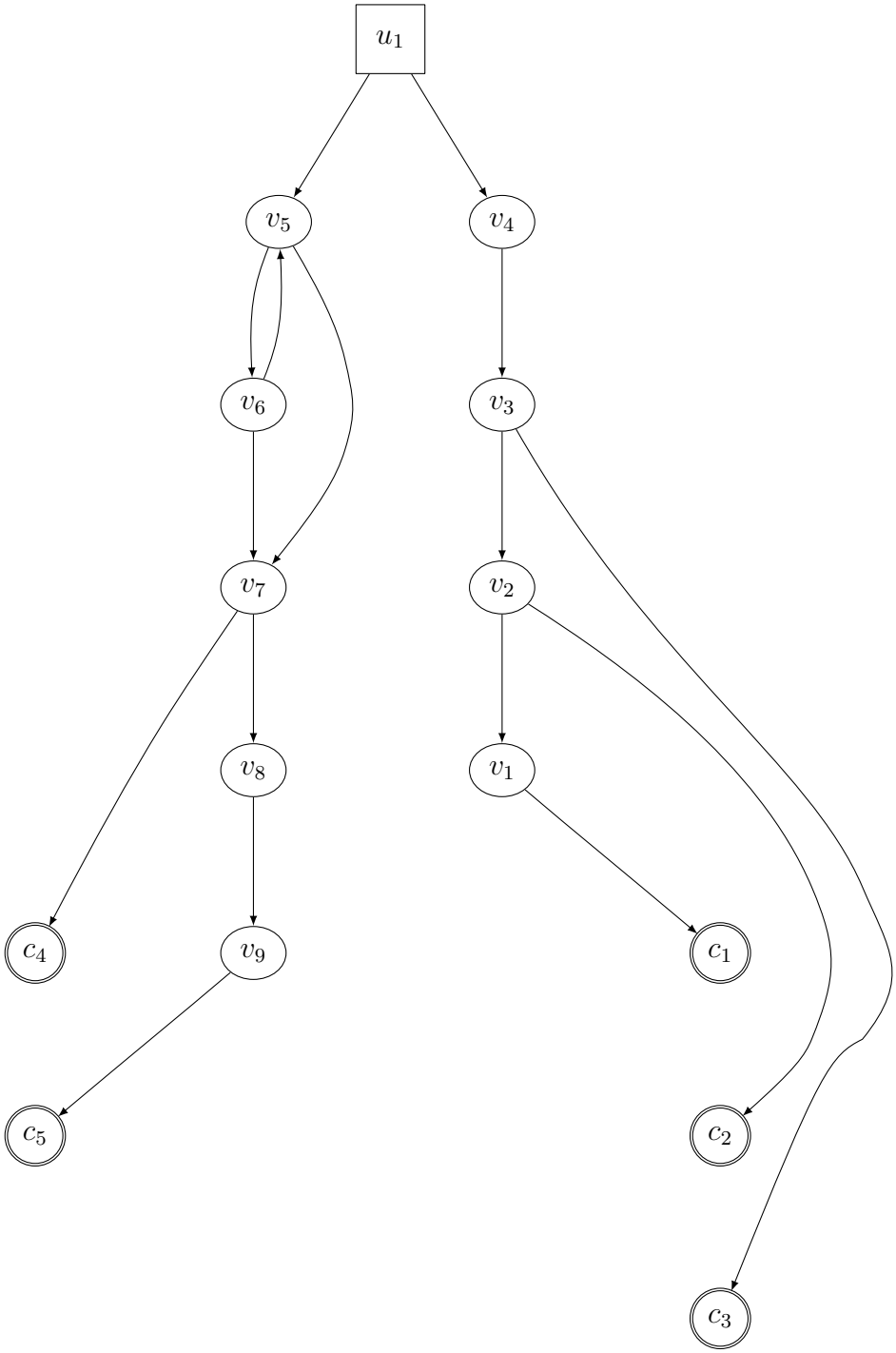

Figure 2.12: System with $\lambda(A, B, C) = 5, \mathrm{gd}(A, B, C) = 4$ and $\mu(A, B, C) = 3$.

in its input size. The corresponding Turing machine is said to be a polynomial time TM.

A TM that solves a problem captures the notion of algorithm. We will use the words "algorithm" and "Turing machine" interchangeably. An *efficient algorithm* means its running time is polynomial in input size.

Optimization problems and decision problems are two type of natural problems that arise in daily life. In an optimization problem, we expect the solution having the sense of being "the best" under some reasonable comparison scheme; and there may be various answers for a particular problem instance. In contrast, either "yes" or "no" can be the final output for a decision problem. However, the above two types of problems can be transformed to each other easily.

Consider the following optimization problem, called the SUBSET-SUM,

Example 2.5.1. You are given a bag of capacity K. A list of items of given weights $\boldsymbol{w} = w_i$'s is known. Maximize the total weights your bag can store.

Mathematically, given a sequence of n positive integers $(w_i)_{i=1}^n$ and an upper limit K, find a sub-sequence (w_{n_k}) such that

$$\sum w_{n_i} \leq K \text{ and } \sum w_{n_i} \text{ is maximum.}$$

and its decision version,

Example 2.5.2. You are given a bag of capacity K. A list of items of given weights w_i's is known. Can your bag hold items with total weight at least k?

Mathematically, given a sequence of n positive integers $(w_i)_{i=1}^n$, an upper limit K and a goal k, is there a sub-sequence (w_{n_k}) such that

$$k \leq \sum w_{n_i} \leq K.$$

We observe that by adding an extra variable (the variable k as in above), an

optimization problem is transformed into a decision problem. For simplicity, we focus only on decision problems in order to discuss *running time*.

An instance of the above decision problem is given by (with weights represented as a vector):

$$\begin{aligned} K &= 10 \\ \boldsymbol{w} &= (1, 2, 3, 1, 1, 1, 2) \\ k &= 11 \end{aligned}$$

The problem instance size (number of bits), ignoring the symbols $K, \boldsymbol{w}, k$ and the equal signs, is the number of bits of the input used to encode the instance. We need $\lceil \lg 10 \rceil$ for K, $\lceil \lg 7 \rceil$ for k and $4 \lceil \lg 1 \rceil + 2 \lceil \lg 2 \rceil + \lceil \lg 3 \rceil$ for $\boldsymbol{w}$, and therefore the problem instance size is $4 + 3 + 4\,(1) + 2\,(2) + 2 = 17$.

It is obvious that $\sum \boldsymbol{w} = 11 > K$, hence the answer for this decision instance is "no". In general, the same heuristic cannot be applied, say when $k = 7$ or $\boldsymbol{w}$ is modified. It is only capable of solving a particular instance of the problem but not the problem itself. What we are interested is a "method" that is capable of solving every instance of a given problem, which is a Turing machine (or an algorithm) that solves this problem. Clearly, we can try every possible way of choosing the items and record the maximum possible weight. In this case, there are in total 2^7 possibilities. In general, there are $2^{|\boldsymbol{w}|}$ possibilities, where $|\boldsymbol{w}|$ denotes the number of items, a quantity always less than or equal to its encoding size. Thus, we know that the running time of checking every possibility has no polynomial upper bound.

Nowadays, it is still open whether such a polynomial time algorithm exists for the SUBSET-SUM problem unless the **P** vs. **NP** problem is answered. Although it is hard to think of an efficient way to solve the SUBSET-SUM problem, one can verify whether a given configuration satisfies the problem requirements easily. For example, let us consider the instance $k = 8$, $K = 10$ and $\boldsymbol{w}$ remains the same. We can reject the

chosen items $(1, 3, 1, 1, 1)$ by realizing the sum is 7. We accept the item list $(1, 2, 3, 2)$ by checking its sum being less than K and equal k. This checking process can be done efficiently (in polynomial time). A *nondeterministic Turing machine* (NTM) is a standard TM with the additional ability to try all configurations at the same time. Its running time is measured as the maximum time spent among every individual trial. For a given problem, assume there is an efficient way to verify whether an arbitrary configuration satisfies the requirements. Then, we can create a polynomial time NTM that try every possible configuration as long as the configurations can be efficiently constructed. A problem belongs to the problem class **NP** if there is a NTM solving it in polynomial time. For example, SUBSET-SUM is **NP**. The problem class **P** consists of all tractable decision problems, or equivalently, those that can be solved by a TM in polynomial time.

Finally, we come to the notion of hardness of a problem among a given problem class. Among the problem class **P** (or **NP**), is there any representative problem in the sense that it is the "hardest" problem in the class? An equivalent question is whether there exists a problem which is at least as hard as all the other problems in a given class. The term "as hard as" is understood as: if there is an algorithm solving the hardest problem, one could have solved all the other problems efficiently in the same class. The Cook-Levin theorem stated that both classes, **P** and **NP**, possess such a representative problem. A problem that is as hard as all problems in a given problem class is said to be a *hard problem* for that class. It is known that SUBSET-SUM is also one such problem for the class **NP**[2]. The hard problem class is denoted by its class name with suffix **HARD**. Hence, SUBSET-SUM is **NP-HARD**. A problem is said to be *complete* in a class if it belongs to the same class and is a hard problem for that class. Similarly, the complete problem class is denoted by its class name with suffix **C**. We saw that SUBSET-SUM is **NP**, hence it is also **NP-C**.

We conclude this chapter by the *boolean satisfiability problems* SAT and 3SAT; and

[2]See [1, Ch. 2] and the chapter exercise.

Karp's many-one reduction.

Definition 2.5.3. Let $x_1, x_2, \ldots, x_n$ be n propositional variables.

A *literal* is of the form either x_k or $\neg x_k$, where $\neg$ is the logical negation. Two literals are *distinct* if they arise from different proposition variables.

A *clause* is a formula which only consists of distinct literals and logical disjunctions ($\vee$), i.e. it is of the form

$$x_{r_1} \vee x_{r_2} \vee \cdots \vee x_{r_p} \vee (\neg x_{s_1}) \vee (\neg x_{s_2}) \vee \cdots \vee \left(\neg x_{s_q}\right),$$

where r_k's and s_k's are distinct integers. With the common operator precedence that $\neg$ has higher priority than $\vee$, we usually drop the parentheses in a clause. Here, we specifically require that every propositional variable appears at most once in a clause. Though, technically, one can relax the restriction and arrive the same results in complexity theory.

A *conjunctive normal form (CNF)* of a boolean formula ϕ over the n propositional variables $x_1, x_2, \ldots, x_n$ is of the form

$$(A_1) \wedge (A_2) \wedge (A_3) \wedge \cdots \wedge (A_m),$$

where A_k's are clauses, called the *clauses* of the formula, and $\wedge$ is the logical conjunction. Equivalently, one can say that a CNF is a conjunction of m clauses. It can be shown that every boolean formula has a CNF.

A *boolean assignment* is a function $\boldsymbol{x}$ from the n propositional variables to $\{0,1\}^n$, i.e. $\boldsymbol{x} : \{x_1, x_2, \ldots, x_n\} \to \{0,1\}^n$. The *truth value* of a formula ϕ under the assignment $\boldsymbol{x}$ is denoted $\phi(\boldsymbol{x})$, which is clearly either 0 or 1. The truth value of a formula is evaluated under usual logical operations.

A formula ϕ is *satisfiable* if there exists a boolean assignment $\boldsymbol{x}$ such that $\phi(\boldsymbol{x}) = 1$. The corresponding assignment is said to be a *satisfiable assignment*. Otherwise, the fomula is said to be *unsatisfiable*.

Example 2.5.4. A satisfiable formula in CNF is $\phi_1 = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$ with satisfiable assignment $\boldsymbol{x} = (1, 0)$. An unsatisfiable formula in CNF is given by $\phi_2 = (x_1) \wedge (\neg x_1)$.

Note that in both formulas, the variable x_1 appears twice in the formulas (and also x_2 for ϕ_1), but it appears only once in each clause. Thus, ϕ_1 and ϕ_2 satisfy the definition of CNF.

With the above notions, we can define the decision problems SAT and 3SAT.

Problem 2.5.5 (SAT)**.** Given a boolean formula ϕ over n propositional variables. Is ϕ a satisfiable formula?

An equivalent formulation is that given m clauses over n propositional variables, does there exist a boolean assignment such that all the m clauses has truth value 1?

Problem 2.5.6 (3SAT)**.** Given a boolean formula ϕ, where each clause consists of exactly 3 distinct literals, over n propositional variables. Is ϕ a satisfiable formula?

The Cook-Levin theorem states that the above two problems are **NP-HARD**. It is obvious that both problems are **NP** by a direct evaluation of any trial boolean assignment.

Theorem 2.5.7 (Cook-Levin)**.** SAT *and* 3SAT *are* ***NP-C****.*

The above result indicates that if we are lucky enough to find an efficient method to tackle the SAT problem, this method automatically gives us another efficient method to solve the SUBSET-SUM problem, though they appear to be completely different in essence. This automatic process is due to Karp's many-one reduction, which is also a key tool to prove that other problems are **NP-HARD** based on a known **NP-HARD** problem.

In order to understand the notion of Karp's many-one reduction, it is important to pay attention to the encoding of a problem. To be precise, the encoding of both SAT and 3SAT are the number of propositional variables n, the number of clauses m of the formula, and the structure of each clause.

Example 2.5.8. One possible encoding for $\phi_1 = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$ is

$$\begin{aligned} n &= 3 \\ m &= 2 \\ A_1 &= (1, 2, -3) \\ A_2 &= (-1, -2, 3)\,. \end{aligned}$$

In short, we can use the express the encoding as $3; 2; (1, 2, -3)\,; (-1, -2, 3)$.

Definition 2.5.9. A *Karp's reduction from problem A to problem B* is a Turing machine T that transforms the encoding of any input x_A of problem A to an input Tx_A of problem B satisfying the following:

1. T runs in polynomial-time (or use "memory" in logarithmic order of input size);
2. The problem instance Tx_A is true in problem B if and only if x_A in problem A is true.

We denote $A \leq_p B$ if there is a Karp's reduction from problem A to problem B.

If T runs in polynomial time, it is said to be a *polynomial time reduction.* If T uses "memory" in logarithmic order, it is said to be a *logspace reduction.* It is known that any TM running in logspace memory also runs in polynomial time[3].

Furthermore, for any two problems A and B, if there is an efficient algorithm κ to solve B and $A \leq_p B$, we can solve problem A by applying the corresponding reduction T to any input instance x; then using the algorithm κ on the transformed input Tx, which is an instance of problem B. The two steps can be done within polynomial time, and hence the composition $\kappa \circ T$ gives an efficient algorithm solving problem A. Usually, in order to show that a problem is **NP-HARD**, it suffices to find a reduction from 3SAT to the problem concerned.

[3]See [1, Ch.4] and [28, Ch. 8]

Finally, it is important to realize the existence of some problems which are **NP-HARD** but not **NP**, namely, the SUCCINCT 3SAT problem [28, Ch. 20]. Thus, it is not surprising that our work only leads to NP-hardness but not completeness.

Chapter 3

Recent work

The application of control theory on complex network was introduced in [17]. Liu focused on structural controllability of the entire system and studied the necessary nodes in the network for achieving controllability of the entire system. In 2014, the method was extended to a preselected subset of nodes [9], which was named *target control*. In 2015, the application to identification of drug targets was studied by [34], in which they considered single-target drugs for biological system. Both [9] and [34] claimed optimality for their solutions obtained. We will discuss the approaches above and analyze their correctness.

In this chapter, only structural properties are of interest. Whenever "controllability" is discussed, it should be understood as structural controllability.

3.1 Controllability of the entire complex network

The fundamental question studied by Liu *et al.* [17] can be described as: given a system, in order to make it controllable, find the necessary nodes to be connected by external inputs.

It is obvious that the system is controllable by connecting to every node a distinct external input, but this solution is practically unrealistic and meaningless. Naturally,

[17] considered the problem as an optimization problem where one minimizes the number of external inputs required and identifies the set of corresponding nodes to be connected which they called *controlled nodes*. Among the controlled nodes found, each external input is paired with a controlled node so that an injection is established. They named the paired controlled nodes *driver nodes*.

In order to search for the minimum set of driver nodes, or equivalently the least number of input nodes, they looked for a maximum directed matching in the given system $G(A)$.

Algorithm 3.1.1. *Given a system $G(A)$, the algorithm runs as follows:*

1. *Construct the bipartite representation $\mathcal{B}(G(A))$ of $G(A)$.*
2. *Obtain a maximum matching $\mathcal{M}$ by any polynomial-time algorithm on $\mathcal{B}(G(A))$.*
3. *If $\mathcal{M}$ is a perfect matching, connect an external input to any vertex.*
4. *Otherwise:*
   (a) *For each unmatched vertex, create an external input vertex and an edge from the input to the unmatched vertex.*
   (b) *For each di-cycle in the matching, add an edge connecting from any one of the input to any vertex of the di-cycle whenever necessary.*

Notice that the first two steps obtain a maximum di-matching in $G(A)$ in polynomial time, which we will omit in further refinement of the algorithm. Clearly, the remaining steps can be done within polynomial time.

As a maximum di-matching corresponds to SCCs, the correctness of the algorithm can be justified by the facts that $G(A, B)$ is controllable if and only if the cacti covering number is the number of state vertices; and that the covering number never decreases if an additional edge from $\mathcal{U}$ to a di-cycle is added.

3.2 Target controllability

In 2014, instead of controllability, Gao *et al.* [9] focused on output controllability. The problem studied can be described as: Given an output system with a target set of vertices, find the necessary nodes to be connected by external inputs so that the system is output controllable. They presented a refined version of the algorithm for controllability of an entire network trying to minimize the number of external inputs. However, a subtle flaw can be found in their proof for both the correctness and optimality of the refined algorithm. We will present counterexamples to the refined algorithm for both cases and study the proof for the algorithm.

Algorithm 3.2.1. *Given a system $G(A)$ and a target set of vertices $\mathcal{C}_{\mathcal{V}}$:*

1. *Set the objective matching vertices $\mathcal{O} = \mathcal{C}_{\mathcal{V}}$ and $k = 0$.*
2. *Obtain a di-matching $\mathcal{M}$ such that maximum number of vertices in $\mathcal{O}$ is matched.*
3. *Let D_k be the unmatched vertices and set $\mathcal{O}$ to be the matched vertices.*
4. *Repeat steps 2 and 3 for each iteration k to obtain the set of unmatched vertex D_k until $\mathcal{O} = \emptyset$, i.e. there are no matched vertices left.*
5. *$D = \bigcup_{k \geq 0} D_k$ is then the driver nodes.*

Note that in their terminology, driver nodes are state vertices which are connected with a new external input.

However, we discover that the algorithm either fails to seek minimum inputs or output controllability. Recall that the output system (A, B, C) appeared in Example 2.4.11 has generic dimension 4.

Our first counterexample to the optimality is constructed based on the example. Consider the system $G(A)$ with the target set $\{v_1, v_2, v_3, v_7\}$. Output controllability can be achieved by using exactly one input connected to $\{v_4, v_5\}$. However, the above algorithm states that one needs to connect to v_4 and v_5 each with a different input.

In this counterexample, the returned set $\{v_4, v_5\}$ is indeed the controlled nodes for the system.

We also construct a similar counterexample to show that the algorithm does not always return a set of controlled nodes of a system. When this is the case, one cannot guarantee output controllability. To illustrate, consider the same system $G(A)$ with target set $\{v_7, v_9\}$. The only possible returned node is v_5 by the fact that $d^-_{G(A)}(v_k) > 0$ for $k > 5$. A simple computation shows that the corresponding generic dimension is always 1. Furthermore, with the target set $\{v_1, v_7, v_9\}$ one can obtain a result with neither minimality nor output controllability.

Notice that the main idea of the algorithm is to produce a graph of maximal linking number greater than or equal to the number of target vertices. However, as discussed in the last chapter, for a general output system there is no equality between the cacti covering number, the generic dimension and the maximal linking number. Thus, possession of a linking structure merely gives no information (in the sense of mathematical rigor) about the generic dimension. In their proof, they mistakenly applied the equality of the four numbers in Theorem 2.4.6 for system (A, B). For a general output system (A, B, C), the quantities are governed by an inequality in Theorem 2.4.9. As stated in the previous chapter, the inequality can be strict. Nevertheless, we believe that the algorithm behaves quite well and closely approximates the generic dimension.

3.3 Drug target identification

Controllability for biomolecular network was studied in [34], in which they called the set of controlled nodes *steering nodes*. The application in biomolecular system is to search for a minimum set of steering nodes to which one can potentially apply drugs to achieve output controllability. They assumed the drugs used are single-target, namely, every column of the input matrix constructed contains exactly one non-zero

entry. They modified the algorithm for entire complex network to a weighted version. Instead of finding a maximum matching, they formulated a weighted bipartite graph and queried for a maximum weight matching. The correctness of the algorithm relies on the existence of a cacti covering to the selected targets, but the existence is not enough for optimality. In Example 2.4.10, we can see that the output system has cacti covering number 1 but its generic dimension is 2. It is because no cacti configurations can cover the selected vertices v_2 and v_4. A counterexample for their algorithm can be constructed based on this observation, by setting the system $G(A)$ with target vertices $\{v_2, v_4\}$.

3.4 Conclusions

We have seen two algorithms which fail to obtain optimal solutions. The main reason behind is merely the assumption of the equality of the quantities: μ, gd and λ, or any equivalent conditions such as the necessity of the existence of a cacti covering or sufficiency of a maximal linking. Our work in the next chapter actually shows that the above problems for selected targets are **NP-HARD** under reasonable assumptions. Unless **P**=**NP**, there is no algorithm that can always find an optimal solution within polynomial time.

Chapter 4

Our results

In this chapter, we will formulate the problems in the previous chapter mathematically. A number of variants will be shown to be **NP-HARD**. The limitations of the techniques applied will also be discussed, hinting that no further results could be obtained from the same reduction.

4.1 Problem Formulation

The mutual aim of the recent work is to maximize "controllability" of a given system subjected to constraints that either minimize costs or side effects.

Problem 4.1.1 (Minimum input)**.** Given matrices A and C of size $n \times n$ and $\ell \times n$ respectively, find a matrix B of size $n \times m$ such that

1. $\mathrm{gd}\,(A, B, C)$ is maximum.

2. m is minimum among all matrices that maximize $\mathrm{gd}\,(A, B, C)$.

Problem 4.1.2 (Minimum affected states)**.** Given matrices A and C of size $n \times n$ and $\ell \times n$ respectively, find a matrix B of size $n \times m$ such that

1. $\mathrm{gd}\,(A, B, C)$ is maximum.

2. The number of non-zero rows in B is minimum among all matrices that maximize $\mathrm{gd}(A, B, C)$.

Their graph equivalent formulations are stated as follows:

Problem 4.1.3 (Minimum $\mathcal{U}$)**.** Given $G(A)$ and $\mathcal{C}_{\mathcal{V}}$. Add vertices $\mathcal{U}$ and edges to $G(A)$ such that

1. $\mathrm{gd}(A, B, C)$ is maximum, where B is the corresponding induced matrix.
2. $\mathcal{U}$ is minimum.

Problem 4.1.4 (Minimum neighbourhood)**.** Given $G(A)$ and $\mathcal{C}_{\mathcal{V}}$. Add vertices $\mathcal{U}$ and add edges to $G(A)$ such that

1. $\mathrm{gd}(A, B, C)$ is maximum, where B is the corresponding induced matrix.
2. $\left|N^1_{\hat{G}}(\mathcal{U})\right|$ is minimum, where $\hat{G}$ is the graph G with additional vertices $\mathcal{U}$ and edges.

Whenever it is clear from the context, we denote $\hat{G}$ the graph G with additional vertices $\mathcal{U}$ and edges, i.e. $\hat{G} = G(A, B, C)$.

We define the notion of optimization for full controllability and target controllability as follows:

Definition 4.1.5. The *Full control optimization problem (FCP)*:

Input: Matrix A of size $n \times n$.

Output: Matrix B of size $n \times m$ such that

1. $\mathrm{gd}(A, B) = n$.
2. m is minimum among all feasible matrices.

Definition 4.1.6. The *Target control optimization problem (TCP)*:

Input: Matrix A of size $n \times n$ and matrix C of size $\ell \times n$, where every row of C has exactly one non-zero entry.

Output: Matrix B of size $n \times m$ such that

1. $\mathrm{gd}(A, B, C) = \ell$.

2. m is minimum among all feasible matrices.

Their corresponding decision versions are formulated naturally. We will focus on their decision versions from now on.

Definition 4.1.7. *Full control problem (FCP):*

Input: Matrix A of size $n \times n$ and an integer k.

Output: Yes iff there exists a matrix B of size $n \times m$, where $m \leq k$, such that $\mathrm{gd}(A, B) = n$.

Definition 4.1.8. *Target control problem (TCP)*:

Input: Matrix A of size $n \times n$, matrix C of size $\ell \times n$, where every row of C has exactly one non-zero entry, and an integer k.

Output: Yes iff there exists a matrix B of size $n \times m$, where $m \leq k$, such that $\mathrm{gd}(A, B, C) = \ell$.

4.2 Complexity for the problems

We will consider the following two variants of TCP. Their decision analogues will be proved to be **NP-HARD**.

Definition 4.2.1. The N-bounded TCP problem (N-TCP) is the TCP problem with the additional constraint that every input has outdegree bounded by N, i.e.

$$d^+_{\hat{G}}(u) \leq N \text{ for all } u \in \mathcal{U}.$$

Definition 4.2.2. The edge TCP optimization problem (ETCP) is the TCP problem with an extra optimization where the number of edges added is minimized among all solutions.

Its corresponding decision version is stated as follows:

Input: Matrix A of size $n \times n$, matrix C of size $\ell \times n$, where every row of C has exactly one non-zero entry, and integers k_1 and k_2.

Output: Yes iff there exists a matrix B of size $n \times m$, where $m \leq k_1$, such that $\mathrm{gd}\,(A, B, C) = \ell$ and $|[\mathcal{U}]| \leq k_2$, where $[\mathcal{U}] = \{uv \mid u \in \mathcal{U}, v \in \mathcal{V}\}$ is the edge-cut set.

We first justify the sufficiency of considering the decision version over the optimization version. Suppose there is an efficient algorithm for the optimization version, we can design an algorithm that invokes the algorithm for the optimization problem and then check the optimized solution to see if it satisfies the requirements of the decision problem. Therefore, any efficient algorithm for an optimization problem would lead to an efficient algorithm to its decision version.

4.2.1 Outdegree bounded TCP

The NP-hardness for N-PCP is shown for $N > 2$ below. With a slight modification of the constructions involved, the cases $N \leq 2$ will be proved.

Theorem 4.2.3. (N+1)-TCP *is* ***NP-HARD*** *for any* $N > 1$.

Proof. Let ϕ be an instance of 3SAT of m clauses $C_1, \ldots, C_m$ over n propositional variables $x_1, x_2, \ldots, x_n$. A few of gadgets are introduced for the reduction to be constructed.

P_j: A di-path of j vertices; P_j^I and P_j^T its initial and terminal vertex respectively.

P_j^k: The k^{th} vertex of the path P_j, where P_j^I is counted as the first vertex.

$\mathcal{D}_1$: A set of n disjoint copies of P_1, i.e. a set of n stable vertices.

$\mathcal{D}_k$: A set of n disjoint copies of P_k for $k > m$.

Similarly, we denote $\mathcal{D}_j^I$ and $\mathcal{D}_j^T$ the set of n initial and n terminal vertices.

$\boldsymbol{X}_i, \neg\boldsymbol{X}_i$: Literal vertices for x_i and $\neg x_i$, $i = 1, 2, \ldots, n$.

$\boldsymbol{T}_i$: The tautology vertex associated with the pair $(\boldsymbol{X}_i, \neg\boldsymbol{X}_i)$, $i = 1, 2, \ldots n$.

$\boldsymbol{CL}_i$: The clause vertex for the i^{th} clause, $i = 1, 2, \ldots, n$.

$\boldsymbol{U}_i$: The i^{th} input vertex, $i = 1, 2, \ldots$.

The reduction T which maps ϕ to an instance $T\phi = (A, C)$ of $(N+1)$-TCP is constructed as follows:

Construction of $G(A)$:

1. Literal construction:

   (a) Construct two literal vertices for each propositional variable x_i, one for itself and the other for its negation.

   (b) Let $\mathcal{X} = \{\boldsymbol{X}_k \mid k = 1, 2, \ldots, n\} \cup \{\neg\boldsymbol{X}_k \mid k = 1, 2, \ldots, n\}$ be the collection of literal vertices. For each literal $\boldsymbol{X} \in \mathcal{X}$:

      i. Construct a path of m vertices P_m , called the *literal path* $P_{\boldsymbol{X}}$ associated with $\boldsymbol{X}$.

      ii. Add an edge $\boldsymbol{X}P^I_{\boldsymbol{X}}$, which connects $\boldsymbol{X}$ to the initial of its literal path.

2. Tautology construction:

   (a) Construct a vertex $\boldsymbol{T}_i$, called the *tautology vertex*, for each propositional variable x_i.

   (b) Let $\mathcal{T} = \{\boldsymbol{T}_i \mid i = 1, 2, \ldots, n\}$ be the collection of tautology vertices. Add edges $\boldsymbol{X}_i\boldsymbol{T}_i$ and $\neg\boldsymbol{X}_i\boldsymbol{T}_i$ for each $\boldsymbol{T}_i \in \mathcal{T}$.

3. Discrete layer construction:

   (a) Construct $\mathcal{D}_1$.

   (b) Construct $\mathcal{D}_k$ for $k = m+3, m+4, \ldots, m+N+1$.

   Let $\mathcal{D} = \bigcup \mathcal{D}^T_k$ be the collection of all terminal vertices constructed.

4. Clause construction:

   (a) For each of the m clauses C_j:

      i. Add vertex $\boldsymbol{CL}_j$.

ii. Add an edge $P^j_{\boldsymbol{X}}\boldsymbol{CL}_j$ for each literal $\boldsymbol{X}$ appearing in the clause C_j, which connects the j^{th} vertex of the literal paths of the three literals of C_j to $\boldsymbol{CL}_j$.

Let $\mathcal{B} = \{\boldsymbol{CL}_j \mid j = 1, 2, \ldots, m\}$.

The above constructed graph $G(A)$ has the following properties:

1. It has at least $nN + 1$ weakly connected component as each $\mathcal{D}_k$ contains n such components and at least 1 for the literal vertices.

2. $d(\boldsymbol{X}_j, T_j) = 1$ and $d(\neg\boldsymbol{X}_j, T_j) = 1$ for $j = 1, 2, \ldots, n$.

3. $d(\boldsymbol{X}, \boldsymbol{CL}_j) = j + 1$ if $\boldsymbol{X}$ appears as a literal of the clause C_j; and ∞ otherwise.

4. The longest path from any literal vertex has length bounded by $m + 1$.

5. Only the discrete layers $\mathcal{D}_k$ contain paths longer than $m + 1$.

6. The longest path is of length $m + N$, which appears in $\mathcal{D}_{m+N+1}$.

Finally, we select the target set $\mathcal{C}_{\mathcal{V}} = \mathcal{T} \cup \mathcal{D} \cup \mathcal{B}$, which has exactly $(N + 1)n + m$ vertices and set the maximum number of inputs to be n.

To be precise, $T\phi$ is the above constructed instance with $G(A)$, $\mathcal{C}_{\mathcal{V}}$ and $k = n$. The construction can be done in logspace memory hence polynomial time. The solution matrix B we are looking for should have rank $(N + 1)n + m$. We will show that ϕ is satisfiable if and only if $T\phi$ has such a matrix B as solution by the below lemmas. □

Examples are given to illustrate the construction stages:

Example 4.2.4. Note that the first two stages of the construction are unique up to m and n. Fig. 4.1 shows the unique digraph after the two stages for any instance of 6 clauses over 4 variables.

Example 4.2.5. Let ϕ be a formula with 3 clauses over 4 variables, where the clauses are given by

$$\phi_1 = x_1 \vee x_2 \vee \neg x_3$$

$$\phi_2 = x_1 \vee x_3 \vee \neg x_4$$

$$\phi_3 = x_2 \vee \neg x_3 \vee x_4.$$

The corresponding constructed digraph (with its discrete layer omitted) is given in Fig. 4.2. Its discrete layer for the case 3TCP (outdegree bounded by $N = 3$) is shown in Fig. 4.3. The target vertices (the clause vertices, tautology vertices and the terminal vertices from all paths in the discrete layers) are drawn in grey. Note that there are exactly $n = 4$ paths in each discrete layer and in total $N = 3$ discrete layers. The digraph of the corresponding system of ϕ is the union of the two figures shown.

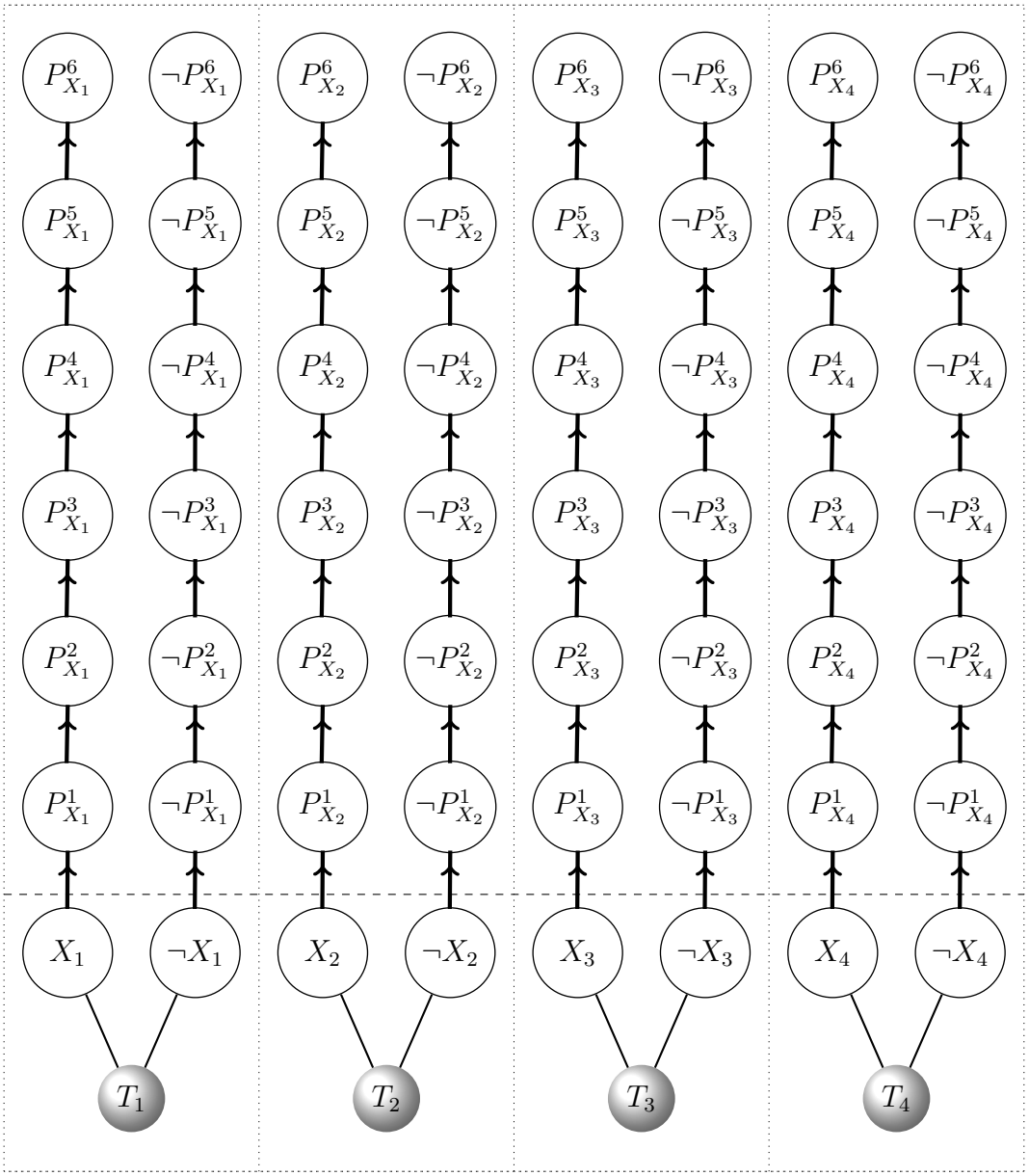

Figure 4.1: Literal and tautology construction for $m = 6$ and $n = 4$.

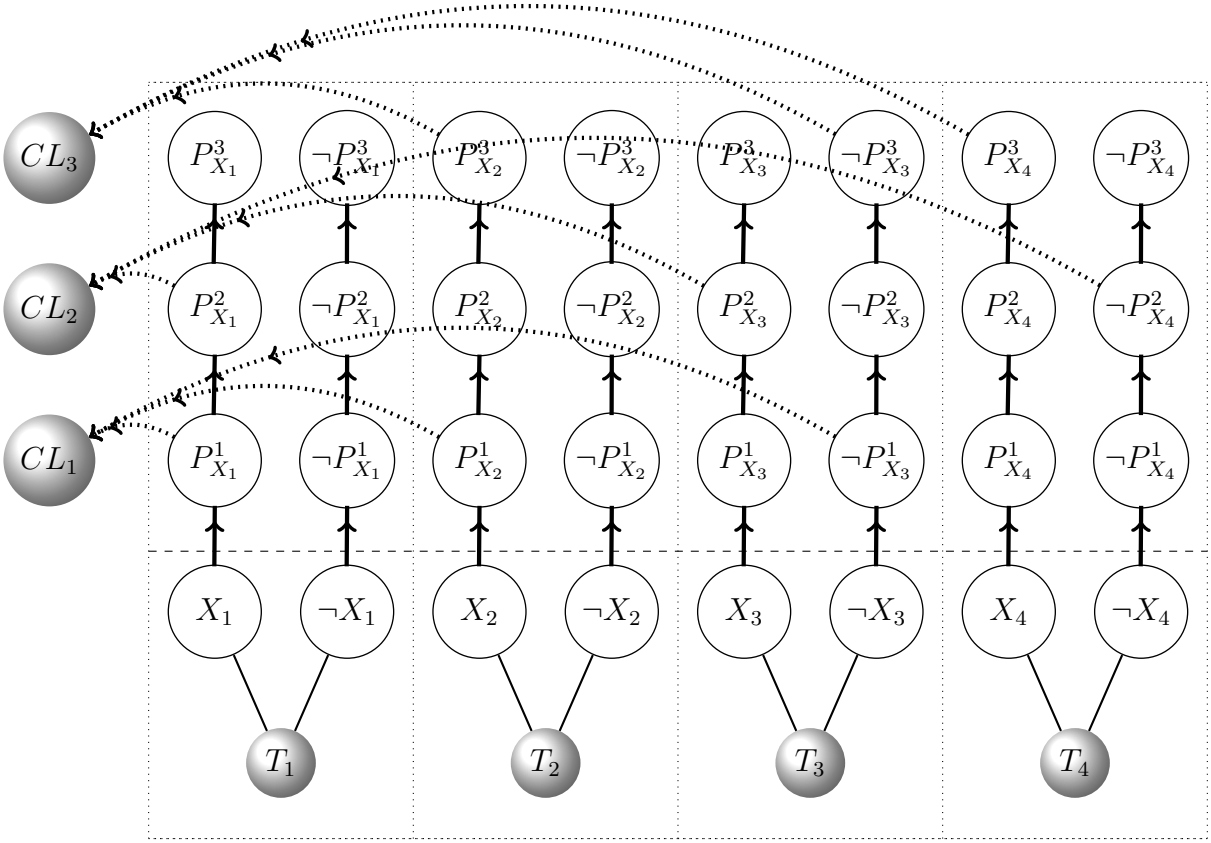

Figure 4.2: Digraph for $\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_3 \vee \neg x_4) \wedge (x_2 \vee \neg x_3 \vee x_4)$ with its discrete layer omitted.

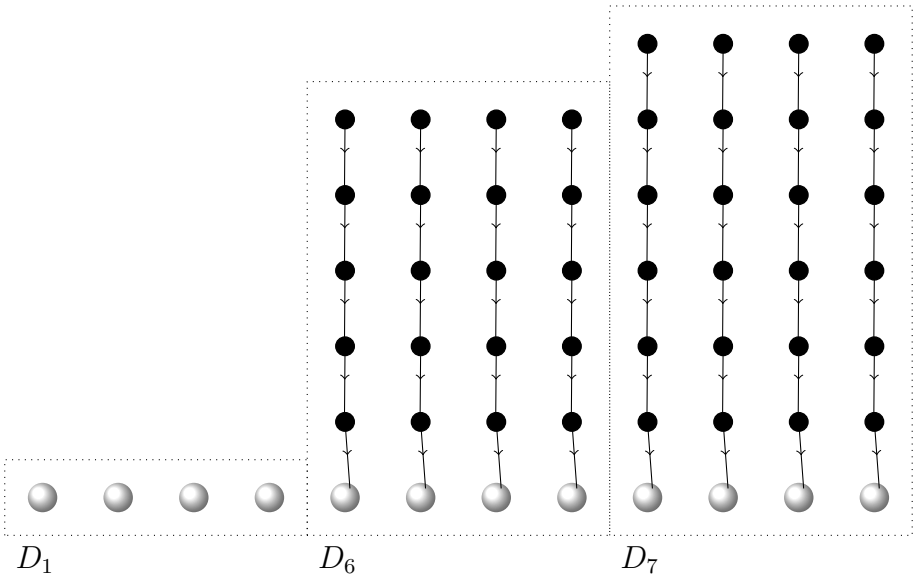

Figure 4.3: Discrete layer for $m = 3$ and $n = 4$ in 3TCP with the bottommost selected.

Lemma 4.2.6. *The system $T\phi$ is solvable in* (N+1)-TCP *if ϕ is satisfiable.*

Proof. Let $\boldsymbol{x}$ be a satisfiable assignment for ϕ. We construct the matrix B by giving the input vertices set $\mathcal{U} = \{\boldsymbol{U}_j \mid j = 1, 2, \ldots, n\}$ and edges as follows:

1. For $k = 1, m+3, m+4, \ldots, m+N+1$:
   (a) Choose any bijection between $\mathcal{U}$ and $\mathcal{D}_k^I$.
   (b) Add edges from $\mathcal{U}$ to $\mathcal{D}_k^I$ to indicate the chosen bijection.
2. According to the satisfiable assignment $\boldsymbol{x}$, add edges connecting from $\boldsymbol{U}_j$ to the corresponding chosen truth value either $\boldsymbol{X}_j$ or $\neg\boldsymbol{X}_j$ of x_j.

Note that $\mathcal{OC}(A, B, C)$ captures walks of different lengths from $\mathcal{U}$ to $\mathcal{C}$. The column vectors and the row vectors represent the corresponding outgoing and incoming vertices with each block representing different lengths of the walks. In particular, CA^jB contains all walks from $\mathcal{U}$ to $\mathcal{C}$ of length $j+2$. With the additional vertices $\mathcal{U}$ and $\mathcal{C}$, any path from $\mathcal{U}$ to $\mathcal{C}$ that induces a path in $G(A)$ must satisfy the same bound added by 2.

By considering the set $\mathcal{D}_{m+N+1}$, we know that the longest possible path in $G(A, B, C)$ has length at most $m+N+2 = (m+N)+1+1$ (1 for the output vertex connected from a terminal and an extra possible 1 for an input connected to an initial). Hence, CA^jB is a zero matrix for $j \geq m+N+1$.

Clearly, there is a unique path (of different lengths) from $\mathcal{U}$ to every vertex $v \in \mathcal{D} \cup \mathcal{B}$. Hence, the corresponding rows in $\mathcal{OC}(A, B, C)$ have exactly one non-zero entry, which are in CA^jB for $j = 0, m+2, m+3 \ldots, m+N$. As the entries in CA^jB are induced from bijections, no column contains more than one 1's. In particular, the rank of CA^jB is n, where the non-zero pivoting elements appear at exactly the rows representing the terminal vertices of $\mathcal{D}_{j+1}$. Thus, we have

$$\operatorname{rank} \mathcal{OC}(A, B, C) \geq Nn.$$

Consider all paths from $\mathcal{U}$ to $\mathcal{C}$ with length 3. By construction, these paths must pass through the tautology vertices $\mathcal{T}$. Furthermore, the satisfiable assignment induces a bijection from $\mathcal{U}$ to $\mathcal{T}$. Similar arguments show that CAB has rank n.

Finally, for each of the m clauses C_j, satisfiability implies the existence of k literals $(1 \leq k \leq 3)$ from which the clause receives the truth value 1. Hence, there exists k paths from $\mathcal{U}$ to $\boldsymbol{CL}_j$. By construction, the paths are all of length $(j+1)+2$ and these are the only paths from $\mathcal{U}$ to $\mathcal{C}$. Therefore, the corresponding matrix $CA^{j+1}B$ consists of exactly k 1's in the row representing the incoming vertex $\boldsymbol{CL}_j$; with the remaining rows being all zero. We can find a sequence of column operations acting on $CA^{j+1}B$ that only affects the corresponding k columns containing the 1's and results in a block with a unique non-zero entry. Thus, we can obtain an equivalent matrix for $\mathcal{OC}(A,B,C)$ with exactly $|\mathcal{C}| = (N+1)n + m$ pivoting vectors, i.e.

$$\operatorname{rank} \mathcal{OC}(A,B,C) = |\mathcal{C}|.$$

Obviously $d^+_{\hat{G}}(\boldsymbol{U}_j) = N+1$, hence the matrix B is a solution for $T\phi$. □

Lemma 4.2.7. *If the system $T\phi$ is solvable in* (N+1)-TCP*, then ϕ is satisfiable.*

Proof. Let B be a matrix satisfying the following:

1. $|\mathcal{U}| \leq n$.
2. $\operatorname{gd}(A,B,C) = |\mathcal{C}|$.
3. $d^+_{\hat{G}}(u) \leq N+1$ for all $u \in \mathcal{U}$.

By Thm. 2.4.9, we have $\operatorname{gd}(A,B,C) \leq \lambda(A,B,C)$. Choose any $(\mathcal{U}, \mathcal{C}_{\mathcal{V}})$-congestion-free linking of size at least $(N+1)n + m$ in $G(A,B)$.

Note that the outdegree bound implies $[\mathcal{U}]$ contains at most $n(N+1)$ edges; and that $G(A)$ has at least $nN+1$ weakly connected components. Among all the weakly connected components, nN of them are from the discrete layers, with each containing

exactly one target vertex. The congestion-free linking implies the existence of a path from $\mathcal{U}$ to each of the target vertices. Thus, at least one edge must be added for each of the nN weakly connected components in $G(A)$ in order to maintain connectivity.

The existence of n linkings from $\mathcal{U}$ to the tautology vertices $\mathcal{T}$ implies further n edges have to be added. In total, at least $n(N+1)$ edges must be added and all the inequalities become equalities.

It suffices to show that no two vertices in $\mathcal{U}$ can connect to the non-distinct literal pairs, and hence induces a boolean assignment.

Consider the stable set $\mathcal{D}_1$, since exactly n edges are added from $\mathcal{U}$ to $\mathcal{D}_1$ and $d(\mathcal{U}, v) = 1$ for any $v \in \mathcal{D}_1$. Thus, all the n paths of length 1 to $\mathcal{C}_\mathcal{V}$ are directed edges connected to $\mathcal{D}_1$.

Consider the tautology set $\mathcal{T}$, any path starting from $\mathcal{U}$ to $\mathcal{T}$ must pass through exactly one literal vertex since $d_{\hat{G}}(\mathcal{U}, \mathcal{T}) \leq 2$ and the stable set $\mathcal{D}_1$ used up all paths of length 1 in order to be congestion-free. In other words, there are no directed edges from $\mathcal{U}$ to $\mathcal{T}$ and the only edges related to $\mathcal{T}$ are connected from $\mathcal{U}$ to the literal vertices. As each tautology vertex has a corresponding pair of literals and only one of which is connected from $\mathcal{U}$, there are exactly n paths of length 2 which induces a boolean assignment $\boldsymbol{x}$.

The corresponding assignment $\boldsymbol{x}$ is a satisfiable assignment. For each of the clauses $\boldsymbol{CL}_j$, we can find a path from $\mathcal{U}$ to $\boldsymbol{CL}_j$. Observe that the path must contain one of the literal appeared in the assignment $\boldsymbol{x}$, which gives a truth value 1 to the clause $\boldsymbol{CL}_j$ by construction.

Hence, the boolean formula ϕ is satisfiable. □

The above results show that T is a Karp's reduction and (N+1)-TCP is **NP-HARD** for $N > 1$.

For the problems 1TCP and 2TCP, one can modify the discrete layer construction stage to arrive at the same NP-hardness.

Theorem 4.2.8. N-TCP *is* ***NP-HARD*** *for any* $N > 0$.

Proof. The cases $N \geq 3$ are shown. For $N = 1$, we simply remove the discrete layer construction. For the case $N = 2$, we only keep the stable set $\mathcal{D}_1$ in the discrete layer. The transformation is still valid as the same arguments apply. □

4.2.2 Edge bounded TCP

We proceed on proving ETCP is also **NP-HARD** by a slightly modified reduction.

Theorem 4.2.9. ETCP *is* ***NP-HARD****.*

Proof. Let ϕ be an instance of 3SAT of m clauses over n variables. The previously used gadgets $\mathcal{D}_k$ are modified (and extended) to contain one fewer copy of di-path for $k = 3, \ldots, m + 2$:

$\mathcal{D}_k$: a set of $n - 1$ disjoint copies of P_k for $2 < k \leq m + 2$.

The main idea for this modification is to associate each clause C_j with $\mathcal{D}_{j+2}$. The reduction is the same as the previous proof except in the discrete layer construction. We construct $\mathcal{D}_1, \mathcal{D}_3, \mathcal{D}_4, \ldots, \mathcal{D}_{m+2}$ instead.

Using the same target vertices set, we set the maximum number of inputs to be n and the maximum number of extra edges to be added be $m\,(n-1) + 2n$. The below lemmas show the equivalence between solvability of $T\phi$ and satisfiability of ϕ. □

Example 4.2.10. Let ϕ be the same formula in the previous example, i.e. $m = 3$ and $n = 4$. The only difference in the corresponding system is the discrete layer, where there are only $n - 1$ paths in $\mathcal{D}_k$ for $k > 1$. Fig. 4.4 shows the corresponding discrete layer in ETCP. Note that the number of discrete layers is $m + 1 = 4$.

Lemma 4.2.11. *The system $T\phi$ is solvable in* ETCP *if ϕ is satisfiable.*

Proof. We choose any satisfiable assignment $\boldsymbol{x}$ for ϕ. Establish edges between $\mathcal{U}$ to $\mathcal{D}_1$ by any bijection. Add edges from $\mathcal{U}$ to the literal vertices according to the assignment. Finally, for each clause C_j:

1. Choose $\boldsymbol{X}$ to be the first literal that gives C_j a truth value 1.

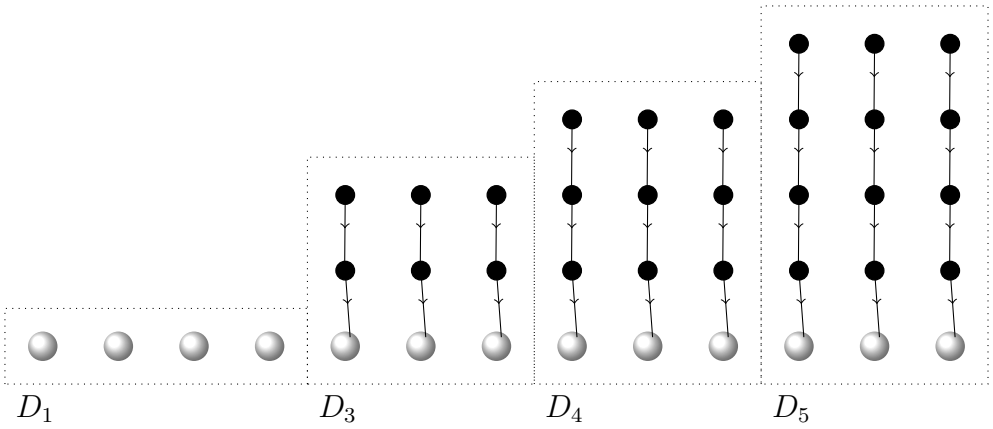

Figure 4.4: Discrete layer for $m = 3$ and $n = 4$ in ETCP.

2. Let $\boldsymbol{U}_i$ be the vertex connected adjacent to $\boldsymbol{X}$. Choose any bijection between $\mathcal{U} \setminus \{\boldsymbol{U}_i\}$ and $\mathcal{D}^I_{j+2}$, add corresponding edges from $\mathcal{U} \setminus \{\boldsymbol{U}_i\}$ to $\mathcal{D}^I_{j+2}$.

The number of edges added is given by $m\,(n-1)+2n$ (n for $\mathcal{D}_1$, n for literal vertices and $m\,(n-1)$ for the last step). In the matrix $\mathcal{OC}\,(A, B, C)$, only the part involving the clauses needs to be justified. The analysis for the other targets remains the same by path uniqueness. Consider the vertex $\boldsymbol{CL}_j$ which has k ($1 \leq k \leq 3$) non-zero entries corresponding to the number of satisfied literals of the clause. The entries appear in the block $CA^{j+1}B$. The block contains exactly one column with a unique non-zero entry and $k-1$ columns with exactly two non-zero entries as every input vertex has an edge to $\mathcal{D}_{j+2}$ except the one $\boldsymbol{U}$ corresponding to the first chosen literal. In additional to the row representing $\boldsymbol{CL}_j$, the $k-1$ non-zero entries appear at distinct rows being part of the bijection from $\mathcal{U} \setminus \{\boldsymbol{U}\}$ to $\mathcal{D}_{j+2}$. Again, we can find a sequence of column operations acting on $CA^{j+1}B$ to get exactly 3 pivoting vectors. Hence, $\mathcal{OC}\,(A, B, C)$ is equivalent to a matrix of rank $|\mathcal{C}|$. □

Lemma 4.2.12. *It the system $T\phi$ is solvable in* ETCP*, then ϕ is satisfiable.*

Proof. The number of weakly connected components in the constructed graph $G\,(A)$ is at least $m\,(n-1)+n+1$. Similar to the argument for N-TCP, from the congestion-free linking obtained, we can ensure $m\,(n-1)+n$ edges have to be added to the

$m\,(n-1)+n$ weakly connected components, corresponding to the $\mathcal{D}_j$ for $j = 1, 3, 4, \ldots, m+2$. The remaining n edges are connected to the literals for each tautology vertex by the exact same path length analysis. Satisfiability is inherited from connectivity between $\mathcal{U}$ and the clause vertices. Thus, we again obtain a satisfiable assignment. □

Therefore, ETCP *is* ***NP-HARD****.*

4.3 Limitations on the construction

In the proofs, the number of edges plays a key role to obtain a boolean assignment for the satisfiability of the boolean formula. A natural question arises, what happens if more edges are allowed? Do the same results hold if one relaxes the (vertices and edges) double optimization problem to a single optimization problem?

The answer is negative if one insists on using the exact same construction.

Example 4.3.1. Consider the boolean formula ϕ with all the 2^3 possible different clauses over 3 propositional variables x_1, x_2 and x_3, i.e all combinations of $(\pm 1, \pm 2, \pm 3)$.

If the number of vertices is relaxed and the edges are limited by the same bound, one can construct a new input vertex to each target and add a direct edge. This shows that even when ϕ is unsatisfiable, $T\phi$ is still solvable.

On the other hand, a solution to $T\phi$ with 3 vertices by using one edge more than the required bound, which is 22 edges, exists. Note that the assignment $\boldsymbol{x} = (1, 1, 1)$ satisfies 7 clauses except the clause $\neg x_1 \vee \neg x_2 \vee \neg x_3$. With $n = 3$ and $m = 7$, the edge bound for the 7 clauses formula is $m\,(n-1) + 2n = 20$. From the 3-vertex-and-20-edge solution for the 7 clauses (which exists and corresponds to $\boldsymbol{x}$), we can construct a 3-vertex-and-23-edge solution for the unsatisfiable formula ϕ as follows:

1. Add an edge from $\boldsymbol{U}_1$ to $\neg \boldsymbol{X}_1$.

2. Add edges according to any bijection from $\{\boldsymbol{U}_2, \boldsymbol{U}_3\}$ to the discrete set of two paths corresponding to the clause $\neg x_1 \vee \neg x_2 \vee \neg x_3$.

The analysis for the targets in the discrete layer is trivial. The justification for the clauses involving x_1 is simple as the newly added edges do not change the corresponding block. For the clauses with literal $\neg x_1$ except the clause $\neg x_1 \vee \neg x_2 \vee \neg x_3$, in the corresponding block, one can eliminate the column corresponding to $\boldsymbol{U}_1$ by using the other literals. Finally, we can see that the existence of a path to the clause $\neg x_1 \vee \neg x_2 \vee \neg x_3$ gives us an extra pivoting vector. Thus, the $m\,(n-1) + 2n$ edge bound is optimal.

By a similar analysis, one can show that the vertex bound is also optimal.

4.4 Equivalence of FCP

We formulated two versions for the FCP problem, one minimizing the number of input vertices; the other minimizing the number of adjacent vertices of the input. They turn out to be equivalent.

The maximal matching algorithm proposed by Liu *et al.* [17] indeed seeks strongly connected components of a graph and gets a cacti configuration. For each strongly connected component, at least one edge must be added. Hence, their algorithm that minimizes the number of input vertices also minimizes the number of adjacent vertices. One can check that their algorithm always returns the minimum set of controlled nodes (see the proof of Liu's Minimum Input Theorem [17]).

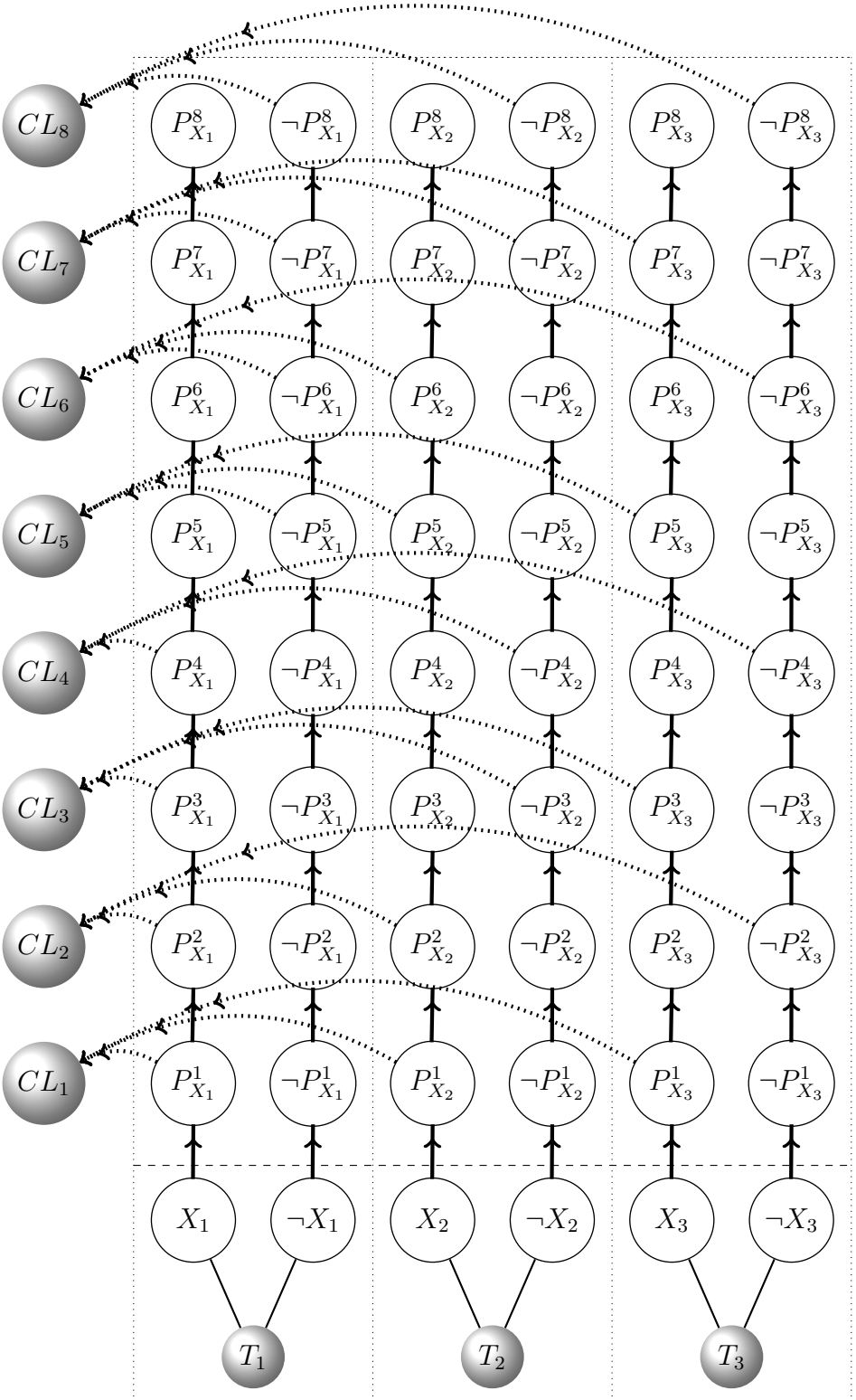

Figure 4.5: The clauses layer for the smallest unsatisfiable formula.

Chapter 5

Discussions

In practice, the number of additional edges or the outdegree of input vertices is usually bounded, especially for the design of single-target-drugs or finite-multiple-target-drugs in a biomolecular system. Hence, the search for controlled nodes for partial controllability of a given network can be considered as an **NP-HARD** problem.

In short, our NP-hardness result implies that one should consider algorithms that leave out optimality or absolute correctness. Potential candidates include the combination of artificial intelligence and repeatedly trained by the not-so-correct maximal linking algorithm. By dropping optimality, a simple modification of the maximal linking algorithm may lead to an absolute correct algorithm. One can consider checking the rank of matrix using Gaussian Elimination algorithm and keeping track of which vectors are eliminated and then attaching extra external inputs to the corresponding vertices. Research of these variant is currently being undertaken by our lab.

Despite the proposed seemingly simple numerical algorithm, the modified approach has to be a probabilistic algorithm due to the fact that structural controllability is defined among the whole equivalent class; and that the computable numbers are countable while the real numbers are uncountable. For example, one cannot replace the non-zero entries with the halting probability (or any non-computable numbers). Unless a symbolic version for the Gaussian Elimination algorithm is applied, one can-

not assert the maximality of the computed rank. However, symbolic computation of the maximum rank for any arbitrary matrix is well-known as a case for matrix completion. It is known that the problem contains the polynomial identity test, which is still open nowadays, as a special case.

At the time of writing this chapter, a new sufficient condition for structural output controllability is found in [21]. The approach of using zero forcing sets for strong structural controllability can be found in [22]. Further algorithmic study with these recent results can be undertaken in the future. Hopefully, these recent conditions can lead to further discovery of computational complexity features for problems in complex networks.

Appendix A

Sage Code for controllability

A symbolic code for checking output controllability is written in sage. The source code is published in github: `https://github.com/izayoi-ami/output-control`

```
class ControlSystem:
    def __init__(self,E,U,T,symbolic=True):
        self.cnt = 0
        self.E=E
        self.U=U
        self.T=T
        self.A=None
        self.B=None
        self.C=None
        self.OC=None
        self.prepare(symbolic)

    def edge_label(self):
        name = "e{}".format(self.cnt)
        self.cnt+=1
```

```
        return SR.symbol(name)

    def prepare(self,symbolic=True):
        E,U,T=self.E,self.U,self.T
        ts = lambda : self.edge_label() if symbolic else 1
        S0 = SR(0)

        n=max(E.keys()+sum(E.values(),[]))+1
        Vs= []
        for k in range(n):
            tmp = [ts() if k in E.keys() and j in E[k] else 0 for j in
             ↪  range(n)]
            Vs += [vector(tmp)]

        Us= []
        for k in U:
            tmp = [ts() if j in k else 0 for j in range(n)]
            Us += [vector(tmp)]
        Ts = []
        for k in (T):
            tmp = [1 if k==j else 0 for j in range(n)]
            Ts += [vector(tmp)]
        A=matrix(Vs).transpose()
        B=matrix(Us).transpose()
        C=matrix(Ts)

        Ms = [C*A^k*B for k in range(n)]
        OCs = []
```

```
        for M in Ms:
            for c in M.columns():
                OCs += [c]
        OC=matrix(OCs).transpose()
        self.A=A
        self.B=B
        self.C=C
        self.OC=OC
```

Example. The computational code for Example 2.4.11 is shown:

```
E={1:[0],2:[1],3:[2],4:[5,6],5:[4,6],6:[7],7:[8]}
U=[[3,4]]
C=[0,1,2,6,8]
S=ControlSystem(E,U,C)
#S.OC.rank() is the rank
```

Appendix B

Rank's criteria for output controllability

Theorem (Kalman)**.** *Let* (A, B, C) *be a system, then*

$$d\,(A, B, C) = \operatorname{rank} \mathcal{OC}\,(A, B, C)\,.$$

Hence, the system (A, B, C) *is output controllable iff* $\operatorname{rank} \mathcal{OC}\,(A, B, C) = \operatorname{rank} C$*.*

Proof. Recall that the system is defined by

$$\begin{cases} \boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t \\ \quad \boldsymbol{y}_t = C\boldsymbol{x}_t \end{cases}$$

and $\boldsymbol{x}_0 = \mathbf{0}$. By a sequence of direct computations, we obtain

$$\begin{aligned} \boldsymbol{x}_1 &= B\boldsymbol{u}_0 \\ \boldsymbol{x}_2 &= AB\boldsymbol{u}_0 + B\boldsymbol{u}_1 \\ &\vdots \\ \boldsymbol{x}_n &= \sum_{k=0}^{n-1} A^{n-1-k} B\boldsymbol{u}_k \end{aligned}$$

and the above summation can be written in matrix form

$$\boldsymbol{x}_n = \begin{bmatrix} A^{n-1}B & A^{n-2}B & \cdots & AB & B \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{n-2} \\ \boldsymbol{u}_{n-1} \end{bmatrix}.$$

Note that $\boldsymbol{y}_n = C\boldsymbol{x}_n$ can also be written as

$$\boldsymbol{y}_n = C\boldsymbol{x}_n = \begin{bmatrix} A^{n-1}B & A^{n-2}B & \cdots & AB & B \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{n-2} \\ \boldsymbol{u}_{n-1} \end{bmatrix}.$$

Writing the above matrix equation in operator form

$$\boldsymbol{y}_n = T\boldsymbol{u},$$

we know that the dimension of image of T is the rank of the matrix representing T.

Next, we observe that the rank is not increased by adding more blocks of the form A^kB for $k \geq n$.

By Cayley-Hamilton theorem, the matrix A satisfies its characteristic polynomial equation

$$\det(\lambda I - A) = 0.$$

In other words, we can express A^k as a linear combination of $A^0, A^1, \ldots, A^{n-1}$ and thus appending A^kB to $\mathcal{OC}(A, B, C)$ does not increases its rank for any $k \geq n$. $\square$

Appendix C

LaTeX code for the construction stages

Codes used to generate the figures in Chapter 4 are shown here. They are available in the same github link.

C.1 Clauses construction

Usage:

```
\clauseStage{m}{n}
```

```
\clauseStage{m}{n}{clausesList}
```

The third parameter is a list of clauses separated by semi-colon. Each clause is in the form: j, k_1, k_2, k_3, where j specifies the index of the clause and the remaining are the corresponding literals (with $-n$ meaning $\neg x_n$).

Figure 4.1 on page 49 is generated by:

```
\clauseStage{6}{4}
```

Figure 4.2 on page 50 is generated by:

```
\clauseStage{3}{4}{1,1,2,-3;2,1,3,-4;3,2,-3,4}
```

```
\DeclareDocumentCommand \clauseStage {m m >{\SplitList{;}}g}{
  \begin{tikzpicture}[x=1.75cm,y=2cm]
    \pgfmathtruncatemacro{\m}{#1}
    \pgfmathtruncatemacro{\n}{#2}
    \tikzset{node distance = 7cm }
    \GraphInit[vstyle=Normal]
    \tikzset{VertexStyle/.append style = {minimum size = 35 pt}}
    \tikzset{->-/.style={
        decoration={
          markings,
          mark=at position ##1 with {\arrow{>}}
        },
        postaction={decorate}
    }}
    \tikzset{EdgeStyle/.append style = {->-=0.75, ultra thick} }
    \draw[dotted] (-0.5,-0.5) rectangle (2*\n-0.5,-0.5);
    \draw[dotted] (-0.5,2+\m-0.5) rectangle (2*\n-0.5,2+\m-0.5);
    \draw[dashed] (-0.5,1.4) -- (2*\n-0.5,1.4);
    \foreach \i [evaluate=\i as \x using -0.5+2*\i] in {0,...,\n}{
      \draw[dotted] (\x,-0.5) -- (\x,2+\m-0.5);
    }
    \foreach \x [evaluate=\x as \i using \x-1] in {1,...,\n}{
      \Vertex[x=0+2*\i,y=1,Math=true,L=X_\x]{X\x10}
      \Vertex[x=1+2*\i,y=1,Math=true,L=\lnot X_\x]{X\x20}
      \tikzset{VertexStyle/.style = {shape = circle, ball
      ↪ color=white}}
      \Vertex[x=0.5+2*\i,y=0,Math=true,L=T_\x]{T}
```

```
    \SetVertexNormal
    \tikzset{VertexStyle/.append style = {minimum size = 35 pt}}
    \foreach \v in {1,2} {\Edge(X\x\v0)(T)}
    \foreach \p [evaluate=\p as \prev using \p-1] in {1,...,\m}{
      \Vertex[x=0+2*\i,y=1+\p,Math=true,L=P_{X_{\x}}^{\p}]{X\x1\p}
      \Vertex[x=1+2*\i,y=1+\p,Math=true,L=\lnot
      ↪ P_{X_{\x}}^{\p}]{X\x2\p}
      \coordinate (pH) at ([shift={(-0.36,0.3)}]X\x1\prev);
      \coordinate (pT) at ([shift={(0,-0.3)}]X\x1\p);
      \draw[->-=0.75, ultra thick] (pH) -- (pT);
      \coordinate (pH) at ([shift={(-0.36,0.3)}]X\x2\prev);
      \coordinate (pT) at ([shift={(0,-0.3)}]X\x2\p);
      \draw[->-=0.75, ultra thick] (pH) -- (pT);
    }
  }
  \DeclareDocumentCommand \addClauseOne {>{\SplitArgument{3}{,}}m}{
    \addClause ##1
  }

  \DeclareDocumentCommand \addClause {m m m m}{
    \tikzset{VertexStyle/.style = {shape = circle, ball
    ↪ color=white}}
    \Vertex[x=-1.25,y=1+##1,Math=true,L=CL_##1]{CL}
    \foreach \x in {##2,##3,##4}{
      \pgfmathtruncatemacro{\n}{abs(\x)}
      \pgfmathtruncatemacro{\b}{1+(\x<0)}
      \Edge[style={bend right,dotted,->}](X\n\b##1)(CL)
    }
```

```
    }
    \IfNoValueF{#3}{
      \ProcessList {#3} {\addClauseOne}
    }
  \end{tikzpicture}
}
```

C.2 Discrete Layer for N-TCP

Usage:

```
\discreteNStagee{m}{n}{N}
```

```
\DeclareDocumentCommand \discreteNStage {m m m}{
  \begin{tikzpicture}[x=1cm,y=1cm]
    \pgfmathtruncatemacro{\m}{#1}
    \pgfmathtruncatemacro{\n}{#2}
    \tikzset{node distance = 2cm }
    \GraphInit[vstyle=Normal]
    \tikzset{->-/.style={
        decoration={
          markings,
          mark=at position ##1 with {\arrow{>}}
        },
        postaction={decorate}
    }}
    \tikzset{EdgeStyle/.append style = {->-=0.75, ultra thick} }
    \tikzset{VertexStyle/.style = {shape = circle, ball color=white}}
```

```
16      \SetVertexNoLabel
17      \foreach \i in {1,...,\n}{
18        \Vertex[x=\i-1,y=0,Math=true,L=D_1^{\i}]{D1\i1}
19      }
20
21      \draw[dotted] (-0.5,-0.5) node[below right]{$D_1$} rectangle
        ↪  (\n-0.5,0.5);
22
23      \foreach \i [evaluate=\i as \mi using \i+\m, evaluate=\i as \mii
        ↪  using \mi+1] in {2,...,#3}{
24      \pgfmathtruncatemacro{\li}{\i+\m+1}
25      \draw[dotted] (-0.5+\n*\i-\n,-0.5) node[below right]{$D_{\li}$}
        ↪  rectangle (-0.5+\n*\i,\mi+0.5);
26        \SetVertexNormal
27        \tikzset{VertexStyle/.style = {shape = circle, ball
          ↪  color=white}}
28        \foreach \j [evaluate=\j as \x using \n*(\i-1)+\j-1] in
          ↪  {1,...,\n}{
29          \Vertex[x=\x,y=0]{D\i\j1}
30        }
31        \SetVertexSimple[MinSize=7pt]
32        \SetVertexNoLabel
33        \foreach \j [evaluate=\j as \x using \n*(\i-1)+\j-1] in
          ↪  {1,...,\n}{
34          \foreach \k [evaluate=\k as \prevk using \k-1] in
            ↪  {2,...,\mii}{
35            \Vertex[x=\x,y=\prevk]{D\i\j\k}
36            \coordinate (pH) at ([shift={(0,0)}]D\i\j\k);
```

```
            \coordinate (pT) at ([shift={(-0.13,0.12)}]D\i\j\prevk);
            \draw[->-=0.5] (pH) -- (pT);
          }
        }
      }
  \end{tikzpicture}
}
```

C.3 Discrete Layer for ETCP

Usage:

```
\discreteEStagee{m}{n}
```

```
\DeclareDocumentCommand \discreteEStage {m m}{
  \begin{tikzpicture}[x=1cm,y=1cm]
    \pgfmathtruncatemacro{\m}{#1}
    \pgfmathtruncatemacro{\n}{#2}
    \pgfmathtruncatemacro{\nLess}{\n-1}
    \tikzset{node distance = 2cm }
    \GraphInit[vstyle=Normal]
    \tikzset{->-/.style={
        decoration={
          markings,
          mark=at position ##1 with {\arrow{>}}
        },
        postaction={decorate}
    }}
```

```
15      \tikzset{EdgeStyle/.append style = {->-=0.75, ultra thick} }
16      \tikzset{VertexStyle/.style = {shape = circle, ball color=white}}
17      \SetVertexNoLabel
18      \foreach \i in {1,...,\n}{
19        \Vertex[x=\i-1,y=0,Math=true,L=D_1^{\i}]{D1\i1}
20      }
21
22      \draw[dotted] (-0.5,-0.5) node[below right]{$D_1$} rectangle
        ↪  (\n-0.5,0.5);
23
24      \foreach \i [evaluate=\i as \mi using \i+2, evaluate=\i as \mii
        ↪  using \mi+1] in {1,...,\m}{
25      \pgfmathtruncatemacro{\li}{\mi}
26      \draw[dotted] (0.5+\nLess*\i,-0.5) node[below right]{$D_{\li}$}
        ↪  rectangle (0.5+\nLess*\i+\nLess,\mi-0.5);
27        \SetVertexNormal
28        \tikzset{VertexStyle/.style = {shape = circle, ball
          ↪  color=white}}
29        \foreach \j [evaluate=\j as \x using (\n-1)*(\i-1)+\j-1] in
          ↪  {1,...,\nLess}{
30          \Vertex[x=\x+\n,y=0]{D\i\j1}
31        }
32        \SetVertexSimple[MinSize=7pt]
33        \SetVertexNoLabel
34        \foreach \j [evaluate=\j as \x using (\n-1)*(\i-1)+\j-1] in
          ↪  {1,...,\nLess}{
35          \foreach \k [evaluate=\k as \prevk using \k-1] in {2,...,\mi}{
36            \Vertex[x=\x+\n,y=\prevk]{D\i\j\k}
```

```
37            \coordinate (pH) at ([shift={(0,0)}]D\i\j\k);
38            \coordinate (pT) at ([shift={(-0.13,0.12)}]D\i\j\prevk);
39            \draw[->-=0.5] (pH) -- (pT);
40          }
41        }
42      }
43    \end{tikzpicture}
44  }
```

References

[1] Arora, Sanjeev and Barak, Boaz, *Computational complexity: a modern approach* (Cambridge University Press, 2009).

[2] Barabási, Albert-László, Gulbahce, Natali and Loscalzo, Joseph, 'Network medicine: a network-based approach to human disease', vol. 12, no. 1, 56–68 (2011-01).

[3] Bollobás, Béla, *Modern graph theory* (Springer, 1998).

[4] Bondy, J. A and Murty, U. S. R, *Graph theory* (Springer, 2008).

[5] Bridges, Douglas S., *Computability: A Mathematical Sketchbook*, 1994th edn. (Springer, 1994-01-14).

[6] Diestel, Reinhard, *Graph theory* (Springer, 1997).

[7] Fournier, Jean-Claude, *Graph theory and applications with exercises and problems* (ISTE ; Wiley, 2009).

[8] Friedberg, Stephen H, Insel, Arnold J and Spence, Lawrence E, *Linear algebra* (Prentice-Hall, 1979).

[9] Gao, Jianxi, Liu, Yang-Yu, D'Souza, Raissa M. and Barabási, Albert-László, 'Target control of complex networks', vol. 5, 5415 (2014-11-12).

[10] Garey, Michael R and Johnson, David S, *Computers and intractability: a guide to the theory of NP-completeness*, OCLC: 4195125 (W.H. Freeman, 1979).

[11] Gu, Shi, Pasqualetti, Fabio, Cieslak, Matthew, Telesford, Qawi K., Yu, Alfred B., Kahn, Ari E., Medaglia, John D., Vettel, Jean M., Miller, Michael B., Grafton, Scott T. and Bassett, Danielle S., 'Controllability of structural brain networks', vol. 6, 8414 (2015-10-01).

[12] Hosoe, S., 'Determination of generic dimensions of controllable subspaces and its application', vol. 25, no. 6, 1192–1196 (1980-12).

[13] Hou, Lvlin, Lao, Songyang, Small, Michael and Xiao, Yandong, 'Enhancing complex network controllability by minimum link direction reversal', vol. 379, no. 20, 1321–1325 (2015-07-03).

[14] Lin, C. t, 'System structure and minimal structural controllability', in *1976 IEEE Conference on Decision and Control including the 15th Symposium on Adaptive Processes*, pp. 879–885 (1976-12).

[15] Lin, Ching-Tai, 'Structural controllability', vol. 19, no. 3, 201–208 (1974-06).

[16] Lipschutz, Seymour, *Schaum's outline of theory and problems of linear algebra* (McGraw-Hill, 1991).

[17] Liu, Yang-Yu, Slotine, Jean-Jacques and Barabási, Albert-László, 'Controllability of complex networks', vol. 473, no. 7346, 167–173 (2011-05-12).

[18] Lovász, László, *Combinatorial problems and exercises* (North-Holland Pub. Co. ; Elsevier North-Holland, sole distributors for the U.S.A. and Canada, 1979).

[19] Matthews, Keith Robert, *Elementary linear algebra* (Dept. of Mathematics, University of Queensland, 1991).

[20] Medaglia, John D., Pasqualetti, Fabio, Hamilton, Roy H., Thompson-Schill, Sharon L. and Bassett, Danielle S., 'Brain and Cognitive Reserve: Translation via Network Control Theory', (2016-04-15).

[21] Monshizadeh, N., Camlibel, K. and Trentelman, H., 'Strong targeted controllability of dynamical networks', in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 4782–4787 (2015-12).

[22] Monshizadeh, Nima, Zhang, Shuo and Camlibel, Kanat, 'Zero forcing sets and controllability of dynamical systems defined on graphs', vol. 59, no. 9, 2562–2567 (2014-09).

[23] Murota, K. and Poljak, S., 'Note on a graph-theoretic criterion for structural output controllability', vol. 35, no. 8, 939–942 (1990-08).

[24] Nasiruzzaman, A. B. M. and Pota, H. R., 'Critical node identification of smart power system using complex network framework based centrality approach', in *North American Power Symposium (NAPS), 2011*, pp. 1–6 (2011-08).

[25] Nepusz, Tamás and Vicsek, Tamás, 'Controlling edge dynamics in complex networks', vol. 8, no. 7, 568–573 (2012-07).

[26] Olshevsky, A., 'Minimal Controllability Problems', vol. 1, no. 3, 249–258 (2014-09).

[27] Pan, Yujian and Li, Xiang, 'Structural Controllability and Controlling Centrality of Temporal Networks', vol. 9, no. 4, e94998 (2014-04-18).

[28] Papadimitriou, Christos H, *Computational complexity* (Addison-Wesley, 1994).

[29] Poljak, S., 'On the generic dimension of controllable subspaces', vol. 35, no. 3, 367–369 (1990-03).

[30] Shields, R. and Pearson, J., 'Structural controllability of multiinput linear systems', vol. 21, no. 2, 203–212 (1976-04).

[31] Sun, Jie and Motter, Adilson E., 'Controllability Transition and Nonlocality in Network Control', vol. 110, no. 20, 208701 (2013-05-14).

[32] Valenza, Robert J, *Linear algebra: an introduction to abstract mathematics* (Springer-Verlag, 1993).

[33] Willems, J. L., 'Structural controllability and observability', vol. 8, no. 1, 5–12 (1986-10).

[34] Wu, L., Shen, Y., Li, M. and Wu, F. X., 'Network Output Controllability-Based Method for Drug Target Identification', vol. 14, no. 2, 184–191 (2015-03).

[35] Yan, Gang, Ren, Jie, Lai, Ying-Cheng, Lai, Choy-Heng and Li, Baowen, 'Controlling Complex Networks: How Much Energy Is Needed?' vol. 108, no. 21, 218703 (2012-05-23).