AMERICAN UNIVERSITY OF BEIRUT

ON THE COMPLEXITY OF THE MAXIMUM INDEPENDENT SET RECONFIGURATION PROBLEM

by
EZZAT ABDEL SALAM CHEBARO

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Department of Computer Science
of the Faculty of Arts and Sciences
at the American University of Beirut

Beirut, Lebanon
August 2022

AMERICAN UNIVERSITY OF BEIRUT

ON THE COMPLEXITY OF THE MAXIMUM INDEPENDENT SET RECONFIGURATION PROBLEM

by

EZZAT ABDEL SALAM CHEBARO

Approved by:

Prof. Amer E. Mouawad, Assistant Professor	Advisor
Computer Science	[illegible]
Prof. Izzat El Hajj, Assistant Professor	Member of Committee
Computer Science	[illegible]
Prof. Wassim El Hajj, Associate Dean	Member of Committee
Computer Science	[illegible]

Date of thesis defense: August 31, 2022

AMERICAN UNIVERSITY OF BEIRUT

THESIS RELEASE FORM

Student Name:	Chebaro	Ezzat	Abdel Salam
	Last	First	Middle

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of my thesis; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes

X **As of the date of submission of my thesis**

___ **After 1 year from the date of submission of my thesis .**

___ **After 2 years from the date of submission of my thesis .**

___ **After 3 years from the date of submission of my thesis .**

[illegible]	September 2, 2022
Signature	Date

This form is dated and signed when asked to submit the final document to ScholarWorks. DELETE THIS NOTE WHEN SIGNED

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to Prof. Amer E. Mouawad for being a great professor and advisor throughout both my undergraduate and graduate degrees. He introduced me to theoretical computer science, and it has quickly become a new passion of mine. I was eventually taught how to research and attempt to answer open problems, which has been an eye opening experience as well as an invaluable skill. Aside from being a teacher, in the traditional sense, he was also extremely understanding when I faced real life adversities, and his support and advice helped me tremendously.

I would also like to offer thanks to my friends and family for being there for me throughout. They too stood by me when things got difficult, and although they were very patient, they always made sure to push me in right direction. Without all these people around me, I would not have made it this far in my academic pursuit.

ABSTRACT
OF THE THESIS OF

Ezzat Abdel Salam Chebaro for Master of Computer Science
Major: Computer Science

Title: On the Complexity of the Maximum Independent Set Reconfiguration Problem

We study the complexity of the polynomially equivalent Minimum Vertex Cover Reconfiguration and Maximum Independent Set Reconfiguration problems on a variety of graph classes, which ask whether there exists a reconfiguration sequence between two minimum vertex covers/maximum independent sets S and T of a graph G. The problems are studied under the token jumping and token sliding models, which turn out to be equivalent in this context. We show that the problems are in **P** when restricted to bipartite graphs, **PSPACE-complete** when restricted to planar graphs, as well as a list of results on a variety of other graph classes.

Table of Contents

Chapter 1

Introduction

A common problem that tends to naturally pose itself is the question of how some system can be reconfigured from one state to another, given rules on how the states can be altered. Some practical examples of such problems would be the Rubik's Cube or the 15-puzzle, which saw mathematical interest in 1879 [1]. Clearly, studying such problems is nothing new, as it was done before the formation of computer science as a field of study, which is reasonable since it is only natural to study solving such puzzles and games optimally. The study of such problems was also always of interest to the computational complexity community. Some early examples are a paper that showed the video game Sokoban is **PSPACE-complete** [2], and a later paper that studied sliding puzzles through the lens of nondeterministic constraint logic [3]. As these sorts of problems gained more interest, they were eventually formalized under the reconfiguration framework [4, 5].

These problems, now known as *reconfiguration problems*, study relationships between solutions of some computational problem, known as a *source problem.* A very commonly studied source problem in this field is the Independent Set problem, or the polynomially equivalent Vertex Cover problem. An independent set is a set of vertices in a graph such that no two vertices in said set are adjacent; a maximum independent set is an independent set of maximum size. A vertex cover of a graph is a set of vertices such that every edge is incident to at least one vertex in said set; a minimum vertex cover is a vertex cover of minimum size. Both source problems are decision problems which ask whether there exists a vertex cover or independent set of a certain size.

Both Independent Set Reconfiguration (ISR) and Vertex Cover Reconfiguration (VCR) are relatively well studied, with results known on a variety of different graph classes (see these results in chapter 2). In the context of this thesis, the complexity of the Maximum Independent Set Reconfiguration (Max-ISR) and Minimum Vertex Cover Reconfiguration (Min-VCR), which turn out to be polynomially equivalent, will be the main focus.

A vertex cover or an independent set can be visualized as a set of tokens that are placed on the vertices of the graph, such that the set of tokens satisfy the conditions of being a vertex cover or an independent set. These tokens represent a *configuration*, also known as a state, and a configuration can be transformed into some other configuration under some reconfiguration model. Three common models are *token sliding* (TS), *token jumping* (TJ), and *token addition/removal* (TAR). TS, introduced by Hearn and Demaine [3], allows a token to be removed from some vertex u and placed on some other vertex v that is adjacent to u. TJ, introduced by Kamiński et al. [6], allows a token to be removed from vertex u and placed on any other vertex v. Finally, TAR, introduced by Ito et al. [4], allows any number of tokens to be added or removed under the constraint that there must be at most/least k tokens on the graph. Note that under any of these reconfiguration models, once a *move* is performed, also known as a *reconfiguration step*, the resulting configuration must maintain the condition of the chosen source problem. So, under MIN-VCR, each configuration must maintain that the set of tokens form a minimum vertex cover. A sequence of configurations, known as a *reconfiguration sequence*, essentially describes the moves needed to transform the configuration at the start of the sequence to the configuration at the end of the sequence. Reconfiguration problems concerned with *reachability* ask whether there exists a reconfiguration sequence such that it starts and ends with some particular configurations. Reconfiguration problems concerned with *optimization* ask whether there exists a reconfiguration sequence of a certain length. The focus of this paper will then be to determine the complexity of reachability for MIN-VCR and MAX-ISR, on a variety of graph classes.

Chapter 2

Literature Review

First consider the below Figure 2.1 which represents the hierarchy of graph classes that are relevant to this thesis. An arrow from a graph class A to a graph class B represents that graph class B is a subset of graph class A. Note that if graph class A is easy to solve, then B is at least as easy to solve, likewise if graph B is hard to solve, then A is at least as hard to solve. Figure 2.1 should help with visualizing these graph classes and how they relate to one another.

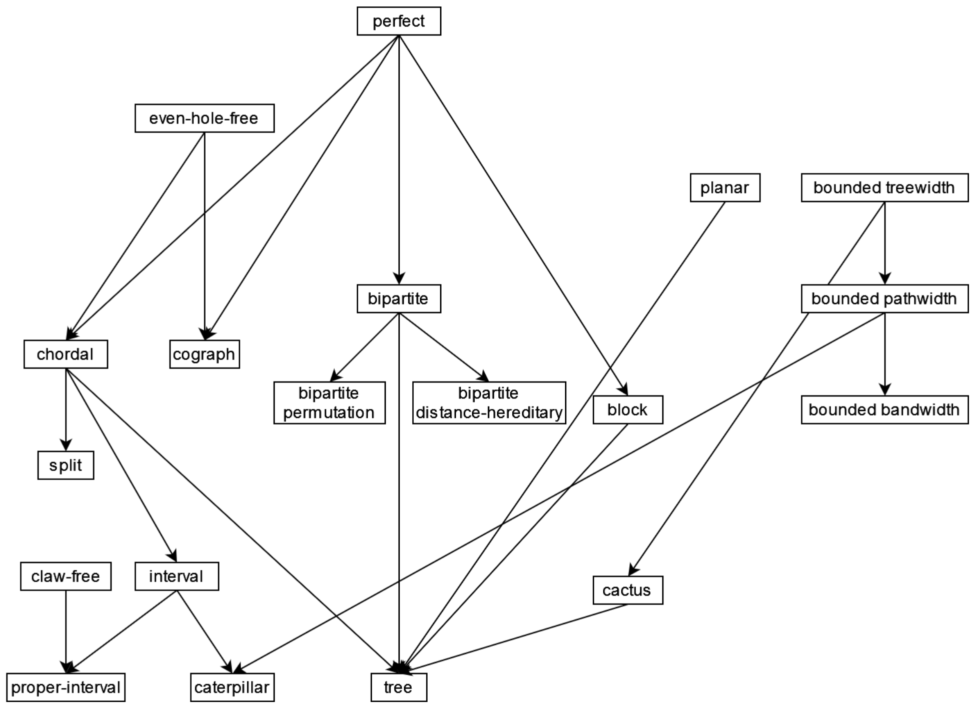


Figure 2.1: Graph class hierarchy

Table 2.1: Results on reachability for ISR/VCR

Class	Source	TS	TAR/TJ
planar (of degree 3)	NP-complete	PSPACE-complete [6]	PSPACE-complete [6]
even-hole-free	?	PSAPCE-complete (since chordal)	P [6, 7, 8]
perfect	P	PSPACE-complete [6]	PSPACE-complete [6]
cograph	P	P [6]	P [9, 10]
chordal	P	PSPACE-complete [11]	P (since even-hole-free)
split	P	PSPACE-complete [11]	P (since even-hole-free)
interval	P	P [12]	P (since even-hole-free)
claw-free	P	P [13]	P [13]
porper interval	P	P [14]	P (since even-hole-free)
caterpillar	P	P [14]	P (since even-hole-free)
tree	P	P [15]	P (since even-hole-free)
bipartite	P	PSPACE-complete [16]	NP-complete [16]
bipartite permutation	P	P [17]	?
bipartite distance hereditary	P	P [17]	?
block	P	P [18]	?
bounded treewidth	P	PSPACE-complete [19]	PSPACE-complete [19]
bounded pathwidth	P	PSPACE-complete [19]	PSPACE-complete [19]
bounded bandwidth	P	PSPACE-complete [19]	PSPACE-complete [19]
cactus	P	P [20]	P [8]

Table 2.1 is due to Nishimura [5]. It shows the complexity results for reachability of ISR and VCR on a variety of graph classes. Question marks denote unknown results. Note that the table has been updated to present recent results for TS-ISR/TS-VCR on chordal and split graphs. Figures 2.2 and 2.3 visualize the graph hierarchy along with the complexity boundary given the results in the table for TS-ISR/TS-VCR and TJ-ISR/TJ-VCR respectively. These results are of interest since the goal of the thesis will be to determine results for Max-ISR/Min-VCR on the same set of graph classes. It will be interesting to see which graph classes will be affected by the minimality/maximality constraints on vertex covers/independent sets such that there is a change in complexity class. Having both results side by side will paint a picture on how the complexity boundary changes given said constraint. These questions should make the motivation of the thesis clear, with implications of such results leading to a clearer picture of the complexities of ISR/VCR.

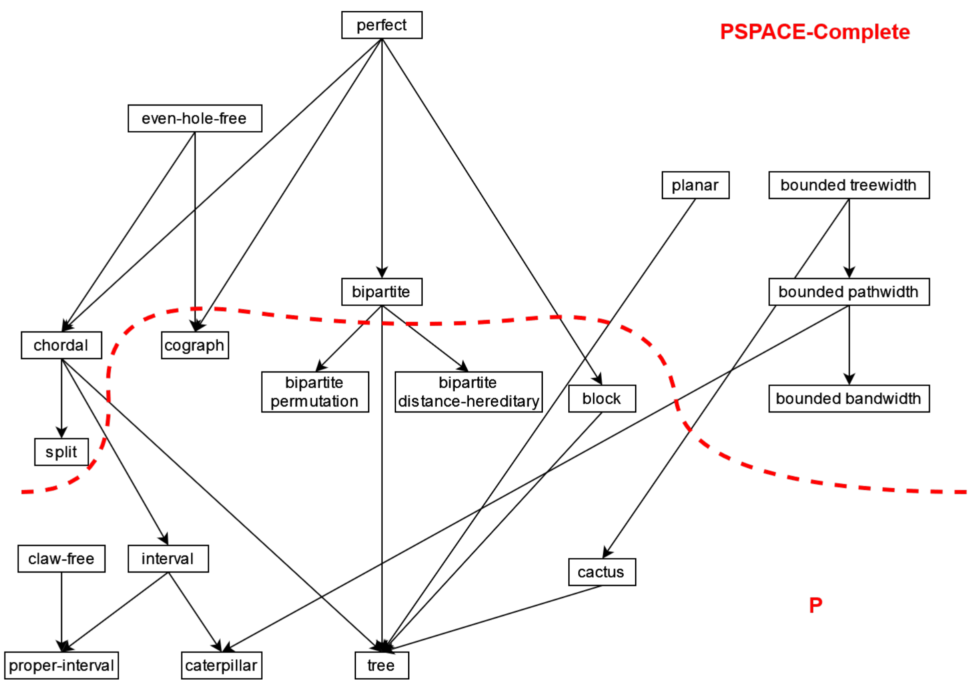


Figure 2.2: Graph class hierarchy with complexity boundary for TS-ISR/TS-VCR

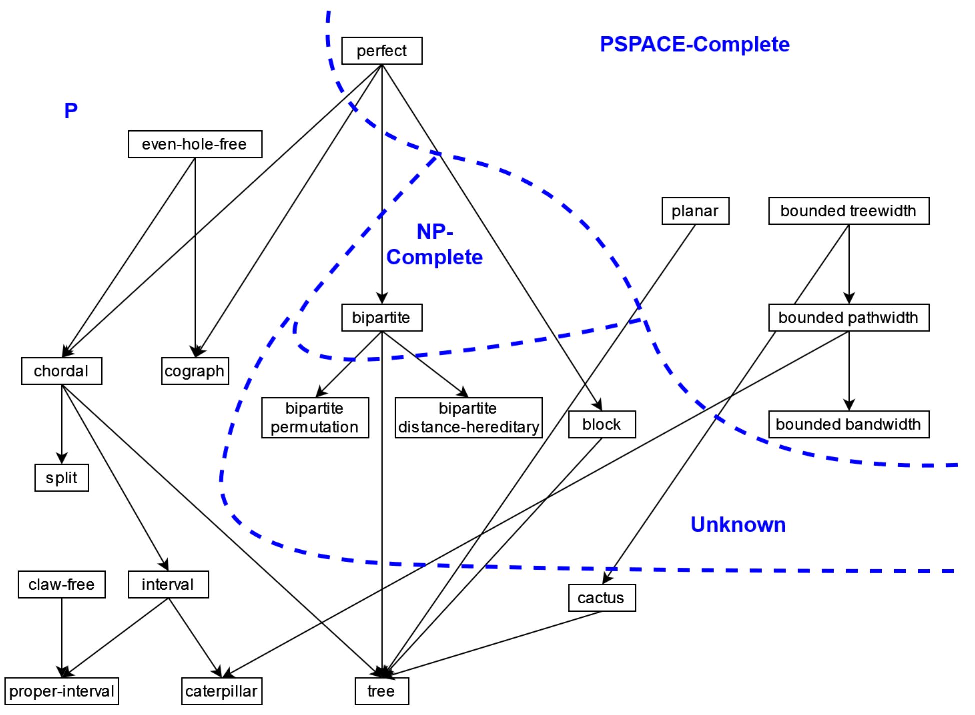


Figure 2.3: Graph class hierarchy with complexity boundary for TJ-ISR/TJ-VCR

Chapter 3

Preliminaries

3.1 Notation

3.1.1. General

We let $[n] = \{1, 2, \ldots, n\}$ such that $n \in \mathbb{N}$, and we let $[i, .., j] = \{i, i+1, \ldots, j-1, j\}$ such that $i, j \in \mathbb{N}$, and $j > i$. It is assumed that every graph G is simple, finite, and undirected, unless stated otherwise. We let $V(G)$ and $E(G)$ be the vertex and edge sets of graph G, respectively. Also, we let $n = |V(G)|$ and $m = |E(G)|$. We denote the *open neighborhood* of some vertex v in graph G by $N_G(v) = \{u \mid uv \in E(G)\}$ and we denote the *closed neighborhood* of some vertex v in graph G by $N_G[v] = N_G(v) \cup v$, the subscript can be dropped when there is only one graph in question. The *degree* of some vertex v is denoted by $deg(v) = |N_G(v)|$. An edge $e \in E(G)$ can also be denoted by uv where $u, v \in V(G)$ are its vertex endpoints. Let $G \setminus \{v\}$ denote the graph obtained by deleting vertex v from G (along with incident edges). Similarly let $G \setminus S$ denote the graph obtained by deleting the set of vertices S from G (along with incident edges).

Given some problem A and some problem B, we write $A \leq_p B$ when there exists a polynomial-time reduction from problem A to B. We say that two problems A and B are *polynomially equivalent* if and only if $A \leq_p B$ and $B \leq_p A$. If two problems are polynomially equivalent, then clearly if some problem is polynomial-time solvable then so is the other (and vice-versa), so proving some result for one would prove it for the other. Similarly, if one problem is NP-hard or PSPACE-hard then so is the other (and vice-versa).

3.1.2. Reconfiguration

Given a *reconfiguration problem*, a *configuration* of some graph is a set of vertices that forms a valid instance of the *source problem*. The vertices in the configuration are normally called tokens. In the case of Minimum Vertex Cover Reconfiguration (Min-VCR), all configurations are minimum vertex covers, and in the case of Maximum Independent Set Reconfiguration (Max-ISR), all configurations are maximum independent sets. Given two configurations C_i and C_j, let $C_i \leftrightarrow C_j$ denote that the configurations are *adjacent*, i.e., reachable from each other via one move. Let $C_i \leftrightsquigarrow C_j$ denote that the configurations are reachable from each other via one or more moves. Moving from one configuration to another will depend on the adjacency model in question. As stated in chapter 1, there are three main models that are commonly studied:

- **Token Sliding (TS):** $C_i \leftrightarrow C_j$ if $|C_i| = |C_j|$ and $v_i \in N_G(v_j)$ where $\{v_i\} = C_i \setminus C_j$ and $\{v_j\} = C_j \setminus C_i$ i.e. two configurations are said to be adjacent if the move involves moving a single token across an edge. This move is also known as a *slide.*
- **Token Jumping (TJ):** $C_i \leftrightarrow C_j$ if $|C_i| = |C_j|$ and $|C_i \setminus C_j| = |C_j \setminus C_i| = 1$ i.e. two configurations are said to be adjacent if the move involves moving a single token from one vertex to another. This move is also known as a *jump.*
- **Token Addition/Removal (TAR):** $C_i \leftrightarrow C_j$ if $|C_i| = |C_j| - 1$, $|C_i \setminus C_j| = 1$, and $|C_j \setminus C_i| = 0$ or if $|C_i| = |C_j| + 1$, $|C_i \setminus C_j| = 0$, and $|C_j \setminus C_i| = 1$ i.e. two configurations are said to be adjacent if the move involves adding or removing a single token.

A sequence of configurations $(C_1, \ldots C_l)$ such that $C_i \leftrightarrow C_{i+1}$ where $i \in [1 \,..\, l-1]$, is known as a *reconfiguration sequence.* A reconfiguration problem can then be stated and there are generally two common forms:

- **Reachability:** Given a graph G, a start configuration S and a target configuration T, is it true that $S \leftrightsquigarrow T$?
- **Optimization:** Given a graph G, some integer l, a start configuration S and a target configuration T, is it true that $S \leftrightsquigarrow T$ such that there exists a configuration sequence $X = (S, \ldots, T)$ where $|X| \leq l$.

Some forms of both these problem variants will also ask to output a valid configuration sequence from S to T if the instance is indeed a yes-instance. This thesis is solely concerned with studying the reachability variant of the problem, so we assume any future mentions of reconfiguration problems are of the reachability variant, unless stated otherwise.

3.2 Problem Definitions

3.2.1. Minimum Vertex Cover Reconfiguration

A vertex cover of a graph G is a set of vertices $X \subseteq V(G)$ such that for any edge $e \in E(G)$ there exists some $v \in X$ where v is an endpoint of e, a minimum vertex cover is one of minimum size. We denote an instance of Minimum Vertex Cover Reconfiguration (Min-VCR) by (G, S, T), where G is the input graph, S is the starting minimum vertex cover and T is the target minimum vertex cover. Now the problem can be studied under the TS and TJ reconfiguration models, denoted by TJ-Min-VCR and TS-Min-VCR respectively. Note that TAR is not considered since when working with any minimum/maximum structures since the addition/removal of any token breaks minimality, and so no two configurations will ever be reachable. An interesting outcome of working with minimum vertex covers is that TJ-Min-VCR and TS-Min-VCR turn out to be polynomially equivalent problems.

Lemma 3.1. TJ-Min-VCR *and* TS-Min-VCR *are polynomially equivalent problems.*

Proof. Before going into the reductions, first it needs to be shown that the only possible move in TJ-Min-VCR are slides. To see this, observe that given some configuration C, then for any token that is on some vertex u, there must exist at least one edge adjacent to it that is only covered by u itself, otherwise C would not be a minimum vertex cover since $C \setminus \{u\}$ is still a vertex cover. Clearly, if the token on u were to be moved it will have to go to some vertex that currently does not have a token on it. So, if the token on u is moved then there would exist at least one vertex v, such that there is no token on v and $v \in N(u)$. This would force the token on u to be placed on v so as to keep the edge uv covered. Since these two vertices are neighbors this move will always be a slide. Now, the reductions goes as follows:

- TJ-Min-VCR $\leq_p$ TS-Min-VCR:

  Given an instance $I = (G, S, T)$ of TJ-Min-VCR, it can be reduced to instance $I' = (G, S, T)$ of TS-Min-VCR such that I is a yes-instance if and only if I' is a yes-instance. For correctness:

  - If I is a yes-instance then there exists a reconfiguration sequence $X = (S, \ldots, T)$. Since the only possible moves in TJ-Min-VCR are slides then X is also a valid solution for I'.
  - If I' is a yes-instance then there exists a reconfiguration sequence $X' = (S, \ldots, T)$. Since all slides are jumps, by definition, then X' is also a valid solution for I.

- TS-Min-VCR $\leq_p$ TJ-Min-VCR:

  Given an instance $I = (G, S, T)$ of TS-Min-VCR, it can be reduced to instance $I' = (G, S, T)$ of TJ-Min-VCR such that I is a yes-instance if and only if I' is a yes-instance. For correctness:

- If I is a yes-instance then there exists a reconfiguration sequence $X = (S, \ldots, T)$. Since all slides are jumps, by definition, then X is also a valid solution for I'.
- If I' is a yes-instance then there exists a reconfiguration sequence $X' = (S, \ldots, T)$. Since the only possible moves in TJ-MIN-VCR are slides then X' is also a valid solution for I.

This completes the proof. □

Now since TAR-MIN-VCR is not considered, and since TJ-MIN-VCR and TS-MIN-VCR are polynomially equivalent problems, because the only possible move for both problems are slides, then moving forward this thesis will only concern itself with TS-MIN-VCR. Any complexity result determined on TS-MIN-VCR will also be shared by TJ-MIN-VCR. Moving forward, TS-MIN-VCR will be denoted by MIN-VCR for simplicity.

3.2.2. Maximum Independent Set Reconfiguration

An independent set of some graph G is a set of vertices $X \subseteq V(G)$ such that for any two vertices $u, v \in X$ it is the case that $uv \notin E(G)$, a maximum independent set is one of maximum size. We denote an instance of MAXIMUM INDEPENDENT SET RECONFIGURATION (MAX-ISR) by (G, S, T), where G is the input graph, S is the starting maximum independent set configuration, and T is the target maximum independent set configuration. The problem will also be studied under the TS and TJ reconfiguration models, denoted by TJ-MAX-ISR and TS-MAX-ISR respectively; working under TAR is not considered since the structure in question is one of maximum size. Similarly to **Lemma 3.1**, an interesting outcome of working with maximum independent sets is that TJ-MAX-ISR and TS-MAX-ISR turn out to be polynomially equivalent problems.

Lemma 3.2. TJ-MAX-ISR *and* TS-MAX-ISR *are polynomially equivalent problems.*

Proof. Before going into the reductions, first it needs to be shown that the only possible move in TJ-MAX-ISR are slides. To see this, first consider some configuration C and assume that there exists a move that is a jump but not a slide. So consider a possible move that involves moving a token on some vertex v to some other vertex u such that $u \notin N(v)$. Observe that the vertex u can not be in the neighborhood of any of the vertices in C, otherwise the move would result in a configuration that is not an independent set. This leads to a contradiction since C is not an independent set of maximum size, to see this consider the independent set $C \cup \{u\}$. So all moves must be slides. Now, the reductions goes as follows:

- TJ-MAX-ISR $\leq_p$ TS-MAX-ISR:

  Given an instance $I = (G, S, T)$ of TJ-MAX-ISR, it can be reduced to instance $I' = (G, S, T)$ of TS-MAX-ISR such that I is a yes-instance if and only if I' is a yes-instance. For correctness:

  - If I is a yes-instance then there exists a reconfiguration sequence $X = (S, \ldots, T)$. Since the only possible moves in TJ-MAX-ISR are slides then X is also a valid solution for I'.
  - If I' is a yes-instance then there exists a reconfiguration sequence $X' = (S, \ldots, T)$. Since all slides are jumps, by definition, then X' is also a valid solution for I.

- TS-MAX-ISR $\leq_p$ TJ-MAX-ISR:

  Given an instance $I = (G, S, T)$ of TS-MAX-ISR, it can be reduced to instance $I' = (G, S, T)$ of TJ-MAX-ISR such that I is a yes-instance if and only if I' is a yes-instance. For correctness:

  - If I is a yes-instance then there exists a reconfiguration sequence $X = (S, \ldots, T)$. Since all slides are jumps, by definition, then X is also a valid solution for I'.
  - If I' is a yes-instance then there exists a reconfiguration sequence $X' = (S, \ldots, T)$. Since the only possible moves in TJ-MAX-ISR are slides then X' is also a valid solution for I.

This completes the proof. □

As before, since TAR-MAX-ISR is not considered, and since TJ-MAX-ISR and TS-MAX-ISR are polynomially equivalent problems, because the only possible move for both problems are slides, then moving forward this thesis will only concern itself with TS-MAX-ISR. Any complexity result determined on TS-MAX-ISR will also be shared by TJ-MAX-ISR. Moving forward, TS-MAX-ISR will be denoted as MAX-ISR for simplicity.

3.2.3. Polynomial Equivalence of Min-VCR *and* Max-ISR

The VERTEX COVER problem asks whether a graph has a vertex cover of size at most k, while the INDEPENDENT SET problem, asks whether a graph has an independent set of size at least k. Similarly, MINIMUM VERTEX COVER and MAXIMUM INDEPENDENT SET ask whether a graph has a vertex cover of minimum size and an independent set of maximum size respectively. It should be clear to see that given some graph G and a vertex cover S of G, then $G - S$ is an independent set. If S is a vertex cover of minimum size then $G - S$ is an independent set of maximum size. It is well known that VERTEX COVER and INDEPENDENT SET are polynomially equivalent to each other, as is the case for MINIMUM VERTEX COVER and MAXIMUM INDEPENDENT SET. It turns out that MIN-VCR and MAX-ISR are also polynomially equivalent. The reductions and proof of correctness are fairly straightforward, although slightly redundant.

Lemma 3.3. Min-VCR *and* Max-ISR *are polynomially equivalent problems.*

Proof. **Min-VCR $\leq_p$ Max-ISR:** Given an instance $I = (G, S, T)$ of Min-VCR, it can be reduced to an instance $I' = (G, S', T')$ of Max-ISR, such that I is a yes-instance if and only if I' is a yes-instance. The reduction goes as follows: Let $S' = V(G') \setminus S$, and $T' = V(G') \setminus T$. Now, to prove correctness:

- If instance I is a yes-instance then there exists a configuration sequence $X = (C_1, \ldots, C_l)$ such that $S = C_1$ and $T = C_l$. Now consider the following configuration sequence $X' = (C'_1, \ldots, C'_l)$ for instance I' such that $C'_i = V(G) \setminus C_i \; \forall i \in [1 \mathinner{..} l]$. Clearly $S' = C'_1$ and $T' = C'_l$, by construction. It should also be clear that every configuration in X' are maximum independent sets. What remains to be shown is that this configuration sequence is a valid one.

  Since $\forall i \in [l]$ $C_i \leftrightarrow C_{i+1}$ then $v_i \in N_G(v_{i+1})$ where $\{v_i\} = C_i \setminus C_{i+1}$ and $\{v_{i+1}\} = C_{i+1} \setminus C_i$. Let $\{v'_i\} = C'_i \setminus C'_{i+1} = \{v_{i+1}\}$ and $\{v'_{i+1}\} = C'_{i+1} \setminus C'_i = \{v_i\}$. So, since v_i and v_{i+1} are neighbors it should be trivial to see that $v'_i \in N_G(v'_{i+1})$, making $C'_i \leftrightarrow C'_{i+1} \; \forall i \in [1, \ldots, l]$. So I' is a yes-instance.

- If instance I' is a yes-instance then there exists a configuration sequence $X' = (C'_1, \ldots, C'_l)$ such that $S' = C'_1$ and $T' = C'_l$. Now consider the following configuration sequence $X = (C_1, \ldots, C_l)$ for instance I such that $C_i = V(G) \setminus C'_i \; \forall i \in [1 \mathinner{..} l]$. Clearly $S = C_1$ and $T = C_l$, by construction. It should also be clear that every configuration in X are minimum vertex covers. What remains to be shown is that this configuration sequence is a valid one.

  Since $\forall i \in [l]$ $C'_i \leftrightarrow C'_{i+1}$ then $v'_i \in N_G(v'_{i+1})$ where $\{v'_i\} = C'_i \setminus C'_{i+1}$ and $\{v'_{i+1}\} = C'_{i+1} \setminus C'_i$. Let $\{v_i\} = C_i \setminus C_{i+1} = \{v_{i+1}\}$ and $\{v_{i+1}\} = C_{i+1} \setminus C_i = \{v_i\}$. So clearly since v'_i and v'_{i+1} are neighbors it should be trivial to see that $v_i \in N_G(v_{i+1})$, making $C_i \leftrightarrow C_{i+1}$. So I is a yes-instance.

Max-ISR $\leq_p$ Min-VCR: Given an instance $I = (G, S, T)$ of Max-ISR, it can be reduced to an instance $I' = (G, S', T')$ of Min-VCR, such that I is a yes-instance if and only if I' is a yes-instance. The reduction goes as follows: Let $S' = V(G') \setminus S$, and $T' = V(G') \setminus T$. Now, to prove correctness:

- If instance I is a yes-instance then there exists a configuration sequence $X = (C_1, \ldots, C_l)$ such that $S = C_1$ and $T = C_l$. Now consider the following configuration sequence $X' = (C'_1, \ldots, C'_l)$ for instance I' such that $C'_i = V(G) \setminus C_i \; \forall i \in [1 \mathinner{..} l]$. Clearly $S' = C'_1$ and $T' = C'_l$, by construction. It should also be clear that every configuration in X' are minimum vertex covers. What remains to be shown is that this configuration sequence is a valid one.

  Since $\forall i \in [l]$ $C_i \leftrightarrow C_{i+1}$ then $v_i \in N_G(v_{i+1})$ where $\{v_i\} = C_i \setminus C_{i+1}$ and $\{v_{i+1}\} = C_{i+1} \setminus C_i$. Let $\{v'_i\} = C'_i \setminus C'_{i+1} = \{v_{i+1}\}$ and $\{v'_{i+1}\} = C'_{i+1} \setminus C'_i = \{v_i\}$. So, since v_i and v_{i+1} are neighbors it should be trivial to see that $v'_i \in N_G(v'_{i+1})$, making $C'_i \leftrightarrow C'_{i+1} \; \forall i \in [1, \ldots, l]$. So I' is a yes-instance.

- If instance I' is a yes-instance then there exists a configuration sequence $X' = \{C'_1, \dots, C'_l\}$ such that $S' = C'_1$ and $T' = C'_l$. Now consider the following configuration sequence $X = (C_1, \dots, C_l)$ for instance I such that $C_i = V(G) \setminus C'_i \; \forall i \in [1 \,..\, l]$. Clearly $S = C_1$ and $T = C_l$, by construction. It should also be clear that every configuration in X are maximum independent sets. What remains to be shown is that this configuration sequence is a valid one.

  Since $\forall i \in [l] \; C'_i \leftrightarrow C'_{i+1}$ then $v'_i \in N_G(v'_{i+1})$ where $\{v'_i\} = C'_i \setminus C'_{i+1}$ and $\{v'_{i+1}\} = C'_{i+1} \setminus C'_i$. Let $\{v_i\} = C_i \setminus C_{i+1} = \{v_{i+1}\}$ and $\{v_{i+1}\} = C_{i+1} \setminus C_i = \{v_i\}$. So clearly since v'_i and v'_{i+1} are neighbors it should be trivial to see that $v_i \in N_G(v_{i+1})$, making $C_i \leftrightarrow C_{i+1}$. So I is a yes-instance.

This completes the proof. □

Since the problems are indeed polynomially equivalent, any complexity result determined on one will be shared by the other. Moving forward, the choice of which problem is used will simply depend on convenience, but the end result will be the same.

Chapter 4

Known Results

4.1 Polynomial Results

Some results for Max-ISR/Min-VCR follow trivially from previously known results. First observe the following:

- All instances of TS-Max-ISR are instances of TS-ISR, so if TS-ISR is easy on a certain graph class then so would TS-Max-ISR. Consequently, given **Lemma 3.3**, TS-Min-VCR is also easy on that same graph class.
- All instances of TJ-Max-ISR are instances of TJ-ISR, so if TJ-ISR is easy on a certain graph class then so would TJ-Max-ISR. Consequently, given **Lemma 3.2** and **Lemma 3.3**, TS-Max-ISR and TS-Min-VCR are also easy on that same graph class.

Given the above, the below statements then hold true:

- Since TS-ISR is in **P** on cographs, interval graphs, claw-free graphs, proper interval graphs, caterpillar graphs, trees, bipartite permutation graphs, bipartite distance hereditary graphs, block graphs, and cactus graphs (results seen in Table 2.1), then Max-ISR and Min-VCR are also in **P** on those same graph classes.
- Since TJ-ISR is in **P** on even-hole-free graphs, chordal graphs, and split graphs (results seen in Table 2.1), then Max-ISR and Min-VCR are also **P** on those same graph classes.

4.2 PSPACE-complete Results

A relevant result in Proposition 5 of Wrochna's paper [19] states that Max-ISR is **PSPACE-complete** on graphs of bounded bandwidth. For the sake of completeness the proof will be restated here. Some definitions will be needed to understand the context. First, the H-Word Reconfiguration problem will need to be defined. Given a tuple $H = (\Sigma, R)$, where Σ is an alphabet and $R \subseteq \Sigma^2$, an *H-word* is a *word*, or a sequence of symbols, over Σ such that every two consecutive symbols are in R. H can be visualized as a directed graph where the symbols are represented by the vertices and R is represented by the directed edges. In this context a word is an H-word if and only if it is a walk in H. A walk is defined as a sequence of adjacent vertices in a graph. The H-Word Reconfiguration problem then asks whether two H-words are reachable from one another. A move in this problem is defined by changing a single symbol in an H-word such that the resultant word is also an H-word. In Wrochna's paper H-Word Reconfiguration is proved to be **PSPACE-complete**.

Now, to define bandwidth. First consider some graph G and some function $f(\cdot)$ that assigns distinct integers for all $v \in V(G)$. The bandwidth of graph G is an integer that is achieved by some labeling $f(\cdot)$ such that $\max\{|f(v)-f(u)| \quad \forall uv \in E(G)\}$ is minimized. Another way to visualize bandwidth is to consider all possible placements of the vertices of G on distinct integer points on the number line. Now, consider the placement that yields the shortest longest edge. The length of that edge is the bandwidth of G. Graphs of bounded bandwidth are simply graphs that have a bandwidth that is bounded by some integer b.

Theorem 4.1. Max-ISR *on graphs of bounded bandwidth is PSPACE-complete.*

Proof. To show that Max-ISR is **PSPACE-complete** on graphs of bounded bandwidth, Wrochna constructs a reduction from the H-Word Reconfiguration problem to the Max-ISR problem on graphs of bounded bandwidth. The reduction goes as follows. Given an instance $I = (H, S, T)$ of H-Word Reconfiguration, where $H = (\Sigma, R)$, S is the starting H-word, and T is the target H-word, construct an instance $I' = (G', S', T')$ of Max-ISR where G' is a graph, S' is the starting maximum independent set, and T' is the target maximum independent set. Let $n = |S| = |T|$ and let $p = 2|\Sigma|$. I' is constructed as follows. $V(G') = \{v_i^a$ for all $i \in [n]$ and for all $a \in \Sigma\}$. Let $V_i = \{v_i^a$ for all $a \in \Sigma\}$ for all $i \in [n]$. $E(G')$ is comprised of edges between every two vertices of V_i for all $i \in [n]$ and edges between vertices $v_i^a v_{i+1}^b$ for all $(a, b) \notin R$ and for all $i \in [n-1]$. Given $S = (a_1, \ldots, a_n)$, construct $S' = \{V_1^{a_1}, \ldots, V_n^{a_n}\}$. Given $T = (a_1, \ldots, a_n)$, construct $T' = \{V_1^{a_1}, \ldots, V_n^{a_n}\}$.

The constructed graph G' has a bandwidth of at most $p = 2|\Sigma|$. This construction also defines a bijection between maximum independent sets and H-words. Notice that given some H-word $(a_1, \ldots, a_n)$, a maximum independent set can be constructed $\{V_1^{a_1}, \ldots, V_n^{a_n}\}$, and vice versa. A maximum independent set in G' is of size n and contains one vertex for each clique V_i. A move in the H-Word Reconfiguration problem is represented by moving a token inside of one of the cliques V_i. S' and T' are indeed maximum independent sets since they are constructed from H-words, given the previously mentioned bijection.

Instances I' and I are then equivalent and so this is a valid reduction from H-Word Reconfiguration to Max-ISR on a graph of bounded bandwidth, making Max-ISR on a graph of bounded bandwidth **PSPACE-complete**. □

Corollary 4.1. Max-ISR *on graphs of bounded pathwidth is PSPACE-complete*

Proof. Since graphs of bounded bandwidth are also graphs of bounded pathwidth and since Max-ISR is **PSPACE-complete** on graphs of bounded bandwidth, then Max-ISR on graphs of bounded pathwidth is indeed **PSPACE-complete**. □

Corollary 4.2. Max-ISR *on graphs of bounded treewidth is PSPACE-complete*

Proof. Since graphs of bounded bandwidth are also graphs of bounded treewidth and since Max-ISR is **PSPACE-complete** on graphs of bounded bandwidth, then Max-ISR on graphs of bounded treewidth is indeed **PSPACE-complete**. □

4.3 Results Thus Far

What remains are unknown results for Max-ISR on planar graphs, perfect graphs, and bipartite graphs. Table 4.1 compiles these results, with question marks denoting unknown results, while Figure 4.1 visualizes the graph class hierarchy along with the complexity boundary for the Max-ISR/Min-VCR problems. The goal of this thesis moving forward will be to determine some of these unknown results.

Table 4.1: Results on reachability for Max-ISR/Min-VCR

Class	Source	TS/TJ
planar (of degree 3)	NP-hard [21]	?
even-hole-free	?	P (since TJ-ISR)
perfect	P [22]	?
cograph	P	P (since TS-ISR)
chordal	P	P (since TJ-ISR)
split	P	P (since TJ-ISR)
interval	P	P (since TS-ISR)
claw-free	P [23]	P (since TS-ISR)
proper interval	P	P (since TS-ISR)
caterpillar	P	P (since TS-ISR)
tree	P	P (since TS-ISR)
bipartite	P	?
bipartite permutation	P	P (since TS-ISR)
bipartite distance hereditary	P	P (since TS-ISR)
block	P	P (since TS-ISR)
bounded treewidth	P [24]	PSAPCE-complete (since bounded bandwidth)
bounded pathwidth	P	PSAPCE-complete (since bounded bandwidth)
bounded bandwidth	P	PSAPCE-complete [19]
cactus	P	P (since TS-ISR)

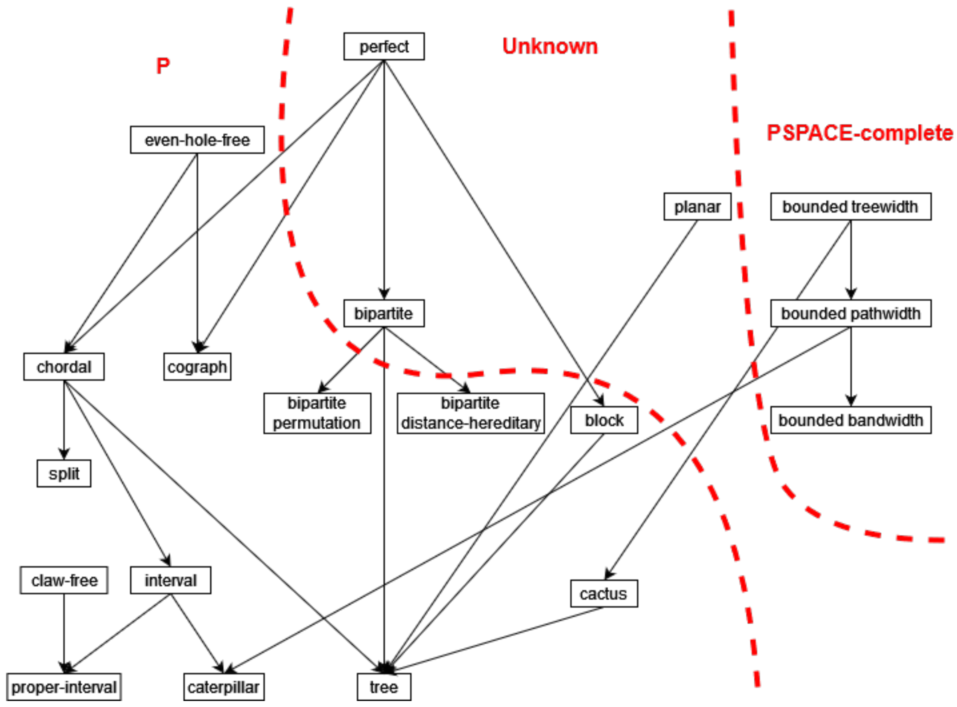


Figure 4.1: Graph class hierarchy with complexity boundary for Max-ISR/Min-VCR

Chapter 5

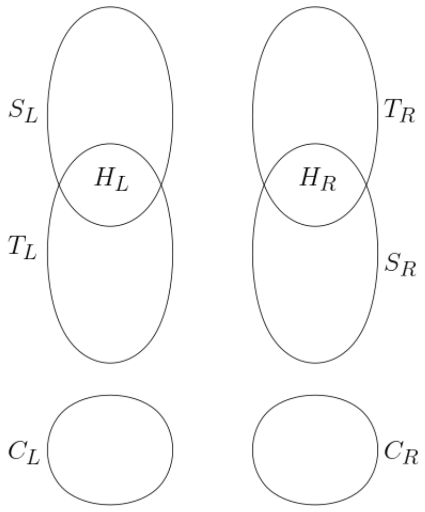
Bipartite Graphs

5.1 Preliminaries

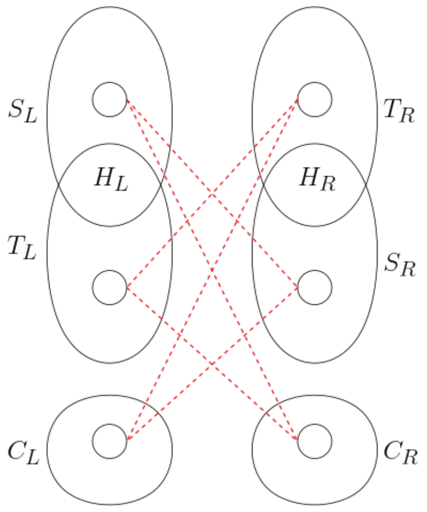
If a graph G is a bipartite graph, then its vertex set can be partitioned into 2 parts which will be denoted by L and R, such that no two vertices in the same part share an edge. In this chapter we prove the below theorem:

Theorem 5.1. Min-VCR *on bipartite graphs is in P.*

Before getting into the proofs, there are a few notations that need to be defined. Given a bipartite graph G, we denote the two parts as L and R. Given an instance (G, S, T) of Min-VCR on bipartite graphs, we define the following: $S_L = S \cap L$, $S_R = S \cap R$, $T_L = T \cap L$, $T_R = T \cap R$, $H = S \cap T$, $H_L = S_L \cap T_L$, $H_R = S_R \cap T_R$, $C = G \setminus (S \cup T)$, $C_L = L \setminus (S_L \cup T_L)$, and $C_R = R \setminus (S_R \cup T_R)$. Figure 5.1a visualizes the described sets.



(a) Sets



(b) Sets along with the illegal edges

Figure 5.1: Visualization of the partitioned bipartite graph

Looking at the relationships between these sets will also be useful. Clearly S_L, H_L, T_L, and C_L do not have any edges between each other, as is the case with S_R, H_R, T_R, and C_R, since this is a bipartite graph. Edges cannot exist between $T_L \setminus H_L$ and $T_R \setminus H_R$, since if such edges did exist then S would not be a valid vertex cover as it would not cover them. Symmetrically, edges cannot exist between $S_L \setminus H_L$ and $S_R \setminus H_R$ since T would not be a valid vertex cover. Edges can not exist between C_L and $T_R \setminus H_R$, since these edges would not be covered by S, as is the case with C_L and $S_R \setminus H_R$, since these edges would not be covered by T. Edges can not exist between neither C_R and $S_L \setminus H_L$, nor C_R and $T_L \setminus H_L$, via a symmetrical argument. Figure 5.1b visualizes the described sets along with the illegal edges that were just described.

5.2 The Algorithm

Lemma 5.1. *$|S_L \setminus H_L| = |T_R \setminus H_R|$ and $|T_L \setminus H_L| = |S_R \setminus H_R|$*

Proof. To show that $|S_L \setminus H_L| = |T_R \setminus H_R|$ assume otherwise:

- If $|S_L \setminus H_L| < |T_R \setminus H_R|$, then $S_L \cup T_L \cup H_R$ is a vertex cover whose size is smaller than the minimum vertex cover T, which is a contradiction. Visualize this construction by considering the vertex cover T and replacing $T_R \setminus H_R$ with $S_L \setminus H_L$.
- If $|S_L \setminus H_L| > |T_R \setminus H_R|$, then $S_R \cup T_R \cup H_L$ is a vertex cover whose size is smaller than the minimum vertex cover S, which is a contradiction. Visualize this construction by considering the vertex cover S and replacing $S_L \setminus H_L$ with $T_R \setminus H_R$.

To show that $|T_L \setminus H_L| = |S_R \setminus H_R|$ use a symmetrical argument, so assume otherwise:

- If $|T_L \setminus H_L| < |S_R \setminus H_R|$, then $S_L \cup T_L \cup H_R$ is a vertex cover whose size is smaller than the minimum vertex cover S, which is a contradiction. Visualize this construction by considering the vertex cover S and replacing $S_R \setminus H_R$ with $T_L \setminus H_L$.
- If $|T_L \setminus H_L| > |S_R \setminus H_R|$, then $S_R \cup T_R \cup H_L$ is a vertex cover whose size is smaller than the minimum vertex cover T, which is a contradiction. Visualize this construction by considering the vertex cover T and replacing $T_L \setminus H_L$ with $S_R \setminus H_R$.

This completes the proof. □

Lemma 5.2. *If an instance (G, S, T) of* MIN-VCR *on bipartite graphs is a yes-instance then there does not exist a reconfiguration sequence $X = (S, \dots, T)$ such that tokens slide between H_L and $T_R \cup S_R$ or between H_R and $S_L \cup T_L$.*

Proof. First it will be shown slides between H_L and $T_R \cup S_R$ are impossible. To show this, consider the first occurrence of a slide that changes the number of tokens contained in $H_L \cup C_R$. Denote A as the configuration before the slide, B as the configuration after the slide, and Z as the set of tokens contained in $H_L \cup C_R$ in configuration B. There are two cases to handle now:

- If $|Z| = |H_L| - 1$ then first consider the vertex cover $Y = T_R \cup S_R \cup H_L$. Visualize constructing Y through modifying S by replacing $(S_L \backslash H_L)$ with $(T_R \backslash H_R)$. Since $|S_L \backslash H_L| = |T_R \backslash H_R|$, given **Lemma 5.1**, then $|Y| = |S|$. Since the edges between H_L and C_R can be covered by Z, and since the edges between H_L and $T_R \cup S_R$ can be covered by $T_R \cup S_R$, then $(Y \setminus H_L) \cup Z$ is a valid vertex cover of size $|Y| - 1 = |S| - 1$. This contradicts that S is a minimum vertex cover.

- If $|Z| = |H_L| + 1$ then the configuration B has $|H_L| + 1$ contained inside $T_R \cup S_R$, but it is enough to use $|H_L|$ tokens, being H_L itself, to cover the edges between H_L and C_R, as well as the edges between H_L and $T_R \cup S_R$. So, consider the vertex cover $(B \setminus Z) \cup H_L$, which is a vertex cover of size $|B| - 1 = |S| - 1$. This contradicts that S is a minimum vertex cover.

Since the only way to change the number of tokens contained in $H_L \cup C_R$ are slides between H_L and $T_R \cup S_R$, then such slides are impossible because they lead to a contradiction.

Now it will be shown slides between H_R and $S_L \cup T_L$ are impossible. To show this, consider the first occurrence of a slide that changes the number of tokens contained in $H_R \cup C_L$. Denote A as the configuration before the slide, B as the configuration after the slide, and Z as the set of tokens contained in $H_R \cup C_L$ in configuration B. There are two cases to handle now:

- If $|Z| = |H_R| - 1$ then first consider the vertex cover $Y = S_L \cup T_L \cup H_R$. Visualize constructing Y through modifying S by replacing $(S_R \backslash H_R)$ with $(T_L \backslash H_L)$. Since $|S_R \backslash H_R| = |T_L \backslash H_L|$, given **Lemma 5.1**, then $|Y| = |S|$. Since the edges between H_R and C_L can be covered by Z, and since the edges between H_R and $S_L \cup T_L$ can be covered by $S_L \cup T_L$, then $(Y \setminus H_R) \cup Z$ is a valid vertex cover of size $|Y| - 1 = |S| - 1$. This contradicts that S is a minimum vertex cover.

- If $|Z| = |H_R| + 1$ then the configuration B has $|H_R| + 1$ contained inside $S_L \cup T_L$, but it is enough to use $|H_R|$ tokens, being H_R itself, to cover the edges between H_R and C_L, as well as the edges between H_R and $S_L \cup T_L$. So, consider the vertex cover $(B \setminus Z) \cup H_R$, which is a vertex cover of size $|B| - 1 = |S| - 1$. This contradicts that S is a minimum vertex cover.

Since the only way to change the number of tokens contained in $H_R \cup C_L$ are slides between H_R and $S_L \cup T_L$, then such slides are impossible because they lead to a contradiction.

This completes the proof. □

To recap, it has just been established that given a yes-instance (G, S, T) of Min-VCR on bipartite graphs there does not exist a reconfiguration sequence $X = (S, \dots, T)$ such that tokens slide between H_L and $T_R \cup S_R$ or between H_R and $S_L \cup T_L$. So tokens within $H_L \cup C_R$ and $H_R \cup C_L$ will always be contained within their respective sets.

Lemma 5.3. *If an instance (G, S, T) of* Min-VCR *on bipartite graphs is a yes-instance then there exists exists a reconfiguration sequence $X = (S, \dots, T)$ such that no token ever leaves H_L nor H_R.*

Proof. Given an instance (G, S, T) of Min-VCR on bipartite graphs, that is a yes-instance, consider some reconfiguration sequence $X = (C_1, \dots, C_l)$, such that $C_1 = S$, $C_l = T$ and $l > 1$. Construct a new reconfiguration sequence $X' = (C'_1, \dots, C'_l)$ such that $C'_i = (C_i \setminus (H \cup C)) \cup (H)$ for all $i \in [l]$. This construction may have consecutive configurations in the sequence that are equivalent. To deal with this simply modify X' by exhaustively searching for two consecutive configurations C'_i and C'_{i+1}, such that $C'_i = C'_{i+1}$, and remove one of them from the sequence. Notice that this construction is a modified version of the original configuration sequence such that it ensures the tokens contained in $H_L \cup C_R$ are always on H_L, and that the tokens contained in $H_r \cup C_L$ are always on H_R, all other moves are conserved. This construction is safe since, given **Lemma 5.2**, tokens within $H_L \cup C_R$ and $H_R \cup C_L$ will always be contained within their respective sets, and tokens outside these sets can never move in, so modifying their moves within these sets has no bearing on the rest of the graph. Every edge in the graph, other than the edges adjacent to H_L and H_R, are guaranteed to be covered, given that they are covered in X. Every configuration in X' also has the edges adjacent to H_L and H_R covered since they all have tokens placed on all of H_L and H_R, by construction. So every configuration in X' remains to be a vertex cover. Also notice that since both $S = C_1$ and $T = C_l$ have their tokens that are contained in $H_L \cup C_R$ and $H_R \cup C_L$ placed on H_L and H_R respectively, then $S = C'_1$ and $T = C'_l$, by construction. So X' is a reconfiguration sequence such that no token ever leaves H_L nor H_R, proving the lemma. □

Corollary 5.1. *An instance (G, S, T) of* Min-VCR *on bipartite graphs is a yes-instance if and only if instance (G', S', T') is a yes-instance where $G' = G \setminus (H \cup C)$, $S' = S \setminus H$, and $T' = T \setminus H$.*

Proof. If (G, S, T) is a yes-instance then, given **Lemma 5.3**, there exists a reconfiguration sequence $X = (C_1, \dots, C_l)$, where $C_1 = S$, $C_l = T$, and $l > 1$, such that no token ever leaves H_L nor H_R. This would mean that in the reconfiguration sequence, there are no slides that involve moving into or out of H and C. So, the reconfiguration sequence $X' = \{C'_1, \dots, C'_l\}$ where $C'_i = C_i \setminus H$ for all $i \in [l]$ can be used to decide instance (G', S', T'). Notice that $S' = C'_1$ and $T' = C'_l$.

If (G', S', T') is a yes-instance then there exists a reconfiguration sequence $X' = (C'_1, \dots, C'_l)$, where $C'_1 = S'$, $C'_l = T'$ and $l > 1$. Now, construct reconfiguration sequence $X = (C_1, \dots, C_l)$ to decide instance (G, S, T) such that $C_i = C'_i \cup H$ for all $i \in [l]$. Notice that $S = C_1$ and $T = C_l$. This is safe since the tokens in H will never be moved and so they are always covering all edges adjacent to H, while the sequence indeed does reconfigure the rest of graph, since X' safely reconfigures (G', S', T'). □

Lemma 5.4. *An instance (G, S, T) of* Min-VCR *on bipartite graphs, such that $H = C = \emptyset$ and such that there exists $v \in S$ where $deg(v) = 1$, is a yes-instance if and only if instance (G', S', T') is a yes-instance, where $G' = G \setminus N[v]$, $S' = S \setminus \{v\}$, and $T' = T \setminus N(v)$.*

Proof. Given an instance (G, S, T) of Min-VCR on bipartite graphs such that, $H = C = \emptyset$ and such that there exists $v \in S$ where $deg(v) = 1$, consider such a vertex along with its only neighbor u. Let $k = |S|$. Now, consider the instance (G', S', T') where $G' = G \setminus \{u, v\}$, $S' = S \setminus \{v\}$, and $T' = T \setminus \{u\}$. Clearly $|S'| = |T'| = |S| - 1 = |T| - 1 = k - 1$, and $N(v) = \{u\}$. It needs to be shown that (G, S, T) is a yes-instance if and only if (G', S', T') is a yes-instance:

- If (G, S, T) is a yes-instance then there exists a reconfiguration sequence $X = (C_1, \ldots, C_l)$ where $S = C_1$, $T = C_l$, and $l > 1$. Notice that since $deg(v) = 1$, there does not exist a configuration in X that has tokens on both v and u, otherwise it would not be a minimum vertex cover - to see this exclude v from such configuration, the resultant configuration is still a vertex cover. Also notice that since $deg(v) = 1$ then every configuration must have a token on exclusively v or u, so that the edge uv is covered. This would mean that $u \in T$, otherwise both v and u would be in S, but that is impossible.

  Now consider the reconfiguration sequence $X' = (C_1 \setminus \{v, u\}, \ldots, C_l \setminus \{v, u\})$ for the (G', S', T') instance. So, X' is constructed from X such that either v or u is removed from each configuration making them each size $k - 1 = |S'|$. Clearly $S' = C_1 \setminus \{v, u\}$ and $T' = C_l \setminus \{v, u\}$, by construction. What is left to be shown is that X' decides instance (G', S', T'). For every configuration C'_i in X', all edges in G' are covered since those edges are covered in the configuration C_i in X. Moves between two consecutive configurations C'_i and C'_{i+1} in X', where $i \in [l - 1]$, fall in one of two cases:

  - If the move between C_i and C_{i+1} does not involve sliding a token from v to u or u to v, then the move between C'_i and C'_{i+1} is the same so its valid.
  - If the move between C_i and C_{i+1} does involve sliding a token from v to u or u to v, then configurations C'_i and C'_{i+1} are equivalent, so no move is done. In this case modify X' by removing either C'_i and C'_{i+1} from the configuration.

  So, X' is a reconfiguration sequence that decides (G', S', T').

- If (G', S', T') is a yes-instance then there exists a reconfiguration sequence $X' = (C'_1, \ldots, C'_l)$ where $S' = C'_1$, $T' = C'_l$, and $l > 1$. Now consider the reconfiguration sequence $X = (S, C'_1 \cup \{u\}, \ldots, C'_l \cup \{u\})$ for the (G, S, T) instance. Clearly $T = C'_l \cup \{u\}$, by construction, and all the configurations are of size $|S| = k$. What is left to be shown is that X decides instance (G, S, T). Notice that it is indeed the case that $S \leftrightarrow C'_1 \cup \{u\}$ since the slide that took place between these two configurations is a token moving from v to u, since u is vertex v's only neighbor all edges remain covered after the move. For the remainder of the reconfiguration sequence, there is always a token on u, and so all edges adjacent to u and v remain covered, and so they can be ignored. The rest of the reconfiguration sequence is then valid because X' itself was a valid reconfiguration sequence. So, X is a reconfiguration sequence that decides (G, S, T).

This completes the proof. $\square$

Lemma 5.5. *There exists a polynomial time algorithm that decides* MIN-VCR *on bipartite graphs.*

Proof. Consider the following algorithm, which decides MIN-VCR on bipartite graphs where $H = C = \emptyset$:

Algorithm 1 An algorithm that decides Min-VCR on bipartite graphs where $H = C = \emptyset$

Input: An instance (G, S, T) of Min-VCR where $H = C = \emptyset$
Output: "yes-instance" or "no-instance"

```
 1: procedure MIN-VCR-BIPARTITE($G, S, T$)
 2:     if $S$ is empty then
 3:         return "yes-instance"
 4:     else
 5:         if there does not exist a vertex $v \in S$ such that $deg(v) = 1$ then
 6:             return "no-instance"
 7:         else
 8:             let $v$ be some vertex in $S$ such that $deg(v) = 1$
 9:             $G' \leftarrow G \setminus N[v]$
10:             $S' = S \setminus \{v\}$                              ▷ $|S'| = |S| - 1$
11:             $T' \leftarrow T \setminus N(v)$                           ▷ $|T'| = |T| - 1$
12:             return MIN-VCR-BIPARTITE($G', S', T'$)          ▷ $|S'| = |T'|$
13:         end if
14:     end if
15: end procedure
```

Let $n = |V(G)|$ and $m = |E(G)|$. The algorithm performs $O(m + n)$ amount of steps to find a vertex of degree 1 in S, before recursing, in which it recurses at most $|S|$ times. The algorithm returns when S is empty, otherwise on every recursion either S is reduced by 1 or the function returns. Since $|S| \leq n$ where $n = |V(G)|$ then this algorithm recurses $O(n)$ times, with $O(m + n)$ work done on each level. So the algorithm runs in $O(n(m + n))$, which is indeed polynomial time. Correctness will be shown via induction on the size of S and T, let $k = |S| = |T|$.

- Base Case: Given an instance (G, S, T) where $H = C = \emptyset$, if $k = 0$ then the algorithm correctly decides MIN-VCR on bipartite graphs. In this case the algorithm will always output that this is a "yes-instance". This is the case since if $S = T = \emptyset$ then it is vacuously true that $S \leftrightsquigarrow T$, since two identical configurations are always reachable - they're the same configuration. This case is handled by lines 2-3 in the algorithm.

- Induction Hypothesis: Given an instance (G, S, T) where $H = C = \emptyset$, if $|S| = |T| = k$, where $k > 0$, then the algorithm correctly decides Min-VCR on bipartite graphs.

- Inductive Step: Given an instance (G, S, T) where $H = C = \emptyset$, if $|S| = |T| = k+1$ then there are two cases to handle:

  – If there does not exist a vertex $v \in S$ such that $deg(v) = 1$ then the instance is a "no-instance" since no configuration that has a token on a vertex in T is reachable from S. To see this, consider any vertex $v \in S$, clearly $deg(v) \geq 2$. Since slides are the only possible moves, then the only possible set of moves for the token on v is to occupy any vertex $u \in N(v)$, but such a slide would always lead to at least one uncovered edge vw where $w \in N(v) \setminus \{u\}$. So all possible slides from v lead to configurations that are not vertex covers. So clearly, no token in S can move to T, since they can not move anywhere. So, no configuration that has a token on a vertex in T is reachable from S, making it impossible to reach T from S. This case is handled by lines 5-6 in the algorithm, where it outputs "no-instance" if there does not exist a vertex $v \in S$ such that $deg(v) = 1$.

  – Without loss of generality, if there exists a vertex $v \in S$ such that $deg(v) = 1$ then consider such a vertex along with its only neighbor u. Now, consider the instance (G', S', T') where $G' = G \setminus \{u, v\}$, $S' = S \setminus \{v\}$, and $T' = T \setminus \{u\}$. Clearly $|S'| = |T'| = |S| - 1 = |T| - 1 = k$. Given the inductive hypothesis, the algorithm correctly decides the instance (G', S', T'). Now, given **Lemma 5.4**, (G, S, T) is a yes-instance if and only if (G', S', T') is a yes-instance. This case is handled by lines 7-12 in the algorithm, where it outputs "yes-instance" if instance (G', S', T') is indeed a yes-instance or outputs "no-instance" if instance (G', S', T') is indeed a no-instance.

- Conclusion: By mathematical induction, since the base case and the inductive hypothesis being true imply that the inductive step is true, then the algorithm correctly decides an instance (G, S, T) where $H = C = \emptyset$ for $|S| = |T|$ of any size.

Notice that this algorithm correctly decides Min-VCR on bipartite graphs in polynomial time only when $H = C = \emptyset$. Via a preprocessing step, any Min-VCR instance on bipartite graphs can be decided in polynomial time. First, consider **Corollary 5.1**, which states that an instance (G, S, T) of Min-VCR on bipartite graphs is a yes-instance if and only if instance (G', S', T') is a yes-instance where $G' = G \setminus (H \cup C)$, $S' = S \setminus H$, and $T' = T \setminus H$. With this in mind any instance (G, S, T) can be converted to an instance (G', S', T') where $G' = G \setminus (H \cup C)$, $S' = S \setminus H$, and $T' = T \setminus H$, in polynomial time. (G', S', T') can then be passed to the above described algorithm. Since (G, S, T) is a yes-instance if and only if (G', S', T') is a yes-instance, so **Theorem 5.1** is trivially true. Min-VCR on bipartite graphs is in **P**. $\square$

Corollary 5.2. *Maximum Independent Set Reconfiguration on bipartite graphs is in P.*

Proof. Since Max-ISR and Min-VCR are polynomially equivalent, given **Lemma 3.1**, and since Min-VCR on bipartite graphs is in **P**, given **Theorem 5.1**, it then follows that Max-ISR on bipartite graphs is also in **P**. □

Chapter 6

Planar Graphs

6.1 Preliminaries

This chapter's result follows from a proof from Hearn and Demain where they showed that TS-ISR is **PSPACE-complete** via a reduction from non-deterministic constraint logic (NCL) configuration-to-edge on AND/OR graphs, which they also show to be **PSPACE-complete** [3]. So, some definitions and background will be needed before moving forward.

NCL is a model of computation, in which a set of decision problems can be defined. NCL is defined by a *constraint graph*, which is an undirected graph with non-negative integers assigned to both the edges and vertices. The integers assigned to the edges are known as *weights*, while the integers assigned to the vertices are known as *minimum in-flow constraints*. A valid configuration of such a graph would be an assigned orientation, or direction, for each edge such that sum of all the incoming edges at every vertex is at least its minimum in-flow constraint. A move is then defined as the flipping of the orientation of some edge such that the resultant graph still satisfies the constraints of every vertex. The reconfiguration problem of interest can now we be defined. The NCL Configuration-to-Edge (NCL CTE) is a decision problem that asks whether given some configuration A there exists a reconfiguration sequence that configures A to some configuration that has some edge E_B in a desired orientation.

A constraint graph of interest is known as the AND/OR constraint graph, the constraint graph is only made up of AND and OR vertices. An AND vertex is a vertex with a minimum in-flow of 2, with 3 edges weights of 1, 1, and 2. An OR vertex is a vertex with a minimum in-flow of 2, with 3 edges all having a weight of 2. The AND vertex can only have its edge with weight 2 oriented outwards if and only if the other two edges are oriented inwards. The OR vertex can only have one of its edges oriented outwards if and only if one of the other two edges is oriented inwards. The vertices have been named this way since they act similarly to the traditional logical AND and OR gates. Figure 6.1 illustrates these vertices.

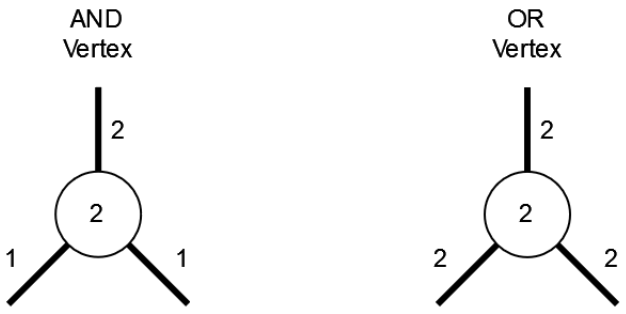


Figure 6.1: AND/OR vertices

Hearn and Demain do show that NCL CTE on AND/OR graphs is **PSPACE-complete**, but notice that AND/OR graphs need not be planar. So, they go on to prove that given any AND/OR graph, an equivalent planar 3-connected AND/OR graph can be constructed in polynomial time. This proof defines a reduction from NCL CTE on AND/OR graphs to NCL CTE on planar AND/OR 3-connected graphs, which proves that NCL CTE on planar AND/OR graphs is also **PSPACE-complete**.

6.2 The Reduction

Theorem 6.1. MAX-ISR *on planar graphs is PSPACE-complete*

Proof. This proof will be a slightly modified version of Hearn and Demaine's proof that showed TS-ISR is **PSPACE-complete** via a reduction from NCL CTE on AND/OR graphs. With some minor tweaks and observations the desired result can be determined.

First consider an instance $I = (G, A, E_B)$ of NCL CTE on planar AND/OR graphs, which is **PSPACE-complete**, such that G is the AND/OR constraint graph, A is the starting configuration, and E_B is an edge with its desired orientation. Now, consider an instance $I' = (G', S', T')$ of MAX-TS-ISR, such that G' is some graph, S' is the starting configuration and T' is the target configuration. To define how to construct G', Hearn and Demaine define gadgets that replace the AND and OR vertices. Figure 6.2 illustrates these gadgets.

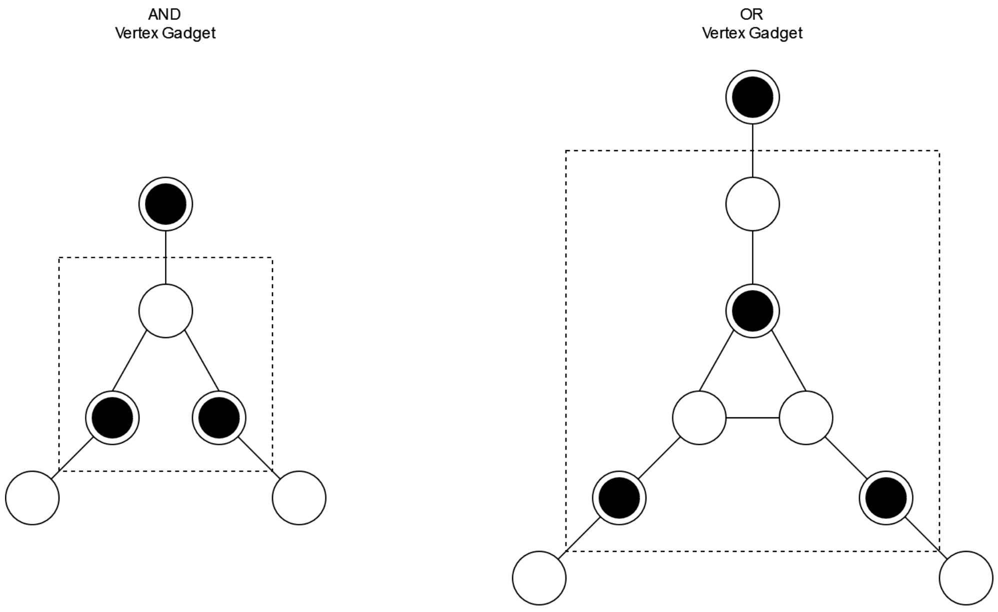


Figure 6.2: AND/OR vertex gadgets

The edges that go past the dotted lines are called *port* edges. The vertices inside the dotted line that are adjacent to a port edge are known as inner port edge vertices, while the vertices outside the dotted line that are adjacent to a port edge are known as outer port edge vertices. The entire structure inside the dotted line represents a single vertex in the NCL AND/OR graph, with a minimum in flow constraint of 2. The port edges represent the edges represent the edges adjacent to the vertices in the NCL AND/OR graph. Notice that in the AND vertex gadget, the top port edge represents the edge with the weight of 1, while all other port edges in both the AND and OR vertex gadgets represent the edges with a weight of 2. A token on an outer port edge vertex represents an inward oriented edge while a token on an inner port edge vertex represents an outward oriented edge. To construct a graph G', given an NCL graph G, replace each AND and OR vertices with their corresponding gadget and connect them via adding edges between two outer port vertices, with the port edges correspond to the right edges in G. When orienting edges in G, place tokens on G' as denoted earlier. Since G is planar G' remains to be planar. Also notice that a token adjacent to a port edge can never leave its port edge.

- AND gadget: This gadget acts like the AND vertex. To see this notice that that token on the upper port vertex can only slide to the lower port vertex when the other two tokens slide to their respective upper port vertices, and vice versa. Also notice the maximum independent set on this gadget is of size 3.
- OR gadget: This gadget acts like the OR vertex. To see this notice that that token on the upper port vertex can only slide to the lower port vertex when either one of the two tokens slide to their respective upper port vertices so that the inner token can slide, and vice versa. Also notice the maximum independent set on this gadget is of size 4.

Now, to see that instance I' works with maximum independent sets notice that the tokens on the AND/OR gadgets are indeed independent sets of maximum sizes. When constructing G' these gadgets are connected to each other by edges. Adding edges to a graph either keeps the maximum independent set of the same size or makes it smaller. In this case, it maintains its size. So TS-MAX-ISR on planar graphs is **PSPACE-complete**, since NCL CTE on planar AND/OR graphs is **PSPACE-complete**. □

Corollary 6.1. MAX-ISR *on planar graphs of degree 3 is PSPACE-complete*

Proof. The proof of **Theorem 6.1** proves a stronger result since the construction of G' ensures that every vertex has a maximum degree of 3 and so MAX-ISR on planar graphs of degree 3 is **PSPACE-complete**. □

Chapter 7

Conclusion

7.1 Final Results

Given the previously known results along with results proved in this paper, the below Table 7.1 compiles said results. As before, question marks denote unknown results. Figure 7.1, below, illustrates the complexity boundary known so far.

Table 7.1: Results on reachability for Max-ISR/Min-VCR

Class	Source	TS/TJ
planar (of degree 3)	NP-hard [21]	P
even-hole-free	?	P
perfect	P [22]	?
cograph	P	P
chordal	P	P
split	P	P
interval	P	P
claw-free	P [23]	P
porper interval	P	P
caterpillar	P	P
tree	P	P
bipartite	P	P
bipartite permutation	P	P
bipartite distance hereditary	P	P
block	P	P
bounded treewidth	P [24]	PSPACE-complete
bounded pathwidth	P	PSPACE-complete
bounded bandwidth	P	PSPACE-complete
cactus	P	P

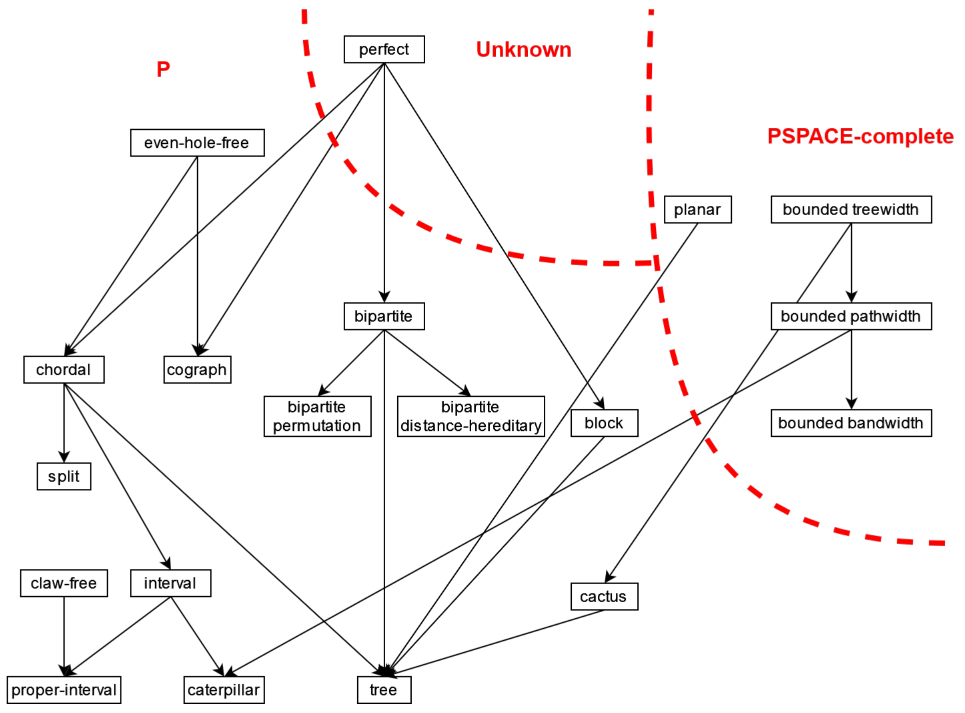


Figure 7.1: Graph class hierarchy with complexity boundary for Max-ISR/Min-VCR

7.2 Open Problems

The most obvious open question left unanswered in this paper is the complexity of Max-ISR/Min-VCR on perfect graphs, this result would complete what the paper sought out to do. The graph classes in this paper were chosen purely based on what has been studied and of relevance for ISR/VCR, but it is always interesting to look for results on other graph classes.

Leaving the realm of Max-ISR/Min-VCR, an interesting endeavor moving forward would be to extend the same activity conducted in this paper on other reconfiguration problems, which is to take a well studied reconfiguration problem and see how the complexity results differ when some constraint is added. Natural targets for problems of interest are any covering/packing problem pairs with a minimum/maximum constraint, seen in the Table 7.2, below.

Table 7.2: Covering/packing problem pairs for reconfiguration

Covering Problem	Packing Problem	Minimum Covering Problem	Maximum Packing Problem
Vertex Cover	Independent Set	Minimum Vertex Cover	Maximum Independent Set
Edge Cover	Matching	Minimum Edge Cover	Maximum Matching
Set Cover	Set Packing	Minimum Set Cover	Maximum Set Packing

Bibliography

[1] W. W. Johnson, W. E. Story, *et al.*, "Notes on the "15" puzzle," *American Journal of Mathematics*, vol. 2, no. 4, pp. 397–404, 1879.

[2] J. C. Culberson, "Sokoban is pspace-complete," *Department of Computing Science, The University of Alberta*, 1997.

[3] R. A. Hearn and E. D. Demaine, "Pspace-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation," *Theor. Comput. Sci.*, vol. 343, no. 1-2, pp. 72–96, 2005.

[4] T. Ito, E. D. Demaine, N. J. A. Harvey, C. H. Papadimitriou, M. Sideri, R. Uehara, and Y. Uno, "On the complexity of reconfiguration problems," *Theor. Comput. Sci.*, vol. 412, no. 12-14, pp. 1054–1065, 2011.

[5] N. Nishimura, "Introduction to reconfiguration," *Algorithms*, vol. 11, no. 4, p. 52, 2018.

[6] M. Kaminski, P. Medvedev, and M. Milanic, "Complexity of independent set reconfigurability problems," *Theor. Comput. Sci.*, vol. 439, pp. 9–15, 2012.

[7] A. E. Mouawad, N. Nishimura, V. Raman, and S. Siebertz, "Vertex cover reconfiguration and beyond," *Algorithms*, vol. 11, no. 2, p. 20, 2018.

[8] T. Ito, H. Nooka, and X. Zhou, "Reconfiguration of vertex covers in a graph," *IEICE Trans. Inf. Syst.*, vol. 99-D, no. 3, pp. 598–606, 2016.

[9] P. S. Bonsma, "Independent set reconfiguration in cographs and their generalizations," *J. Graph Theory*, vol. 83, no. 2, pp. 164–195, 2016.

[10] M. Bonamy and N. Bousquet, "Reconfiguring independent sets in cographs," *CoRR*, vol. abs/1406.1433, 2014.

[11] R. Belmonte, E. J. Kim, M. Lampis, V. Mitsou, Y. Otachi, and F. Sikora, "Token sliding on split graphs," *Theory Comput. Syst.*, vol. 65, no. 4, pp. 662–686, 2021.

[12] M. Bonamy and N. Bousquet, "Token sliding on chordal graphs," in *Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers* (H. L. Bodlaender and G. J. Woeginger, eds.), vol. 10520 of *Lecture Notes in Computer Science*, pp. 127–139, Springer, 2017.

[13] P. S. Bonsma, M. Kaminski, and M. Wrochna, "Reconfiguring independent sets in claw-free graphs," in *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings* (R. Ravi and I. L. Gørtz, eds.), vol. 8503 of *Lecture Notes in Computer Science*, pp. 86–97, Springer, 2014.

[14] T. Yamada and R. Uehara, "Shortest reconfiguration of sliding tokens on a caterpillar," in *WALCOM: Algorithms and Computation - 10th International Workshop, WALCOM 2016, Kathmandu, Nepal, March 29-31, 2016, Proceedings* (M. Kaykobad and R. Petreschi, eds.), vol. 9627 of *Lecture Notes in Computer Science*, pp. 236–248, Springer, 2016.

[15] E. D. Demaine, M. L. Demaine, E. Fox-Epstein, D. A. Hoang, T. Ito, H. Ono, Y. Otachi, R. Uehara, and T. Yamada, "Linear-time algorithm for sliding tokens on trees," *Theor. Comput. Sci.*, vol. 600, pp. 132–142, 2015.

[16] D. Lokshtanov and A. E. Mouawad, "The complexity of independent set reconfiguration on bipartite graphs," *ACM Trans. Algorithms*, vol. 15, no. 1, pp. 7:1–7:19, 2019.

[17] E. Fox-Epstein, D. A. Hoang, Y. Otachi, and R. Uehara, "Sliding token on bipartite permutation graphs," in *Algorithms and Computation - 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings* (K. M. Elbassioni and K. Makino, eds.), vol. 9472 of *Lecture Notes in Computer Science*, pp. 237–247, Springer, 2015.

[18] D. A. Hoang, E. Fox-Epstein, and R. Uehara, "Sliding tokens on block graphs," in *WALCOM: Algorithms and Computation, 11th International Conference and Workshops, WALCOM 2017, Hsinchu, Taiwan, March 29-31, 2017, Proceedings* (S. Poon, M. S. Rahman, and H. Yen, eds.), vol. 10167 of *Lecture Notes in Computer Science*, pp. 460–471, Springer, 2017.

[19] M. Wrochna, "Reconfiguration in bounded bandwidth and tree-depth," *J. Comput. Syst. Sci.*, vol. 93, pp. 1–10, 2018.

[20] D. A. Hoang and R. Uehara, "Sliding tokens on a cactus," in *27th International Symposium on Algorithms and Computation, ISAAC 2016, December 12-14, 2016, Sydney, Australia* (S. Hong, ed.), vol. 64 of *LIPIcs*, pp. 37:1–37:26, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[21] M. R. Garey and D. S. Johnson, "The rectilinear steiner tree problem in NP complete," *SIAM Journal of Applied Mathematics*, vol. 32, pp. 826–834, 1977.

[22] M. Grötschel, L. Lovász, and A. Schrijver, "Polynomial algorithms for perfect graphs," in *Topics on Perfect Graphs* (C. Berge and V. Chvátal, eds.), vol. 88 of *North-Holland Mathematics Studies*, pp. 325–356, North-Holland, 1984.

[23] G. J. Minty, "On maximal independent sets of vertices in claw-free graphs," *J. Comb. Theory, Ser. B*, vol. 28, no. 3, pp. 284–304, 1980.

[24] B. K. Bhattacharya, M. De, S. C. Nandy, and S. Roy, “Maximum independent set for interval graphs and trees in space efficient models,” in *Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada, 2014*, Carleton University, Ottawa, Canada, 2014.