The Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)

Reconfiguring Shortest Paths in Graphs*

Kshitij Gajjar[1], Agastya Vibhuti Jha[2], Manish Kumar[3], Abhiruk Lahiri[4]

[1] National University of Singapore, Singapore
[2] École polytechnique fédérale de Lausanne, Switzerland
[3] Ben-Gurion University of the Negev, Israel
[4] Ariel University, Israel
kshitijgajjar@gmail.com, agastya.jha@epfl.ch, manishk@post.bgu.ac.il, abhiruk@ariel.ac.il

Abstract

Reconfiguring two shortest paths in a graph means modifying one shortest path to the other by changing one vertex at a time, so that all the intermediate paths are also shortest paths. This problem has several natural applications, namely: (a) revamping road networks, (b) rerouting data packets in a synchronous multiprocessing setting, (c) the shipping container stowage problem, and (d) the train marshalling problem.

When modelled as graph problems, (a) is the most general case while (b), (c) and (d) are restrictions to different graph classes. We show that (a) is intractable, even for relaxed variants of the problem. For (b), (c) and (d), we present efficient algorithms to solve the respective problems. We also generalize the problem to when at most k (for some $k \geq 2$) contiguous vertices on a shortest path can be changed at a time.

1 Introduction

A *reconfiguration problem* asks computational questions of the following type. Given two different configurations of a system, is it possible to gradually transform one to the other? The two most popular examples of reconfiguration problems are the 15-puzzle (Ratner and Warmuth 1986; Goldreich 2011) and the Rubik's cube (Demaine et al. 2011; Demaine, Eisenstat, and Rudoy 2018). In both, we want to determine how to reach a "solved" final configuration using a sequence of "moves", starting from a given initial configuration. Recently, a lot of research has gone into the study of different types of reconfiguration problems on graphs (Mouawad et al. 2017; Lokshtanov and Mouawad 2018; Lokshtanov et al. 2018; Mouawad et al. 2018).

In this paper, we undertake a theoretical study of the reconfiguration problem on shortest paths, known as the *Shortest Path Reconfiguration* problem (abbreviated as SPR), introduced by (Kaminski, Medvedev, and Milanic 2010).

Definition 1. *Given an undirected, unweighted graph G with a source vertex s and a target vertex t, we say that two s–t shortest paths P and Q in G are* reconfigurable *if there is a sequence of s–t shortest paths $(P_0, P_1, \ldots, P_{k-1}, P_k)$ where $P_0 = P$ and $P_k = Q$ (for some positive integer k) such that P_i and P_{i+1} (for each $i \in \{0, 1, \ldots, k-1\}$) differ in only one vertex. (See Figure 1 for an example.)*

Technically, SPR is the decision problem of checking whether two given shortest paths in a graph are reconfigurable. Additionally, one can ask questions of the following form. If two paths are reconfigurable, is the reconfiguration sequence short enough? If so, is the sequence efficiently computable?

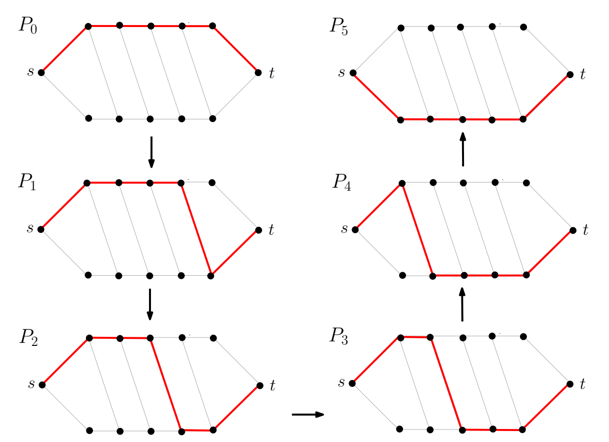

Figure 1: Reconfiguring a path $P = P_0$ to another path $Q = P_5$ by changing one vertex at a time. Note that all paths in the sequence $(P_0, P_1, P_2, P_3, P_4, P_5)$ are s–t shortest paths.

SPR has several real-world applications, some of which we describe in Subsection 1.2. Despite these numerous applications, SPR has not received its fair share of attention from the theoretical standpoint. This is because when research on reconfiguration began almost forty years ago, the main motivation behind studying the problem was in the context of coordinated motion planning of robots (Hopcroft, Schwartz, and Sharir 1984). Large swarms of robots are operated by a central algorithm, which gives specific instructions to each robot so that they can function as a team to solve a given task. Given the initial and final configurations of the robots (a configuration is simply a snapshot of the positions of the robots), the goal of the algorithm is to mod-

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 682203-ERC-[Inf-Speed-Tradeoff].

ify the positions of the robots in a sequential, step-by-step manner so that they can reach the final configuration without bumping into each other. (A possible usage of the robots in this setting is to manage a warehouse or inventory.)

Soon thereafter, it was shown that coordinated motion planning of robots is PSPACE-complete (Hopcroft and Wilfong 1986), implying that there is no polynomial-time algorithm for it unless P = PSPACE. Another closely related problem that was studied roughly around the same time was known as *2-dimensional planar linkage* (Hopcroft, Joseph, and Whitesides 1984). Although it was not explicitly stated, it is easy to observe that 2-dimensional planar linkage is essentially a problem about reconfiguring paths of a fixed length on a graph, which is a special case of SPR. For two decades after that, this observation went unexplored and theoretical research in SPR remained dormant. Recently however, there has been a flurry of papers on SPR (Kaminski, Medvedev, and Milanic 2010; Bonsma 2013, 2017; Asplund et al. 2018; Asplund and Werner 2020).

Prior to this work, (Bonsma 2013) showed that SPR is PSPACE-complete in general. A careful look at their proof further tells us the following.

Observation 1. SPR *is* PSPACE*-complete even when the input graphs are restricted to be bipartite.*

On the positive side, it known that SPR is solvable in polynomial time for certain graph classes such as planar graphs (Bonsma 2017), grid graphs (Asplund et al. 2018), claw-free graphs and chordal graphs (Bonsma 2013).

In this paper, we further investigate the complexity of SPR, particularly focusing on graph classes that model real-world problems.

1.1 Our Contributions

Our contributions are twofold. First, we study SPR for various graph classes. And second, we introduce a generalized variant of SPR.

SPR: circle graphs, Boolean hypercube, bridged graphs. For circle graphs, permutation graphs and the Boolean hypercube, we provide a complete characterisation of shortest paths and their reconfigurability for SPR. This automatically yields polynomial-time algorithms for them. For the Boolean hypercube, we show that every shortest path corresponds to a permutation. In fact, the length of the shortest reconfiguration sequence between two shortest paths is precisely the *Kendall tau distance* (Sedgewick and Wayne 2016) between their respective permutations. See Subsection 3.3 for more details. The characterisation for circle graphs and permutation graphs is a bit more technically involved. Details can be found in Subsection 3.1. For a subclass of metric graphs called bridged graphs, we show that SPR can be solved in polynomial time (Subsection 3.2). Finally, for graphs of bounded diameter, we observe that SPR can be solved in polynomial time (Subsection 3.4).

k-SPR: hardness and optimization variants. We introduce a novel generalisation of SPR called k-SPR, in which we are allowed to change at most k successive vertices[1] (instead of only one vertex) at a time. We show that k-SPR is PSPACE-complete when $k = O(1)$, and can be solved in polynomial time when $k \geq n/2$ (Subsection 2.1).

It is known that SPR can be solved in polynomial time for line graphs (Bonsma 2013). We show that k-SPR is PSPACE-complete for line graphs for all $k \geq 2$ (Subsection 2.1), demonstrating that k-SPR can be significantly harder than SPR. We also use a "lift-and-project" type proof to show that SPR is PSPACE-complete for graph powers (Subsection 2.2) by using the PSPACE-hardness of k-SPR.

We also study a few optimisation variants of k-SPR, and show that there is no polynomial-time algorithm to approximate k-SPR within a factor of $2^{O(n^2)}$, unless PSPACE = P (Section 4). Finally, we examine the gradation of the maximum number of different shortest paths in n-vertex graphs as the distance between s and t varies from $O(1)$ to $\Omega(n)$ (Section 5).

1.2 Applications

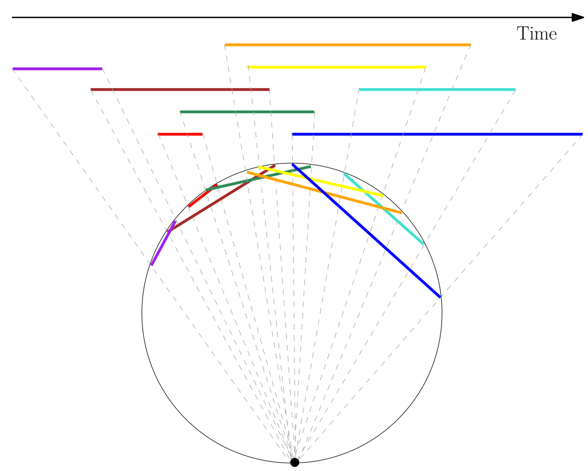

Figure 2: An overlap graph (above) for a cargo ship, and its corresponding circle graph (below). Each interval in the overlap graph represents a cargo container, its two end points being the loading/unloading times of the container on the ship. *Figure inspired by a similar figure from (Gavril 1973).*

The shipping container stowage problem. SPR for circle graphs is applicable in maritime transport. Around 80% of all traded goods are transported by sea (Review of Maritime Transport 2018). Cargo shipping a billion dollar industry which leaves a considerable carbon footprint on the environment (Organisation Co-operation and Development 2021). Therefore, an efficient process for stowing freight containers on cargo ships is desirable. The process of shifting these containers is an expensive, time-consuming and

[1]More formally, let $(v_1, v_2, \ldots, v_r)$ be an s–t shortest path. Then for each $1 \leq i < j \leq r$ such that $j - i < k$, one may replace the subpath $(v_i, v_{i+1}, \ldots, v_j)$ by a completely new subpath $(u_i, u_{i+1}, \ldots, u_j)$ in a single reconfiguration step of k-SPR.

delicate task. The problem of minimizing the amount of shifting, given a ship's voyage plan, is known as the *container stowage problem*. Owing to its importance, this problem has been studied extensively (Wilson and Roach 2000; Avriel et al. 1998; Avriel, Penn, and Shpirer 2000; Tierney, Pacino, and Jensen 2014; Gajjar and Radhakrishnan 2017). A slight variation of this problem, called the *blocks relocation problem* has also been studied (Caserta, Voß, and Sniedovich 2011; Caserta, Schwarze, and Voß 2012).

One can model the container stowage problem as a graph by representing each container as a vertex, wherein two vertices are adjacent if and only if loading one container necessitates unloading the other. These graphs are called *overlap graphs*. In fact, a graph is an overlap graph if and only if it is a *circle graph* (Gavril 1973) (Figure 2). Using this, it was shown that it is NP-complete to minimize the amount of unloading/reloading of containers (Avriel, Penn, and Shpirer 2000; Tierney, Pacino, and Jensen 2014). However, there are two heuristics that give an approximate solution efficiently (Wilson and Roach 2000; Caserta, Schwarze, and Voß 2012). One heuristic uses a shortest path-based solution (Caserta, Schwarze, and Voß 2012), while the other reshuffles the containers in a smart way while limiting the number of possible moves for each container (Avriel et al. 1998).

When the containers are reshuffled at a port, a major operational challenge is to maintain the quality of the solution. Unloading a container (say C) at its destination port requires removing the containers stowed above it (called overstowed containers). As all these overstowed containers are adjacent to the vertex C in the overlap graph, a good strategy is to maintain a path from C to the vertex that corresponds to the container at the top of C's stack at each port.

The train marshalling problem. We solve SPR for circle graphs by solving SPR for a subclass of circle graphs called *permutation graphs*, and then generalizing our solution to circle graphs. We do this by considering all possible "equators" of the circle graph. Thus, our algorithm for permutation graphs is more efficient than our algorithm for circle graphs. In fact, permutation graphs themselves model a problem similar to container stowage called the *train marshalling problem* (Dahlhaus et al. 2000; Jaehn, Rieder, and Wiehl 2015; Rinaldi and Rizzi 2017; Dörpinghaus and Schrader 2018; Falsafain and Tamannaei 2020). Both permutation graphs and circle graphs also have applications in memory allocation for system programs (Even and Itai 1971; Even, Pnueli, and Lempel 1972). For a comprehensive survey on permutation graphs and circle graphs, see (Golumbic 1980; Brandstädt, Le, and Spinrad 1999).

Data packet rerouting. In an efficient synchronous multiprocessing environment, it is widely assumed that there is a common memory and processors having sequential capabilities can access it simultaneously and almost arbitrarily (Valiant and Brebner 1981). Such a network of processors had been realised in a *d-dimensional Boolean hypercube* (Hayes et al. 1986). The routing of message packets in such a network happens via a greedy scheme which follows shortest paths (Stamoulis and Tsitsiklis 1994). The main challenge here is to perform routing in a congestion-free manner, and a lot of research had gone into this (Pifarré et al. 1994; Grammatikakis, Hsu, and Sibeyn 1998). A natural solution is to gradually reroute the packets to a different route (Greenberg and Hajek 1992), which is precisely the SPR problem on the Boolean hypercube.

Revamping road networks. k-SPR has a natural application in restructuring road networks. Suppose you are a city planner and your city's road network needs to be revamped to better serve the requirements of its residents. For this, you want to change the route between two point locations s and t in the city. It is not possible to change the entire route in one go, as laying out new roads takes resources, effort and time. Furthermore, this transition should be smooth. You do not want your ongoing renovation project to cause undue congestion on some roads, leading to a disruption in the overall flow of traffic. In other words, your job is to alter the s–t route gradually (one road at a time), whilst ensuring that road commuters do not have to undertake a longer route from s to t during the process.

A similar scenario arises in road accidents (Wang et al. 2016). This can sometimes lead to a certain road becoming inoperable, leading to bottleneck situations that could increase the travel times of the commuters. In this case, it should be possible to quickly find a way to reroute the traffic gradually and efficiently.

In SPR, only one vertex can be changed at each reconfiguration step, by definition. This condition can be sometimes too restrictive for practical purposes. When a graph is used to model a road network, roads are generally represented by simple induced paths, and vertices on the path represent various landmarks like bus stops, gas stations, shops, etc. (Bast, Funke, and Matijevic 2006; Bast et al. 2007; Bauer and Delling 2009; Goldberg, Kaplan, and Werneck 2006).

To model the fact that all these consecutive vertices can be changed in one go, we introduce the k-SPR problem, where one can change at most k (for some fixed positive integer k) contiguous vertices at each reconfiguration step. We study the optimization variant of k-SPR, where each road has a "cost of construction" associated with it and the aim is to produce a reconfiguration sequence whose total construction cost is close to optimal.

We study k-SPR for line graphs and graph powers. These graph classes give us interesting theoretical results that enhance our understanding of SPR. Optimization variants of other types of reconfiguration problems (e.g., reconfiguring swarm robots) have also been studied previously (Kirkpatrick and Liu 2016; Demaine et al. 2019).

2 Hardness Results

Note that k-SPR for $k = 1$ is precisely the SPR problem, which is known to be PSPACE-complete (Bonsma 2013). Note that the PSPACE-hardness of SPR does not straightaway imply the PSPACE-hardness of k-SPR. We show that k-SPR is PSPACE-complete, even for a restricted graph class called line graphs.

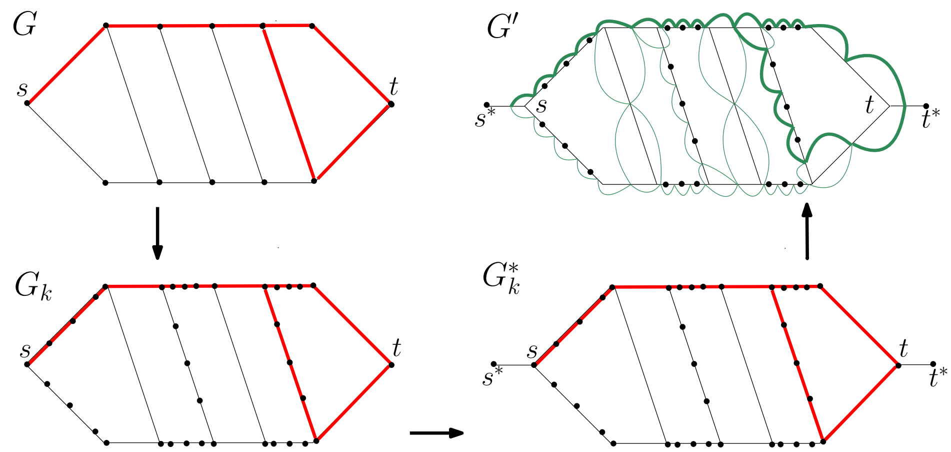

Figure 3: The construction of $G' = L(G_k^*)$ to show PSPACE-hardness of k-SPR for $k = 5$. The two s–t shortest paths denoted in bold red in G differ in only 1 vertex. The corresponding s^*–t^* shortest paths denoted in bold green in G' differ in 5 vertices.

2.1 Hardness of k-SPR for Line Graphs

In this section, we will see that k-SPR (for $k \geq 2$) can be significantly harder than SPR. In particular, we show that k-SPR is PSPACE-complete for line graphs. In contrast, (Bonsma 2013) showed that SPR can be solved in polynomial time for line graphs (in fact, for claw-free graphs, a superclass of line graphs).

Definition 2. *Given a graph G on m edges, its line graph $L(G)$ is an m-vertex graph where each vertex of $L(G)$ corresponds to an edge of G, such that two vertices of $L(G)$ are adjacent if and only if their corresponding edges in G share a vertex (see Figure 4 for an example).*

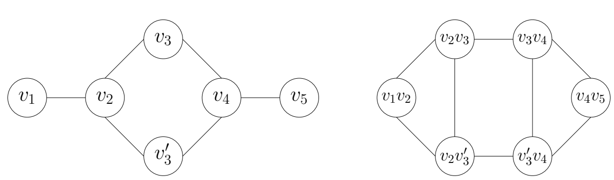

Figure 4: A graph (left) and its line graph (right)

Lemma 1. *k-SPR is PSPACE-complete for all fixed (constant) integers $k \geq 2$, even when the input graphs are restricted to line graphs.*

Proof. Fix an integer $k \geq 2$. We reduce SPR on general graphs to k-SPR on line graphs.

For our proof, we study graphs in their *layered representation*. Such representations have been studied in the context of shortest paths in the past (Dinic 1970; Dinitz 2006). It is instructive to follow Figure 3 while reading this proof. Suppose we are given an SPR instance (G, s, t, P, Q), where P and Q are s–t shortest paths in G. The goal is to check whether P and Q are reconfigurable in G. From the SPR instance (G, s, t, P, Q), we will construct a k-SPR instance (G', s', t', P', Q'), where P' and Q' are s'-t' shortest paths in G', such that P' and Q' are k-reconfigurable in G' if and only if P and Q are reconfigurable in G. Also, G' is a line graph that can be constructed from G in polynomial time in three steps (i), (ii), (iii), as explained below.

Step (i): Consider the layered graph representation of G, with s being the zeroth layer and t being the last layer. This can be done by constructing a BFS tree rooted at s. Now replace every "even-odd" edge (i.e., every edge connecting a vertex in layer i to a vertex in layer $i+1$, for every even i) by a path on k vertices between the two end points of the edge. Note that if $k = 2$, then this last operation does nothing. Let this new graph be G_k, and the new paths corresponding to P and Q in G_k be P_k and Q_k, respectively.

Step (ii): Add two vertices s^* and t^* to G_k such that s^* is adjacent only to s, and t^* is adjacent only to t. Let this new graph be G_k^*. The start vertex of G_k^* is s^* and the target vertex of G_k^* is t^*. Thus, each s–t shortest paths of G corresponds to an s^*–t^* shortest paths of G_k^* whose first edge is always (s^*, s) and last edge is always (t, t^*).

Step (iii): Let $G' = L(G_k^*)$. Since G' is the line graph of G_k^*, each vertex of G' is labelled by two vertices of G_k^*. That is, a vertex xy in G' (where x and y are two adjacent vertices of G_k^*) corresponds to an edge (x, y) in G^*. The vertex s^*s is our start vertex s' and the vertex tt^* is our target vertex t'.

This completes our construction of G'. The paths P' and Q' in G' have their first vertex as s^*s, and their last vertex as tt^*. Their remaining vertices are the edges on the paths P_k and Q_k, respectively. Given the fact that P and Q are s–t shortest paths in G, it is easy to check that P' and Q' are s'–t' shortest paths in G'. This completes the definition of the k-SPR instance (G', s', t', P', Q'). We make the following claim, whose proof will complete our proof of Lemma 1.

Claim 1. *(G, s, t, P, Q) is a yes-instance of SPR $\iff$ (G', s', t', P', Q') is a yes-instance of k-SPR.*

$\Rightarrow$ direction: Every reconfiguration step in G changes some vertex u_i in layer i to a vertex v_i in the same layer, where $(u_{i-1}, u_i, u_{i+1}, v_i)$ is a 4-cycle in G. Note that u_i can never be s or t, so it cannot be present in the zeroth or last layer of G. Thus, the graph G_k^* contains a vertex u_{i-2} (possibly s^*) and a vertex u_{i+2} (possibly t^*). Both $u_{i-2}u_{i-1}$ and

$u_{i+1}u_{i+2}$ are vertices in the line graph $G' = L(G_k^*)$. Among the two edges (u_{i-1}, u_i) and (u_i, u_{i+1}) in G, one is retained as an edge in G_k and one is converted to a path on k vertices in G_k (depending on whether i is odd or even). The retained edge contributes to a single vertex in the line graph G', and the path on k vertices (or $k-1$ edges) contributes $k-1$ vertices to the line graph G'. Thus, there are $1 + (k-1) = k$ vertices between $u_{i-2}u_{i-1}$ and $u_{i+1}u_{i+1}$ on the path P' in G'. These k vertices are reconfigured to another set of k vertices on the path Q' in G'.

$\Leftarrow$ direction: Consider a reconfiguration step in G' which replaces a subpath of j (where $j \leq k$) vertices on a shortest s'–t' path by another subpath of j vertices. Since G' is the line graph of G_k^*, these j vertices of G' can be mapped back to a subpath of j edges in G_k^* (i.e., a subpath of $j+1$ vertices in G_k^*). Let x and y be the first and last vertices of the subpath comprised by these $j+1$ vertices in G_k^*. It is easy to see that neither x nor y are not changed by mapping the reconfiguration step in G' back to a reconfiguration step in G_k^*. Note that x is adjacent to at least two vertices in the next layer in G_k^* (thus $x \neq s^*$ and so $x \in G_k$) and y is adjacent to at least two vertices in the previous layer in G_k^* (thus $y \neq t^*$ and so $y \in G_k$). Therefore, x and y can be mapped back to vertices $\mathsf{im}(x)$ and $\mathsf{im}(y)$ in G, because all "new" vertices of G_k are adjacent to only one vertex in the next layer and only one vertex in the previous layer. Finally, if $\mathsf{im}(x)$ is in layer i of G (for some i), then $\mathsf{im}(y)$ must be in layer $i+2$ of G. This is because if $\mathsf{im}(y)$ lies in a layer before $i+2$ (i.e., $i+1$), then $(\mathsf{im}(x), \mathsf{im}(y))$ would be a multiple edge in G, which is a contradiction. And if $\mathsf{im}(y)$ lies in a layer after $i+2$, then x and y would have more than k edges between them in G_k. This is also a contradiction, since $j \leq k$. □

2.2 Hardness of SPR for Graph Powers

We show that it is possible to use the PSPACE-hardness of k-SPR to prove PSPACE-hardness of SPR for some graph classes, namely graph powers.

Definition 3. *The k-th power of a graph G is obtained by making all vertices u, v such that $d(u,v) \leq k$ adjacent.*

Theorem 1. SPR *is* PSPACE*-complete for graph powers.*

Our proof technique is as follows. Let G^k be the k-th graph power of G. We use the PSPACE-hardness of $(2k-1)$-SPR for G to show the PSPACE-hardness of SPR (or 1-SPR) for G^k. A proof of this theorem can be found in the full version of our paper (Gajjar et al. 2021).

2.3 Gradation of the Complexity of k-SPR

In this section, we show that for a fixed n, the complexity of k-SPR can decrease as k increases.

Theorem 2. *For every fixed (constant) integer $k \geq 1$, there exists an n-vertex graph G and two reconfigurable s–t shortest paths P_1 and P_2 in G such that the length of every reconfiguration sequence from P_1 to P_2 is at least* $\exp\left(\Omega\left(\sqrt{\frac{n}{k}}\right)\right)$.

Theorem 3. k-SPR *can be solved in polynomial time when $k \geq n/2$.*

Detailed proofs of Theorem 2 and Theorem 3 can be found in the full version of our paper (Gajjar et al. 2021).

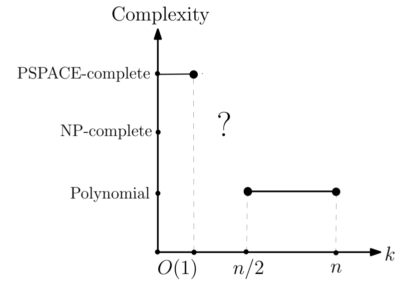

Figure 5: Complexity of k-SPR, as k varies from 1 to n

3 Polynomial-time Solvable SPR Problems

In this section, we present polynomial-time algorithms for SPR on permutation graphs, circle graphs, bridged graphs, Boolean hypercubes and graphs of constant diameter.

3.1 SPR for Permutation and Circle Graphs

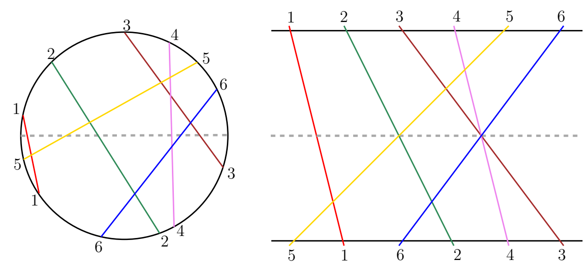

Figure 6: A circle graph with an equator (left) and the permutation graph isomorphic to the circle graph (right)

In this section, we will show that SPR can be solved in linear time for *circle graphs*.

Definition 4. *A graph is called a circle graph if its vertices can be represented by the chords of a circle such that two vertices have an edge in the graph if and only if their corresponding chords intersect.*

Given a graph, its circle representation can be constructed in quadratic time, if it exists (Golumbic 1980).

Circle graphs and permutation graphs. A circle graph is called *equatorial* if an additional chord can be drawn in the circle that intersects all the other given chords. The additional chord is called the equator. It is easy to see that a graph is an equatorial circle graph if and only if it is a permutation graph (see Figure 6 for an example). Thus, permutations graphs constitute a subclass of circle graphs.

For the purpose of our proof, we devise a novel two-step labelling scheme for the vertices of circle graphs. In step one of our labelling, we label each vertex with its BFS level (i.e., a vertex is labelled i if it appears on i^{th} level of the BFS tree rooted at s). A chord labelled i intersects chords labelled $i-1$ and $i+1$ (possibly also other chords labelled i, but we ignore those). Then, we orient the chord i **from** the point of intersection of the $i-1$ chord on chord i **to** the point of intersection of the $i+1$ chord on the chord i.

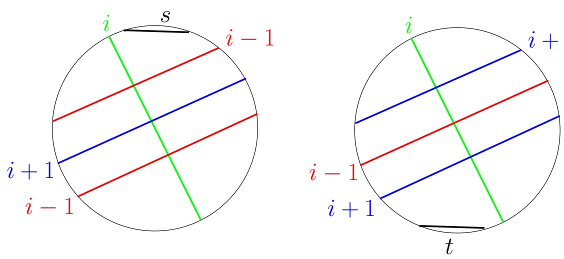

Figure 7: These two figures show a contradiction if a chord has both orientations, proving Lemma 2. (Left) s cannot reach the $i-1$ below without intersecting $i+1$. (Right) The $i+1$ above cannot reach t without intersecting $i-1$.

Lemma 2. *The orientation of the chords is unambiguous, i.e., there is no chord with both possible orientations.*

(The idea behind our proof of Lemma 2 is illustrated by Figure 7.) In step two of our labelling, we define a labelling scheme of the chords based on their orientation by step one. This labelling is the basis of our proof of our main theorem.

Theorem 4. *Two s–t shortest paths in a circle graph are reconfigurable if and only they have the same label. Furthermore, the reconfiguration sequence can be obtained in linear time (if it exists).*

Detailed proofs of Lemma 2 and Theorem 4 can be found in the full version of our paper (Gajjar et al. 2021).

3.2 SPR for Bridged Graphs

We begin this section with some definitions. Let G be a graph, and u, v be two vertices of G. Their *interval* $I(u, v)$ is the set of all vertices of G that lie on at least one shortest u-v path. More formally,

$$I(u, v) = \{w \in V(G) \colon d(u, w) + d(w, v) = d(u, v)\}.$$

A subset of vertices H of $V(G)$ is called *convex* if for each pair of vertices $(u, v) \in H \times H$, their interval $I(u, v) \subseteq H$.

Definition 5. *A graph G is called a* bridged *graph if the neighbourhood of every convex set in G is also convex.*

It is known that bridged graphs are precisely the graphs in which all isometric cycles have length three (Soltan and Chepoi 1983; Farber and Jamison 1987). In particular, all chordal graphs are bridged. Bonsma (Bonsma 2013) showed that SPR can be solved in polynomial time for chordal graphs. We extend Bonsma's result to bridged graphs.

Let us now look at some properties of bridged graphs. A graph is called *weakly modular* if it satisfies the following two conditions (Bandelt and Chepoi 1996; Chepoi 1989).

- **Quadrangle condition:** $\forall\ u, v, w, z \in V(G)$ with $k := d(u, v) = d(u, w), d(u, z) = k + 1$ and $vz, wz \in E(G)$, $\exists\ x \in V(G)$ such that $d(u, x) = k - 1$ and $xv, xw \in E(G)$.
- **Triangle condition:** $\forall\ u, v, w \in V(G)$ with $k := d(u, v) = d(u, w)$ and $vw \in E(G)$, $\exists\ x \in V(G)$ such that $d(u, x) = k - 1$ and $xv, xw \in E(G)$.

Bridged graphs are weakly modular graphs with no induced cycle of length four or five (Chepoi 1989). We essentially present a polynomial-time algorithm for SPR for weakly modular graphs. Our algorithm recursively uses the triangle condition from the above definition. For general graphs, such a recursion would make the running time exponential. We use a suitable data structure (apposite for weakly modular graphs) to make the running time polynomial.

We denote a shortest path between s and t going through the vertex w by $P_{s,w,t}$.

Algorithm 1: SPR for weakly modular graphs

Input: G, paths $P_{s,u_\ell,t}$ and $P_{s,v_\ell,t}$

if $w_{\ell-1} \in P_{s,u_\ell,t}$ **then**
  Output $u_l \to v_l$, SPR$(G, P_{s,w_{\ell-1},v_\ell}, P_{s,v_{\ell-1},v_\ell})$
end if
if $w_{\ell-1} \in P_{s,v_\ell,t}$ **then**
  Output SPR$(G, P_{s,u_{\ell-1},u_\ell}, P_{s,w_{\ell-1},u_\ell})$, $u_l \to v_l$
end if
if $w_{\ell-1} \notin P_{s,u_\ell,t}, P_{s,v_\ell,t}$ **then**
  Output SPR$(G, P_{s,u_{\ell-1},u_\ell}, P_{s,w_{\ell-1},u_\ell})$, $u_l \to v_l$, SPR$(G, P_{s,w_{\ell-1},v_\ell}, P_{s,v_{\ell-1},v_\ell})$
end if

Lemma 3. *Algorithm 1 solves* SPR *on weakly modular graphs in $O(2^\ell n)$ time, where $\ell = d(s, t)$.*

The running time of Algorithm 1 is clearly exponential in n when $\ell = \Theta(n)$. This can be improved. Consider the following data structure.

Definition 6. Lookup(u_i, v_i)

- ***Input:*** *Two vertices u_i and v_i from layer i of the* BFS *tree rooted at s.*
- ***Output:*** *A vertex w_{i-1} from layer $i - 1$ of the* BFS *tree rooted at s, such that $(w_{i-1}, u_i), (w_{i-1}, v_i) \in E(G)$.*

We construct Lookup(u_i, v_i) by searching for common parents for every pair of vertices in a BFS layer. Implementing Lookup(u_i, v_i) takes $O(n^3)$ space. Finding a w at each step using this data structure requires only a constant amount of time. Finally, the BFS naturally partitions the vertices of G into layers, reducing the running time to $O(n^2)$.

Theorem 5. SPR *can be solved in $\mathcal{O}(n^2)$ time for weakly modular graphs.*

Corollary 1. SPR *can be solved in $\mathcal{O}(n^2)$ time for bridged graphs.*

Detailed proofs of Lemma 3 and Theorem 5 can be found in the full version of our paper (Gajjar et al. 2021).

3.3 SPR for Boolean Hypercubes

A d-dimensional Boolean hypercube is a graph whose vertex set is $\{0, 1\}^d$, and two vertices are adjacent if and only if their corresponding bit strings differ by exactly one bit (Figure 8). As an input of SPR, we are given two paths P_1 and P_2 of length k with terminal vertices s and t.

Let $\ell = s \oplus t$ where $\oplus$ denotes the bitwise XOR operation. We write $\bar{\ell}$ to denote the indices where $\ell_i = 1$.

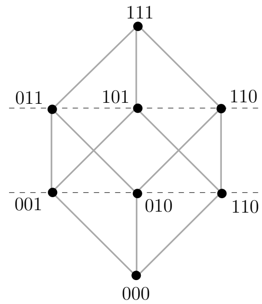

Figure 8: The 3-dimensional Boolean hypercube

For example, if $d = 5$, $s = 00101$ and $t = 10011$, then $\ell = 10110$ and $\bar{\ell} = \{1, 3, 4\}$. Any shortest path from s to t has to change these indices. This can be done in $3! = 6$ ways: $(134), (143), (314), (341), (413), (431)$. In other words, there are six possible s–t shortest paths in this example. So we represent s–t shortest paths as permutations.

Observation 2. *An s–t shortest path given by the permutation $(i_1 i_2 \ldots i_{j-1} i_j i_{j+1} i_{j+2} \ldots i_k)$ can be reconfigured to another s–t shortest path given by the permutation $(i_1\ i_2 \ldots i_{j-1} i_{j+1} i_j i_{j+2} \ldots i_k)$ in a single reconfiguration step.*

Algorithm 2: SPR for Boolean hypercubes

Input: Permutations P_1 and P_2

```
for (i, j) such that 1 ≤ i < j ≤ k do
    if P_2^{-1}(P_1[i]) > P_2^{-1}(P_1[j]) then
        Swap P_1[i] and P_1[j] in P_2
    end if
end for
```

Theorem 6. *Algorithm 2 reconfigures two given s–t shortest paths P_1 and P_2 in a Boolean hypercube in the minimum number of reconfiguration steps.*

A detailed proof of Theorem 6 can be found in the full version of our paper (Gajjar et al. 2021).

3.4 SPR for Constant Diameter Graphs

Theorem 7. *Let G be an n-vertex graph such that $d(s,t) = c$. Then* SPR *can be solved in $n^{\mathcal{O}(c)}$ time for G.*

The following is well-known and easy to see.

Observation 3. *Every split graph and every co-bipartite graph has diameter at most 3.*

This implies $d(s,t) \leq 3$ for split graphs and co-bipartite graphs, leading to the following corollary of Theorem 7.

Corollary 2. SPR *can be solved in polynomial-time for split graphs and co-bipartite graphs.*

4 Optimization Variants of SPR

In this section, we introduce three variants of the SPR problem. In this new setting, we are allowed to change any number of vertices at a time. But change comes at a cost. We pay a price of p_i for changing i vertices on a path. Furthermore,

$$p_1 \leq p_2 \leq \cdots \leq p_{n-1} \leq p_n.$$

Definition 7 (MinSumSPR). *Given (G, s, t, P_1, P_2), an instance of* SPR*, output a reconfiguration sequence from P_1 to P_2 (if it exists) that minimises the **total** cost of reconfiguration.*

Definition 8 (MinMaxSPR). *Given (G, s, t, P_1, P_2), an instance of* SPR*, output a reconfiguration sequence from P_1 to P_2 (if it exists) that minimises the **maximum** cost of reconfiguration.*

Generalizing these two definitions, we get the following.

Definition 9 (MinTop-ℓ-SPR). *Given (G, s, t, P_1, P_2), an instance of* SPR*, output a reconfiguration sequence from P_1 to P_2 (if it exists) that minimises the **sum total of the maximum ℓ (or top-ℓ)** costs of reconfiguration.*

Note that MinSumSPR is a special case of MinTop-ℓ-SPR with $\ell = \infty$ and MinMaxSPR is a special case of MinTop-ℓ-SPR with $\ell = 1$. The following is easy to see.

Observation 4. *For every positive integer n, the number of s–t shortest paths in every n-vertex graph is at most 2^n.*

Theorem 8. *There is no polynomial-time algorithm that approximates MinTop-ℓ-*SPR *to within a factor of $O(2^{n^2})$, unless* PSPACE = P.

A proof of Theorem 8 can be found in the full version of our paper (Gajjar et al. 2021).

5 Length versus Number of Shortest Paths

In this section, we study how the number of s–t shortest paths ($|V_{SPR}|$) varies with the s–t distance ($d(s,t)$). Let $f(x)$ be the maximum value of $|V_{SPR}|$ when $d(s,t) = x$.

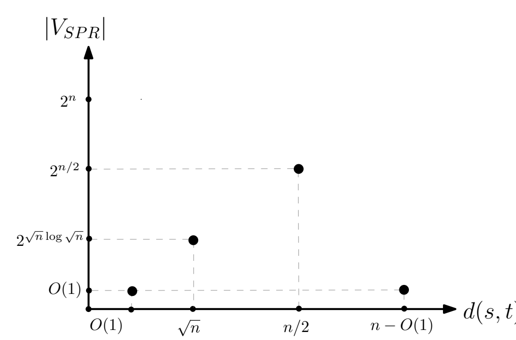

Figure 9: The maximum possible number of s–t shortest paths, as the distance between s and t varies from 0 to $n-1$

It is easy to see that $f(x) \leq 2^n$ (Lemma 4) for all $0 \leq d(s,t) \leq n-1$. For other specific values of $d(s,t)$, we have the following stronger bounds, represented by Figure 9.

Lemma 4.

$$d(s,t) = x = n/2 \quad \Rightarrow \quad f(x) = \Omega(2^{n/2});$$
$$d(s,t) = x = \Theta(\sqrt{n}) \quad \Rightarrow \quad f(x) = \Omega\left(2^{\sqrt{n}\log\sqrt{n}}\right);$$
$$d(s,t) = x = n - O(1) \quad \Rightarrow \quad f(x) = O(1).$$

A proof of this lemma can be found in the full version of the paper (Gajjar et al. 2021).

Acknowledgements

Kshitij Gajjar's research at National University of Singapore is supported by an NUS ODPRT Grant, WBS No. R-252-000-A94-133. A part of this work was also done when he was a Post-Doctoral Fellow at Technion, Israel.

Agastya Vibhuti Jha would like to thank Dr. Jatin Batra for introducing him to Ordered Optimization, which gave the idea of k-SPR.

Abhiruk Lahiri's research is supported by an Ariel University Post-Doctoral Fellowship, Israel Science Foundation, Grant Nos. 592/17 and 822/18, and the ERC-CZ project LL2005. He would like to thank the Caesarea Rothschild Institute and the Department of Computer Science, University of Haifa, for providing its facilities to carry out his research.

References

Asplund, J.; Edoh, K. D.; Haas, R.; Hristova, Y.; Novick, B.; and Werner, B. 2018. Reconfiguration graphs of shortest paths. *Discret. Math.*, 341(10): 2938–2948.

Asplund, J.; and Werner, B. 2020. Classification of reconfiguration graphs of shortest path graphs with no induced 4-cycles. *Discret. Math.*, 343(1): 111640.

Avriel, M.; Penn, M.; and Shpirer, N. 2000. Container ship stowage problem: complexity and connection to the coloring of circle graphs. *Discret. Appl. Math.*, 103(1-3): 271–279.

Avriel, M.; Penn, M.; Shpirer, N.; and Witteboon, S. 1998. Stowage planning for container ships to reduce the number of shifts. *Ann. Oper. Res.*, 76: 55–71.

Bandelt, H.; and Chepoi, V. 1996. A Helly theorem in weakly modular space. *Discret. Math.*, 160(1-3): 25–39.

Bast, H.; Funke, S.; and Matijevic, D. 2006. Ultrafast Shortest-Path Queries via Transit Nodes. In *The Shortest Path Problem, Proceedings of a DIMACS Workshop, Piscataway, New Jersey, USA, November 13-14, 2006*, volume 74 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 175–192.

Bast, H.; Funke, S.; Matijevic, D.; Sanders, P.; and Schultes, D. 2007. In Transit to Constant Time Shortest-Path Queries in Road Networks. In *Proceedings of the Nine Workshop on Algorithm Engineering and Experiments, ALENEX 2007, New Orleans, Louisiana, USA, January 6, 2007*. SIAM.

Bauer, R.; and Delling, D. 2009. SHARC: Fast and robust unidirectional routing. *ACM J. Exp. Algorithmics*, 14.

Bonsma, P. S. 2013. The complexity of rerouting shortest paths. *Theor. Comput. Sci.*, 510: 1–12.

Bonsma, P. S. 2017. Rerouting shortest paths in planar graphs. *Discret. Appl. Math.*, 231: 95–112.

Brandstädt, A.; Le, V. B.; and Spinrad, J. P. 1999. *Discrete Mathematics and Applications*. SIAM. ISBN 978-0-898714-32-6.

Caserta, M.; Schwarze, S.; and Voß, S. 2012. A mathematical formulation and complexity considerations for the blocks relocation problem. *Eur. J. Oper. Res.*, 219(1): 96–104.

Caserta, M.; Voß, S.; and Sniedovich, M. 2011. Applying the corridor method to a blocks relocation problem. *OR Spectr.*, 33(4): 915–929.

Chepoi, V. 1989. Classification of graphs by means of metric triangles. *Methdy Diskret. Analiz.*, 96: 75–93.

Dahlhaus, E.; Horák, P.; Miller, M.; and Ryan, J. F. 2000. The train marshalling problem. *Discret. Appl. Math.*, 103(1-3): 41–54.

Demaine, E. D.; Demaine, M. L.; Eisenstat, S.; Lubiw, A.; and Winslow, A. 2011. Algorithms for Solving Rubik's Cubes. In *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, volume 6942 of *Lecture Notes in Computer Science*, 689–700.

Demaine, E. D.; Eisenstat, S.; and Rudoy, M. 2018. Solving the Rubik's Cube Optimally is NP-complete. In *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, 24:1–24:13.

Demaine, E. D.; Fekete, S. P.; Keldenich, P.; Meijer, H.; and Scheffer, C. 2019. Coordinated Motion Planning: Reconfiguring a Swarm of Labeled Robots with Bounded Stretch. *SIAM J. Comput.*, 48(6): 1727–1762.

Dinic, E. A. 1970. Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation. In *Soviet Math. Doklady*, volume 11, 1277–1280.

Dinitz, Y. 2006. Dinitz' Algorithm: The Original Version and Even's Version. In *Theoretical Computer Science, Essays in Memory of Shimon Even*, 218–240.

Dörpinghaus, J.; and Schrader, R. 2018. A Graph-Theoretic Approach to the Train Marshalling Problem. In *Proceedings of the 2018 Federated Conference on Computer Science and Information Systems, FedCSIS 2018, Poznań, Poland, September 9-12, 2018*, volume 15 of *Annals of Computer Science and Information Systems*, 227–231.

Even, S.; and Itai, A. 1971. Queues Stacks and Graphs. In *Theory of Machines and Computations: Proceedings of an International Symposium on the Theory of Machines and Computations*, 71–86. Academic Press.

Even, S.; Pnueli, A.; and Lempel, A. 1972. Permutation Graphs and Transitive Graphs. *J. ACM*, 19(3): 400–410.

Falsafain, H.; and Tamannaei, M. 2020. A Novel Dynamic Programming Approach to the Train Marshalling Problem. *IEEE Trans. Intell. Transp. Syst.*, 21(2): 701–710.

Farber, M.; and Jamison, R. E. 1987. On local convexity in graphs. *Discret. Math.*, 66(3): 231–247.

Gajjar, K.; Jha, A. V.; Kumar, M.; and Lahiri, A. 2021. Reconfiguring Shortest Paths in Graphs. *CoRR, arXiv preprint arXiv:2112.07499.*

Gajjar, K.; and Radhakrishnan, J. 2017. Distance-Preserving Subgraphs of Interval Graphs. In Pruhs, K.; and Sohler, C., eds., *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 39:1–39:13. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-049-1.

Gavril, F. 1973. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3(3): 261–273.

Goldberg, A. V.; Kaplan, H.; and Werneck, R. F. 2006. Reach for A*: Efficient Point-to-Point Shortest Path Algorithms. In *Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments, ALENEX 2006, Miami, Florida, USA, January 21, 2006*, 129–143. SIAM.

Goldreich, O. 2011. Finding the Shortest Move-Sequence in the Graph-Generalized 15-Puzzle Is NP-Hard. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, volume 6650 of *Lecture Notes in Computer Science*, 1–5. Springer.

Golumbic, M. C. 1980. *Algorithmic graph theory and perfect graphs*. Academic Press. ISBN 978-0-12-289260-8.

Grammatikakis, M. D.; Hsu, D. F.; and Sibeyn, J. F. 1998. Packet Routing in Fixed-Connection Networks: A Survey. *J. Parallel Distributed Comput.*, 54(2): 77–132.

Greenberg, A. G.; and Hajek, B. E. 1992. Deflection routing in hypercube networks. *IEEE Trans. Commun.*, 40(6): 1070–1081.

Hayes, J. P.; Mudge, T. N.; Stout, Q. F.; Colley, S.; and Palmer, J. 1986. A Microprocessor-based Hypercube Supercomputer. *IEEE Micro*, 6(5): 6–17.

Hopcroft, J. E.; Joseph, D.; and Whitesides, S. 1984. Movement Problems for 2-Dimensional Linkages. *SIAM J. Comput.*, 13(3): 610–629.

Hopcroft, J. E.; Schwartz, J. T.; and Sharir, M. 1984. On the Complexity of Motion Planning for Multiple Independent Objects; PSPACE-Hardness of the "Warehouseman's Problem". *The International Journal of Robotics Research*, 3(4): 76–88.

Hopcroft, J. E.; and Wilfong, G. T. 1986. Reducing Multiple Object Motion Planning to Graph Searching. *SIAM J. Comput.*, 15(3): 768–785.

Jaehn, F.; Rieder, J.; and Wiehl, A. 2015. Minimizing delays in a shunting yard. *OR Spectr.*, 37(2): 407–429.

Kaminski, M.; Medvedev, P.; and Milanic, M. 2010. Shortest Paths between Shortest Paths and Independent Sets. In *Combinatorial Algorithms - 21st International Workshop, IWOCA 2010, July 26-28, 2010, Revised Selected Papers*, volume 6460 of *Lecture Notes in Computer Science*, 56–67.

Kirkpatrick, D.; and Liu, P. 2016. Characterizing minimum-length coordinated motions for two discs. In *Proceedings of the 28th Canadian Conference on Computational Geometry, CCCG 2016, August 3-5, 2016, Simon Fraser University, Vancouver, British Columbia, Canada*, 252–259.

Lokshtanov, D.; and Mouawad, A. E. 2018. The complexity of independent set reconfiguration on bipartite graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, 185–195. SIAM.

Lokshtanov, D.; Mouawad, A. E.; Panolan, F.; Ramanujan, M. S.; and Saurabh, S. 2018. Reconfiguration on sparse graphs. *J. Comput. Syst. Sci.*, 95: 122–131.

Mouawad, A. E.; Nishimura, N.; Pathak, V.; and Raman, V. 2017. Shortest Reconfiguration Paths in the Solution Space of Boolean Formulas. *SIAM J. Discret. Math.*, 31(3): 2185–2200.

Mouawad, A. E.; Nishimura, N.; Raman, V.; and Siebertz, S. 2018. Vertex Cover Reconfiguration and Beyond. *Algorithms*, 11(2): 20.

Organisation Co-operation and Development. 2021. Organisation for Economic Co-operation and Development, Ocean shipping and shipbuilding. URL: https://www.oecd.org/ocean/topics/ocean-shipping/. Accessed: 8th September 2021.

Pifarré, G. D.; Gravano, L.; Felperin, S. A.; and Sanz, J. L. C. 1994. Fully Adaptive Minimal Deadlock-Free Packet Routing in Hypercubes, Meshes, and other Networks: Algorithms and Simulations. *IEEE Trans. Parallel Distributed Syst.*, 5(3): 247–263.

Ratner, D.; and Warmuth, M. K. 1986. Finding a Shortest Solution for the N × N Extension of the 15-PUZZLE Is Intractable. In *Proceedings of the 5th National Conference on Artificial Intelligence. Philadelphia, PA, USA, August 11-15, 1986. Volume 1: Science*, 168–172.

Review of Maritime Transport. 2018. United Nations Conference on Trade and Development. URL: https://unctad.org/webflyer/review-maritime-transport-2018. Accessed: 8th September 2021.

Rinaldi, F.; and Rizzi, R. 2017. Solving the train marshalling problem by inclusion-exclusion. *Discret. Appl. Math.*, 217: 685–690.

Sedgewick, R.; and Wayne, K. 2016. *Algorithms*. Addison-Wesley. ISBN 978-0-1343-8468-9.

Soltan, V.; and Chepoi, V. 1983. Conditions for invariance of set diameters under d-convexification in a graph. *Cybernetics*, 19(6): 750–756.

Stamoulis, G. D.; and Tsitsiklis, J. N. 1994. The efficiency of greedy routing in hypercubes and butterflies. *IEEE Trans. Commun.*, 42(11): 3051–3061.

Tierney, K.; Pacino, D.; and Jensen, R. M. 2014. On the complexity of container stowage planning problems. *Discret. Appl. Math.*, 169: 225–230.

Valiant, L. G.; and Brebner, G. J. 1981. Universal Schemes for Parallel Communication. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing, May 11-13, 1981, Milwaukee, Wisconsin, USA*, 263–277. ACM.

Wang, S.; Djahel, S.; Zhang, Z.; and McManis, J. 2016. Next Road Rerouting: A Multiagent System for Mitigating Unexpected Urban Traffic Congestion. *IEEE Trans. Intell. Transp. Syst.*, 17(10): 2888–2899.

Wilson, I. D.; and Roach, P. A. 2000. Container stowage planning: a methodology for generating computerised solutions. *J. Oper. Res. Soc.*, 51(11): 1248–1255.