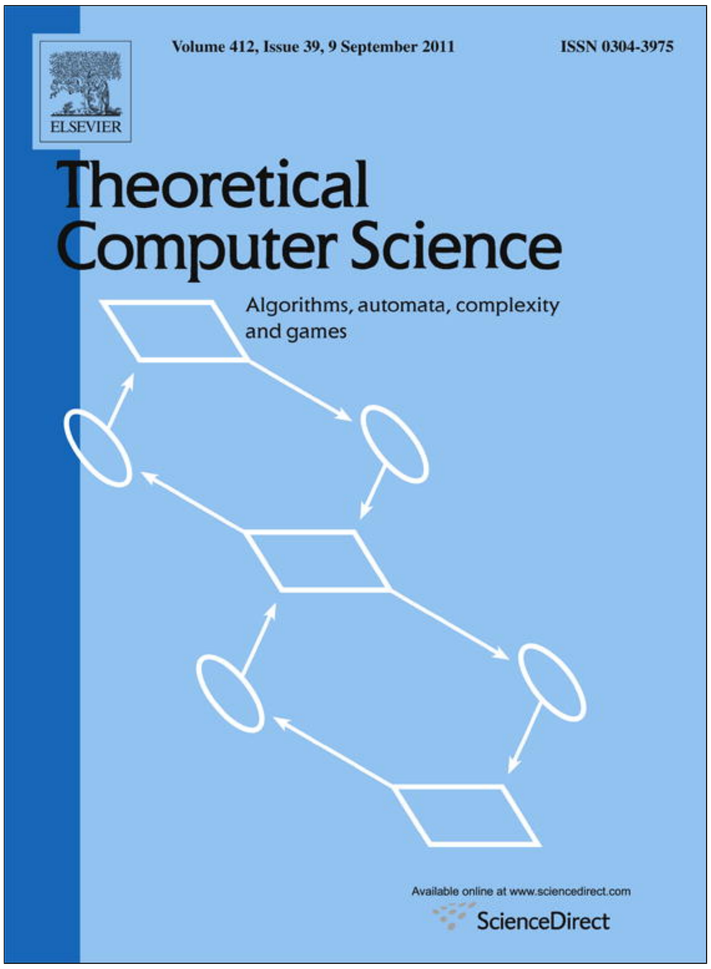
Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.


Volume 412, Issue 39, 9 September 2011
ISSN 0304-3975
ELSEVIER
Theoretical Computer Science
Algorithms, automata, complexity and games
Available online at www.sciencedirect.com
ScienceDirect

This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

http://www.elsevier.com/copyright

Author's personal copy

Theoretical Computer Science 412 (2011) 5205–5210

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Shortest paths between shortest paths[☆]

Marcin Kamiński [a,1], Paul Medvedev [b,*], Martin Milanič [c]

[a] *Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium*
[b] *Department of Computer Science, University of Toronto, Toronto, Canada*
[c] *FAMNIT and PINT, University of Primorska, Koper, Slovenia*

ARTICLE INFO

Article history:
Received 26 August 2010
Received in revised form 7 February 2011
Accepted 11 May 2011
Communicated by G. Italiano

Keywords:
Reconfiguration
Shortest path
Reconfigurability
NP-hard
Diameter

ABSTRACT

We study the following problem on reconfiguring shortest paths in graphs: Given two shortest s–t paths, what is the minimum number of steps required to transform one into the other, where each intermediate path must also be a shortest s–t path and must differ from the previous one by only one vertex. We prove that the shortest reconfiguration sequence can be exponential in the size of the graph and that it is NP-hard to compute the shortest reconfiguration sequence even when we know that the sequence has polynomial length.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

One of the biggest impacts of algorithmic graph theory has been its usefulness in modeling real-world problems, where the domain of the problem is modeled as a graph and the constraints on the solution define feasible solutions. For example, consider the problem of routing a certain commodity between two nodes in a transportation network, using as few hops as possible. The transportation network can be modeled as a graph, each route can be modeled as a path, and the feasible solutions are all the shortest paths between the two nodes. Traditionally, the real-world user first defines a problem instance and then uses an algorithm to find a feasible solution which she then "implements" in the real world. However, some real-world situations do not follow this simple paradigm and are more dynamic because they allow the solution to "evolve" over time. For example, consider the situation where the commodity is already being transferred along a shortest route, but the operator has been instructed to use a different route, which is also a shortest path. She can physically switch the route only one node at a time, but does not wish to interrupt the transfer. Thus, she would like to switch between the two routes in as few steps as possible, while maintaining a shortest path route at every intermediate step.

In general, this type of situation gives rise to a *reconfiguration* framework, where we consider an algorithmic problem $\mathcal{P}$ and a way of transforming one feasible solution of an instance I of $\mathcal{P}$ to another (*reconfiguration rule*). Given two feasible solutions s_1, s_k of I, we want to find a *reconfiguration sequence* $s_1, \ldots, s_k$ such that each s_i $(1 \leq i \leq k)$ is a feasible solution of I, and the transition between s_i and s_{i+1} is allowed by the reconfiguration rule. An alternate definition is via the *reconfiguration graph*, where the vertices are the feasible solutions of I, and two solutions are adjacent if and only if one can be obtained from the other by the reconfiguration rule. The reconfiguration sequence is then a path between s_1 and s_k in the reconfiguration

[☆] A version of this work has appeared in the Proceedings of IWOCA 2010, 21st International Workshop on Combinatorial Algorithms, London, 26–28 July 2010 (Marcin Kamiński et al., 2011) [10].

[*] Corresponding author. Tel.: +1 416 946 3924.
E-mail addresses: Marcin.Kaminski@ulb.ac.be (M. Kamiński), pashadag@cs.toronto.edu (P. Medvedev), martin.milanic@upr.si (M. Milanič).

[1] Chargé de Recherches du FRS-FNRS.

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.05.021

Author's personal copy

graph. We can then ask for the shortest reconfiguration sequence, or, in the *reconfigurability problem*, to simply check if the two solutions are *reconfigurable* (i.e., if such a sequence exists).

The reconfiguration framework has recently been applied in a number of settings, including vertex coloring [4,5,3,2], list-edge coloring [9], clique, set cover, integer programming, matching, spanning tree, matroid bases [8], block puzzles [7], independent set [7,8,10], and satisfiability [6]. In the well-studied vertex coloring problem, for example, we are given two k-colorings of a graph, and the reconfiguration rule allows us to change the color of a single vertex. In a different example, we are given two independent sets, which we imagine to be two sets of tokens placed on the vertices, and the reconfiguration rule is to slide a single token along an edge.

A topic certainly related to graph reconfiguration problems is the reconfiguration of Boolean formulas studied in [6]. The authors define a class of Boolean formulas that can be built from *tight relations* and prove three interesting dichotomy theorems. First, they consider the problem of checking if two assignments of a Boolean formula are connected. They show that for the formulas built from tight relations the question can be answered in linear time and is PSPACE-complete for the formulas not in that class. Then, they study the problem of determining if the reconfiguration graph of a Boolean formula is connected. The set of formulas that can be built from tight relations gives instances that are in coNP; the problem is PSPACE-complete for the formulas not in that class. Finally, they focus on the diameter of connected components of the reconfiguration graph. They show that the diameter is linear for the formulas that can be built from tight relations; and can be exponential otherwise.

Though the complexities of each of the many reconfiguration problems may each be studied independently, a fundamental question is whether there exists any systematic relationship between the complexity of the original problem and that of its reconfigurability problem. To this end, current studies have revealed a pattern where most "natural" problems in P have their reconfigurability problems in P as well, while problems whose reconfigurability versions are at least NP-hard are NP-complete. For example, spanning tree, matching, and matroid problems in general (all in P) lead to polynomially solvable reconfigurability problems when using the most natural reconfiguration rule, while the reconfigurability of independent set, set cover, and integer programming (all NP-complete) are PSPACE-complete [8].

Ito et al. [8] have conjectured that this relationship is not true in general, and that there exist problems in P which give rise, in a natural way, to NP-hard reconfigurability problems. Indeed, the problem of deciding whether two k-colorings are reconfigurable is PSPACE-complete for (i) bipartite graphs and $k \geq 4$, and (ii) planar graphs, for $4 \leq k \leq 6$ [2]. Clearly, 4-coloring of bipartite or planar graphs is in P. However, these are not "natural" problems in the sense that the colorings used in the PSPACE-hardness proof constructions are not optimal. It is interesting to ask if there exists a "natural" problem in P whose reconfiguration version is NP-hard.

Another systematic relationship that has been pursued is between the complexity of a reconfigurability problem and the diameter of the reconfiguration graph. When the diameter is polynomial, a reconfiguration sequence is a trivial certificate for the reconfigurability of two instances, guaranteeing that the problem is in NP. However, current evidence further suggests that for reconfigurability problems that are solvable in polynomial time, the diameter is also polynomial. In the study of k-coloring, it was found that for $k \leq 3$, the reconfigurability problem is solvable in polynomial time and the diameter of the reconfiguration graph is at most quadratic in the number of vertices of the colored graph. For satisfiability, the formulas built from tight relations (whose reconfigurability is polynomial) lead to reconfiguration graphs with linear diameter [6]. We are not aware of any natural problems with the property that the diameter can be exponential while reconfigurability can be decided in polynomial time[2]; however, such an example, if found, would indicate that the diameter cannot serve as a reliable indicator of the reconfigurability complexity.

In this paper, we introduce the reconfiguration version of the shortest path problem, which arises naturally, such as in the routing example above. We show (in Section 2) that the reconfiguration graph can have exponential diameter, implying that the shortest path reconfiguration problem probably breaks one of the two established patterns described above. On the one hand, if reconfigurability of shortest paths can be decided in polynomial time, then it is the first example of a reconfigurability problem in P with exponential diameter. On the other hand, if it is NP-hard, it is the first example of a "natural" problem in P whose reconfigurability version is NP-hard. For these reasons, we believe that shortest path reconfiguration is an important problem to study, not only for its practical application but also for our understanding of the systematic relationship between the hardness of a problem, the diameter of its reconfiguration graph, and the hardness of its reconfigurability problem. Towards this end, we give (in Section 3) a reduction from SAT to show that it is NP-hard to find the shortest reconfiguration sequence between two shortest paths.

2. Instances with exponential diameter

We define the reconfiguration rule for shortest paths in the natural way: two shortest (s, t)-paths are adjacent in the reconfiguration graph of shortest (s, t)-paths if and only if they differ, as sequences, in exactly one vertex.

We now present a family of graphs G^k whose size is linear in k but the diameter of the reconfiguration graph is $\Omega(2^k)$. The graph G^1 contains vertices $\{x_i^1 \mid 1 \leq i \leq 7\} \cup \{y_i^1 \mid 1 \leq i \leq 6\} \cup \{s, t\}$ and edges $\{(x_i^1, y_i^1), (x_{i+1}^1, y_i^1), (y_i^1, t) \mid i \leq 6\} \cup \{(s, x_i^1) \mid$

[2] For a very artificial one, consider the problem in which instances are n-bit words and two instances are adjacent if they differ by 1 modulo 2^n. The diameter of the reconfiguration graph is 2^{n-1} but all pairs of instances are reconfigurable.

Author's personal copy

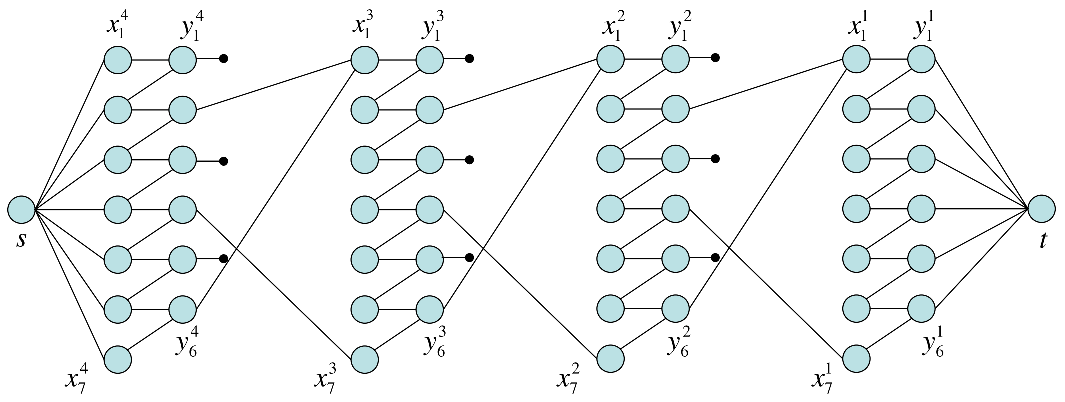


Fig. 1. The graph G^k for $k = 4$, where the reconfiguration distance between $p_b^k = s, x_1^k, y_1^k, \ldots, x_1^1, y_1^1, t$ and $p_e^k = s, x_7^k, y_6^k, x_1^{k-1}, x_1^{k-1}, \ldots, x_1^1, y_1^1, t$ is $\Theta(2^k)$. An edge with a circle end means that the vertex is connected to all the vertices in the next layer.

$1 \le i \le 7\}$. The graph G^k is defined recursively with vertices $\{x_i^k \mid 1 \le i \le 7\} \cup \{y_i^k \mid 1 \le i \le 6\} \cup V(G^{k-1})$ and the edges $\{(x_i^k, y_i^k), (x_{i+1}^k, y_i^k) \mid i \le 6\} \cup \{(y_i^k, x_j^{k-1}) \mid i \in \{1, 3, 5\}, j \le 7\} \cup \{(y_2^k, x_1^{k-1}), (y_4^k, x_7^{k-1}), (y_6^k, x_1^{k-1})\} \cup E(G^{k-1} \setminus \{s\}) \cup \{(s, x_i^k) \mid 1 \le i \le 7\}$ (see Fig. 1). Let $p_b^k = s, x_1^k, y_1^k, \ldots, x_1^1, y_1^1, t$, and let $p_e^k = s, x_7^k, y_6^k, x_1^{k-1}, x_1^{k-1}, \ldots, x_1^1, y_1^1, t$. We will consider the problem of reconfiguring p_b^k to p_e^k in G^k.

Lemma 1. *Let p be a shortest path in G^k that goes through y_1^k, and let q be a path that goes through y_6^k. Then the reconfiguration distance between p and q is at least* $9(2^k - 1)$.

Proof. We prove by induction on k, where the base case is clear. Let $\rho = p_1, \ldots, p_n$ be the shortest reconfiguration sequence between p and q. First, let i' be the smallest integer such that $p_{i'+1}$ contains y_4^k, and let $i \le i'$ be the smallest integer such that every path $p_i, \ldots, p_{i'}$ contains y_3^k. By construction, we know that p_{i-1}, and hence p_i, contains y_1^{k-1} and $p_{i'+1}$, and hence $p_{i'}$, contains y_6^{k-1}. Hence, by the induction hypothesis, the length of this first phase, $i' - i + 1$, is at least $9(2^{k-1} - 1)$.

Next, let j' be the smallest integer such that $p_{j'+1}$ contains y_6^k, and let $j \le j'$ be the smallest integer such that every path $p_j, \ldots, p_{j'}$ contains y_5^k. By construction, we know that p_{j-1}, and hence p_j, contains y_6^{k-1} and $p_{j'+1}$, and hence $p_{j'}$, contains y_1^{k-1}. Hence, by the induction hypothesis, the length of this second phase, $j' - j + 1$, is at least $9(2^{k-1} - 1)$.

Observe from the graph construction that ρ must always visit y_{x-1}^k before visiting y_x^k, hence $i' < j$, and so the length of ρ is at least the sum of the two phases plus the moves of the first and second vertex of the path necessary to percolate y_1^k down to y_6^k, proving the lemma. □

On the other hand, there exists an asymptotically matching lower bound:

Lemma 2. *The reconfiguration distance between p_b^k and p_e^k is at most* $11(2^k - 1)$.

Proof. We prove by induction on k, where the base case is clear. It will be helpful to formally treat a reconfiguration sequence not as a sequence of paths but as a sequence of vertices, each one representing the switched vertex at that step. Applying the induction hypothesis, let ρ be the shortest reconfiguration sequence in G^{k-1}, and let $rev(\rho)$ be that sequence in the reverse direction (from p_e^{k-1} to p_b^{k-1}). We construct the sequence as $\rho' = x_2^k, y_2^k, x_3^k, y_3^k, \rho, x_4^k, y_4^k, x_5^k, y_5^k, rev(\rho), x_6^k, y_6^k, x_7^k$. This sequence of moves reconfigures p_b^k into p_e^k with the number of steps satisfying the lemma. □

We therefore have the following theorem:

Theorem 1. *The reconfiguration distance in G^k between p_b^k and p_e^k is in* $\Theta(2^k)$.

3. NP-hardness of MIN-SPR

Given (G, s, t, p_b, p_e, k), where p_b and p_e are shortest (s, t)-paths and k is an integer, the MIN-SPR problem is to determine whether there is a reconfiguration sequence between p_b and p_e of length at most k. Let ϕ be a Boolean formula in conjunctive normal form with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$. We will create an instance $(G_\phi, s, t, p_b, p_e, 2m(n+2))$ and show that ϕ is satisfiable if and only if this instance is in MIN-SPR. For ease of presentation, the graph G_ϕ will be directed. However, our result holds for undirected graphs because the directed shortest (s, t)-paths in G_ϕ are exactly the shortest paths in the underlying undirected graph of G_ϕ.

For every variable x_ℓ and its possible value $vs \in \{0, 1\}$, we build a gadget $G(\ell, vs)$. The vertex set is $\{v(\ell, vs, cs, d) \mid cs \in \{0, 1\}, 1 \le d \le 2m\}$. The values ℓ, vs, cs, and d for a vertex are referred to as its *level*, *v-state* (short for variable state), *c-state* (short for clause state), and *depth*, and denoted by $\ell(v)$, $vs(v)$, $cs(v)$, and $d(v)$, respectively. For every $1 \le d \le 2m - 1$, and every cs, there is an edge from $v(\ell, vs, cs, d)$ to $v(\ell, vs, cs, d+1)$. For all $1 \le d \le m - 1$, there is an edge from $v(\ell, vs, 0, 2d)$ to $v(\ell, vs, 1, 2d+1)$, and from $v(\ell, vs, 1, 2d)$ to $v(\ell, vs, 0, 2d+1)$. We also add edges, called formula edges, that are formula

Author's personal copy

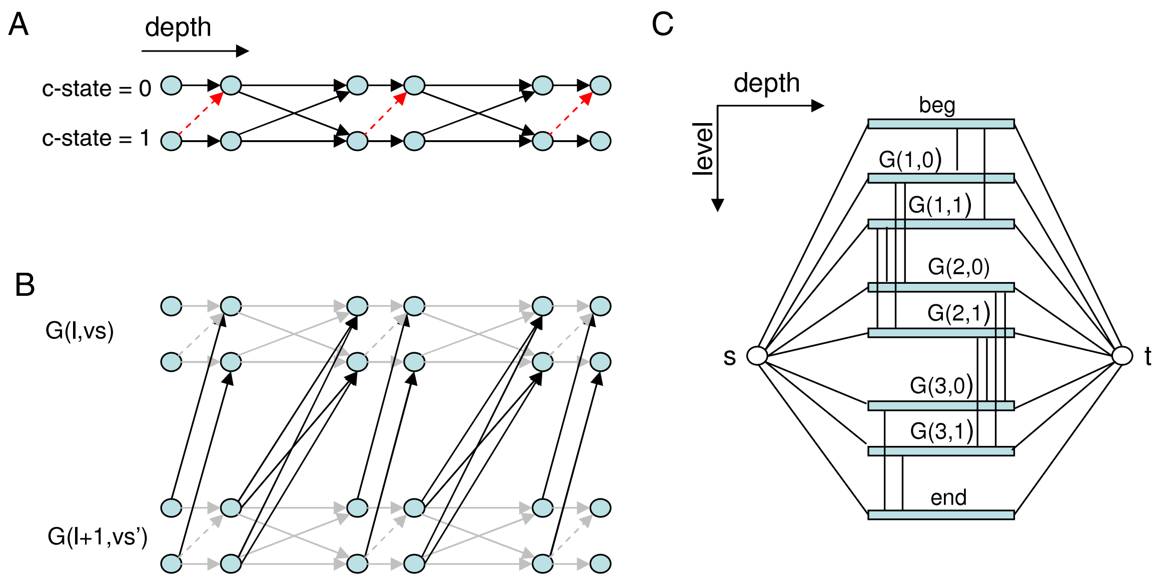


Fig. 2. The reduction from a formula ϕ to a graph G_ϕ for the case of three clauses and three variables. Panel (A) shows the internal connections of a gadget, with the potential formula edges that depend on ϕ given in red (dashed). Panel (B) shows the way we connect two given gadgets, while (C) shows the structure of the whole graph. Each of the rectangles represents a gadget, with the lines showing which gadgets are connected together.

dependent. For all d, if $x_\ell = vs$ satisfies C_d, we add an edge from $v(\ell, vs, 1, 2d-1)$ to $v(\ell, vs, 0, 2d)$. This gadget is shown in Fig. 2A.

We now connect some of these gadgets together. The gadgets we connect are $G(\ell, vs)$ to $G(\ell+1, 0)$ and to $G(\ell+1, 1)$, for all $\ell \leq n-1$ and all vs. Given two gadgets, $G(\ell, vs)$ and $G(\ell+1, vs')$, the meaning of connecting $G(\ell, vs)$ to $G(\ell+1, vs')$ is given as follows (shown in Fig. 2BC). For all $d \leq 2m-1$ and cs, there is an edge from $v(\ell+1, vs', cs, d-1)$ to $v(\ell, vs, cs, d)$. Also, for all $d \leq m-1$ and cs, there is an edge from $v(\ell+1, vs', cs, 2d)$ to $v(\ell, vs, 1-cs, 2d+1)$.

We next add a begin and end gadget to the graph, consisting of vertices beg_d and end_d, respectively, for $1 \leq d \leq 2m$. These are connected in a path, with edges (beg_d, beg_{d+1}) and (end_d, end_{d+1}) for $d \leq 2m-1$. The level of the vertices in the begin (end) gadget is 0 ($n+1$), the c-state is 0 (1), and the depth of beg_d or end_d is d. For all vs, $d \leq 2m-1$, there is an edge from $v(1, vs, 0, d)$ to beg_{d+1}, and from end_d to $v(n, vs, 1, d+1)$.

Finally, we add vertices s and t to the graph, and make an edge from s to every depth 1 vertex, and from every depth $2m$ vertex to t. The depth of s (t) is defined to be 0 ($2m+1$). We call the resulting directed graph G_ϕ. Let $p_b = s, beg_1, \ldots, beg_{2m}, t$ and $p_e = s, end_1, \ldots, end_{2m}, t$ be two paths in this graph. Then, $(G_\phi, s, t, p_b, p_e, 2m(n+2))$ is the instance of the Min-SPR problem that we will consider here.

The intuition behind the reduction is that in order for the path to percolate down from p_b to p_e in a minimal number of steps, it must pass consecutively through exactly one of $G(\ell, 0)$ or $G(\ell, 1)$ for every variable x_ℓ. The choice of which one corresponds to assigning x_ℓ the corresponding value. Furthermore, each shortest path that goes through a gadget can visit the vertex at depth $2d$ with a c-state of 0 or 1. This corresponds to having the d^{th} clause satisfied or not. Initially, the path goes only through vertices with c-state 0, and the only way to switch the c-state at a given depth is via a formula edge. By going through a gadget $G(\ell, vs)$, there is an opportunity to use the formula edges to switch the c-state of all clauses that $x_\ell = vs$ would satisfy. In order to reach the final path p_e, the c-state of all the vertices in the path must become 1, hence all the clauses must be satisfied.

Each edge (a, b) is considered to be either *inter-clause* or *intra-clause*, depending on the parity of $d(a)$. The edge is inter-clause if $d(a)$ is even. We call edges that connect vertices on the same level (exactly those that belong to the same gadget) as *intra-level*, while the edges that connect vertices on different levels are called *inter-level*. The following facts about G_ϕ follow from definitions and capture most of the properties of the reduction that are needed to prove completeness and soundness.

Fact 1. *Let $e = (a, b)$ be an edge in G_ϕ. The statements below follow directly from the construction:*

1. $\ell(b) \leq \ell(a) \leq \ell(b) + 1$.
2. *If e is an intra-level intra-clause edge, $cs(a) = 0$ implies that $cs(b) = 0$.*
3. *If e is a non-formula intra-clause edge, then $cs(a) = cs(b)$.*
4. *If e is intra-level, then $vs(a) = vs(b)$.*

First, we will show that the reduction is sound. Let $p = s, v_1, \ldots, v_{2m}, t$ be a shortest path and consider an arbitrary move that switches v_d with v'_d. The *move graph* is the subgraph induced by $v_{d-1}, v_d, v'_d, v_{d+1}$, referred to by the tuple $(v_{d-1}, v_d, v'_d, v_{d+1})$.

Lemma 3. *The length of a reconfiguration sequence is at least $2m(n+2)$. Moreover, each move in a sequence that has this length must either increase the c-state and leave the level of the switched vertex unchanged, or increase the level by one but leave the c-state unchanged.*

Author's personal copy

Proof. A single move cannot increase the level or the c-state by more than one (Fact 1.1). Moreover, we claim that it cannot increase both of these at the same time. Let m be an arbitrary move with move graph (a, b, c, d). If m increases the level by one, then the properties of the construction (Fact 1) applied to the move graph imply that the c-state does not increase. Specifically, for the case that the depth of b is odd, Fact 1.1 applied to the edges (b, d) and (c, d) implies that they are intra- and inter-level, respectively. If the c-state of b is 1 then it trivially cannot increase, but if it is zero then Fact 1.2 implies that $cs(d) = 0$, and Fact 1.3 implies that $cs(c) = 0$. A similar argument can be applied to the edges (a, b) and (a, c) for the case the depth of b is even. Thus, the level and c-state cannot both increase.

The sum of the levels and c-states in the starting path p_b is 0 and in the final path p_e is $2m(n + 2)$. Since we showed that the sum cannot increase by more than one in a single move, a reconfiguration sequence of length at $2m(n+2)$ must increase this sum by exactly one each move, and a shorter reconfiguration sequence is not possible. □

Lemma 4. *No path can contain two vertices with the same level but different v-state.*

Proof. In any path, the level of the vertices is non-increasing (by Fact 1.1). Therefore, all the vertices that have the same level must appear consecutively in the path. Since the edges connecting them are intra-level, Fact 1.4 implies that their v-states are identical. □

We say that a reconfiguration sequence ρ *visits* a vertex if there exists $p \in \rho$ that contains that vertex.

Lemma 5. *Suppose there exists a reconfiguration sequence ρ of length $2m(n + 2)$. Then ρ visits at least one vertex at every level, and all the vertices that it visits at a given level have the same v-state.*

Proof. First, since the level of a switched vertex can never increase by more than one, p_b has the vertices of the smallest level and p_e of the biggest level, ρ must visit at least one vertex at every level.

Next, consider all the paths in ρ that contain a level ℓ vertex, for some ℓ. We claim that these paths form a contiguous subsequence of ρ. After ρ visits its first level ℓ vertex, it can never have a path with just lower level vertices (by Lemma 3), so the next time it reaches a path with no level ℓ vertex, all the path's vertices will have a higher level. After that point, ρ can never visit a level ℓ vertex again (by Lemma 3).

Now, for the sake of contradiction, suppose that ρ visits two vertices of the same level but different v-states. Consider the first time this happens, going from a path p to p' via move (a, b, c, d). We know that b and c are the only level ℓ vertices in p and p', respectively (by Lemma 4), and that they have different c-states (by Lemma 3). Since the levels of a and d are not ℓ, the c-states of b and d must be the same (Fact 1.3), a contradiction. □

Suppose there exists a reconfiguration sequence ρ of length $2m(n + 2)$. Lemma 5 allows us to build a truth assignment $\theta \in \{0, 1\}^n$ by assigning θ_ℓ the v-state of the vertices of level ℓ in ρ.

Lemma 6. *The assignment θ is satisfying for ϕ.*

Proof. Consider an arbitrary clause C_d, and the vertices at depth $2d - 1$. Each $p \in \rho$ contains exactly one vertex at this depth. In p_b, the c-state of this vertex is 0, while in p_e it is 1, so there exists some first move (a, b, c, d) at depth $2d - 1$ that increases the c-state. By Lemma 3, this move cannot also change the level. Therefore, either (b, d) or (c, d) is a formula edge, otherwise Fact 1.3 would give us the contradiction $0 = cs(b) = cs(d) = cs(c) = 1$. Since the c-state of b is 0, (b, d) cannot be a formula edge, meaning (c, d) is a formula edge. Then we know from the construction that $x_{l(c)} = vs(c)$ satisfies C_d, meaning C_d is satisfied by $\theta_{\ell(c)} = vs(c)$. □

We now prove that the reduction is complete.

Lemma 7. *If ϕ is satisfiable, then there exists a reconfiguration sequence of length at most $2m(n + 2)$.*

Proof. Let $\theta \in \{0, 1\}^n$ be a satisfying truth assignment. Let $sat(\ell, d) = 1$ if C_d is satisfied by $\theta_1, \ldots, \theta_\ell$, and 0 otherwise. We define a path $p(\ell, \ell')$ which goes through the vertices of the gadget $G(\ell, \theta_\ell)$ with c-states at depth $2d$ and $2d - 1$ derived from $sat(\ell', d)$. Specifically, let $p(\ell, \ell') = s, v(\ell, \theta_\ell, sat(\ell', 1), 1), v(\ell, \theta_\ell, sat(\ell', 1), 2), \ldots, v(\ell, \theta_\ell, sat(\ell', d), 2d - 1), v(\ell, \theta_\ell, sat(\ell', d), 2d), \ldots, t$. We will build a reconfiguration sequence ρ that starts from p_b, and then goes to $p(\ell, \ell - 1)$ and $p(\ell, \ell)$ for every ℓ, finally finishing with p_e.

Let us fill in the intermediate moves of ρ. The vertices of $p(\ell, \ell)$ can be switched in order of increasing depth using inter-level edges to get $p(\ell + 1, \ell)$ in $2m$ steps. The paths $p(\ell, \ell - 1)$ and $p(\ell, \ell)$ are different only when C_d is satisfied by θ_ℓ, in which case there is a formula edge from $v(\ell, \theta_\ell, 1, 2d - 1)$ to $v(\ell, \theta_\ell, 0, 2d)$. Using these edges, the vertices of $p(\ell, \ell - 1)$ can be switched in order of increasing depth to get $p(\ell, \ell)$.

The number of moves required to switch between $p(\ell, \ell)$ and $p(\ell + 1, \ell)$ is $2m$. The total number of moves to switch between $p(\ell, \ell - 1)$ and $p(\ell, \ell)$ is $2k$, where k is the number of clauses satisfied by θ_ℓ but not satisfied by $\theta_1, \ldots, \theta_{\ell-1}$. When summed over ρ, these add up to at most $2m$, since each clause can become satisfied for the first time only once. Finally, we can switch between p_b and $p(1, 0)$ and between $p(n, n)$ and p_e using $2m$ steps each. The length of ρ is therefore $2m(n+2)$. □

Combining Lemmas 6 and 7 together with the fact that the reduction can be clearly done in polynomial time, we have the following theorem.

Theorem 2. *The* MIN-SPR *problem is NP-hard, even if k is polynomial in $|V(G)|$.*

Author's personal copy

4. Concluding remarks

In this paper, we studied the reconfiguration variant of the shortest path problem. We believe that the major open problem is to determine the complexity of deciding whether two shortest paths are reconfigurable. If the problem is NP-hard, then it will be the first example of a "natural" problem in P whose reconfigurability version is NP-hard. If the problem is polynomially solvable, then it will be the first example of an efficiently solvable reconfigurability problem with reconfiguration graphs of large diameter.

Our results are somewhat orthogonal to previous research on reconfiguration since we consider the length of a shortest path between two instances in the reconfiguration graph. If we assume that the bound k on the reconfiguration sequence length is given in unary, Min-SPR is in NP and Theorem 2 says it is NP-complete. It would be interesting to analyze whether similar results hold for other problems that have been studied in the context of reconfiguration.

Acknowledgements

We are grateful to Paul Bonsma, Takehiro Ito and Daniel Pellicer for interesting and fruitful discussions. We also thank the referees for comments that helped to improve the paper.

The third author was supported in part by "Agencija za raziskovalno dejavnost Republike Slovenije", research program P1-0285.

Addendum

While this paper was under review, Paul Bonsma proved that shortest path reconfigurability is PSPACE-complete [1].

References

[1] Paul S. Bonsma, Shortest path reconfiguration is PSPACE-hard, 2010. arXiv:1009.3217v1 [cs.CC].
[2] Paul S. Bonsma, Luis Cereceda, Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances, Theoret. Comput. Sci. 410 (50) (2009) 5215–5226.
[3] Paul S. Bonsma, Luis Cereceda, Jan van den Heuvel, Matthew Johnson, Finding paths between graph colourings: computational complexity and possible distances, Electron. Notes Discrete Math. 29 (2007) 463–469.
[4] Luis Cereceda, Jan van den Heuvel, Matthew Johnson, Connectedness of the graph of vertex-colourings, Discrete Math. 308 (5–6) (2008) 913–919.
[5] Luis Cereceda, Jan van den Heuvel, Matthew Johnson, Mixing 3-colourings in bipartite graphs, European J. Combin. 30 (2009) 1593–1606.
[6] Parikshit Gopalan, Phokion G. Kolaitis, Elitza N. Maneva, Christos H. Papadimitriou, The connectivity of Boolean satisfiability: computational and structural dichotomies, SIAM J. Comput. 38 (6) (2009) 2330–2355.
[7] Robert A. Hearn, Erik D. Demaine, PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation, Theoret. Comput. Sci. 343 (1–2) (2005) 72–96.
[8] Takehiro Ito, Erik D. Demaine, Nicholas J.A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, Yushi Uno, On the complexity of reconfiguration problems, in: ISAAC, in: Lecture Notes in Computer Science, vol. 5369, Springer, 2008, pp. 28–39.
[9] Takehiro Ito, Marcin Kamiński, Erik D. Demaine, Reconfiguration of list edge-colorings in a graph, in: WADS, in: Lecture Notes in Computer Science, vol. 5664, Springer, 2009, pp. 375–386.
[10] Marcin Kamiński, Paul Medvedev, Martin Milanič, Shortest paths between shortest paths and independent sets, in: IWOCA, in: Lecture Notes in Computer Science, vol. 6460, Springer, 2011, pp. 56–67.