On the Complexity of Reconfiguration Problems

Takehiro Ito[1], Erik D. Demaine[2], Nicholas J. A. Harvey[2],
Christos H. Papadimitriou[3], Martha Sideri[4], Ryuhei Uehara[5], and Yushi Uno[6]

[1] Graduate School of Information Sciences, Tohoku University,
Aoba-yama 6-6-05, Sendai, 980-8579, Japan.
[2] MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar St., Cambridge, MA 02139, USA.
[3] Computer Science Division, University of California at Berkeley,
Soda Hall 689, EECS Department, Berkeley, CA 94720, USA.
[4] Department of Computer Science,
Athens University of Economics and Business,
Patision 76, Athens 10434, Greece.
[5] School of Information Science, JAIST,
Asahidai 1-1, Nomi, Ishikawa 923-1292, Japan.
[6] Graduate School of Science, Osaka Prefecture University,
1-1 Gakuen-cho, Naka-ku, Sakai 599-8531, Japan.
takehiro@ecei.tohoku.ac.jp, edemaine@mit.edu, nickh@mit.edu,
christos@cs.berkeley.edu, sideri@aueb.gr, uehara@jaist.ac.jp,
uno@mi.s.osakafu-u.ac.jp

Abstract. Reconfiguration problems arise when we wish to find a step-by-step transformation between two feasible solutions of a problem such that all intermediate results are also feasible. We demonstrate that a host of reconfiguration problems derived from NP-complete problems are PSPACE-complete, while some are also NP-hard to approximate. In contrast, several reconfiguration versions of problems in P are solvable in polynomial time.

1 Introduction

Consider the bipartite graph with weighted vertices in Fig. 1(a) (both solid and dotted edges). It models a situation in which power stations with fixed capacity (the square vertices) provide power to customers with fixed demand (the round vertices). It can be seen as a feasible solution of a particular instance of a search problem which we may call the POWER SUPPLY problem [10, 11]: Given a bipartite graph $G = (U, V, E)$ with weights on the vertices, is there a forest covering all vertices in G, and with exactly one vertex from U in each component, such that the sum of the demands of the V vertices (customers) in each component is no more than the capacity of the U vertex (power station) in it?

But suppose now that we are given *two* feasible solutions of this instance (the leftmost and rightmost ones in Fig. 1), and we are asked: Can the solution on the left be transformed into the solution on the right *by moving only one customer at a time, and always remaining feasible?* This problem, which we call the POWER

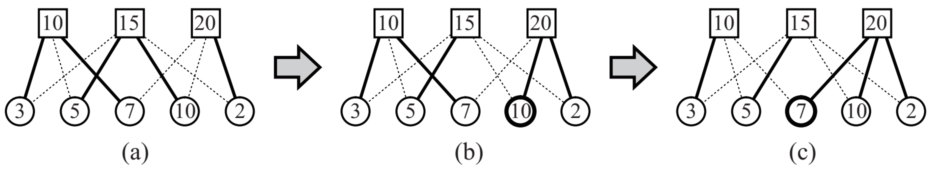

Fig. 1. A sequence of feasible solutions for the POWER SUPPLY problem.

SUPPLY RECONFIGURATION problem, is an exemplar of the kind of problems we discuss in this paper. (In this particular instance, it turns out that the answer is "yes"; see Fig. 1.) As one may have expected, the most basic reconfiguration problem is the SATISFIABILITY RECONFIGURATION problem: Given a CNF formula and two satisfying truth assignments $\mathbf{s}_0$ and $\mathbf{s}_t$, are these connected in the subgraph of the hypercube induced by the satisfying truth assignments? This problem has been shown PSPACE-complete in [6].

In more generality, *reconfiguration problems* have the following structure: Fix a search problem $\mathcal{S}$ (a polynomial-time algorithm which, on instance I and candidate solution y of length polynomial in that of I, determines whether y is a feasible solution of I); and fix a polynomially-testable symmetric *adjacency relation* A on the set of feasible solutions, that is, a polynomial-time algorithm such that, given an instance I of $\mathcal{S}$ and two feasible solutions y' and y'' of I, it determines whether y' and y'' are adjacent. (In almost all problems discussed in this paper, the feasible solutions can be considered as sets of elements, and two solutions are adjacent if their symmetric difference has size 1 — or, in some cases such as POWER SUPPLY RECONFIGURATION, 2.) The RECONFIGURATION PROBLEM FOR $\mathcal{S}$ AND A is the following computational problem: Given instance I of $\mathcal{S}$ and two feasible solutions y_0 and y_t of I, is there a sequence of feasible solutions $y_0, y_1, \ldots, y_t$ of I such that y_{i-1} and y_i are adjacent for $i = 1, 2, \ldots, t$?

Reconfiguration problems can also arise from optimization problems, if one turns the optimization problem into a search problem by giving a threshold. For example, the CLIQUE RECONFIGURATION problem is the following: Given a graph G, an integer k, and two cliques C_0 and C_t of G, both of size at least k, is there a way to transform C_0 into C_t via cliques, each of which results from the previous one by adding or subtracting one node of G, without ever going through a clique of size less than $k-1$?

Reconfiguration problems are useful and entertaining, have been coming up in recent literatures [1, 6, 9], and are interesting for a variety of reasons. First, they may reflect, as in the POWER SUPPLY RECONFIGURATION problem above, a situation where we actually seek to implement such a sequence of elementary changes in order to transform the current configuration to a more desirable one, in a context in which intermediate steps must also be fully feasible, and only restricted changes can occur — in our example, no two customers can change providers simultaneously, and we certainly do not wish customers to be without power. In a complex, dynamic environment in which changing circumstances affect the feasible solution of choice, determining whether such adaptation is possible may be crucial. Reconfiguration problems also model questions of *evolvability*: Can genotype y_0 evolve into genotype y_t via individual mutations which are each of adequate fitness? Here a genotype is considered feasible if its fitness is above a threshold, and two genotypes are considered adjacent if one is a simple mutation of the other. Finally, reconfiguration versions of constraint satisfaction problems (the first kind studied in the literature [6]) yield insights into the structure of the solution space, and heuristics, such as survey propagation, whose performance depends crucially on connectivity and other properties of the solution space.

In this paper we embark on a systematic investigation of the complexity of reconfiguration problems. Our main focus is showing that a host of reconfiguration problems (including all those mentioned above and many more) are PSPACE-complete. The proof for the POWER SUPPLY RECONFIGURATION problem and those for certain other problems are explained in Section 2. In Section 3 we point out that certain reconfiguration problems arising from problems in P (such as the MINIMUM SPANNING TREE and MATCHING problems) can be solved in polynomial time, and in Section 4 we show certain approximability and inapproximability results for reconfiguration problems.

2 PSPACE-completeness

In this section we show that a host of reconfiguration problems are PSPACE-complete. We first give a proof for the POWER SUPPLY RECONFIGURATION problem in Subsection 2.1, and then give proof sketches for certain other reconfiguration problems in Subsection 2.2.

2.1 Power supply reconfiguration

The POWER SUPPLY RECONFIGURATION problem was defined informally in the Introduction. An instance is given in terms of a bipartite graph $G = (U, V, E)$, where each vertex in U is called a *supply vertex* and each vertex in V is called a *demand vertex*. Each supply vertex $u \in U$ is assigned a positive integer $\sup(u)$, called the *supply of* u, while each demand vertex $v \in V$ is assigned a positive integer $\mathrm{dem}(v)$, called the *demand of* v. We wish to find a forest which covers all vertices in G such that each tree T in the forest has exactly one supply vertex whose supply is at least the sum of demands of all demand vertices in T. We call an assignment $f : V \to U$ a *configuration of* G if there is an edge $\big(v, f(v)\big) \in E$ for each demand vertex $v \in V$. A configuration f of G is called *feasible* if the following condition holds: for each supply vertex $u \in U$,

$$\sup(u) \geq \sum \big\{ \mathrm{dem}(v) \mid v \in V \text{ such that } f(v) = u \big\}.$$

The *adjacency relation* on the set of feasible configurations is defined as follows: two feasible configurations f and f' are *adjacent* if $\big|\{v \in V : f(v) \neq f'(v)\}\big| = 1$, that is, f' can be obtained from f by changing the assignment of a single demand vertex. Then, for given a bipartite graph $G = (U, V, E)$ and two feasible configurations f_0 and f_t of G, the POWER SUPPLY RECONFIGURATION problem is to determine whether there is a sequence of feasible configurations $f_0, f_1, \ldots, f_t$ of G such that f_{i-1} and f_i are adjacent for $i = 1, 2, \ldots, t$.

Fig. 1 illustrates three feasible configurations of a bipartite graph G, where each supply vertex is drawn as a square, each demand vertex as a round, and the supply or demand is written inside. Fig. 1 also illustrates an example of a transformation from the feasible configuration in Fig. 1(a) to one in Fig. 1(c), where the demand vertex whose assignment was changed from the previous one is depicted by a thick round. The optimization problem for finding a certain configuration of a given graph has been studied in [10, 11].

Theorem 1. Power supply reconfiguration *is PSPACE-complete.*

Proof. It is easy to see that this problem, as well as any reconfiguration version of a problem in NP, can be solved in (most conveniently, nondeterministic [13]) polynomial space.

We give a reduction to this problem from the satisfiability reconfiguration problem, which was recently shown to be PSPACE-complete [6]. In that problem we are given a Boolean formula ϕ in conjunctive normal form, say with n variables $x_1, x_2, \ldots, x_n$ and m clauses $C_1, C_2, \ldots, C_m$, and two satisfying truth assignments $\mathbf{s}_0$ and $\mathbf{s}_t$ of ϕ. Then, we are asked whether there is a sequence of satisfying truth assignments, starting with $\mathbf{s}_0$ and ending in $\mathbf{s}_t$, and each differing from the previous one in only one variable. Let c be the maximum number of clauses in which a literal occurs, and hence no literal appears in more than c clauses in ϕ.

Given such an instance of satisfiability reconfiguration, we construct an instance of power supply reconfiguration as follows. We first make a *variable gadget* G_{x_i} for each variable x_i, $1 \le i \le n$; G_{x_i} is a binary tree with three vertices as illustrated in Fig. 2(a); the root F_i is a demand vertex of demand c, and the two leaves x_i and $\bar{x}_i$ are supply vertices of supply c. Then the corresponding bipartite graph G_ϕ is constructed as follows. For each variable x_i, $1 \le i \le n$, put the variable gadget G_{x_i} to the graph, and for each clause C_j, $1 \le j \le m$, put a demand vertex C_j of demand 1 to the graph. Finally, for each

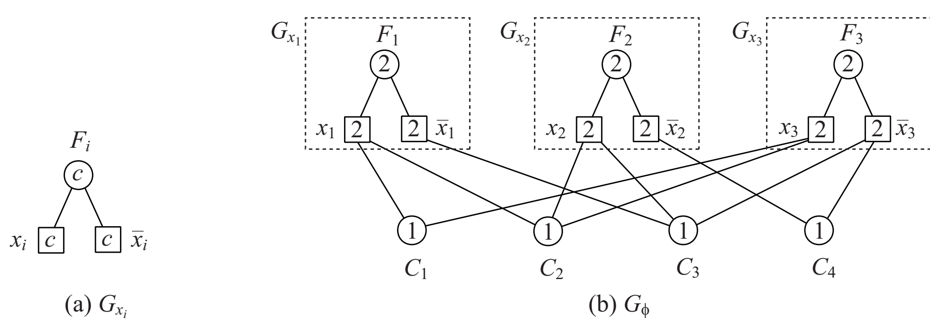

Fig. 2. (a) Variable gadget G_{x_i}, and (b) bipartite graph G_ϕ corresponding to a Boolean formula ϕ with four clauses $C_1 = (x_1 \vee x_3)$, $C_2 = (x_1 \vee x_2 \vee x_3)$, $C_3 = (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$ and $C_4 = (\bar{x}_2 \vee \bar{x}_3)$, and hence $c = 2$.

clause C_j, $1 \le j \le m$, join a supply vertex x_i (or $\bar{x}_i$) in G_{x_i}, $1 \le i \le n$, with the clause demand vertex C_j if and only if the literal x_i (respectively, $\bar{x}_i$) is in the clause C_j. (See Fig. 2(b) as an example.) Clearly, G_ϕ is a bipartite graph.

Consider a feasible configuration of G_ϕ. Then each demand vertex F_i, $1 \le i \le n$, must be assigned to one of x_i and $\bar{x}_i$; *a literal is considered false if F_i is assigned to the corresponding supply vertex.* Notice that, since supply vertices have supply c and the F_i's have demand c, a false-literal supply vertex cannot provide power to any of the other demand vertices. Hence, all clause demand vertices C_j, $1 \le j \le m$, must be assigned to true-literal supply vertices that occur in them. Since each literal x_i (or $\bar{x}_i$), $1 \le i \le n$, appears in at most c clauses in ϕ, the corresponding supply vertex x_i (respectively, $\bar{x}_i$) in G_{x_i} can provide power to all clause demand vertices C_j whose corresponding clauses have x_i (respectively, $\bar{x}_i$).

To complete the reduction, we now create two feasible configurations f_0 and f_t of G_ϕ corresponding to the satisfying truth assignments $\mathbf{s}_0$ and $\mathbf{s}_t$ of ϕ, respectively. Each demand vertex F_i, $1 \le i \le n$, is assigned to the supply vertex whose corresponding literal is false, while each clause demand vertex C_j, $1 \le j \le m$, is assigned to an arbitrary true-literal supply vertex adjacent to C_j. Clearly, f_0 and f_t are feasible configurations of G_ϕ. This completes the construction of the corresponding instance of the POWER SUPPLY RECONFIGURATION problem.

We know that a feasible configuration of G_ϕ corresponds to a satisfying truth assignment of ϕ *plus* an assignment of each clause to a true literal. It is easy to see that this correspondence goes backwards: every satisfying truth assignment of ϕ can be mapped to at least one (in general, to exponentially many) feasible configurations of G_ϕ.

How about adjacent configurations — defined to be configurations differing in the assignment of just one demand vertex? One can easily observe that there are only two types of reassignments to go from a feasible configuration of G_ϕ to an adjacent one, as follows:

(1) One could change the assignment of a demand vertex F_i from x_i to $\bar{x}_i$, or vice versa, *if any clause demand vertex is currently assigned to neither supply vertices x_i nor $\bar{x}_i$.*
(2) Alternatively, if a clause demand vertex C_j is adjacent to more than one true-literal supply vertices, then one could change the assignment of C_j from the current one to another.

Therefore, any sequence of adjacent feasible configurations of G_ϕ can be broken down to subsequences, intermittently with a reassignment of type (1) above; in each subsequence, every two adjacent configurations can go from one to another via a reassignment of type (2) above. Therefore, all feasible configurations in each subsequence correspond to the same satisfying truth assignment of ϕ, while any two consecutive subsequences correspond to adjacent satisfying truth assignments (namely, differing in only one variable). Conversely, for given any sequence of adjacent satisfying truth assignments of ϕ, there is a corresponding sequence of adjacent feasible configurations of G_ϕ, obtained as follows: Consider a flip of a variable x_i from true to false. (A flip of x_i from false to true is similar.) Then we wish to change the assignment of the demand vertex F_i from the supply vertex $\bar{x}_i$ to x_i. (Remember that the literal to which F_i is assigned is considered false.) We first change the assignments of all clause demand vertices, which are currently assigned to x_i, to another true-literal supply vertex: since we are about to flip the variable x_i and we know that the truth assignment of ϕ after the flip will be also satisfying, there must be a second true-literal supply vertex for every clause demand vertex currently assigned to x_i. After all such reassignments, we finally change the assignment of F_i from $\bar{x}_i$ to x_i.

It is easy now to see that there is a sequence of adjacent satisfying truth assignments of ϕ from $\mathbf{s}_0$ to $\mathbf{s}_t$ if and only if there is a sequence of adjacent feasible configurations of G_ϕ from f_0 to f_t. This completes a proof of Theorem 1. □

2.2 Other Intractable Reconfiguration Problems

There is a wealth of reconfiguration versions of NP-complete problems which can be shown PSPACE-complete via extensions, often quite sophisticated, of the original NP-completeness proofs; in this subsection we only sample the realm of possibilities.

We have already defined the CLIQUE RECONFIGURATION problem in the Introduction as an example of a general scheme whereby any optimization problem can be transformed into a reconfiguration problem by giving a threshold (upper bound for minimization problems, lower bound for maximization problems) for the allowed values of the objective function of the intermediate feasible solutions; the INDEPENDENT SET RECONFIGURATION and VERTEX COVER RECONFIGURATION problems are defined similarly. In the INTEGER PROGRAMMING RECONFIGURATION problem, we are given a 0-1 linear program seeking to maximize cx subject to $Ax \le b$, and we consider two solutions adjacent if they only differ in one variable.

Theorem 2. *The following problems are PSPACE-complete*: INDEPENDENT SET RECONFIGURATION*,* CLIQUE RECONFIGURATION*,* VERTEX COVER RECONFIGURATION*,* SET COVER RECONFIGURATION*,* INTEGER PROGRAMMING RECONFIGURATION*.*

Proof sketch. We sketch a proof for the INDEPENDENT SET RECONFIGURATION problem. The reduction can be obtained by extending the well-known reduction from the 3SAT problem to the INDEPENDENT SET problem [12]. We construct a graph $\rho(\phi)$ from a given 3SAT formula ϕ with n variables and m clauses, as follows. For each variable x in ϕ, we put an edge to the graph; the two endpoints are labeled x and $\bar{x}$. Then, for each clause C in ϕ, we put a clique of size $|C|$ to the graph; each node in the clique corresponds to a literal in the clause C. Finally, we add an edge between two nodes in different components if and only if the nodes correspond to opposite literals. Then, any maximum independent set in $\rho(\phi)$ contains at least n nodes; the n nodes are chosen from the endpoints of edges corresponding to the variables; *a literal is considered true if* *the corresponding endpoint is chosen.* Clearly, $\rho(\phi)$ has a maximum independent set of size $k = n + m$ if and only if ϕ is satisfiable. Consider all independent sets of size k in $\rho(\phi)$; they can be partitioned into subclasses of the form $\rho(\mathbf{s})$ corresponding to the satisfying truth assignments $\mathbf{s}$ of ϕ (the various independent sets in the subclass $\rho(\mathbf{s})$ correspond to the different possible ways to satisfy each clause by $\mathbf{s}$). It is easy to see that all independent sets in $\rho(\mathbf{s})$ are connected via intermediate independent sets of size at least $k-1$. Therefore, by similar arguments in the proof of Theorem 1, one can easily observe that telling whether two independent sets of size k in $\rho(\phi)$ can be transformed into one another via intermediate independent sets of size at least $k-1$ is PSPACE-complete.

Similarly as the NP-completeness proofs [5, §3.1.3], the result for INDEPENDENT SET RECONFIGURATION yields those for CLIQUE RECONFIGURATION and VERTEX COVER RECONFIGURATION. Then, the result for SET COVER RECONFIGURATION is immediate since it is a generalization of VERTEX COVER RECONFIGURATION. INTEGER PROGRAMMING RECONFIGURATION generalizes CLIQUE RECONFIGURATION via the well-known integer program for CLIQUE. □

3 Reconfiguration Problems in P

Reconfiguration problems arise in relation to polynomially solvable problems as well. For example, in the MINIMUM SPANNING TREE RECONFIGURATION problem, we are given an edge-weighted graph G, a threshold k, and two spanning trees of G, both of weight at most k, and wish to transform one tree into another via edge exchanges, without ever getting into a tree with weight $> k$. The MATCHING RECONFIGURATION problem is defined similarly (the formal definition will be given later). We show in this section that both problems can be solved in polynomial time.

The result for the MINIMUM SPANNING TREE RECONFIGURATION problem can be obtained from the following more general proposition.

Proposition 1. *Given a weighted matroid* $\mathbf{M}$ *and two bases* B_0 *and* B_t *of* $\mathbf{M}$*, both of weight at most* k*, there always exists a sequence of* $|B_0 \setminus B_t|$ *exchanges that transforms one into the other without ever exceeding weight* k*.*

Proof sketch. For an unweighted matroid, this result follows trivially from the properties of a base family [15, §39.5]. For a weighted matroid $\mathbf{M}$, we outline a proof for the case when B_0 and B_t are both of maximum weight. Then, the result follows from the fact that the set of *maximum weight bases* of $\mathbf{M}$ also form the base family of another matroid [2, p. 287] [3, p. 130]. By generalizing this proof appropriately, one can obtain the full result. (Due to the page limitation, we omit the details.) □

In the MATCHING RECONFIGURATION problem, we are given an unweighted graph G, a threshold k, and two matchings M_0 and M_t of G, both of size at least k, and we are asked whether there is a sequence of matchings of G, starting with M_0 and ending in M_t, and each resulting from the previous one by either addition or deletion of an edge in G, without ever going through a matching of size less than $k-1$.

Proposition 2. Matching reconfiguration *can be solved in polynomial time.*

Proof sketch. Since the adjacency relation is symmetric, we may assume without loss of generality that $|M_0| \leq |M_t|$. Consider the subgraph H of G induced by all edges in $(M_0 \setminus M_t) \cup (M_t \setminus M_0)$. Then, H consists of single edges, and alternating paths and cycles with respect to M_0 and M_t. The *greedy algorithm* for transforming M_0 into M_t is the following. Divide the components of H into the following four categories: (1) single edges of $M_t \setminus M_0$; (2) alternating paths starting with an edge of $M_t \setminus M_0$; (3) alternating cycles; and (4) all the rest. In this category order, transform M_0 into M_t by repeatedly adding edges of $M_t \setminus M_0$ and deleting edges of $M_0 \setminus M_t$ along each component of H. Notice that, after exchanging the edges in Categories (1) and (2), the obtained matching M has size at least $|M_t|$ ($\geq |M_0|$). Therefore, one can easily observe that intermediate matchings have size at least $|M_0| - 1$ for exchanging edges in Category (2), and have size at least $|M_t| - 2$ for exchanging edges in Categories (3) and (4).

For the case $|M_t| \geq k+1$, the greedy algorithm always transforms M_0 into M_t without ever going through a matching of size less than $k-1$. For the case $|M_0| = |M_t| = k$, there does not always exist a desired sequence of matchings if H has components of Category (3). Nonetheless, existence can be determined in polynomial time, as follows. If M_0 and M_t are *not* maximum matchings of G, we first transform M_t into a matching M'_t of size $k+1$ along an arbitrary augmenting path with respect to M_t; then, the greedy algorithm works for transforming M_0 into M'_t. Therefore, a desired sequence always exists for this subcase. If M_0 and M_t are maximum matchings of G and H contains alternating cycles, we have the following lemma, whose proof is omitted due to the page limitation.

Lemma 1. *There is a sequence of adjacent matchings from M_0 to M_t such that all intermediate matchings have size at least $k-1$ if and only if every cycle in H contains a vertex that begins an even-length alternating path in G with respect to M_0 ending at an unmatched vertex by M_0.*

By Lemma 1 one can easily determine whether there exists a desired sequence for this subcase in polynomial time. □

We note in passing that the matching reconfiguration problem for edge-weighted graphs seems quite a bit more complicated; however, we conjecture that it also can be solved in polynomial time.

Besides minimum spanning tree reconfiguration and matching reconfiguration, it turns out that all polynomial-time solvable special cases of satisfiability, as characterized by Schaefer [14], give rise to polynomially solvable reconfiguration problems:

Theorem 3 ([6]). Satisfiability reconfiguration *for linear, Horn, dual Horn and 2-literal clauses are all in* P.

4 Approximation

We have seen that an optimization problem gives rise to a reconfiguration problem by bounding the objective of intermediate configurations. In turn, we can get a natural optimization problem if we try to *optimize the worst objective among all configurations* in the reconfiguration path. For example, in the problem that we call the MAXMIN CLIQUE RECONFIGURATION problem, we are given a graph and two cliques C_0 and C_t, and we are asked to transform C_0 into C_t by a sequence of additions and removals of nodes so that the minimum size of any clique in the sequence is as large as possible.

Theorem 4. MAXMIN CLIQUE RECONFIGURATION *cannot be approximated within any constant factor unless* P = NP.

Proof. We give a reduction in an approximation-preserving manner from the CLIQUE problem to this problem. For a given graph G with n nodes, we construct a new graph G' with $3n$ nodes as a corresponding instance of MAXMIN CLIQUE RECONFIGURATION: a set of n nodes is connected as G, while two new sets of n nodes are connected each as a clique (these two cliques of G' are called C_0 and C_t); finally, there are edges in G' between each new node and each node in G.

Consider any sequence of cliques of G', each resulting from the previous one by insertion or deletion of a node, starting from C_0 and ending in C_t. We claim that one of them will be a clique of G — this follows directly from the absence of any edges from C_0 to C_t. Conversely, for any clique C of G, there exists a sequence from C_0 to C_t via C (add the nodes of C to the clique C_0, then remove those of C_0, then add those of C_t). Therefore, the minimum clique size in the sequence is the size of C, and hence solving (or approximating) this instance of MAXMIN CLIQUE RECONFIGURATION is the same as solving (respectively, approximating) the CLIQUE problem for G. Since it is known that CLIQUE cannot be approximated within any constant factor unless P = NP [7], the result follows. $\square$

A similar argument establishes the following:

Theorem 5. MAXMIN MAXSAT RECONFIGURATION *cannot be approximated within a factor better than* $\frac{15}{16}$ *unless* P = NP.

Proof. We reduce in an approximation-preserving manner the MAXSAT problem to this problem. Suppose that we are given an instance ϕ of MAXSAT with n variables $x_1, x_2, \ldots, x_n$ and m clauses $C_1, C_2, \ldots, C_m$. We construct a new instance ϕ' in which each clause C_j, $1 \leq j \leq m$, is replaced by $(C_j \vee y \vee z)$ where y and z are new variables, and the additional clause $(\bar{y} \vee \bar{z})$ *with weight* m. Note that the truth assignments $\mathbf{s}_0 : z = 1, y = 0, x_1 = x_2 = \cdots = x_n = 1$ and $\mathbf{s}_t : z = 0, y = 1, x_1 = x_2 = \cdots = x_n = 0$ are both satisfying all $2m$ clauses.

Consider now an optimal path in the hypercube between $\mathbf{s}_0$ and $\mathbf{s}_t$. Since at $\mathbf{s}_0 : z = 1, y = 0$ and at $\mathbf{s}_t : z = 0, y = 1$, there must exist a truth assignment on this path such that $y = z$. Since the clause $(\bar{y} \vee \bar{z})$ has weight m and the path is assumed optimal, it must be that $y = z = 0$. Thus, the remaining variables must spell an optimum satisfying truth assignment of the original formula ϕ. Hence, from an optimum path for the corresponding instance of MAXMIN MAXSAT RECONFIGURATION, we can obtain an optimum truth assignment for the original instance of MAXSAT. Similarly, from an α-approximation for MAXMIN MAXSAT RECONFIGURATION, it is easy to see that we get a $(2\alpha - 1)$-approximation of the MAXSAT instance. Since it is known that MAXSAT cannot be approximated within a factor better than $\frac{7}{8}$ unless P = NP [8], the result follows. $\square$

By a similar maneuver, it can be shown that the MINMAX SET COVER RECONFIGURATION problem cannot be approximated within a factor better than $o(\log n)$ unless NP is contained in $\mathrm{DTIME}\big(n^{O(\log\log n)}\big)$ [4].

Returning to the POWER SUPPLY problem, there is a natural optimization version of the problem, in which the constraint that the total demand of all demand vertices in each tree T be within the supply of the supply vertex in T is replaced by a "soft" criterion: we allow that the total demand in T exceeds the supply in T, but wish to minimize the sum of the "deficient power" of all supply vertices in the graph.

We now define the MINMAX POWER SUPPLY RECONFIGURATION problem. For a configuration f of a bipartite graph $G = (U, V, E)$ and a supply vertex $u \in U$, the *deficient power* $d(f, u)$ *of* u *on* f is defined as follows:

$$d(f, u) = \sum \big\{\mathrm{dem}(v) \mid v \in V \text{ such that } f(v) = u\big\} - \mathrm{sup}(u).$$

If f is infeasible, then there is at least one supply vertex u such that $d(f, u) > 0$. On the other hand, if f is feasible, then $d(f, u) \leq 0$ for all supply vertices $u \in U$; in fact, a nonpositive deficient power $d(f, u)$ represents the *marginal power* of u on f. The *cost* $c(f)$ *of a configuration* f is defined to be $c(f) = \sum_{u \in U} |d(f, u)|$. Clearly, $c(f) = \sum_{u \in U} \mathrm{sup}(u) - \sum_{v \in V} \mathrm{dem}(v)$ for every feasible configuration f of G. In the problem that we call the MINMAX POWER SUPPLY RECONFIGURATION problem, we are given a bipartite graph $G = (U, V, E)$ and two feasible configurations f_0 and f_t of G, and we are asked to transform f_0 into f_t by a sequence of reassignments of single demand vertices so that the maximum cost of any configuration in the sequence is as small as possible. It is easy to see that a sequence $f_0, f_1, \ldots, f_t$ which consists of only feasible configurations is optimum, and the optimum value is $\sum_{u \in U} \mathrm{sup}(u) - \sum_{v \in V} \mathrm{dem}(v)$.

One can observe that the MINMAX POWER SUPPLY RECONFIGURATION problem is strongly NP-hard (by a reduction from the 3-PARTITION problem [5], for example). However, the problem can be solved in linear time for the following special case. Suppose in the remainder of this section that we are given a bipartite graph $G = (U, V, E)$ having exactly two supply vertices. For a configuration f of G, let $W(f) = \{v \in V \mid f(v) \neq f_t(v)\}$, that is, $W(f)$ is the set of demand vertices which are assigned to "wrong" supply vertices on f. Note that all (demand) vertices in $W(f)$ are adjacent to both the two supply vertices. For a given initial configuration f_0 of G, let v^* be a demand vertex in $W(f_0)$ having the maximum demand, that is, $\mathrm{dem}(v^*) = \max\{\mathrm{dem}(v) \mid v \in W(f_0)\}$. Then, we have the following lemma.

Lemma 2. *If $c(f_0) \geq 2 \cdot \mathrm{dem}(v^*)$, then the optimum sequence for* MINMAX POWER SUPPLY RECONFIGURATION *consists of only feasible configurations, and it can be found in linear time.*

Proof. Suppose without loss of generality that $W(f_0) \neq \emptyset$. If all demand vertices in $W(f_0)$ are assigned to the same supply vertex, then we just change the assignments of all demand vertices in $W(f_0)$ from the current supply vertex to the other. Since both f_0 and f_t are feasible, all intermediate configurations are also feasible. Therefore, we assume in the following that each of the two supply vertices has at least one demand vertex in $W(f_0)$.

Since f_0 is feasible, the cost $c(f_0)$ denotes the sum of marginal powers of the two supply vertices. Moreover, since the sum is at least $2 \cdot \mathrm{dem}(v^*)$, one of the two supply vertices has marginal power of at least $\mathrm{dem}(v^*)$. Therefore, we can change the assignment of at least one demand vertex $v \in W(f_0)$ from the "wrong" supply vertex to the "correct" one, since $\mathrm{dem}(v) \leq \mathrm{dem}(v^*)$. Clearly, the resulting configuration f_1 is also feasible, and satisfies $c(f_1) \geq 2 \cdot \mathrm{dem}(v^*)$. By repeatedly executing such a reassignment, we can obtain a desired sequence $f_0, f_1, \ldots, f_t$ which consists of only feasible configurations. Therefore, the sequence is an optimum solution. The length of the sequence is $|W(f_0)|$ $(\leq |V|)$ since each demand vertex in $W(f_0)$ moves exactly once and any of the other demand vertices does not move in the sequence. We can thus find an optimum solution in linear time. □

Theorem 6. *There is a linear-time 2-approximation algorithm for* MINMAX POWER SUPPLY RECONFIGURATION *having exactly two supply vertices.*

Proof. Let OPT be the optimum value for an instance of MINMAX POWER SUPPLY RECONFIGURATION. Since we have to change the assignment of the demand vertex v^* for obtaining the target configuration f_t, we have $\mathrm{OPT} \geq \mathrm{dem}(v^*)$.

By Lemma 2 it suffices to consider the case $c(f_0) < 2 \cdot \mathrm{dem}(v^*)$. In this case, consider a slightly modified instance in which the supplies of the two supply vertices are increased so that the total supply is equal to $2 \cdot \mathrm{dem}(v^*)$. In the modified instance, both the configurations f_0 and f_t remain feasible and $c(f_0) = 2 \cdot \mathrm{dem}(v^*)$. Therefore, by Lemma 2 we can find in linear time an optimum sequence which consists of only feasible configurations for the modified instance; the optimum value is thus $2 \cdot \mathrm{dem}(v^*)$. Note that some configurations in the sequence may be infeasible for the original instance. We take the sequence as our approximation solution for the original instance, and hence our approximate value A is $A = 2 \cdot \mathrm{dem}(v^*) \leq 2 \cdot \mathrm{OPT}$. □

5 Open Problems

There are many open problems raised by this work, and we mention some of these below:

– Do all problems in P give rise, in a natural way, to polynomially solvable reconfiguration problems? We conjecture that the answer is negative, but we have yet to identify a counterexample (even a conjectured one).
– Is the TRAVELING SALESMAN RECONFIGURATION problem (where two tours are adjacent if they differ in two edges) PSPACE-complete?
– Are there better approximation algorithms for the MINMAX POWER SUPPLY RECONFIGURATION problem? Lower bounds?
– Are the problems in Section 4 PSPACE-complete to approximate (not just NP-hard)?

References

1. P. Bonsma and L. Cereceda, Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances, Proc. of MFCS2007, LNCS 4708 (2007) 738–749.
2. W. J. Cook, W. H. Cunningham, W. R. Pulleyblank and A. Schrijver, Combinatorial Optimization, Wiley, 1997.
3. J. Edmonds, Matroids and the greedy algorithm, Math. Programming 1 (1971) 127–136.
4. U. Feige, A threshold of $\ln n$ for approximating set cover, J. ACM 45 (1998) 634–652.
5. M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.
6. P. Gopalan, P. G. Kolaitis, E. N. Maneva and C. H. Papadimitriou, The connectivity of Boolean satisfiability: computational and structural dichotomies, Proc. of ICALP 2006, LNCS 4051 (2006) 346–357.
7. J. Håstad, Clique is hard to approximate within $n^{1-\varepsilon}$, Acta Mathematica 182 (1999) 105–142.
8. J. Håstad, Some optimal inapproximability results, J. ACM 48 (2001) 798–859.
9. R. A. Hearn and E. D. Demaine, PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation, Theoretical Computer Science 343 (2005) 72–96.
10. T. Ito, X. Zhou and T. Nishizeki, Partitioning trees of supply and demand, International J. Foundations of Computer Science 16 (2005) 803–827.
11. T. Ito, E. D. Demaine, X. Zhou and T. Nishizeki, Approximability of partitioning graphs with supply and demand, Proc. of ISAAC 2006, LNCS 4288 (2006) 121–130.
12. C. H. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.
13. W. J. Savitch, Relationships between nondeterministic and deterministic tape complexities, J. of Computer and System Sciences 4 (1970) 177–192.
14. T. J. Schaefer, The complexity of satisfiability problems, Proc. of 10th ACM Symposium on Theory of Computing, pp. 216–226, 1978.
15. A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Springer-Verlag, 2003.