Journal of Combinatorial Optimization manuscript No.
(will be inserted by the editor)

Approximability of the Subset Sum Reconfiguration Problem

Takehiro Ito · Erik D. Demaine

Received: date / Accepted: date

Abstract The SUBSET SUM problem is a well-known NP-complete problem in which we wish to find a packing (subset) of items (integers) into a knapsack with capacity so that the sum of the integers in the packing is at most the capacity of the knapsack and at least a given integer threshold. In this paper, we study the problem of reconfiguring one packing into another packing by moving only one item at a time, while at all times maintaining the feasibility of packings. First we show that this decision problem is strongly NP-hard, and is PSPACE-complete if we are given a conflict graph for the set of items in which each vertex corresponds to an item and each edge represents a pair of items that are not allowed to be packed together into the knapsack. We then study an optimization version of the problem: we wish to maximize the minimum sum among all packings in a reconfiguration. We show that this maximization problem admits a polynomial-time approximation scheme (PTAS), while the problem is APX-hard if we are given a conflict graph.

Keywords approximation algorithm · PTAS · reachability on solution space · subset sum

1 Introduction

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. Ito *et al.* [12] proposed a framework of reconfiguration problems, and gave complexity and approximability results for reconfiguration problems derived from several well-known problems, such as INDEPENDENT SET, CLIQUE, MATCHING, etc. In this paper, we study two reconfiguration problems derived from the SUBSET SUM problem.

T. Ito
Graduate School of Information Sciences, Tohoku University,
Aoba-yama 6-6-05, Sendai, 980-8579, Japan.
E-mail: takehiro@ecei.tohoku.ac.jp

E. D. Demaine
MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar St., Cambridge, MA 02139, USA.
E-mail: edemaine@mit.edu

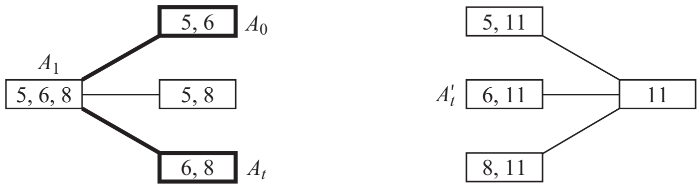

Fig. 1 All packings of total size at least $k = 10$ for $A = \{5, 6, 8, 11\}$ and $c = 20$.

The SUBSET SUM problem is a well-known NP-complete problem, defined as follows [17]. Suppose that we are given a knapsack with a nonnegative integer capacity c, and a set A of n items $a_1, a_2, \ldots, a_n$, each of which has a nonnegative integer size $s(a_i)$, $1 \leq i \leq n$. We call a subset A' of A a *packing* if the total size of A' does not exceed the capacity c, that is, $\sum_{a \in A'} s(a) \leq c$. Given an integer threshold k, the SUBSET SUM problem is to find a packing A' whose total size is at least k, that is, $k \leq \sum_{a \in A'} s(a) \leq c$. For a knapsack with capacity $c = 20$ and a set $A = \{5, 6, 8, 11\}$, there are 8 packings of total size at least $k = 10$, as illustrated in Fig. 1, where each packing is surrounded by a box. Our definition of SUBSET SUM is known as the decision version of the MAXIMUM SUBSET SUM problem in which we wish to find a packing whose total size is maximum [17].[1]

Suppose now that we are given *two* packings A_0 and A_t, both of total size at least k, and we are asked whether we can transform one into the other via packings by moving (namely, either adding or removing) a single item to/from the previous one without ever going through a packing of total size less than k. We call this decision problem the SUBSET SUM RECONFIGURATION problem. For two packings $A_0 = \{5, 6\}$ and $A_t = \{6, 8\}$ in Fig. 1, the answer is "yes" since they can be transformed into each other via $A_1 = \{5, 6, 8\}$; in Fig. 1, two packings (boxes) are joined by a line if and only if one packing can be obtained from the other by moving a single item.

Obviously, we cannot always find such a transformation. For example, there is no transformation between $A_0 = \{5, 6\}$ and $A'_t = \{6, 11\}$ in Fig. 1 if we are allowed to use only packings of total size at least $k = 10$. On the other hand, the answer is always "yes" if $k = 0$: we first remove all items of A_0, and obtain the empty packing; and then, add all items of A_t to the knapsack. In turn, we can get a natural optimization problem

[1] Note that SUBSET SUM in [6] is slightly different from our definition: SUBSET SUM in [6] is defined as the problem of finding a packing whose total size is exactly k.

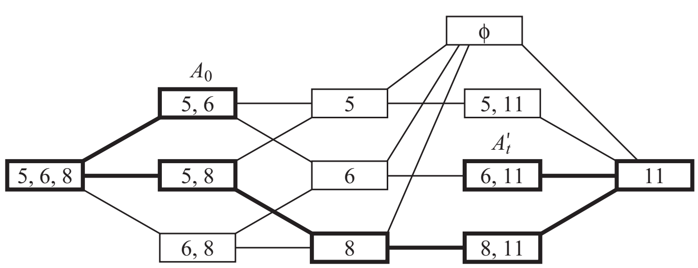

Fig. 2 All packings for $A = \{5, 6, 8, 11\}$ and $c = 20$.

if we wish to maximize the minimum total size among all packings in a transformation between A_0 and A_t. We call this maximization problem the MAXMIN SUBSET SUM RECONFIGURATION problem. The sequence of packings emphasized by thick lines in Fig. 2 is an optimal solution for $A_0 = \{5, 6\}$ and $A'_t = \{6, 11\}$; its objective value is 8.

Reconfiguration problems have been studied extensively in recent literature, such as SAT RECONFIGURATION [7,18,19], INDEPENDENT SET RECONFIGURATION [9,10,12,16], SHORTEST PATH RECONFIGURATION [3,15], VERTEX-COLORING RECONFIGURATION [2,4], LIST EDGE-COLORING RECONFIGURATION [13,14], etc. However, reconfiguration problems for SUBSET SUM have not been studied yet. One can easily imagine a variety of practical scenarios, where a packing (e.g., representing a feasible display of electronic advertisements on a Web browser) needs to be changed (to show other advertisements) by individual changes (appealing to the user by showing one by one) while maintaining both threshold and capacity of the allowed area on the Web browser (in order to maintain both advertiser and user satisfactions during the transformation). Similar situations can be found when moving products in automated warehouse, changing displays at a supermarket, etc. Reconfiguration problems are also interesting in general because they provide a new perspective and deeper understanding of the solution space and of heuristics that navigate that space.

Several variants have been studied for (ordinary) SUBSET SUM and MAXIMUM SUBSET SUM [17]. In particular, MAXIMUM SUBSET SUM with "conflict graph" [21] is an important variant, because this variant has been studied for several other problems, such as BIN PACKING [5] and SCHEDULING under makespan minimization [1]. In the variant, we are given a *conflict graph* for a set A of items in which each vertex corresponds to an item in A, and each edge represents a pair of items in A that are not allowed to be packed together into the knapsack.

In this paper, we first show that SUBSET SUM RECONFIGURATION is strongly NP-hard, and is PSPACE-complete for the variant with conflict graph. We then show that MAXMIN SUBSET SUM RECONFIGURATION with conflict graph is APX-hard, and hence there is no polynomial-time approximation scheme (PTAS) for this variant unless P = NP. (Remember that, for any ε, $0 < \varepsilon < 1$, a PTAS for a maximization problem finds a solution with objective value APX such that $\mathrm{APX} \geq (1-\varepsilon)\mathrm{OPT}$ in polynomial time when ε is regarded as a fixed constant, where OPT is the optimal objective value.) In contrast, we give a PTAS for the original version of MAXMIN SUBSET SUM RECONFIGURATION. Note that, since this maximization problem is strongly NP-hard, the problem does not admit a fully polynomial-time approximation scheme (FPTAS) unless P = NP; in this sense, a PTAS is the best approximation algorithm we can expect for the problem [20,23]. (Remember that an FPTAS is a PTAS which runs in time polynomial in both the input size and $1/\varepsilon$.) We also remark that, as far as we know, this is the first PTAS obtained for this kind of reconfiguration problems. An early version of the paper has been presented in [11].

Our main result of this paper is a PTAS for MAXMIN SUBSET SUM RECONFIGURATION. The strategy of our PTAS is the following: we divide a set A of items into two groups, one is the set of items having "large" sizes, and the other consists of items having "small" sizes; and we deal with the two groups separately. Because such an approximation technique is fairly standard, especially for MAXIMUM SUBSET SUM and BIN PACKING [17,23], one might think that our PTAS could be obtained directly by extending several known FPTASs or PTASs [17,23]. However, this is not the case, because the focus of reconfiguration problems is different from the ordinary problems: we seek the *reachability* between two feasible solutions, and hence the placement of items is the central matter. For example, two packings $\{5, 6\}$ and $\{11\}$ in Fig. 1 have the same total size 11, and hence we can regard them as an "equivalent" packing in the ordinary SUBSET SUM problem. However, we must identify these two packings in the reconfiguration problems; for example, $\{11\}$ can be transformed into $\{6, 11\}$, but $\{5, 6\}$ cannot, when $k = 10$. (See Fig. 1.) We thus introduce a "configuration graph" which represents the placements of items and their connectivity. (A formal definition will be given in Section 3, but an example is already shown in Fig. 2.) Our main idea is to approximate the configuration graph appropriately.

2 Complexity and Inapproximability

Before showing our results, we introduce some terms and define the problems more formally. In Introduction, we have defined a packing A_i as a subset of items in a set A such that the total size of A_i is at most the capacity c of a knapsack; the total size of a packing A_i is denoted simply by $s(A_i)$, that is, $s(A_i) = \sum_{a \in A_i} s(a)$. Note that a packing does not necessarily satisfy a threshold k. We say that two packings A_i and A_j of A are *adjacent* if their symmetric difference is of cardinality 1, that is, $|A_i \bigtriangleup A_j| = \big|(A_i \setminus A_j) \cup (A_j \setminus A_i)\big| = 1$; the item a in $A_i \bigtriangleup A_j$ is said to be *moved* between A_i and A_j. A *reconfiguration sequence* between two packings A_0 and A_t is a sequence of packings $A_0, A_1, \ldots, A_t$ such that A_{i-1} and A_i are adjacent for $i = 1, 2, \ldots, t$. For a reconfiguration sequence $\mathcal{P}$, we denote by $f(\mathcal{P})$ the minimum total size among all packings in $\mathcal{P}$, that is, $f(\mathcal{P}) = \min\{s(A_i) : A_i \in \mathcal{P}\}$. Then, for two packings A_0 and A_t, let

$$\mathrm{OPT}(A_0, A_t) = \max\{f(\mathcal{P}) : \mathcal{P} \text{ is a reconfiguration sequence between } A_0 \text{ and } A_t\}.$$

Given an integer threshold k and two packings A_0 and A_t, the SUBSET SUM RECONFIGURATION problem is to determine whether $\mathrm{OPT}(A_0, A_t) \geq k$. On the other hand, its optimization version is defined as follows: Given two packings A_0 and A_t, the MAXMIN SUBSET SUM RECONFIGURATION problem is to compute $\mathrm{OPT}(A_0, A_t)$. Note that we are asked simply to compute the optimal value $\mathrm{OPT}(A_0, A_t)$, and hence we need *not* to find an actual reconfiguration sequence.

We first have the following theorem.

Theorem 1 SUBSET SUM RECONFIGURATION *is strongly NP-hard.*

Proof We give a polynomial-time reduction from the 3-PARTITION problem [6] to our problem. In 3-PARTITION, we are given a positive integer bound b, and a set U of $3m$ elements $u_1, u_2, \ldots, u_{3m}$; each element $u_i \in U$ has a positive integer size $s(u_i)$ such that $b/4 < s(u_i) < b/2$ and such that $\sum_{u \in U} s(u) = mb$. Then, the 3-PARTITION problem is to determine whether U can be partitioned into m disjoint subsets $U_1, U_2, \ldots, U_m$ such that $\sum_{u \in U_j} s(u) = b$ for each j, $1 \leq j \leq m$. It is known that 3-PARTITION is strongly NP-complete [6].

Given an instance of 3-PARTITION, we construct the corresponding instance of SUBSET SUM RECONFIGURATION. The set A consists of $4m$ items $a_1, a_2, \ldots, a_{3m}, b_1, b_2, \ldots, b_m$: let $s(a_i) = s(u_i)$ for each i, $1 \leq i \leq 3m$, and let $s(b_j) = b$ for each j, $1 \leq j \leq m$. Then, each item a_i corresponds to the element u_i in U. The knapsack is of capacity $c = mb$, and set the threshold $k = (m-1)b$. Finally, the two packings A_0 and A_t are defined as follows: $A_0 = \{b_1, b_2, \ldots, b_m\}$ and $A_t = U$, and hence both A_0 and A_t are of total size mb $(> k)$.

Since $k = (m-1)b$ and $s(b_j) = b$ for all j, $1 \leq j \leq m$, it is easy to see that there exists a desired partition $\{U_1, U_2, \ldots, U_m\}$ if and only if there exists a reconfiguration sequence $\mathcal{P}$ between A_0 and A_t with $f(\mathcal{P}) = k = (m-1)b$ and hence $\mathrm{OPT}(A_0, A_t) = (m-1)b \geq k$. □

Theorem 1 immediately implies the following corollary.

Corollary 1 MAXMIN SUBSET SUM RECONFIGURATION *is strongly NP-hard.*

We then consider the variant with conflict graph. Notice that every feasible packing of A induces an independent set in the conflict graph. We have the following theorem.

Theorem 2 SUBSET SUM RECONFIGURATION *with conflict graph is PSPACE-complete.*

Proof We first show that the problem is in PSPACE. Since PSPACE = NPSPACE [22], it suffices to show that the problem can be solved in nondeterministic polynomial space. Clearly, a packing can be described in linear number of bits, simply by specifying whether each item in A is contained in the packing or not. Furthermore, since we can move only a single item from the current packing, the number of packings adjacent to the current one is at most n, and hence all the adjacent packings can be enumerated in polynomial time. Therefore, we can traverse the packings, at each step nondeterministically choosing an adjacent packing, and maintaining the current packing together with checking whether or not the packing is equal to the target packing A_t. Thus, the problem is in NPSPACE, and hence in PSPACE.

We then show that SUBSET SUM RECONFIGURATION with conflict graph is PSPACE-hard by giving a polynomial-time reduction from the INDEPENDENT SET RECONFIGURATION problem [12]. Given a graph G of n nodes, an integer threshold k', and two independent sets I_0 and I_t of G, both of cardinality at least k', the INDEPENDENT SET RECONFIGURATION problem asks whether we can transform I_0 into I_t via independent sets of G, each of which results from the previous one by either adding or removing a single node of G, without ever going through an independent set of cardinality less than $k' - 1$. This problem is known to be PSPACE-complete [12].

We now construct the corresponding instance of SUBSET SUM RECONFIGURATION with conflict graph. The set A contains n items, and let $s(a) = 1$ for all items a in A. Each item in A corresponds to a node of G, and the conflict graph for A is connected as G. The knapsack is of capacity $c = n$, and let the threshold $k = k' - 1$. Finally, the two packings A_0 and A_t consist of the items which correspond to the nodes in I_0 and I_t, respectively; and hence both A_0 and A_t are of total size at least $k' = k + 1$.

Since every feasible packing of total size at least k induces an independent set in G of cardinality at least $k = k' - 1$, it is obvious that there is a desired transformation between I_0 and I_t if and only if $\mathrm{OPT}(A_0, A_t) \geq k$. □

We note in passing that SUBSET SUM RECONFIGURATION with conflict graph is strongly NP-hard even if a conflict graph is either empty or a star: if a conflict graph is empty, then the problem is the original version of SUBSET SUM RECONFIGURATION; and we can easily obtain a star from an empty conflict graph, with keeping the feasibility, by adding a dummy item of size equal to the capacity c and joining it with each of the vertices in the empty conflict graph.

We finally have the following inapproximability result.

Theorem 3 Maxmin subset sum reconfiguration *with conflict graph is APX-hard, and cannot be approximated within any constant factor unless* $\mathrm{P} = \mathrm{NP}$.

Proof We give a polynomial-time reduction in an approximation-preserving manner from the (ordinary) independent set problem to our problem. In independent set, we are given a graph G with n nodes, and we are asked to compute the maximum cardinality $\mathrm{OPT}_I(G)$ of independent sets in G. It is known that independent set is APX-complete [20] and cannot be approximated within any constant factor unless $\mathrm{P} = \mathrm{NP}$ [8].

Given an instance of independent set, we now construct the corresponding instance of maxmin subset sum reconfiguration with conflict graph. The set A consists of $n+2$ items $a_1, a_2, \ldots, a_n, b_1, b_2$: we set $s(a_i) = 1$ for each i, $1 \leq i \leq n$, and set $s(b_1) = s(b_2) = n+1$; each item a_i corresponds to a node in G. In the conflict graph, the n nodes corresponding to $a_1, a_2, \ldots, a_n$ are connected as G; the conflict graph consists of three components G, $\{b_1\}$ and $\{b_2\}$. (If required, we can easily obtain a connected conflict graph by adding a dummy item of size equal to the capacity c and joining it with each vertex in the conflict graph.) The knapsack is of capacity $c = 2n + 1$, and the two packings A_0 and A_t are defined as follows: $A_0 = \{b_1\}$ and $A_t = \{b_2\}$.

We now show that the reduction above preserves approximability. Consider any reconfiguration sequence $\mathcal{P}$ between A_0 and A_t. Let $A_{\min}$ be the packing in $\mathcal{P}$ having the minimum total size, and hence $f(\mathcal{P}) = s(A_{\min})$. Since $s(b_1) + s(b_2) = 2n + 2 > c$, there must exist a packing in $\mathcal{P}$ which contains neither b_1 nor b_2. Since $s(b_1) = s(b_2) = n + 1$, we thus know that neither b_1 nor b_2 are contained in $A_{\min}$. Therefore, $A_{\min}$ induces an independent set in G of cardinality $s(A_{\min})$. Conversely, for every independent set I of G, there exists a reconfiguration sequence $\mathcal{P}_I$ from A_0 to A_t via the packing A_I corresponding to I: add to the packing A_0 the items corresponding to the nodes in I one by one (then, we obtain $A_0 \cup A_I$), remove b_1 (then, obtain the packing A_I), add b_2, and remove the items in A_I one by one (then, obtain A_t). Since $s(b_1) = s(b_2) = n + 1$, we clearly have $f(\mathcal{P}_I) = s(A_I)$. Therefore, solving (or, approximating) this instance of maxmin subset sum reconfiguration with conflict graph is equivalent to solving (respectively, approximating) independent set for G. Thus, the results follow. □

3 Polynomial-Time Approximation Scheme

Since maxmin subset sum reconfiguration with conflict graph is APX-hard, this variant does not admit a PTAS unless $\mathrm{P} = \mathrm{NP}$. In this section, we give a PTAS for the original version, as in the following theorem.

Theorem 4 *There is a polynomial-time approximation scheme for* maxmin subset sum reconfiguration.

In the remainder of this section, as a proof of Theorem 4, we give an algorithm which actually finds a reconfiguration sequence $\mathcal{P}$ between two given packings A_0 and A_t such that $f(\mathcal{P}) \geq (1 - \varepsilon')\mathrm{OPT}(A_0, A_t)$ in time polynomial in n (but, exponential in

$1/\varepsilon'$) for any fixed constant ε', $0 < \varepsilon' < 1$, where n is the number of items in the set A. Therefore, our approximate objective value $\mathrm{APX}(A_0, A_t)$ is $f(\mathcal{P})$, and hence the error is bounded by $\varepsilon'\mathrm{OPT}(A_0, A_t)$, that is,

$$\mathrm{OPT}(A_0, A_t) - \mathrm{APX}(A_0, A_t) = \mathrm{OPT}(A_0, A_t) - f(\mathcal{P}) \leq \varepsilon'\mathrm{OPT}(A_0, A_t).$$

As we have mentioned in Introduction, the placement of items is the central matter in the reconfiguration problem. Therefore, we construct an edge-weighted graph, called a configuration graph, which represents all (feasible) packings together with their adjacency. For a set A of items and a knapsack of capacity c, a *configuration graph* $\mathcal{C} = (\mathcal{V}, \mathcal{E})$ is defined as follows: each vertex in $\mathcal{V}$ corresponds to a packing A_i, and two vertices are joined by an edge e in $\mathcal{E}$ if and only if the corresponding two packings A_i and A_j are adjacent; the *weight* $\omega(e)$ of e is defined as follows: $\omega(e) = \min\{s(A_i), s(A_j)\}$. Notice that the weight $\omega(e)$ of an edge e corresponds to the objective value $f(\mathcal{P}_{i,j})$ for the reconfiguration sequence $\mathcal{P}_{i,j} = \langle A_i, A_j \rangle$ along e. Figure 2 illustrates the configuration graph for a set $A = \{5, 6, 8, 11\}$ and a knapsack of capacity $c = 20$, where each vertex is drawn as a box and each edge as a line. From now on, we may call a packing simply a vertex of a configuration graph if it is clear from the context. Since there is a vertex corresponding to the empty packing, a configuration graph is always connected. Then, MAXMIN SUBSET SUM RECONFIGURATION can be seen as the problem of maximizing the minimum edge-weight in a path between A_0 and A_t in $\mathcal{C}$. It is easy to see that the problem can be solved in time polynomial in $|\mathcal{V}|+|\mathcal{E}|$, by the following naive algorithm: delete all edges having the smallest weight from $\mathcal{C}$, and check whether the two vertices A_0 and A_t remain in the same connected component of the resulting graph; if so, let $\mathcal{C}$ be the resulting graph and repeat. Note that, however, the size $|\mathcal{V}| + |\mathcal{E}|$ of $\mathcal{C}$ can be exponential in n.

We now briefly explain our PTAS together with the organization of this section. For a fixed constant ε', $0 < \varepsilon' < 1$, let

$$\varepsilon = \frac{1}{2}\varepsilon'. \tag{1}$$

(The reason why the coefficient above is 1/2 will be explained in Section 3.4.) Given a set A of items and the fixed constant ε, we divide the items of A into two groups: an item a is called a *large item* if $s(a) \geq \varepsilon c/2$; otherwise the item is called a *small item*. We show in Section 3.1 that the problem can be optimally solved in polynomial time if A contains only large items; in this case, the number of packings (and hence the number of vertices in the configuration graph) can be bounded by a polynomial in n. In Section 3.2 we then explain that small items can be moved greedily with only small error. In Section 3.3 we finally deal with a general instance by combining the techniques above, without losing the reachability and with keeping the small error. Section 3.4 gives the analysis of our algorithm.

3.1 Large items

In this subsection, we show that MAXMIN SUBSET SUM RECONFIGURATION can be optimally solved in polynomial time if the given set A contains only large items. It suffices to show that we can construct the corresponding configuration graph $\mathcal{C} = (\mathcal{V}, \mathcal{E})$ in polynomial time for such an instance, and that the size $|\mathcal{V}| + |\mathcal{E}|$ of $\mathcal{C}$ is a polynomial in n. Formally, we have the following lemma.

Lemma 1 *For a fixed constant $\varepsilon > 0$, suppose that every item in the set A is of size at least $\varepsilon c/2$, where c is the capacity of the knapsack. Then,* MAXMIN SUBSET SUM RECONFIGURATION *can be optimally solved in polynomial time.*

Proof Since $s(a) \geq \varepsilon c/2$ for all items $a \in A$, the number of items in any (feasible) packing is bounded by $\lfloor 2/\varepsilon \rfloor$. Let $\gamma = \lfloor 2/\varepsilon \rfloor$, then γ is a fixed constant. Since A contains n items, it is easy to see that the number of all packings for A and c can be bounded by $\binom{n+\gamma}{\gamma}$. Therefore, $|\mathcal{V}|$ is a polynomial in n, and hence we can construct $\mathcal{C}$ in time polynomial in n. Since the size $|\mathcal{V}| + |\mathcal{E}|$ of $\mathcal{C}$ is a polynomial in n, we can solve the problem optimally in polynomial time. □

3.2 Small items

Suppose in this subsection that the given set A may contain small items. Then, the number of items in a packing cannot be bounded by a constant, and hence the number $|\mathcal{V}|$ of vertices in the configuration graph $\mathcal{C} = (\mathcal{V}, \mathcal{E})$ cannot be always bounded by a polynomial in n; more specifically, $|\mathcal{V}|$ can be $O(2^n)$. Therefore, we will later (in Section 3.3) construct an "approximate configuration graph $\mathcal{C}_A$," whose size is bounded by a polynomial in n.

We now explain how to find a reconfiguration sequence greedily when $A_0 \triangle A_t$ contains only small items for two given packings A_0 and A_t. Let L_ε be the set of large items in A, that is, $L_\varepsilon = \{a \in A \mid s(a) \geq \varepsilon c/2\}$, and let $S_\varepsilon = A \setminus L_\varepsilon$. We have the following lemma.

Lemma 2 *Let A_0 and A_t be an arbitrary pair of packings such that $A_0 \triangle A_t \subseteq S_\varepsilon$. Then, there exists a reconfiguration sequence $\mathcal{P}_s$ between A_0 and A_t such that*
(a) no item in L_ε is moved in $\mathcal{P}_s$; and
(b) $f(\mathcal{P}_s) \geq (1-\varepsilon)\min\{s(A_0), s(A_t)\}$.
Moreover, such a reconfiguration sequence $\mathcal{P}_s$ can be found in linear time.

Proof We give an $O(n)$-time algorithm which finds a reconfiguration sequence $\mathcal{P}_s$ between A_0 and A_t satisfying (a) and (b), as follows.

Case (i): $s(A_0 \cup A_t) \leq c$.

In this case, we first add all items in $A_t \setminus A_0$ one by one, and obtain the packing $A_0 \cup A_t$; and then, delete all items in $A_0 \setminus A_t$ one by one, and obtain A_t. Note that $A_t \setminus A_0 \subseteq A_0 \triangle A_t \subseteq S_\varepsilon$ and $A_0 \setminus A_t \subseteq A_0 \triangle A_t \subseteq S_\varepsilon$, and hence no item in L_ε is moved in this reconfiguration sequence $\mathcal{P}_s$. We clearly have

$$f(\mathcal{P}_s) = \min\{s(A_0), s(A_t)\} > (1-\varepsilon)\min\{s(A_0), s(A_t)\}.$$

Therefore, $\mathcal{P}_s$ satisfies both (a) and (b). Moreover, $\mathcal{P}_s$ can be found in linear time since we move each item in $A_0 \triangle A_t$ only once.

Case (ii): $s(A_0 \cup A_t) > c$.

In this case, we first add items in $A_t \setminus A_0$ one by one in arbitrary order as long as the total size is at most c; let A_j be the current packing. Then, $s(A_j) > (1 - \frac{\varepsilon}{2})c$; otherwise we can add more items to A_j since $s(a) < \varepsilon c/2$ for all items $a \in A_t \setminus A_0$. We then delete items in $A_0 \setminus A_t$ one by one in arbitrary order until we obtain a packing A'_j such that

$$(1-\varepsilon)c < s(A'_j) \le \left(1 - \frac{\varepsilon}{2}\right) c. \tag{2}$$

Since $s(a) < \varepsilon c/2$ for all items $a \in A_0 \setminus A_t$, we can always find such a packing A'_j. If $s(A'_j \cup A_t) \le c$, then go to Case (i) above; otherwise, repeat Case (ii). Note that, in this reconfiguration sequence $\mathcal{P}_s$, every addition is executed for an item in $A_t \setminus A_0$ $\big(\subseteq S_\varepsilon\big)$ and every deletion is done for an item in $A_0 \setminus A_t$ $\big(\subseteq S_\varepsilon\big)$. Thus, $\mathcal{P}_s$ satisfies (a). Furthermore, since each item in $A_0 \vartriangle A_t$ is moved exactly once, $\mathcal{P}_s$ can be found in linear time. We now show that (b) holds for $\mathcal{P}_s$. By Eq. (2) we have

$$f(\mathcal{P}_s) \ge \min\Big\{(1-\varepsilon)c, \min\{s(A_0), s(A_t)\}\Big\}.$$

Since $c \ge \min\{s(A_0), s(A_t)\}$, we have $f(\mathcal{P}_s) \ge (1-\varepsilon)\min\{s(A_0), s(A_t)\}$. $\square$

3.3 General instance

We finally deal with a general instance, that is, a set A may contain small items and two packings A_0 and A_t do not necessarily satisfy $A_0 \vartriangle A_t \subseteq S_\varepsilon$. Our idea is to construct an *approximate configuration graph* $\mathcal{C}_A$, as follows.

Step 1: Configuration graph for L_ε

We first construct a configuration graph $\mathcal{C}_{L_\varepsilon} = (\mathcal{V}_{L_\varepsilon}, \mathcal{E}_{L_\varepsilon})$ for the large item set L_ε of A and the capacity c. Then, as in Lemma 1, $\mathcal{C}_{L_\varepsilon}$ can be constructed in time polynomial in n, and the size $|\mathcal{V}_{L_\varepsilon}| + |\mathcal{E}_{L_\varepsilon}|$ of $\mathcal{C}_{L_\varepsilon}$ can be bounded by a polynomial in n. Figure 3(a) illustrates the configuration graph for L_ε of A, where each box corresponds to a packing consisting only of large items. Note that $\mathcal{C}_{L_\varepsilon}$ contains the vertex corresponding to the empty packing, and hence $\mathcal{C}_{L_\varepsilon}$ is connected.

Step 2: Small items

We then expand $\mathcal{C}_{L_\varepsilon}$ into the approximate configuration graph $\mathcal{C}_A = (\mathcal{V}_A, \mathcal{E}_A)$, as illustrated in Fig. 3(b). For each edge in $\mathcal{C}_{L_\varepsilon}$ joining two vertices A_i^L and A_j^L (that consist only of large items), we replace it with an edge e that joins two new vertices $A_{i,x}$ and $A_{j,y}$, called *gate vertices* or *gate packings*, defined as follows. Assume without loss of generality that $A_j^L = A_i^L \cup \{a\}$ for some large item a in L_ε, and hence A_j^L can be obtained by adding one large item a to A_i^L. To extend A_j^L to the gate packing $A_{j,y}$ containing small items, we find a packing $A_j^S \subseteq S_\varepsilon$ of small items for the remaining space $c - s(A_j^L)$ of the knapsack; we employ an FPTAS for the ordinary MAXIMUM SUBSET SUM problem [17] for the fixed constant ε. Then, let $A_{j,y} = A_j^L \cup A_j^S$ and let $A_{i,x} = A_{j,y} \setminus \{a\}$. Note that $A_{i,x} \vartriangle A_{j,y} = \{a\}$ and hence $A_{i,x}$ and $A_{j,y}$ are adjacent. We call the edge $e = (A_{i,x}, A_{j,y})$ an *external edge*, and the weight $\omega(e)$ is defined as follows:

$$\omega(e) = \min\{s(A_{i,x}), s(A_{j,y})\} = s(A_{i,x}).$$

In Fig. 3(b), each gate packing is represented by a circle, triangle, square, pentagon, or hexagon, colored with white; all gate packings represented by the same symbol have the same placement of large items; and each external edge is drawn as a (non-dotted) line.

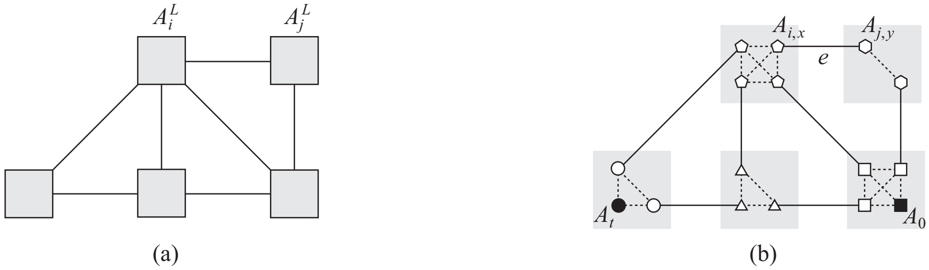

Fig. 3 (a) Configuration graph $\mathcal{C}_{L_\varepsilon}$ for the large item set L_ε of A, and (b) approximate configuration graph $\mathcal{C}_A$ for A.

For each vertex A_i^L in $\mathcal{C}_{L_\varepsilon}$, we have thus created the number $d(A_i^L)$ of new gate vertices $A_{i,1}, A_{i,2}, \ldots, A_{i,d(A_i^L)}$, where $d(A_i^L)$ is the degree of A_i^L in $\mathcal{C}_{L_\varepsilon}$. Note that these gate packings $A_{i,1}, A_{i,2}, \ldots, A_{i,d(A_i^L)}$ are not necessarily distinct, and $A_{i,x} \cap L_\varepsilon = A_i^L$ holds for each index x, $1 \leq x \leq d(A_i^L)$. The original vertex A_i^L is deleted, and we connect the $d(A_i^L)$ gate vertices so that they form a clique. For each pair of vertices $A_{i,x}$ and $A_{i,z}$, the edge joining them is called an *internal edge*; in Fig. 3(b), each internal edge is drawn as a dotted line. It should be noted that $A_{i,x}$ and $A_{i,z}$ are not necessarily adjacent in $\mathcal{C}$ although they are joined by an internal edge. However, using Lemma 2 we can regard such an internal edge as a reconfiguration sequence $\mathcal{P}_s$ between $A_{i,x}$ and $A_{i,z}$ such that $f(\mathcal{P}_s) \geq (1-\varepsilon)\min\{s(A_{i,x}), s(A_{i,z})\}$. Therefore, the weight $\omega(e)$ of e is defined as follows:

$$\omega(e) = \begin{cases} \min\{s(A_{i,x}), s(A_{i,z})\} & \text{if } A_{i,x} = A_{i,z}, \text{ or } A_{i,x} \text{ and } A_{i,z} \text{ are adjacent;} \\ (1-\varepsilon)\min\{s(A_{i,x}), s(A_{i,z})\} & \text{otherwise.} \end{cases} \tag{3}$$

Step 3: A_0 and A_t

The current graph above does not always contain the vertices corresponding to given packings A_0 and A_t. If the graph does not contain A_0, then we add a new vertex A_0 to the graph, and join it with each gate vertex having the same placement $A_0 \cap L_\varepsilon$ of large items by an internal edge. (The case for A_t is similar.) This completes the construction of the approximate configuration graph $\mathcal{C}_A = (\mathcal{V}_A, \mathcal{E}_A)$.

Clearly, a path between the two vertices A_0 and A_t in $\mathcal{C}_A$ corresponds to a reconfiguration sequence between the two packings A_0 and A_t. Since $|\mathcal{V}_A| \leq 2|\mathcal{E}_{L_\varepsilon}| + 2$ and $|\mathcal{E}_{L_\varepsilon}|$ is bounded by a polynomial in n, the size $|\mathcal{V}_A| + |\mathcal{E}_A|$ of $\mathcal{C}_A$ is bounded by a polynomial in n. Therefore, we can find in polynomial time a path between A_0 and A_t whose minimum edge-weight is maximum in $\mathcal{C}_A$; we take the corresponding reconfiguration sequence $\mathcal{P}$ as our approximate solution.

3.4 Analysis of the algorithm

We have shown in Section 3.3 that our algorithm finds a reconfiguration sequence $\mathcal{P}$ between A_0 and A_t in time polynomial in n (but, exponential in $1/\varepsilon$). In this subsection,

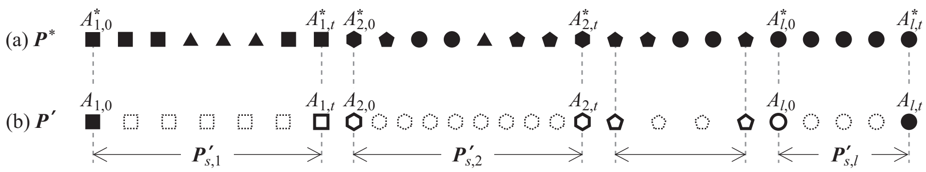

Fig. 4 (a) Optimal reconfiguration sequence $\mathcal{P}^*$, and (b) reconfiguration sequence $\mathcal{P}'$ such that $\mathcal{C}_A$ contains the path corresponding to $\mathcal{P}'$.

we show that $\mathcal{P}$ satisfies $f(\mathcal{P}) \geq (1-\varepsilon')\mathrm{OPT}(A_0, A_t)$ for a fixed constant ε', $0 < \varepsilon' < 1$, as required.

Let $\mathcal{P}^* = \langle A^*_0, A^*_1, \ldots, A^*_t \rangle$ be an arbitrary optimal reconfiguration sequence between A_0 and A_t, where $A^*_0 = A_0$ and $A^*_t = A_t$. Figure 4(a) illustrates the optimal reconfiguration sequence $\mathcal{P}^*$, where each black symbol corresponds to a packing A^*_i in $\mathcal{P}^*$ and all packings represented by the same symbol have the same placement of large items. Let $A^*_{\min}$ be a packing in $\mathcal{P}^*$ whose total size is minimum, and hence $f(\mathcal{P}^*) = s(A^*_{\min})$. Then, we have

$$s(A^*_{\min}) = \mathrm{OPT}(A_0, A_t), \tag{4}$$

and

$$s(A^*_i) \geq s(A^*_{\min}) \tag{5}$$

for each packing A^*_i, $0 \leq i \leq t$.

From now on, we transform $\mathcal{P}^*$ into another reconfiguration sequence $\mathcal{P}'$ between A_0 and A_t so that $\mathcal{C}_A$ contains the path corresponding to $\mathcal{P}'$. Remember that our algorithm finds a reconfiguration sequence $\mathcal{P}$ between A_0 and A_t which is optimal in $\mathcal{C}_A$, and hence we have

$$\mathrm{APX}(A_0, A_t) = f(\mathcal{P}) \geq f(\mathcal{P}'). \tag{6}$$

We first find the "last" packing in $\mathcal{P}^*$ for each large-item placement. This can be done by the following algorithm: let $A^*_{1,0} = A^*_0$; at the i-th step, $i \geq 1$, find the last packing A^*_x in $\mathcal{P}^*$ such that $A^*_x \cap L_\varepsilon = A^*_{i,0} \cap L_\varepsilon$; set $A^*_{i,t} = A^*_x$ and $A^*_{i+1,0} = A^*_{x+1}$, and repeat. Let ℓ be the number of iterations of the algorithm, and hence $A^*_{\ell,t} = A^*_t = A_t$. (See Fig. 4(a).)

For each i, $1 \leq i \leq \ell - 1$, we then find the gate packings $A_{i,t}$ and $A_{i+1,0}$ in $\mathcal{C}_A$ which correspond to $A^*_{i,t}$ and $A^*_{i+1,0}$, respectively. Note that $A^*_{1,0} = A_0$ and $A^*_{\ell,t} = A_t$, and hence both $A^*_{1,0}$ and $A^*_{\ell,t}$ are contained in $\mathcal{C}_A$. Remember that the two packings $A^*_{i,t}$ and $A^*_{i+1,0}$ are adjacent; moreover, the item moved between them is a large item $a \in L_\varepsilon$. Therefore, $\mathcal{C}_A$ contains the external edge $e = (A_{i,t}, A_{i+1,0})$ which corresponds to moving the item a from the large-item placement $A^*_{i,t} \cap L_\varepsilon$ to $A^*_{i+1,0} \cap L_\varepsilon$. We may regard that the two endpoints (gate packings) $A_{i,t}$ and $A_{i+1,0}$ of e correspond to $A^*_{i,t}$ and $A^*_{i+1,0}$, respectively. Of course, the gate packings $A_{i,t}$ and $A_{i+1,0}$ are not always the same as $A^*_{i,t}$ and $A^*_{i+1,0}$, respectively, and hence they are depicted by (non-dotted) white symbols in Fig. 4(b). However, it should be noted that $A^*_{i,t} \cap L_\varepsilon = A_{i,t} \cap L_\varepsilon$ and $A^*_{i+1,0} \cap L_\varepsilon = A_{i+1,0} \cap L_\varepsilon$, and hence $A_{i,t}$ and $A_{i+1,0}$ in Fig. 4(b) are depicted by the same symbols as $A^*_{i,t}$ and $A^*_{i+1,0}$, respectively. For the sake of notational convenience, let $A_{1,0} = A^*_{1,0} = A_0$ and $A_{\ell,t} = A^*_{\ell,t} = A_t$.

We finally define the reconfiguration sequence $\mathcal{P}'$ between A_0 and A_t. Since both $A_{i,0}$ and $A_{i,t}$, $1 \leq i \leq \ell$, have the same large-item placement and are contained in $\mathcal{C}_A$, there is the internal edge $e_{s,i}$ joining them; let $\mathcal{P}'_{s,i}$ be the reconfiguration sequence between $A_{i,0}$ and $A_{i,t}$ corresponding to $e_{s,i}$. Let $\mathcal{P}' = \mathcal{P}'_{s,1} \cup \mathcal{P}'_{s,2} \cup \cdots \cup \mathcal{P}'_{s,\ell}$, as illustrated in Fig. 4(b). Note that the intermediate packings in $\mathcal{P}'_{s,i}$ are not necessarily contained in $\mathcal{P}^*$, and hence they are represented by dotted white symbols in Fig. 4(b). Furthermore, by Lemma 2(a) all packings in $\mathcal{P}'_{s,i}$ have the same placement of large items, and hence they are depicted by the same symbol in Fig. 4(b). By Eq. (3) we have

$$f(\mathcal{P}'_{s,i}) = \omega(e_{s,i}) \geq (1-\varepsilon)\min\{s(A_{i,0}), s(A_{i,t})\} \tag{7}$$

for each i, $1 \leq i \leq \ell$. Note that $f(\mathcal{P}') = \min\{f(\mathcal{P}'_{s,i}) : 1 \leq i \leq \ell\}$.

This completes the construction of $\mathcal{P}'$.

We now show the following lemma.

Lemma 3 *$s(A_{i,0}) > (1-\varepsilon)s(A^*_{i,0})$ and $s(A_{i,t}) > (1-\varepsilon)s(A^*_{i,t})$ for each i, $1 \leq i \leq \ell$.*

Proof Since $A_{1,0} = A^*_{1,0} = A_0$ and $A_{\ell,t} = A^*_{\ell,t} = A_t$ (see also Fig. 4), the lemma clearly holds for $A_{1,0}$ and $A_{\ell,t}$. Therefore, we prove the lemma for packings $A_{i,t}$ and $A_{i+1,0}$, $1 \leq i \leq \ell-1$. Then, since they are gate packings, $\mathcal{C}_A$ has the corresponding external edge $(A_{i,t}, A_{i+1,0})$. Assume without loss of generality that $A_{i+1,0} = A_{i,t} \cup \{a\}$ for some large item $a \in L_\varepsilon$, that is, $A_{i+1,0}$ can be obtained by adding the large item a to $A_{i,t}$.

We first consider $A_{i+1,0}$. Let $A^L_{i+1,0} = A_{i+1,0} \cap L_\varepsilon$ and $A^S_{i+1,0} = A_{i+1,0} \cap S_\varepsilon$. Remember that $A^S_{i+1,0}$ was obtained by using an FPTAS for the set S_ε, the remaining capacity $c - s(A^L_{i+1,0})$ and the fixed constant ε. Let $\mathrm{OPT}^S_{i+1,0}$ be the optimal value for the ordinary MAXIMUM SUBSET SUM problem for the set S_ε and the capacity $c - s(A^L_{i+1,0})$. Then,

$$s(A^S_{i+1,0}) \geq (1-\varepsilon)\mathrm{OPT}^S_{i+1,0}, \tag{8}$$

and hence

$$\begin{aligned} s(A_{i+1,0}) &= s(A^L_{i+1,0}) + s(A^S_{i+1,0}) \\ &\geq s(A^L_{i+1,0}) + (1-\varepsilon)\mathrm{OPT}^S_{i+1,0} \\ &> (1-\varepsilon)\big(s(A^L_{i+1,0}) + \mathrm{OPT}^S_{i+1,0}\big). \end{aligned} \tag{9}$$

Note that $\mathrm{OPT}^S_{i+1,0}$ is the maximum total size of small-item placements under the constraint that the knapsack has the large-item placement $A^L_{i+1,0}$. Since $A^*_{i+1,0} \cap L_\varepsilon = A^L_{i+1,0}$, we thus have $s(A^L_{i+1,0}) + \mathrm{OPT}^S_{i+1,0} \geq s(A^*_{i+1,0})$. Therefore, by Eq. (9) we have $s(A_{i+1,0}) > (1-\varepsilon)s(A^*_{i+1,0})$, as required.

We then consider $A_{i,t}$. Let $A^L_{i,t} = A_{i,t} \cap L_\varepsilon$ and $A^S_{i,t} = A_{i,t} \cap S_\varepsilon$. Since $A_{i,t} = A_{i+1,0} \setminus \{a\}$, we have $A^L_{i,t} = A^L_{i+1,0} \setminus \{a\}$ and $A^S_{i,t} = A^S_{i+1,0}$. By Eq. (8) we have

$$\begin{aligned} s(A_{i,t}) &= s(A^L_{i,t}) + s(A^S_{i,t}) \\ &= s(A^L_{i,t}) + s(A^S_{i+1,0}) \\ &\geq s(A^L_{i,t}) + (1-\varepsilon)\mathrm{OPT}^S_{i+1,0} \\ &> (1-\varepsilon)\big(s(A^L_{i,t}) + \mathrm{OPT}^S_{i+1,0}\big). \end{aligned} \tag{10}$$

Since $A^*_{i,t} \cap L_\varepsilon = A^L_{i,t}$ and we will add the large item a to $A^*_{i,t}$ to obtain $A^*_{i+1,0}$, we have

$$s(A^*_{i,t} \cap S_\varepsilon) \leq c - \bigl(s(A^L_{i,t}) + s(a)\bigr) = c - s(A^L_{i+1,0}).$$

Remember that $\mathrm{OPT}^S_{i+1,0}$ is the maximum total size of small-item placements for the remaining capacity $c - s(A^L_{i+1,0})$. Therefore, we have

$$s(A^L_{i,t}) + \mathrm{OPT}^S_{i+1,0} \geq s(A^L_{i,t}) + s(A^*_{i,t} \cap S_\varepsilon) = s(A^*_{i,t}).$$

By Eq. (10) we thus have $s(A_{i,t}) > (1-\varepsilon)s(A^*_{i,t})$, as required. □

Assume that $\mathcal{P}'_{s,k}$ contains the packing whose total size is minimum in $\mathcal{P}'$. Then, by Eq. (7) we have

$$\begin{aligned} f(\mathcal{P}') &= f(\mathcal{P}'_{s,k}) \\ &\geq (1-\varepsilon)\min\{s(A_{k,0}), s(A_{k,t})\}. \end{aligned}$$

Therefore, by Lemma 3 and Eqs. (4) and (5) we have

$$\begin{aligned} f(\mathcal{P}') &> (1-\varepsilon)^2 \min\{s(A^*_{k,0}), s(A^*_{k,t})\} \\ &\geq (1-\varepsilon)^2 s(A^*_{\min}) \\ &> (1-2\varepsilon)s(A^*_{\min}) \\ &= (1-2\varepsilon)\mathrm{OPT}(A_0, A_t). \end{aligned} \tag{11}$$

By Eqs. (1), (6) and (11) we have

$$\mathrm{APX}(A_0, A_t) \geq f(\mathcal{P}') > (1-2\varepsilon)\mathrm{OPT}(A_0, A_t) = (1-\varepsilon')\mathrm{OPT}(A_0, A_t).$$

This completes the proof of Theorem 4. □

4 Concluding Remarks

In this paper, we showed that both SUBSET SUM RECONFIGURATION and MAXMIN SUBSET SUM RECONFIGURATION are strongly NP-hard. However, we do not know whether they are PSPACE-complete, or belong to NP. In particular, it is not clear whether the diameter of the configuration graph $\mathcal{C}$ can be bounded by a polynomial in the input size n.

In the ordinary KNAPSACK problem [6,17], each item is assigned not only a size but also a *profit*, and we wish to find a packing whose total profit is at least a given threshold. Consider the two reconfiguration problems for KNAPSACK, called KNAPSACK RECONFIGURATION and MAXMIN KNAPSACK RECONFIGURATION, which are defined similarly as SUBSET SUM RECONFIGURATION and MAXMIN SUBSET SUM RECONFIGURATION, respectively. Because they are generalizations of our reconfiguration problems for SUBSET SUM, the complexity and inapproximability results in Section 2 hold also for them. It remains open to obtain a PTAS for MAXMIN KNAPSACK RECONFIGURATION.

Acknowledgments We thank the referees for their helpful comments and suggestions. The first author's work is partially supported by JSPS KAKENHI Grant Number 22700001. The second author's work is supported in part by NSF grant CCF-1161626 and DARPA/AFOSR grant FA9550-12-1-0423.

References

1. H. L. Bodlaender and K. Jansen, On the complexity of scheduling incompatible jobs with unit-times, Proc. of MFCS 1993, LNCS 711 (1993) 291–300.
2. M. Bonamy, M. Johnson, I. Lignos, V. Patel and D. Paulusma, On the diameter of reconfiguration graphs for vertex colourings, Electronic Notes in Discrete Mathematics 38 (2011) 161–166.
3. P. Bonsma, The complexity of rerouting shortest paths, Proc. of MFCS 2012, LNCS 7464 (2012) 222–233.
4. P. Bonsma and L. Cereceda, Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances, Theoretical Computer Science 410 (2009) 5215–5226.
5. L. Epstein and A. Levin, On bin packing with conflicts, Proc. of WAOA 2006, LNCS 4368 (2006) 160–173.
6. M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.
7. P. Gopalan, P. G. Kolaitis, E. N. Maneva and C. H. Papadimitriou, The connectivity of Boolean satisfiability: computational and structural dichotomies. SIAM J. Computing 38 (2009) 2330–2355
8. J. Håstad, Clique is hard to approximate within $n^{1-\varepsilon}$, Acta Math. 182 (1999) 105–142.
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. R. A. Hearn and E. D. Demaine, Games, Puzzles, and Computation, A K Peters (2009)
11. T. Ito and E. D. Demaine, Approximability of the subset sum reconfiguration problem, Proc. of TAMC 2011, LNCS 6648 (2011) 58–69.
12. T. Ito, E. D. Demaine, N. J. A. Harvey, C. H. Papadimitriou, M. Sideri, R. Uehara and Y. Uno, On the complexity of reconfiguration problems, Theoretical Computer Science 412 (2011) 1054–1065.
13. T. Ito, M. Kamiński and E. D. Demaine, Reconfiguration of list edge-colorings in a graph, Discrete Applied Mathematics 160 (2012) 2199–2207.
14. T. Ito, K. Kawamura and X. Zhou, An improved sufficient condition for reconfiguration of list edge-colorings in a tree, IEICE Trans. on Information and Systems E95-D (2012) 737–745.
15. M. Kamiński, P. Medvedev and M. Milanič, Shortest paths between shortest paths, Theoretical Computer Science 412 (2011) 5205–5210.
16. M. Kamiński, P. Medvedev and M. Milanič, Complexity of independent set reconfigurability problems, Theoretical Computer Science 439 (2012) 9–15.
17. H. Kellerer, U. Pferschy and D. Pisinger, Knapsack Problems, Springer-Verlag, 2004.
18. K. Makino, S. Tamaki and M. Yamamoto, On the Boolean connectivity problem for Horn relations, Discrete Applied Mathematics 158 (2010) 2024–2030.
19. K. Makino, S. Tamaki and M. Yamamoto, An exact algorithm for the Boolean connectivity problem for k-CNF, Theoretical Computer Science 412 (2011) 4613–4618.
20. C. H. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.
21. U. Pferschy and J. Schauer, The knapsack problem with conflict graphs, J. Graph Algorithms and Applications 13 (2009) 233–249.
22. W. J. Savitch, Relationships between nondeterministic and deterministic tape complexities, J. Computer and System Sciences 4 (1970) 177–192.
23. V. V. Vazirani, Approximation Algorithms, Springer-Verlag, Berlin, 2001.