Everything I know about Subset Sum Reconfiguration

Robin Houston · July 18, 2013

This is a brief summary of everything I know about the subset sum reconfiguration problem as of the evening of Thursday the 18th of July.

Proposition 1. *Subset sum reconfiguration is weakly NP-hard.* (*Demaine–Ito*)

Proof. The ordinary subset sum problem may be reduced to subset sum reconfiguration. Given elements $a_1, \dots, a_m$ with target sum t, build an SSR instance with elements $a_1, \dots, a_m$ having the corresponding weights, t_1 and t_2 having weight t, threshold values $k = t$ and $c = 2t$, starting configuration t_1 and ending configuration t_2. At some point element t_1 must move from inside to outside: the first time it does, the inside set during the move must consist of some collection of a elements that sum to t. ☐

Proposition 2. *For all $n > 1$, there is an n-element SSR instance whose configuration graph has diameter $2n - 2$.*

Proof. We exhibit in terms of a parameter m:

- an instance with $2m + 2$ elements and a shortest solution with $4m + 2$ steps;
- if $m > 0$, an instance with $2m + 1$ elements and a shortest solution with $4m + 1$ steps.

The instance with $2m + 2$ elements has elements $a_1, \dots, a_m$, $b_1, \dots, b_m$, x and y. For the weights, let $w(a_i) = m + 1$ and $w(b_i) = m + 2$ for all i. Let $w(x) = w(y) = m(m + 2)$. The starting configuration is $a_1, \dots, a_m, x$ and the target configuration is $a_1, \dots, a_m, y$. For threshold values let $k = m(m + 2)$ and $c = 2m(m + 2)$.

The first time x moves, the weights of the remaining elements inside must sum to $m(m + 2)$. In particular, these weights must sum to m modulo $m + 1$, hence the inside elements at this point must include all the b's. Since the weight of x plus the b's equals the capacity c, the elements inside at this point must therefore be precisely all the b's. So all the a's and b's have to move twice, and x and y once each, for a total of $4m + 2$ moves.

To obtain the instance with $2m + 1$ elements, remove the element a_1. ☐

Remark 3. *The lower bound exhibited in Proposition 2 is optimal for $n < 8$. I have verified this by exhaustive enumeration using the program `ssr_graphs.py`.*

Proposition 4. *For all m, there is a $(5m + 3)$-element SSR instance that has an element that must move at least $2m + 1$ times in any solution sequence.*

Proof. The elements are $x_1, \dots, x_{2m+1}$ of weight 3, $a_1, \dots, a_{3m+1}$ of weight 2, and b of weight 1. Let $k = 6m$ and $c = 6m + 3$. The starting configuration is $x_1, \dots, x_{2m+1}$, and the target is $a_1, \dots, a_{3m+1}, b$. For $i = 0, 1, \dots, 2m$, consider the first move after which there are precisely $2m - i$ of the x elements inside. That means the a's and possibly b inside at this point must sum to $3i$. If i is odd then, since the a's all have even weight, b must be inside at this point; similarly if i is even then b must be outside. So element b moves at least $2m + 1$ times. ☐

The construction of Proposition 4 can be modified to give a quadratic lower bound for the diameter of the reconfiguration graph.

Proposition 5. *For all m, there is a $(6m+2)$-element SSR instance that requires $2m^2+6m+2$ moves.*

Proof. The elements are $x_1, \dots, x_{2m+1}$ of weight $3m$, $a_1, \dots, a_{3m+1}$ of weight $2m$, and $b_1, \dots, b_m$ of weight 1. Let $k = 6m^2$ and $c = 6m^2 + 3m$. The starting configuration is $x_1, \dots, x_{2m+1}$, and the target is $a_1, \dots, a_{3m+1}, b_1, \dots, b_m$. For $i = 0, 1, \dots, 2m$, consider the first move after which there are precisely $2m - i$ of the x elements inside. That means the a's and b's inside at this point must sum to $3im$. If i is odd then, since the a's all weigh $2m$, all the b's must be inside at this point; similarly if i is even then the b's must all be outside. So the b elements each move at least $2m + 1$ times. Therefore the total number of moves must be at least $(2m + 1) + (3m + 1) + (2m + 1)m$, which is equal to $2m^2 + 6m + 2$ as required. □

On the other hand, the instances constructed in Propositions 4 and 5 have a property that makes them easy to recognise as solvable – in linear time, if we assume the elements are already sorted by weight, or $O(n \log n)$ time if we have to sort them – without the need to construct an actual solution.

Lemma 6. *Suppose we have an SSR instance with element set E, threshold k and capacity c. By abuse of notation we shall also refer to the whole instance as E. Let a and z be elements of minimal and maximal weight respectively. If $w(z) > c - k$ or $w(a) < c - k - w(z) + 2$ then either E is trivially unsolvable or else there is an element $e \in E$ and a (smaller) instance with element set $E' = E - \{e\}$ that is equivalent to the original in the sense that E' is solvable iff E is.*

Proof. Let e be any element that has weight z.

If $w(z) > c - k$ then our element z cannot move. If z moves between the start and target packings, then E is trivially unsolvable. Otherwise we let $E' = E - \{z\}$. If z is inside in both the start and target packing, let the threshold and capacity of our new instance be $k' = k - w(z)$ and $c' = c - w(z)$, and remove z from the start and target configurations. If z is outside in both the start and target, let $k' = k$ and $c' = c$. Every solution of E' is a solution of E, and vice versa – by adding z to all packings in the solution sequence in the former case, or by literally using the same solution sequence in the latter.

If $w(a) < c - k - w(z) + 2$ then let $E' = E - \{a\}$. Look at whether or not a belongs to the start configuration: if it does, let $k' = k - w(a)$ and $c' = c$; if not, let $k' = k$ and $c' = c + w(a)$. In both cases we have $c' - k' = c - k + w(a)$. Any solution of E can be converted to a solution of E' just by removing element a from all packings in the sequence and eliminating adjacent duplicates. The interesting part is the converse, converting a solution of E' into a solution of E. Let $A'_0, \dots, A'_m$ be a sequence of packings that solves E'. We shall show how to insert element a, with some additional moves of a, to obtain a solution of E.

Let us say that a packing A' of E' *requires a* if $w(A') < k' + w(a)$, and that it *rejects a* if $w(A') > c' - w(a)$. Now we want to argue that in a solution sequence of E' there are never two adjacent packings such that one requires a and the other rejects it. The gap between the bound for rejecting a and the bound for accepting it is $c' - w(a) - (k' + w(a)) = c' - k' - 2w(a) = c - k - w(a)$, so it is not possible to cross this gap without the weight changing by at least $c - k - w(a) + 2$. Two adjacent packings differ in weight by at most $w(z)$, and since we are supposing that $w(a) < c - k - w(z) + 2$ it follows that they differ by less than $c - k - w(a) + 2$, as required.

Finally we can construct a solution sequence for E by inserting a where it is required and removing it where it is rejected. We describe an algorithm that reads $A'_0, \dots A'_m$ in order and outputs a solution of E. The algorithm keeps track of whether a is currently inside or outside the packing as it goes along. If a belongs to the start packing then initially a is inside, otherwise initially a is outside. Then for each i from 0 to m:

- if a is currently inside and A'_i rejects a, output A'_{i-1} followed by A'_i. (Note this case cannot occur when $i = 0$.) Let a be outside.

- if a is currently outside and A'_i requires a, output $A'_{i-1} \cup \{a\}$ followed by $A'_i \cup \{a\}$. (This case cannot occur when $i = 0$ either.) Let a be inside.
- otherwise, output A'_i if a is outside, and output $A'_i \cup \{a\}$ if a is inside.

□

In some cases we can iterate Lemma 6 to reduce the instance to the trivial one-element case.

Proposition 7. *Suppose we have an SSR instance with element set E, threshold k and capacity c. Suppose further that no element weighs more than $c - k$. Let $E = e_0, \dots, e_n$ in nondecreasing order by weight. If*

$$w(e_i) < c - k + \sum_{j<i} w(e_j) - w(e_n) + 2$$

for all i then the instance is solvable. Note that this condition does not mention the start or target packings, so in fact such instances have the property that any valid packing is reachable from any other.

Remark 8. *Let us say that an instance satisfying the property of Proposition 7 is completely reducible. The instances exhibited in Propositions 4 and 5 are completely reducible.*