Theory of Computing Systems (2021) 65:884–900
https://doi.org/10.1007/s00224-021-10034-z

The subset sum game revisited

Astrid Pieterse[1] · Gerhard J. Woeginger[2]

Accepted: 16 January 2021 / Published online: 13 February 2021

Abstract
We discuss a game theoretic variant of the subset sum problem, in which two players compete for a common resource represented by a knapsack. Each player owns a private set of items, players pack items alternately, and each player either wants to maximize the total weight of his own items packed into the knapsack or to minimize the total weight of the items of the other player. We show that finding the best packing strategy against a hostile or a selfish adversary is PSPACE-complete, and that against these adversaries the optimal reachable item weight for a player cannot be approximated within any constant factor (unless P=NP). The game becomes easier when the adversary is short-sighted and plays greedily: finding the best packing strategy against a greedy adversary is NP-complete in the weak sense. This variant forms one of the rare examples of pseudo-polynomially solvable problems that have a PTAS, but do not allow an FPTAS (unless P=NP).

Keywords Multi-agent optimization · Computational complexity · Approximability

1 Introduction

The subset sum game is a combinatorial game for two players A and B with perfect information. An instance of the game consists of $m + n$ items and a knapsack of capacity c. The A-items have weights $a_1, a_2, \ldots, a_m$ and belong to player A, while the B-items have weights $b_1, b_2, \ldots, b_n$ and belong to player B. Throughout we

An extended abstract of this paper has appeared in the Proceedings of the the 5th International Conference on Algorithmic Decision Theory (ADT-2017).

✉ Gerhard J. Woeginger
woeginger@algo.rwth-aachen.de

Astrid Pieterse
astrid.pieterse@informatik.hu-berlin.de

[1] Department of Computer Science, Humboldt University Berlin, Berlin, Germany

[2] Department of Computer Science, RWTH Aachen, Aachen, Germany

assume that every item weight is bounded by the knapsack capacity c. The players move alternately, and the instance specifies whether player A or player B makes the first move. In every move, the active player picks one of his items (which has not been picked in any earlier move) and puts it into the knapsack. As usual, an item can only be added to the knapsack, if the overall weight of all packed items does not exceed the knapsack capacity c. A player may pass on a move, in case none of his items fits. The game ends as soon as none of the remaining unpacked items fits into the knapsack.

We will always look at this game through the eyes of player A, whose goal is simply to maximize the total weight of A-items in the knapsack. Player B will be considered our adversary and enemy, who might behave in one of the following ways.

- **Hostile:** The objective of adversary B is to hurt player A as much as possible, and to minimize the total weight of A-items in the knapsack.
- **Selfish:** The objective of adversary B is to get as much profit for himself as possible, and hence to maximize the total weight of B-items in the knapsack.
- **Greedy:** The (short-sighted) objective of adversary B is to pack in every single move a B-item of largest possible weight.

While the behavior of the greedy adversary is easy to understand (and easy to predict), the behavior of the two other adversaries needs a more precise mathematical definition that considers the game in extensive form. The hostile adversary and the selfish adversary are defined via the underlying game tree; this tree is an acyclic directed graph whose vertices correspond to the possible game configurations. A configuration is fully specified by the current contents of the knapsack. For every possible move in the game, the game tree contains a corresponding arc between the two corresponding configurations. The initial configuration (with empty knapsack) is denoted p_0. Final configurations (where none of the remaining unpacked items fits into the knapsack) have no out-going arcs.

Let us first specify the hostile adversary against some fixed (deterministic) strategy σ of player A. For evaluating a final configuration p, we look at the contents of the knapsack and use $a(p)$ to denote the total weight of packed A-items. For evaluating a configuration q somewhere in the middle of the game, we enumerate all configurations $q_1, \dots, q_k$ that can be reached from q in a single move. If it is player A's turn then his strategy σ will lead him to a well-defined configuration q_j, and we define $a(q) = a(q_j)$. If it is player B's turn then $a(q) = \min_i a(q_i)$. When the game terminates, player A will end up with a total weight of $a(p_0)$.

Next let us specify the selfish adversary against a fixed (deterministic) strategy σ of player A. For evaluating a final configuration p, we denote by $a(p)$ the total weight of packed A-items and by $b(p)$ the total weight of packed B-items. For evaluating a configuration q in the middle of the game, let $q_1, \dots, q_k$ denote the configurations that can be reached from q in a single move.

- If it is player A's turn, then strategy σ leads him from configuration q to a well-defined configuration q_j. We set $a(q) = a(q_j)$ and $b(q) = b(q_j)$.
- If it is player B's turn, then $b(q) = \max_i b(q_i)$. To make the game determinate, we furthermore set $a(q) = \max_k a(q_k)$ with $k \in \arg\max_i b(q_i)$.

This means that whenever the adversary may choose between several moves that yield the same profit for himself, he will always pick the move that is best for player A. We stress that all our results can be carried over to the variant where the selfish adversary picks the move that is worst for player A. When the game terminates, player A will reach a weight of $a(p_0)$ and player B will reach a weight of $b(p_0)$.

In this paper, we will study certain algorithmic questions centered around such subset sum games. The central algorithmic decision problem is defined as follows:

Instance: A knapsack of capacity c; positive integer weights $a_1, a_2, \ldots, a_m$ and $b_1, b_2, \ldots, b_n$; a starting player (A or B); a positive integer bound α; an adversary type (hostile, selfish, greedy).

Question: Does player A have a deterministic strategy that allows him to pack A-items of total weight at least α into the knapsack, if he plays the game against a player B of the given adversary type?

The resulting three variants of the subset sum game will be denoted SSG-hostile, SSG-selfish, and SSG-greedy. The respective optimization versions of the game ask to find the largest possible weight α^* that player A can pack, and the respective approximation versions ask to find an approximation of this largest possible weight α^*.

Known and related results Motivated by certain applications in the area of operations research, Darmann, Nicosia, Pferschy & Schauer [5] introduced the subset sum game variant against the selfish adversary. They analyze a number of intuitive strategies for the game, that are either pure greedy approaches or greedy-based strategies that use some kind of bounded look-ahead. Among other results, they show that a certain greedy strategy reaches a worst case ratio of 2 when it is applied against a selfish adversary. We stress that this result assumes an oracle-access to the selfish adversary; it works move by move through the entire game, and guarantees that at the very end the weight of the packed item set is at least 50% of the weight reached by an optimal strategy. As the selfish adversary has high computational complexity, this approach does not yield a polynomial time algorithm in the classical sense, but just a policy that can be applied while playing the game. In strong contrast to this, in the current paper we will analyze these packing games by purely looking at the given instance, and we do not assume cheap oracle-access to the computationally expensive behavior of the adversary.

The combinatorics of the subset sum game is far from trivial and sometimes shows a quite counter-intuitive behavior. For instance, our intuition tells us that it should always be better to pack large items before small items. However, in [5] it is demonstrated that for certain instances of SSG-selfish it might be optimal for player A to first pack some smaller items and only later on pack large items. Another example for this phenomenon can be found in our Example 2.2 presented in Section 2.

Caprara, Carvalho, Lodi & Woeginger [2, 3] study three packing games that are centered around bilevel variants of the knapsack problem. These games consist of only two rounds; the first player (called leader) packs some items in the first round, and then the second player (called follower) reacts by packing some items in the second round. The objective value of the leader depends on the profits of all items in the final packing. All bilevel packing games considered in [2, 3] are Σ_2^p-complete, most of them are inapproximable, and only one of them has a PTAS. Further knapsack variants with game-theoretic flavor have been studied by Brotcorne, Hanafi & Mansi [1], Carvalho, Lodi & Marcotte [4], Dempe & Richter [6], Fischer & Woeginger [7], Nicosia, Pacifici & Pferschy [9], Pferschy, Nicosia & Pacifici [10], and Qiu & Kern [11].

Our results We provide a complete picture of the computational complexity and the approximability landscape of the subset sum game against the three adversaries types.

- The games against the hostile and selfish adversaries are both PSPACE-complete. Unless $P = NP$, these games do not allow any polynomial time approximation algorithm with constant worst case guarantee.
- The game against the greedy adversary is weakly NP-hard and pseudo-polynomially solvable. This game yields one of the rare pseudo-polynomially solvable problems that have a PTAS, but do not allow an FPTAS.

The rest of the paper is organized as follows. Section 2 states several technical observations. Section 3 proves the inapproximability results for SSG-hostile and SSG-selfish (no constant factor approximation) and SSG-greedy (no FPTAS). Section 4 pinpoints the computational complexity of SSG-greedy, and Section 5 derives a PTAS for SSG-greedy. Finally Section 6 derives the PSPACE-completeness of SSG-hostile and SSG-selfish.

2 Technical Preliminaries

When we introduced the subset sum game in the first paragraph of this paper, we stated that every instance of the game explicitly specifies whether player A or player B makes the first move. The following observation shows that from the computational complexity point of view, there is no difference whether player A or player B starts the game. In other words, this observation allows us to shift the first move from player A to player B and vice versa.

Observation 2.1 *For the games SSG-hostile, SSG-selfish, SSG-greedy, the computational complexity of the variant where player A has the first move coincides with the computational complexity of the variant where the adversary B has the first move.*

Proof Take an arbitrary instance of the game where one of the two players is designated to make the first move. Increase the knapsack capacity to $c' := 2c + 1$, create one additional A-item of weight $c + 1$ and one additional B-item of weight $c + 1$ to the item lists, and allow the other (non-designated) player to make the first move. It is easily seen that in the new game, the only good move for the other player (no matter whether he is hostile, selfish, or greedy) consists in packing the new item of weight

$c + 1$. Afterwards, the remaining knapsack capacity drops down to c and we are back at the original instance of the game with the same designated player to move. □

The definitions of the two games SSG-hostile and SSG-selfish look very similar to each other, and it might not be clear at first sight that these two definitions actually yield two *different* games. The following instance illustrates that these two games indeed are different.

Example 2.2 Consider the subset sum game with B-items 4,4,4,7,7, with A-items 6,6,11, and with a knapsack of capacity $c = 24$. The first move belongs to the adversary B.

Figure 1 lists the full game tree for the game in Example 2.2. Every directed arc describes a possible move. The label of the arc states the active player (A or B), followed by the weight of the packed item; a dash indicates that the player passes (as none of his remaining items can be packed). The number pairs in the rectangular boxes indicate the values of the current configuration for the two players. The first number gives the highest reachable packed weight for player A. The second number gives the packed weight for player B; it turns out (and the reader may want to verify this) that with the sole exception of the root configuration, the hostile adversary and the selfish adversary assign equal values to every configuration.

In the root configuration, the adversary B has to choose between packing an item of weight 4 (which eventually leads to profits of 12 for both players) and packing

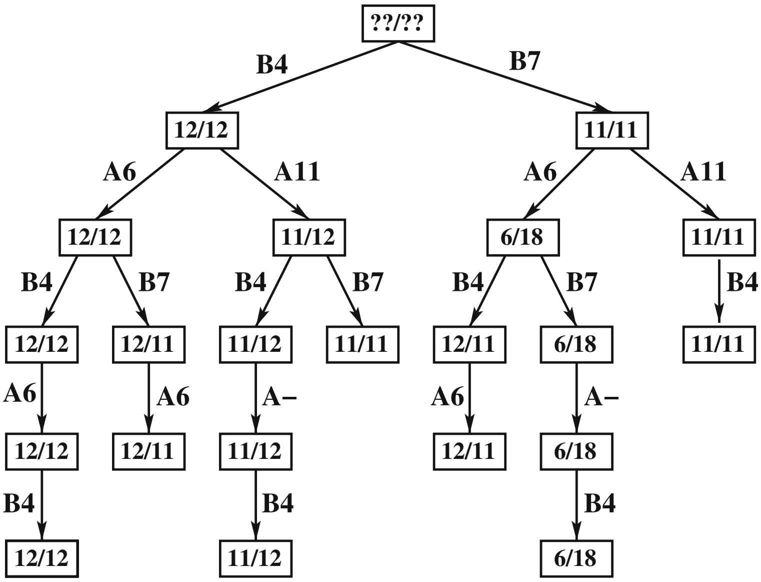

Fig. 1 The game tree for the instance in Example 2.2

an item of weight 7 (which eventually leads to profits of 11 for both players). Hence player A will make profit 11 against the hostile adversary and profit 12 against the selfish adversary.

3 Inapproximability results

In this section, we derive two inapproximability results for the subset sum game. Both results are derived by means of reductions from the NP-complete Particion problem; see Garey & Johnson [8].

Problem: Particion
Instance: An integer $U \geq 3$; positive integers $u_1, \ldots, u_t$ with $\sum_{i=1}^{t} u_i = 2U$.
Question: Does there exist a subset $J \subseteq \{1, \ldots, t\}$ such that $\sum_{i \in J} u_i = U$?

3.1 The hostile and the selfish adversary

Our first reduction from Particion will simultaneously settle the inapproximability for the hostile and for the selfish adversary. Suppose for the sake of contradiction that the optimal value α^* in SSG-hostile or in SSG-selfish can be approximated in polynomial time within a factor of r for some fixed real number r with $0 < r < 1$. Fix an integer R with $R > 1/r$. We take an arbitrary instance of Particion, and construct the following instance of SSG-hostile and SSG-selfish from it.

- For $i = 1, \ldots, tR$ there is an A-item of weight 1.
- For $i = 1, \ldots, t$, there is a B-item of weight $tR\,u_i$.
- The capacity of the knapsack is $c = tRU + t$.
- Player A has the first move.

Lemma 3.1 *If the* Particion *instance has answer YES, then player A packs a total weight of at most t against a hostile or selfish adversary.*

Proof Let $J \subseteq \{1, \ldots, t\}$ with $\sum_{i \in J} u_i = U$. During the first t moves, player A packs an A-weight of exactly t, while the adversary B packs the items of weight $tR\,u_i$ with $i \in J$ into the knapsack. As the B-weight in the knapsack is $tR\,U$, the knapsack is full and the game terminates. □

Lemma 3.2 *If the* Particion *instance has answer NO, then player A can pack a total weight of tR.*

Proof The B-weight in the knapsack is always a multiple of tR. The largest such multiple is $tR\,U$, which cannot be reached as the Particion instance has answer NO. Hence the B-weight is at most $tR\,(U-1)$, and the remaining capacity left for player A's items is at least $c - tR\,(U-1) \geq tR$. □

According to Lemmas 3.1 and 3.2, an approximation algorithm with worst case guarantee $r > 1/R$ can distinguish in polynomial time between YES-instances and NO-instances of the PARTICION problem.

Theorem 3.3 *Unless $P = NP$, the problems SSG-hostile and SSG-selfish do not allow any polynomial time approximation algorithm whose worst case guarantee is a fixed real number (that is independent of the instance size).*

3.2 The greedy adversary

Now let us turn to our second reduction, to prove that the game against the greedy adversary has no FPTAS assuming $P \neq NP$. We take an arbitrary instance of PARTICION, and construct the following instance of SSG-greedy from it.

- For $i = 1, \ldots, t$, there is one corresponding dummy A-item of weight $4U$ and one corresponding standard A-item of weight $4U + u_i$. Furthermore, there is one special A-item of weight $3U - 1$.
- For $i = 1, \ldots, t$ there is a B-item of weight $3U$.
- The capacity of the knapsack is $c = (7t + 1)U - 1$.
- Player A has the first move.

Lemma 3.4 *If the* PARTICION *instance has answer YES, then player A can pack a total weight of* $(4t + 4)U - 1$.

Proof Let $J \subseteq \{1, \ldots, t\}$ with $\sum_{i \in J} u_i = U$. Then in his first t moves, player A packs the $|J|$ standard items of weight $4U + u_i$ with $i \in J$ together with $t - |J|$ dummy items of weight $4U$. Consider the time point just after the t-th move of A (and just before the t-th move of B): then the total A-weight in the knapsack equals $(4t + 1)U$, the total B-weight in the knapsack is $(3t - 3)U$ as all B-items have identical weight, and the remaining knapsack capacity is $3U - 1$. Player B must pass, as none of his items fits. Player A packs his special item $3U - 1$ in his $(t + 1)$-th move and thereby reaches a total packed A-weight of $(4t + 4)U - 1$. □

Lemma 3.5 *If the* PARTICION *instance has answer NO, then player A can never pack a total weight above* $(4t + 2)U$.

Proof Consider the time point just after the t-th move of player A (and before the t-th move of player B), and let W denote the total A-weight in the knapsack at that moment. We distinguish three cases.

First assume $W \leq (4t + 1)U - 1$. Then B will pack his t-th item in his t-th move and bring the packed B-weight up to $3tU$. Then A will never bring the packed A-weight above $c - 3tU = (4t + 1)U - 1$ in this case.

In the second case assume $W \geq (4t + 1)U + 1$. Note that $W \leq (4t + 2)U$, which is the weight of the t heaviest standard A-items. Player B cannot pack his t-th item and the packed B-weight stays at $(3t - 3)U$. The remaining knapsack capacity is

$c - (3t-3)U - W \le 3U - 2$. As no further A-item fits, the packed A-weight cannot exceed $(4t+2)U$ in this case.

In the third case assume $W = (4t+1)U$. The special A-item $3U - 1$ cannot contribute to W, as the total weight of the $t-1$ heaviest standard items is at most $(4t-2)U$. Let $J \subseteq \{1, \dots, t\}$ denote the set of all i for which the standard A-item $4U + u_i$ is in the knapsack. It is easy to see that this implies $\sum_{i \in J} u_i = U$. Hence the PARTICION instance has answer YES, so that the third case leads to a contradiction. □

Now suppose for the sake of contradiction that SSG-greedy allows an FPTAS. For any instance of SSG-greedy and for any $\varepsilon > 0$, the FPTAS takes a running time that is polynomially bounded in the instance size and in $1/\varepsilon$, and then outputs an approximation value α that is at least $1-\varepsilon$ times the optimal value α^*. We execute this FPTAS with precision $\varepsilon = 1/(4t+4)$ on the instance constructed above. The resulting running time is polynomially bounded in the instance size. If the PARTICION instance has answer YES, then by Lemma 3.4 we have

$$\alpha \;\ge\; (1-\varepsilon)\,\alpha^* \;\ge\; \frac{4t+3}{4t+4}\,((4t+4)U - 1) \;>\; (4t+3)U - 1.$$

If the PARTICION instance has answer NO, then by Lemma 3.5 we have $\alpha \le \alpha^* \le (4t+2)U$. Hence, by analyzing the value α generated by the FPTAS, we would be able to separate in polynomial time the YES-instances from the NO-instances of the PARTICION problem.

Theorem 3.6 *Unless $P = NP$, problem SSG-greedy does not possess an FPTAS.*

4 Complexity of the game against the greedy adversary

As problem PARTICION is weakly NP-hard, the reduction in Section 3.2 implies that also problem SSG-greedy is weakly NP-hard. Our main goal in this section is to show that SSG-greedy is pseudo-polynomially solvable. Note that for SSG-greedy, a strategy of player A is fully specified by the ordered list of packed A-items. The following lemma will be useful.

Lemma 4.1 *(Darmann & al [5]) Against the greedy adversary, there exists an optimal strategy for player A that packs the items in non-increasing order of weight.*

Proof Consider an optimal strategy of player A that (without loss of generality) packs the items in order $a_1, a_2, \dots, a_s$. If the item weights in this list are not non-increasing, consider the smallest index k with $a_k < a_{k+1}$. If we swap items k and $k+1$ in the list, the reactions of the greedy adversary will not change. We may repeat this swapping step until the weights are non-increasing. □

Let us assume that the A-items are ordered as $a_1 \geq a_2 \geq \cdots \geq a_m$ and that the B-items are ordered as $b_1 \geq b_2 \geq \cdots \geq b_n$. By Lemma 4.1 there exists an optimal strategy for player A that packs items in order of non-increasing weight. For $i = 0, \ldots, m+1$, for $j = 0, \ldots, n+1$, and for $W_A = 0, \ldots, c$ and $W_B = 0, \ldots, c$, we introduce a corresponding state $[i, j, W_A, W_B]$ that encodes the following configuration that might potentially arise in some run of the game:

> In configuration $[i, j, W_A, W_B]$ player A is to move next. In his last move, player A has packed the i'th A-item (where $i = 0$ means that no A-item has been packed yet, and where $i = m+1$ means that no further A-item fits). Similarly, the greedy adversary B has packed the j'th B-item in his last move. The total weight of the packed A-items equals W_A and the total weight of the packed B-items equals W_B.

For two configurations $S' = [i', j', W'_A, W'_B]$ and $S = [i, j, W_A, W_B]$, it is easy to check whether S can be reached from S' by means of a move of player A and a following counter-move by player B. In this case we must have $i > i'$, $j > j'$, $W_A = W'_A + a_i$ and $W_B = W'_B + b_j$. Furthermore $W_A + W_B \leq c$, and b_j must be the largest available B-item of size $\leq c - (W'_A + W'_B + a_i)$.

This suggests the following approach. We generate all possible states $[i, j, W_A, W_B]$ where the variables i, j, W_A, W_B may take all possible values from their ranges as listed above. Altogether this yields $O(mnc^2)$ states. Then we compute for every pair S and S' of states, whether S can be reached from S'. Next, we determine (by a standard depth-first-search traversal) all states that are reachable from the starting configuration of the game. The maximum value W_A among all reachable states yields the largest possible weight α^* that player A can pack. We summarize the findings of this section.

Theorem 4.2 *Problem SSG-greedy is weakly NP-complete and solvable in pseudo-polynomial time $O(m^2n^2c^4)$.*

The literature contains routine approaches that automatically translate certain types of pseudo-polynomial algorithms into an FPTAS. These pseudo-polynomial algorithms are usually based on dynamic programs, and the FPTAS rounds and simplifies the state space of the DP so that the running time becomes polynomial, whereas the error introduced by the rounding can be kept small; see for instance Woeginger [14]. Unfortunately, the above pseudo-polynomial algorithm for SSG-greedy does not fall into that category and does not allow such an automatic translation. The main obstacle is the strict capacity constraint on the values W_A and on W_B, which would only allow us to round one of these two values (if we want to be able to control the introduced error), so that the running time does not become truly polynomial. Note furthermore that Theorem 3.6 explicitly excludes the existence of an FPTAS (unless $P = NP$). The following section derives the strongest approximation result possible under these circumstances.

5 The Approximation Scheme

In this section, we derive a polynomial time approximation scheme (PTAS) for problem SSG-greedy. Our PTAS is based on the standard approaches from the literature for approximating knapsack and subset sum problems; see for instance Shmoys & Williamson [13]. Against the greedy adversary, the large A-items are handled by the usual enumeration approach, whereas the smaller A-items are packed with a greedy strategy. Throughout this section we will assume without loss of generality that player A has the first move. The following technical lemma will be crucial for the error analysis of the PTAS.

Lemma 5.1 *Consider an instance of SSG-greedy, and let $a_{\max}$ denote the weight of the largest A-item. Let α^* denote the total weight player A packs under an optimal strategy, and let α^G denote the total weight player A packs when he applies the greedy strategy. Then*

$$\alpha^* - a_{\max} \;<\; \alpha^G.$$

Proof Consider the first moment in time, where the greedy strategy for player A cannot pack the largest available A-item (whose weight we denote by a^G_{q+1}). Let $a^G_1 \ge a^G_2 \ge \cdots \ge a^G_q$ denote the weights of the packed A-items at that moment, and let $b^G_1 \ge b^G_2 \ge \cdots \ge b^G_q$ denote the weights of the packed B-items at that moment. Note that $a_{\max} \le c$ implies $q \ge 1$. Furthermore we have

$$a_{\max} \;\ge\; a^G_{q+1} \;>\; c - \sum_{i=1}^{q} a^G_i - \sum_{i=1}^{q} b^G_i \;\ge\; c - \alpha^G - \sum_{i=1}^{q} b^G_i. \tag{1}$$

Now we turn to another run of the game. Let $a^*_1 \ge a^*_2 \ge \cdots$ denote the weights of the packed A-items if player A follows an optimal strategy; let $b^*_1 \ge b^*_2 \ge \cdots$ denote the weights of the B-items packed by the greedy adversary B against the optimal strategy for player A, and let β^* denote the overall weight of these items. We claim that

$$\beta^* \;\ge\; \sum_{i=1}^{q} b^G_i. \tag{2}$$

Indeed, consider the smallest index r that satisfies $b^G_r \ne b^*_r$. We may assume $r \le q$, as otherwise the inequality in (2) trivially holds. Note that $\sum_{i=1}^{r-1} b^G_i = \sum_{i=1}^{r-1} b^*_i$ by the definition of r. Since $a^G_1, \ldots, a^G_r$ are the r largest available A-item weights, we furthermore have $\sum_{i=1}^{r} a^G_i \ge \sum_{i=1}^{r} a^*_i$. This yields

$$c - \sum_{i=1}^{r} a^G_i - \sum_{i=1}^{r-1} b^G_i \;\le\; c - \sum_{i=1}^{r} a^*_i - \sum_{i=1}^{r-1} b^*_i. \tag{3}$$

Consider the moment where the greedy adversary B in his game against the optimal strategy of player A decides to pack item weight b^*_r. By (3), at that moment the remaining knapsack capacity is large enough to accommodate the B-item of weight b_r^G. As $b_r^G \neq b_r^*$ this implies

$$b_r^G \ < \ b_r^*. \tag{4}$$

Next consider the moment where the greedy adversary B in his game against the greedy strategy of player A decides to pack the B-item of weight b_r^G. At that moment, the remaining knapsack capacity is too small to accommodate the also available but larger B-item of weight b_r^*. Since this remaining knapsack capacity is at least $\sum_{i=r}^q b_i^G$, we conclude $\sum_{i=r}^q b_i^G < b_r^*$. From this we derive

$$\beta^* \ \geq \ \sum_{i=1}^{r} b_i^* \ = \ \sum_{i=1}^{r-1} b_i^* + b_r^* \ = \ \sum_{i=1}^{r-1} b_i^G + b_r^* \ > \ \sum_{i=1}^{r-1} b_i^G + \sum_{i=r}^{q} b_i^G, \tag{5}$$

which completes the proof of (2). Finally, by combining $\alpha^* + \beta^* \leq c$ with (1) and (2) we also complete the proof of the lemma.

We turn to the description of the PTAS. We assume that the A-items are ordered as $a_1 \geq a_2 \geq \cdots \geq a_m$ and the B-items are ordered as $b_1 \geq b_2 \geq \cdots \geq b_n$. We will furthermore assume by Lemma 4.1 that both players pack their items in order of non-decreasing weight. Let ε with $0 < \varepsilon < 1$ be a small positive real number; for the sake of simplicity we will assume that the reciprocal value $1/\varepsilon$ is integer.

Let S be a subset of A-items with $|S| \leq 1/\varepsilon$, and let a_k denote the lowest weight A-item in S (that is, the item with largest index in S). We run the following two-phase procedure on S by simulating a game of player A against the greedy adversary.

- The first phase consists of the first $|S|$ moves. Player A packs the items in S in non-increasing order of weight.
- The second phase consists of the remaining moves. Player A ignores all A-items with indices up to k, and plays greedily with the A-items with indices at least $k + 1$.

In case we cannot pack all the items of S in the first phase, we call set S infeasible and ignore it. Otherwise, set S is feasible and the procedure yields a certain total packed weight for player A that we denote $\alpha(S)$. The PTAS outputs the maximum value $\alpha(S)$ over all feasible sets S.

Now consider an optimal strategy for player A that packs a sequence of items with weights $a_1^* \geq a_2^* \geq \cdots \geq a_t^*$. If $t \leq 1/\varepsilon$, then the items in the sequence form a feasible set S^*. In this case our PTAS analyzes S^* in the first phase, and eventually outputs the optimal objective value $\alpha^* = \alpha(S^*)$. If $t > 1/\varepsilon$, then define S^* as the set of the $1/\varepsilon$ largest items in the sequence; note that S^* is a feasible set. What does our two-phase procedure do with S^*?

- The first phase simply follows the first $1/\varepsilon$ moves of the optimal strategy: player A packs the first $1/\varepsilon$ items from the optimal sequence, and the greedy adversary picks the largest B-items that fit. As player A altogether packs $1/\varepsilon$ items during the first phase, the smallest item weight in S^* is at most $\varepsilon\alpha^*$.
- The second phase is only played with the A-items that are smaller than the smallest item in S^*, and hence have weight at most $\varepsilon\alpha^*$. The knapsack capacity is the remaining capacity that has not been used in the first phase.

Let α_1^+ denote the total A-weight packed during the first phase, and let α_2^+ denote the total A-weight packed during the second phase. Let α_1^* denote the weight of the items in S^*, and let $\alpha_2^* = \alpha^* - \alpha_1^*$ denote the weight of the remaining items in the optimal packing for A. Clearly $\alpha_1^* = \alpha_1^+$, and Lemma 5.1 yields that $\alpha_2^* - \varepsilon\alpha^* \le \alpha_2^+$. These two inequalities imply

$$\alpha(S^*) \;=\; \alpha_1^+ + \alpha_2^+ \;\ge\; \alpha_1^* + (\alpha_2^* - \varepsilon\alpha^*) \;=\; (1-\varepsilon)\,\alpha^*.$$

As the PTAS yields a strategy with profit at least $\alpha(S^*)$, this yields the desired approximation guarantee. Up to polynomial factors, the time complexity of the PTAS is proportional to the number of analyzed sets S, which is bounded by $m^{1/\varepsilon}$. This completes our analysis and yields the following theorem.

Theorem 5.2 *Problem SSG-greedy has a polynomial time approximation scheme.*

6 The PSPACE Completeness Result

In this section we determine the computational complexity of the problem variants with a hostile or selfish adversary. Both problems are PSPACE-complete, and the proof is done by means of a polynomial time reduction from the following variant of the quantified satisfiability problem.

Problem: QUANTIFIED 1-IN-3-SAT
Instance: Two sets $X = \{x_1, \ldots, x_s\}$ and $Y = \{y_1, \ldots, y_s\}$ of Boolean variables. A Boolean formula ϕ over $X \cup Y$ in conjunctive normal form with clauses $C_1, \ldots, C_t$; every (disjunctive) clause C_j consists of exactly three literals.
Question: Assuming that a clause in ϕ is satisfied if and only if it contains *exactly* one true literal, is $\forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots \forall x_s \exists y_s\ \phi$ true?

As usual, we interpret a quantified formula as a game between a universal player (controlling the universal quantifiers) and an existential player (controlling the existential quantifiers). The goal of the existential player is that in the end formula ϕ evaluates to true, whereas the universal player wants to prevent this.

Lemma 6.1 QUANTIFIED 1-IN-3-SAT *is PSPACE-complete.*

Proof The reduction is done from the classic QUANTIFIED 3-SATISFIABILITY problem, whose PSPACE-completeness was established by Schaefer [12]:

Problem: QUANTIFIED 3-SATISFIABILITY
Instance: Two sets $X' = \{x_1', \ldots, x_r'\}$ and $Y' = \{y_1', \ldots, y_r'\}$ of Boolean variables. A Boolean formula ϕ' over $X' \cup Y'$ in conjunctive normal form, where every (disjunctive) clause consists of exactly three literals.
Question: Assuming that a clause in ϕ is satisfied if and only if it contains *at least* one true literal, is $\forall x_1' \exists y_1' \forall x_2' \exists y_2' \ldots \forall x_r' \exists y_r'\ \phi'$ true?

We start from an arbitrary instance of QUANTIFIED 3-SATISFIABILITY, and translate it into an equivalent instance of QUANTIFIED 1-IN-3-SAT. The new instance contains all variables in X' and Y'. For every clause $(x \vee y \vee z)$ in ϕ', we introduce twelve new variables $a_1, a_2, a_3, b_1, b_2, b_3$, and $d_1, d_2, d_3, d_4, d_5, d_6$ together with the following four clauses: $(\neg x \vee a_1 \vee b_1)$, $(\neg y \vee a_2 \vee b_2)$, $(\neg z \vee a_3 \vee b_3)$, $(a_1 \vee a_2 \vee a_3)$. Note that whenever at least one of x, y, z is true, there is a way of choosing a_1, a_2, a_3 and b_1, b_2, b_3, so that each of the four clauses contains exactly one true literal; and vice versa, whenever each of the four clauses contains exactly one true literal, then at least one of x, y, z must be true. This is essentially the classic reduction for 1-IN-3-SAT of Schaefer; see Garey & Johnson [8]. Note furthermore that the d_i are dummy variables, that do not occur in any clause.

The string of quantifiers in the QUANTIFIED 1-IN-3-SAT instance starts with the string $\forall x_1 \exists y_1 \ldots \forall x_s \exists y_s$, followed by an alternating string of universally quantified dummy variables d_i and existentially quantified variables a_i and b_i. The formula ϕ consists of all the four-tuples of clauses introduced above. It can be seen that the considered instance of QUANTIFIED 3-SATISFIABILITY has answer YES, if and only if the newly constructed instance of QUANTIFIED 1-IN-3-SAT has answer YES. □

Now we turn to the PSPACE-hardness proofs for SSG-hostile and SSG-selfish. We present a single reduction that simultaneously settles both cases. We start from an arbitrary instance of QUANTIFIED 1-IN-3-SAT with $2s$ variables $x_1, \ldots, x_s$ and $y_1, \ldots, y_s$ and with t clauses, and we construct the following subset sum game from it.

All item weights will be specified in decimal representation, and will all have at most $2s + t + 3$ digits. The first $2s$ digits (in the high positions) form the so-called *variable piece*; the $(2i - 1)$th such digit from the left corresponds to the Boolean variable x_i, and the $(2i)$th digit from the left corresponds to the Boolean variable y_i. The three digits after the variable piece form the so-called *middle piece*. The last t digits (in the low positions) form the so-called *clause piece*; the jth such digit from the left in the clause piece corresponds to clause C_j. The item weights are defined as follows.

- For every literal $\ell \in \{x_i, \overline{x_i}\}$ with $1 \leq i \leq s$ there is a corresponding item $B(\ell)$. The decimal representation of its weight has a 1-digit in the position corresponding to x_i in the variable piece. The middle piece digits and all other digits in the variable piece are 0. Furthermore, the clause piece has a digit 1 in the position corresponding to clause C_j if ℓ occurs in C_j; otherwise, the digit is 0.
- Symmetrically, for every literal $\ell \in \{y_i, \overline{y_i}\}$ with $1 \leq i \leq s$ there is a corresponding item $A(\ell)$. The decimal representation of its weight has a 1-digit in the position corresponding to y_i in the variable piece. The middle piece digits and all other digits in the variable piece are 0. Furthermore, the clause piece has a digit 1 in the position corresponding to clause C_j if ℓ occurs in C_j; otherwise, the digit is 0.
- For every variable x_i and every variable y_i, there is a corresponding threat item $T(x_i)$ respectively $T(y_i)$. The decimal representation of the weight has a 1-digit

in the position corresponding to that variable in the variable piece. All other digits are 0.

- Furthermore, there are four verification items V_1, V_2, V_3, U with weights $w(V_1)$, $w(V_2)$, $w(V_3)$, $w(U)$. All digits in the variable pieces of these item weights are 0. The middle pieces of $w(V_1)$, $w(V_2)$, $w(V_3)$, $w(U)$ respectively are 111, 100, 010, 011. All other digits in the four decimal representations are 0, with the sole exception of the lowest digit in the weight of V_1, which is set to 1.

We stress the following property of our construction: if we add up the weights of any subset of items in decimal, then there will be no carry overs from one position to the next one. The knapsack capacity c has digits 1 in all $2s + t + 3$ positions. The goal weight α of player A is defined as follows: The decimal representation of α has a digit 1 in the even positions in the variable piece (that is, in the positions corresponding to variables y_i), a middle piece 011, and digits 0 in the remaining positions. Finally, we define an auxiliary number β (with digits 1 in middle piece and clause piece, and 0s in the variable piece). The decimal representations of all the introduced numbers are summarized in Figure 2.

All items $A(y_i)$ and $A(\overline{y_i})$, all threat items $T(x_i)$, and the verification item U belong to player A. All items $B(x_i)$ and $B(\overline{x_i})$, all threat items $T(y_i)$, and the verification items V_1, V_2, V_3 belong to player B. Player B has the first move. The question is to decide whether player A can pack a total weight of at least α (in which case we say that player A wins the game).

Lemma 6.2 *Assume that player A and the (hostile or selfish) adversary B both apply optimal strategies. Then in round $2i - 1$ (with $1 \le i \le s$), the adversary B either*

	Owner	Variable piece	Middle piece	Clause piece
		$x_1\, y_1 \dots x_i\, y_i \dots x_s y_s$		$C_1 \dots C_j \dots C_t$
$B(x_i)$	B	0 0 … 1 0 … 0 0	000	0…010…00
$B(\overline{x_i})$	B	0 0 … 1 0 … 0 0	000	0…010…00
$A(y_i)$	A	0 0 … 0 1 … 0 0	000	0…010…00
$A(\overline{y_i})$	A	0 0 … 0 1 … 0 0	000	0…010…00
$T(x_i)$	A	0 0 … 1 0 … 0 0	000	0…000…00
$T(y_i)$	B	0 0 … 0 1 … 0 0	000	0…000…00
V_1	B	0 0 … 0 0 … 0 0	111	0…000…01
V_2	B	0 0 … 0 0 … 0 0	100	0…000…00
V_3	B	0 0 … 0 0 … 0 0	010	0…000…00
U	A	0 0 … 0 0 … 0 0	011	0…000…00
c	–	1 1 … 1 1 … 1 1	111	1…111…11
α	–	0 1 0 1 … 0 1 0 1	011	0…000…00
β	–	0 0 … 0 0 … 0 0	111	1…111…11

Fig. 2 Summary of the decimal representation of the item weights

packs item $B(x_i)$ or item $B(\overline{x_i})$. In round $2i$ (with $1 \le i \le s$), player A either packs item $A(y_i)$ or item $A(\overline{y_i})$.

Proof We assume inductively that up to round $2k$ (with $0 \le k \le s-1$), the statement holds and both players have followed the described moves. Let d denote the current contents of the knapsack at th beginning of round $2k$. Then the decimal representation of d has digits 1 in the first $2k$ positions of the variable piece. This implies that none of the remaining items $A(y_i)$, $A(\overline{y_i})$, $T(y_i)$, $B(x_i)$, $B(\overline{x_i})$, $T(x_i)$ with $1 \le i \le k$ can be picked anymore: their weight is so large that they do not fit into the remaining empty part of the knapsack. All other items (the four verification items and the items corresponding to variables x_i or y_i with $i \ge k+1$) are small enough to fit.

First we consider a hostile adversary B. If B neither packs item $B(x_k)$ nor $B(\overline{x_k})$ in round $2k+1$, then player A can react in the next round by packing the threat item $T(x_k)$. This immediately brings A's weight above the threshold α, so that A wins the game. Hence the hostile B must pack $B(x_k)$ or $B(\overline{x_k})$ in round $2k+1$. If A then does neither pack $A(y_1)$ nor $A(\overline{y_1})$ in round $2k+2$, the adversary will pack the threat item $T(y_k)$ in his next move. In this case A will never be able to pack $A(y_k)$ or $A(\overline{y_k})$ or $T(x_k)$ in the future, and his weight will permanently stay below α. Hence the claimed statement also holds up to round $2k+2$.

Next we consider a selfish adversary B. If B neither packs item $B(x_k)$ nor $B(\overline{x_k})$ in round $2k+1$, then player A can react in the next round by packing the threat item $T(x_k)$. Exactly as in the hostile case, the weight of player A then exceeds the threshold α, and A wins the game. On the other hand, the selfish player B will never be able to compensate for the loss of $B(x_k)$ and $B(\overline{x_k})$. All in all, this means that the selfish B must pack item $B(x_k)$ or $B(\overline{x_k})$ in round $2k+1$. Finally, we may argue exactly as in the hostile case that player A then must pack item $A(y_1)$ or $A(\overline{y_1})$ in round $2k+2$, Hence, also in the selfish case the claimed statement holds up to round $2k+2$. □

Lemma 6.3 *Let c' denote the remaining knapsack capacity at the end of round $2s$. Let $w(A)$ and $w(B)$ denote the weight packed by player A and the (hostile or selfish) adversary B in rounds $2s+1, 2s+2, 2s+3$.*

(S1) *If $c' \ge w(V_1)$ then $w(A) = 0$ and $w(B) = w(V_1)$.*
(S2) *If $c' = w(V_1) - 1$ then $w(A) = w(U)$ and $w(B) = w(V_2)$ or $w(B) = w(V_3)$.*
(S3) *If $c' \le w(V_1) - 2$ then $w(A) = 0$ and $w(B) = w(V_2) + w(V_3)$.*

Proof By the preceding lemma, after the first $2s$ rounds the knapsack will contain exactly one of the items $B(x_i)$ and $B(\overline{x_i})$ and exactly one of the items $A(y_i)$ and $A(\overline{y_i})$ for $1 \le i \le s$. This implies $c' \le \beta$. Note that player B is to move in round $2s+1$. Note furthermore that $w(B) \le w(V_1)$.

Now if $c' \ge w(V_1)$, then the (hostile or selfish) adversary B will pack verification item V_1 in round $2s+1$. Thereby the selfish adversary B reaches his maximal possible profit, and the hostile adversary B prevents player A from packing further items.

Once B has packed item V_1, the game is over as no further items fit into the knapsack. This establishes (S1). Next, if $c' = w(V_1) - 1$ then item V_1 does not fit anymore into the knapsack. Player B packs item V_2 (or as hostile adversary, B perhaps packs item V_3). Player A reacts by packing U. The rest capacity of the knapsack is now smaller than any remaining item, and the game is over. This shows (S2). Finally, if $c' \leq w(V_1) - 2$ then the (hostile or selfish) adversary B will pack item V_2 in round $2s + 1$. This brings the remaining knapsack capacity below $w(U)$, so that player A must pass in round $2s + 2$. Then player B packs item V_3 in round $2s + 3$, and the game is over. This establishes (S3). □

By Lemmas 6.2 and 6.3, player A will win the game if configuration (S2) occurs after round $2s$, and he will lose the game under configurations (S1) and (S3). If player B is hostile, then his goal will be to avoid configuration (S2). If player B is selfish, then he will also avoid configuration (S2), as both (S1) and (S3) yield a better profit for him. Hence in either case, the objective of player A is to reach configuration (S2) and the objective of the adversary is exactly to avoid this configuration (S2).

Now let us finally connect our analysis to the considered instance of QUANTIFIED 1-IN-3-SAT. By Lemma 6.2, after the first $2s$ rounds the knapsack contains exactly one of items $B(x_i)$ and $B(\overline{x_i})$ and exactly one of items $A(y_i)$ and $A(\overline{y_i})$ for $1 \leq i \leq s$. We construct the following truth setting T^* from this packing: If the knapsack contains $B(x_i)$ (respectively $B(\overline{x_i})$) then variable x_i is set to true (respectively false), and if the knapsack contains $A(y_i)$ (respectively $A(\overline{y_i})$) then variable y_i is set to true (respectively false). The remaining knapsack capacity c' at the end of round $2s$ equals $w(V_1) - 1$, if and only if under truth setting T^* every clause in formula ϕ contains exactly one true literal. In other words, for player A reaching configuration (S2) in the subset sum game is equivalent to reaching a satisfying truth setting T^* for the QUANTIFIED 1-IN-3-SAT instance.

This yields a natural bijection between the moves of the players in the QUANTIFIED 1-IN-3-SAT instance and the moves of the players in the constructed instance of SSG-hostile and SSG-selfish. This implies PSPACE-hardness of these games. Furthermore, these games can be fully analyzed with polynomial space by a depth-first-search traversal of the underlying game tree; this yields containment in PSPACE. We summarize our findings in the following theorem.

Theorem 6.4 *The games SSG-hostile and SSG-selfish both are PSPACE-complete.*

7 Final Remarks

We have analyzed the three variants SSG-hostile, SSG-selfish, and SSG-greedy of the subset sum game. Our analysis fully describes the complexity and the approximability landscape of these three games.

The two games SSG-hostile and SSG-selfish look and behave very similarly. In fact, a single proof (for Theorem 6.4) suffices to settle the complexity status of both problems, and a single proof (for Theorem 3.3) settles their approximability status. We do not have a good understanding of the actual differences between these two games. In particular, the following question (motivated by the game instance in Example 2.2) remains open: What is the computational complexity of deciding for a given instance of the subset sum game, whether player A can enforce a strictly larger profit against a selfish adversary than against a hostile adversary? We suspect this problem to be computationally intractable.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Brotcorne, L., Hanafi, S., Mansi, R.: A dynamic programming algorithm for the bilevel knapsack problem. Oper. Res. Lett. **37**, 215–218 (2009)
2. Caprara, A., Carvalho, M., Lodi, A., Woeginger, G.J.: A complexity and approximability study of the bilevel knapsack problem. SIAM J. Optim. **24**, 823–838 (2014)
3. Caprara, A., Carvalho, M., Lodi, A., Woeginger, G.J.: Bilevel knapsack with interdiction constraints. INFORMS J. Comput. **28**, 319–333 (2016)
4. Carvalho, M., Lodi, A., Marcotte, P.: A polynomial time algorithm for a continuous bilevel knapsack problem. Oper. Res. Lett. **46**, 185–188 (2018)
5. Darmann, A., Nicosia, G., Pferschy, U., Schauer, J.: The subset sum game. Eur. J. Oper. Res. **233**, 539–549 (2014)
6. Dempe, S., Richter, K.: Bilevel programming with Knapsack constraint. CEJOR **8**, 93–107 (2000)
7. Fischer, D., Woeginger, G.J.: A faster algorithm for the continuous bilevel knapsack problem. Oper. Res. Lett. **48**, 784–786 (2020)
8. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco (1979)
9. Nicosia, G., Pacifici, A., Pferschy, U.: A Stackelberg knapsack game with weight control. Theor. Comput. Sci. **799**, 149–159 (2019)
10. Pferschy, U., Nicosia, G., Pacifici, A.: On a Stackelberg subset sum game. Electron Notes Discrete Math. **69**, 133–140 (2018)
11. Qiu, X., Kern, W.: Improved approximation algorithms for a bilevel knapsack problem. Theor. Comput. Sci. **595**, 120–129 (2015)
12. Schaefer, T.J.: On the complexity of some two-person perfect-information games. J. Comput. Syst. Sci. **16**, 185–225 (1978)
13. Williamson, D.P., Shmoys, D.B.: The design of approximation algorithms. Cambridge University Press, Cambridge (2011)
14. Woeginger, G.J.: When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? INFORMS J. Comput. **12**, 57–75 (2000)