The Subset Sum Game

Andreas Darmann[a], Gaia Nicosia[b], Ulrich Pferschy[c], Joachim Schauer[c]

[a] *University of Graz, Institute of Public Economics, Universitaetsstr. 15, A-8010 Graz, Austria,* `andreas.darmann@uni-graz.at`
[b] *Dipartimento di Informatica e Automazione, Università degli Studi "Roma Tre", via della Vasca Navale 79, 00146 Rome, Italy,* `nicosia@dia.uniroma3.it`
[c] *University of Graz, Department of Statistics and Operations Research, Universitaetsstr. 15, A-8010 Graz, Austria,* {`pferschy, joachim.schauer`}`@uni-graz.at`

Abstract

In this work we address a game theoretic variant of the Subset Sum problem, in which two decision makers (agents/players) compete for the usage of a common resource represented by a knapsack capacity. Each agent owns a set of integer weighted items and wants to maximize the total weight of its own items included in the knapsack. The solution is built as follows: Each agent, in turn, selects one of its items (not previously selected) and includes it in the knapsack if there is enough capacity. The process ends when the remaining capacity is too small for including any item left.

We look at the problem from a single agent point of view and show that finding an optimal sequence of items to select is an $\mathcal{NP}$-hard problem. Therefore we propose two natural heuristic strategies and analyze their worst-case performance when (1) the opponent is able to play optimally and (2) the opponent adopts a greedy strategy.

From a centralized perspective we observe that some known results on the approximation of the classical Subset Sum can be effectively adapted to the multi-agent version of the problem.

Keywords: Subset Sum problem, multi-agent optimization, performance analysis, Game Theory.

1. Introduction

In the Computer Science literature of the last two decades several classical Combinatorial Optimization problems have been revisited in a game theoretic setting where multiple deciders take over the role of a single decision maker. This new field of research is receiving growing attention and led to the emergence of *Algorithmic Game theory* [15].

In this context we address the problem of two agents competing for a shared resource. It can be described as a game theoretic variant of the classical *Subset Sum problem*, in which there are two players, or agents, called P_a and P_b, and a given amount c of a shared resource. Each agent owns a set of items with non-negative weights and knows the other agent's item set, i.e. there is perfect information.

The *Subset Sum game* works as follows: Starting with P_a, the agents take turns to select exactly one of their items which was not selected before. The total weight of all selected items must not exceed the capacity c at any time. The aim of the game is, for each agent, to select a subset of its items with maximum total weight. This problem is new and has not been studied in the literature before.

1.1. Related Works

Several problems strictly related to the Subset Sum game have been considered in the literature. In particular, due to its simple structure, the $\{0,1\}$-Knapsack problem was frequently considered in a game-theoretic context. Recently, [13, 14] considered a knapsack-type scenario with unitary weights in which the decision process is performed in rounds and managed by a central decision mechanism (arbitrator). In every round each of the two agents selects exactly one of its items and submits the item for possible inclusion in the knapsack, then the arbitrator chooses one of the items as "winner" of the round. The winning item is permanently included in the knapsack. The process goes on as long as there is enough capacity.

In the so called *Knapsack Sharing* problem studied by several authors (see for instance [9, 11]), a single objective function balancing the profits among the agents is considered in a centralized perspective. Another interesting game, based on the maximum $\{0,1\}$-Knapsack, interpreted as a special on-line problem, is addressed in [12] where a two person zero-sum game, called *Knapsack Game*, is considered. Knapsack problems are also addressed in the context of auctions. See, for example, [1, 3].

Kindred problems are the so-called *Bin Packing Games*. There, the set of agents consists of k agents representing bins and of n agents representing items of given size. The value function of a coalition of bins and items is the maximum total size of items in the coalition that can be packed into the bins of the coalition. This class of problems was introduced in [7, 8] where several different results are provided. The *Selfish Bin Packing* is another interesting game theoretic variant, where each item is controlled by a selfish agent who pays a cost proportional to the ratio between its own item size and the total weight of the items packed in the same bin. In [6], the authors study different equilibria and the associated quality measures, namely Price of Anarchy and Price of Stability[1].

A significant application closely related to our problem is the so called *Admission Control* problem (ACP) which, in a wide sense, refers to the design of

[1]Surprisingly, the investigation of this problem from a game-theoretic perspective allowed to establish new results on the approximation ratio of a heuristic algorithm for the classical bin packing problem [5].

mechanisms for managing traffic requests in communication systems (for a comprehensive survey, see for instance [2, 10]). In bandwidth-managed networks, it is required to evaluate (*i*) if bandwidth is available to service a new potential user and (*ii*) the QoS (quality of service) that can be provided to this user. Only if bandwidth is available, the new user is admitted. Clearly, new users can be viewed as selfish users competing for network bandwidth, i.e., resource capacity. In this context, Game Theory techniques have been used to design management protocols to monitor, control, and enforce the use of shared resources and services in networks. For instance, in [18] the authors propose pricing schemes influencing users in their decision to take part or not in a wireless channel. The equilibria induced by these schemes and their performance are evaluated showing their potential to produce high quality outcomes in an incentive-compatible way. Analogously, the issue of designing resource allocation mechanisms that produce efficient throughput and congestion allocations despite the selfish users' behavior is discussed in [17].

1.2. Our Contributions

The above mentioned works follow two established approaches in classical and algorithmic game theory and focus on (*i*) finding equilibria, i.e., solutions where each agent obtains no benefit by moving from them, and/or (*ii*) designing mechanisms, i.e. protocols leading the (selfish) agents to solutions which are either globally optimal or follow some intuitive, easy to compute rule. The problem we address in this paper is indeed a multi-deciders version of the well known Subset Sum problem where two agents compete for the capacity of a common knapsack in presence of a very simple mechanism (round robin). However, we adopt a different perspective here, namely we look at the problem from the point of view of one agent and seek strategies optimizing her payoff depending on the behavior of the other agent.

In this paper we show that it is $\mathcal{NP}$-hard to compute an optimal strategy for one agent by a simple reduction from the standard Subset Sum problem. Hence, we introduce heuristic approaches and analyze two very natural strategies based on a greedy concept which would be intuitive rules of thumb for any practical game scenario (Section 2.2).

The first strategy is the pure greedy algorithm, which maximizes in each round the weight of the selected item. The second strategy is an extension which tries to take the subsequent round into account in the decision. Assuming that one agent adopts the proposed strategy, we analyze the performances when (*i*) the opponent is able to play optimally and (*ii*) the opponent also follows the greedy strategy. In particular, we are able to show that the first heuristic has a performance ratio of 1/2 both against an optimal opponent and against a greedy opponent (Section 3.1), while the second proposed algorithm has a performance ratio of 2/3 against the optimal opponent (see Section 3.2). Moreover, we show that a natural generalization of these greedy based strategies invoking a farther reaching consideration of future rounds does not allow a further improvement of the 2/3 performance ratio (Section 3.3).

Furthermore, we observe that from a centralized perspective, some known results on the approximation of the classical Subset Sum can be effectively adapted to the multi-agent version of the problem (Section 4). We conclude by showing that two natural extensions of the proposed algorithms to a Knapsack Game problem do not provide a bounded performance ratio and, finally, put forward directions for future research (Section 5).

1.3. Formal Problem Setting

Let P_a and P_b indicate the two agents. Agent P_x owns a set N_x of n_x items, where item i of agent P_x has a non-negative weight x_i, with $x = a, b$. We assume that $N_a \cap N_b = \emptyset$ and that there is perfect information, so agents know each other's item sets.

The game can be seen as a sequence of *rounds*, where in each round P_a selects an item from N_a followed by the selection of an item from N_b by agent P_b. The total weight of all selected items must not exceed the capacity c at any time. If at any point of the game one agent is unable to select any more items because the remaining capacity is too small, the agent just remains idle and the other agent can continue to select items.

It is easy to show that the associated decision problem, namely whether both agents can reach a certain total weight, is $\mathcal{NP}$-complete.

> Subset Sum Game Decision (SSGD): Given a_j and b_j, $j = 1, \dots, n$, and two positive values Q_a and Q_b. Is there an outcome of the Subset Sum game such that P_a gains a total weight $\geq Q_a$ and P_b gains $\geq Q_b$?

Proposition 1. *Problem SSGD is $\mathcal{NP}$-complete.*

Proof. Reduction from the Subset Sum problem (SSP): Given n integer numbers $w_1, \dots, w_n$ and a value W, is there a subset S of items with total weight equal to W?

To answer the decision version of SSP consider the following instance I of SSGD: Agent P_a tries to solve SSP, i.e. $a_i = w_i$ for $i = 1, \dots, n$. Agent P_b is negligible with $b_i = \varepsilon$ for all i and $c = W + n\varepsilon$, where $\varepsilon < 1/n$. Set $Q_a = W$ and $Q_b = \varepsilon$. It is easy to see that the strategies pursued by the two agents do not matter at all since P_b always can select all its items while P_a never can exploit the capacity $c - W < 1$. P_a can reach $Q_a = W$ iff SSP is a YES-instance. □

2. Strategies for one Agent

For notational convenience, we address the problem from the point of view of agent P_a. For agent P_b, the perspective is completely the same after subtracting from c the weight of the item selected by P_a in the first round.

In this paper, with a slight abuse of terminology, a *strategy*[2] S of an agent indicates a rule, or an algorithm, that specifies which item to select in any round depending on the capacity and the sets of selected and still available items of both agents. Since the outcome of the game depends on the strategies employed by the two agents, if P_a follows a strategy S and P_b a strategy Z, then we denote the total weights obtained by agent P_a and P_b as A_{SZ} and B_{SZ}, respectively. Clearly, for every pair of feasible strategies S and Z, $A_{SZ} + B_{SZ} \leq c$.

2.1. Optimal Strategy

In general, the optimal strategy of an agent depends not only on the outcome of previous rounds but also on the future decisions of the other agent. If the strategy of P_b is not known, there is no way for P_a to always make optimal decisions. Else, if agent P_a knows the strategy Z of agent P_b, it can compute an *optimal strategy* O, maximizing the total weight of the selected items. This can be done by modeling the Subset Sum game as a *game in extensive form* and representing it by a *game tree* as it is usually done in game theory (see e.g. [16, Sec. 5]).

A game tree represents all possible decisions of both agents in sequential form. Each node of the tree corresponds to the decision of an agent in a certain round, such that every possible outcome of this decision is represented by a child node. Thus, the root of the tree (i.e. a node in level 1) corresponds to the decision of P_a in the first round and has n_a child nodes, one for each possible item selected by P_a. Each such child node (i.e. a node in level 2) represents the decision of P_b in the first round. Clearly, feasible selections are those where the total weight reached at the current node does not exceed the capacity c.

Considering an arbitrary node in level ℓ, $\ell \geq 2$, in the tree, one could easily determine all previous decisions by moving upwards along the unique path to the root of the tree. Thus, the remaining capacity and the set of not yet selected items of the current agent (P_a if $\ell \equiv 1 \bmod 2$, P_b otherwise) are known and one can establish which items could still be selected at this node. Each of them gives rise to a child node in level $\ell + 1$. It is convenient to assume that an agent selects an artificial item of weight 0 if it cannot select any other item, but the other agent still has items to select. Every leaf of this game tree describes a feasible outcome of the game and yields a pair of total weights (A, B) obtained by the two agents.

Now an *optimal strategy* can be determined for both agents by settling all decisions by *backward induction*. This means that for each node, whose child nodes are all leaves, the associated agent can reach a final decision by simply choosing the best of all child nodes w.r.t. their allocated total weight. Then these leaf nodes can be deleted and the pair of gained weights of the chosen leaf is moved to its parent node. In this way, we can move upwards in the tree towards the root and settle all decisions along the way.

[2] In game theory, a strategy would denote the set of all decisions for any possible scenario of the game. This would be the result of a strategy in our more algorithmic terminology.

If an agent, say P_b, follows a certain strategy S, then P_b will execute a certain decision in each of its nodes. Thus, each of its nodes only has one child node. P_a can determine an optimal answer against strategy S by following the same procedure as above.

Unfortunately, this procedure cannot be easily used in practice due to the exponential number of nodes in the game tree. In particular, it can be easily concluded from the proof of Proposition 1, that it is $\mathcal{NP}$-hard to compute the optimal strategy for an agent against a given strategy of the other agent. To escape this intractability, it is a reasonable approach to make use of heuristic algorithms as strategies (see in Sections 2.2 and 3).

In the terminology of Game Theory an optimal strategy as described above is by definition a *Nash equilibrium* and also a so-called *subgame perfect equilibrium* (a slightly stronger property), since the decisions made in the above backward induction procedure are also optimal for every subtree (see [16, Sec. 5] for more details). Clearly, the optimal strategy and hence the equilibrium of the game is not unique, since there may well be several different leafs of the game tree yielding the same weights for both agents.

2.2. Heuristic Algorithms

Because it is $\mathcal{NP}$-hard to compute an optimal strategy, we will consider heuristic strategies for the agents. A simple greedy algorithm is a very natural choice for the Subset Sum game. In this case an agent simply selects in every round the item with largest weight that does not violate the capacity constraint. In this way, the capacity available for the other agent is (at least locally) minimized, which seems to be an intuitively appealing approach. In Section 3.1 it is shown that such a greedy algorithm may reach only half of the weight obtained by an optimal strategy but can not do worse than that. For sake of completeness we also give a formal description in Algorithm 1.

Algorithm 1 Greedy G

$N := N_a; \bar{c} := c;$
while $\min\{a_j \mid j \in N\} \leq \bar{c}$ **do**
  $a_{\max} := \max\{a_j \mid a_j \leq \bar{c}, j \in N\}$ attained for $\tilde{\jmath}$;
  select item $\tilde{\jmath}$; $N := N - \{\tilde{\jmath}\}$;
  selection of an item b' by P_b;
  $\bar{c} := \bar{c} - a_{\tilde{\jmath}} - b'$
end while

An effective improvement over this simple greedy mechanism is the following Look-ahead Greedy algorithm L, which tries to avoid the shortsightedness of Greedy at least to some extend. Motivated by the worst-case example given in Section 3.1 (proof of Theorem 5) it considers in every round all *feasible pairs* of items and picks the pair with highest total weight. The larger item of this pair is then selected in the current round. In this context a pair of items is feasible if P_a can be sure to be able to select the two items in the current and the subsequent round, no matter what P_b does in the current round, i.e. even if P_b selects the largest item that fits. A formal description is given in Algorithm 2. To avoid tedious special cases we assume that there are always sufficient items available for P_a to consider a pair of items in every round. This can be achieved by adding dummy items with weight 0.

Algorithm 2 LOOK-AHEAD GREEDY L

$N := N_a; \bar{c} := c;$
while $\min\{a_j \mid j \in N\} \leq \bar{c}$ **do**
  $p_{\max} := 0;$
  for every pair (i, j) in N with $a_i \geq a_j$ **do**
    $b_{\max} := \max\{b_\ell \mid b_\ell \leq \bar{c} - a_i\}$
    if $a_i + b_{\max} + a_j \leq \bar{c}$ **then**
      $p_{\max} := \max\{p_{\max}, a_i + a_j\}$
    end if
  end for
  let $p_{\max}$ be attained for $(\tilde{\imath}, \tilde{\jmath})$;
  select item $\tilde{\imath}$; $N := N - \{\tilde{\imath}\}$;
  selection of an item b' by P_b;
  $\bar{c} := \bar{c} - a_{\tilde{\imath}} - b'$
end while

It should be noted that algorithm LOOK-AHEAD GREEDY may also select an item different from $\tilde{\jmath}$ in the next round if a different pair of items without $\tilde{\jmath}$ turns out to be the best choice in the next round or if P_b does not select the maximal possible weight $b_{\max}$ in the current round.

A natural generalization of LOOK-AHEAD GREEDY looks even farther into the future and considers more than two items as a look ahead. The resulting k-LOOK-AHEAD GREEDY algorithm (k-L) determines, in each round, the best combination of k items for P_a that cannot be "blocked" by the opponent P_b. Then the largest item of this k-tuple is selected in this round. Clearly, the above algorithm L arises as the special case of 2-L. Note that by definition of this algorithm, any possibility by P_b to block the k-tuple considered by P_a is taken into account. This "blocking strategy" of P_b may be quite different from the greedy strategy which always puts $b_{\max}$ against 2-L.

While it is easy to see that LOOK-AHEAD GREEDY may perform better than GREEDY (see e.g. Example 3), and also 3-LOOK-AHEAD GREEDY may perform better than 2-LOOK-AHEAD GREEDY (e.g. in Example 5), we can show that it may also be the other way round (see e.g. Example 1). In conclusion, there is no strict dominance between the different extensions of the GREEDY strategy.

Example 1.
Consider the following instance of the Subset Sum game with capacity $12 + 5\delta$ *where* $\delta \ll \varepsilon$.

item	1	2	3	4	5	6	7	8
N_a	10	$4-\varepsilon$	$4-\varepsilon$	$4-\varepsilon$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
N_b	δ	δ	δ	δ	δ			

Observe that the items in N_b can all be packed and can be neglected in the selection of P_a. 3-Look-ahead Greedy *identifies the three items with weight $4-\varepsilon$ as the best triple and selects one of them. In round 2, only the remaining two items of this triple can be packed, and the algorithm continues to gain a total weight of $12-3\varepsilon$.*

An optimal selection of P_a would start with item 1 and continues to pack all four items of weight $\frac{1}{2}$ ending up with a total weight of 12. Note that Greedy *and* 2-Look-ahead Greedy *both pick this optimal strategy.*

A simple variation of this example where the three items of weight $4-\varepsilon$ are replaced by two items of weight $6-\varepsilon$ shows by an analogous reasoning that 2-Look-ahead Greedy *may be stuck with $12-2\varepsilon$ while the optimal solution identified by* Greedy *obtains 12.*

3. Performance Analysis

To analyze the performance of the heuristics for the Subset Sum game we follow a worst-case perspective taking agent P_a's point of view. As in the performance analysis of classical approximation algorithms, we consider the worst solution scenario over all possible instances, i.e. sets of input data, and compare the solution value derived by a heuristic with the solution value attained by the optimal strategy.

Differently from classical optimization problems we also have to include the strategy of agent P_b in our analysis. In the following we define as *performance bound* $\rho_{HS} \in [0,1]$ a bound on the ratio between the solution value of a heuristic H and the solution value of an optimal strategy for agent P_a both playing against a specified strategy S of the adversary agent P_b, i.e.

$$\rho_{HS} \leq \frac{A_{HS}}{A_{OS}} \quad \text{for all inputs}\,. \tag{1}$$

As usual, we call a performance bound ρ_{HS} *tight*, if no larger value than ρ_{HS} exists which fulfills (1).

Assuming that P_b does not act in a completely arbitrary or self destructive way, the most plausible strategy S of P_b is the optimal response. That is, knowing the strategy H of P_a, P_b maximizes its total weight (cf. Section 2.1) neither trying to help nor harm P_a. By maximizing its own total weight, the capacity remaining to be utilized by P_a is automatically minimized, which fits well together with the notion of a worst-case analysis. However, note that an optimal strategy of P_b does not necessarily yield the worst possible outcome for a strategy of P_a. This can be seen—after exchanging the roles of P_a and P_b—from Example 4, where the switching from the optimal to the Greedy strategy of one agent generates a worse outcome for the optimal strategy of the other agent.

To avoid clumsy notation we assume w.l.o.g. that the items of both agents are numbered in the order they are selected during the game, i.e. agent P_a selects a_j in round j. Items not selected are numbered arbitrarily with indices higher than the selected items.

The following proposition will be useful in the proofs below.

Proposition 2. *It can always be assumed that an optimal strategy of P_b selects items in* nonincreasing *order of weights against the* GREEDY *strategy of P_a.*

Proof. Assume contrary to the statement that there exists a selected item b_j such that $b_{j-1} < b_j$, $j \geq 2$. Clearly, there must be

$$\sum_{k=1}^{j} b_k \leq c - \sum_{k=1}^{j} a_k \Longleftrightarrow a_j \leq c - \sum_{k=1}^{j-1} a_k - \sum_{k=1}^{j} b_k \, . \tag{2}$$

Since a_j was computed as the maximum over all remaining items with capacity at most $c - \sum_{k=1}^{j-1} a_k - \sum_{k=1}^{j-1} b_k$, and since a_j also fulfills the stricter condition (2), it follows that a_j is also the maximum over all remaining items with the intermediate capacity at most $c - \sum_{k=1}^{j-1} a_k - \sum_{k=1}^{j-2} b_k - b_j$. Therefore, the GREEDY algorithm of P_a does not change its selection in round j even if P_b selects b_j in round $j-1$ and b_{j-1} in round j. □

Note that Proposition 2 does not hold for the LOOK-AHEAD GREEDY strategy, as shown in the following example:

Example 2. *Consider the following instance $\mathcal{I}$ of the Subset Sum game with capacity $c = 19 + 3\varepsilon$.*

item	1	2	3	4
N_a	6	6	$3+\varepsilon$	$3+\varepsilon$
N_b	$5+4\varepsilon$	5	2	3ε

In the first round P_a computes the largest pair consisting of items 1 and 2 and thus chooses item 1 with weight 6 leaving the residual capacity $\bar{c} = 13 + 3\varepsilon$. P_b has four possibilities to react. The resulting games are listed as columns in Table 1.

In order to achieve the maximum total weight for herself, P_b has to choose first the item with weight 2 and then the one with weight 5. In some sense P_b can "threaten" to use its largest item 1 in the next round and thereby forces P_a to choose the larger item with weight 6 instead of items 3 and 4. Thereby LOOK-AHEAD GREEDY *leaves room for the final item 3ε of P_b. If P_b chose items in decreasing order of their weights, this would permit a better choice for P_a and would reduce the total weight attained by P_b.*

By going through all possible solutions it can be checked that in the above example the strategy of P_a is optimal.

round 1 of P_b	$5+4\varepsilon$	5	2	3ε
$\bar{c}$	$8-\varepsilon$	$8+3\varepsilon$	$11+3\varepsilon$	13
best pair of P_a	$6, 0$	$3+\varepsilon, 3+\varepsilon$	$6, 0$	$3+\varepsilon, 3+\varepsilon$
round 2 of P_a	6	$3+\varepsilon$	6	$3+\varepsilon$
$\bar{c}$	$2-\varepsilon$	$5+2\varepsilon$	$5+3\varepsilon$	$10-\varepsilon$
round 2 of P_b	3ε	2	5	$5+4\varepsilon$
$\bar{c}$	$2-4\varepsilon$	$3+2\varepsilon$	3ε	$5-5\varepsilon$
round 3 of P_a	-	$3+\varepsilon$	0	$3+\varepsilon$
round 3 of P_b	-	-	3ε	-
A_{LO}	12	$12+2\varepsilon$	12	$12+2\varepsilon$
B_{LO}	$5+7\varepsilon$	7	$7+3\varepsilon$	$5+7\varepsilon$

Table 1: Resulting games in instance $\mathcal{I}$, after P_a chose item 1 in the first round.

We summarize the results of Example 2 in the following proposition.

Proposition 3. *The optimal strategy of agent P_b against* Look-ahead Greedy *or against an optimal strategy of agent P_a may select items in a* non monotone *order of weights.* □

3.1. Performance of the Greedy *Algorithm*

The following technical lemma will be used to prove the performance bound of Greedy both against an optimal and against a greedy strategy of P_b.

Lemma 4. *Let agent P_a use the* Greedy *strategy G and let S be the strategy used by P_b. Assume that* Greedy *is able to select the $j-1$ largest items of N_a and fails to pack the j-th largest item in round j against S. If $S \in \{O, G\}$ then*

$$A_{OS} \leq c - \sum_{k=1}^{j-1} b_k$$

where $b_1, \ldots, b_{j-1}$ are the items selected by P_b in the first $j-1$ rounds following strategy S against the Greedy *strategy G of P_a.*

Proof. Clearly, $j \geq 2$ holds. Since $A_{OS} \leq c - B_{OS}$, it is sufficient to show that for $S \in \{O, G\}$ we have $B_{OS} \geq \sum_{k=1}^{j-1} b_k$.

Let $S = O$, i.e., P_b uses its optimal strategy. Then, even an optimal strategy of P_a cannot avoid that P_b reaches a total weight B_{OO} of at least $\sum_{k=1}^{j-1} b_k$, which P_b managed to achieve after $j-1$ rounds even against the $j-1$ largest items of N_a. Let $S = G$. Assume that $B_{OG} < \sum_{k=1}^{j-1} b_k$. Let $\tilde{b}_1, \tilde{b}_2, \ldots, \tilde{b}_{j-1}$ be the items selected in the first $j-1$ rounds by G of P_b against O of P_a. Then, in order to satisfy $B_{OG} < \sum_{k=1}^{j-1} b_k$,

$$\sum_{k=1}^{j-1} \tilde{b}_k < \sum_{k=1}^{j-1} b_k \tag{3}$$

must hold. Since P_b uses the GREEDY strategy in both scenarios (against G resp. O of P_a), there is a unique smallest index i, $1 \leq i \leq j-1$, such that $\tilde{b}_i \neq b_i$. Note that because of $i \leq j-1$, $c - \sum_{k=1}^{i} \tilde{a}_k \geq c - \sum_{k=1}^{i} a_k$ holds, where $\tilde{a}_1, \ldots, \tilde{a}_i$ and $a_1, \ldots, a_i$ denote the items selected by P_a according to O and G respectively. If $\tilde{b}_i < b_i$, then in round i it would have been possible for P_b to pack the larger item b_i against O of P_a, which contradicts the fact that P_b uses the greedy strategy. Hence, $\tilde{b}_i > b_i$ holds. However, because of (3) and the definition of i, $\sum_{k=i}^{j-1} b_k > \sum_{k=i}^{j-1} \tilde{b}_k$ holds. This implies $\sum_{k=i}^{j-1} b_k > \tilde{b}_i$, and hence, when playing against G of P_a, P_b could have packed $\tilde{b}_i > b_i$ in round i. Again, this contradicts the GREEDY strategy. Therewith, $B_{OG} \geq \sum_{k=1}^{j-1} b_k$ holds. □

Theorem 5. *The* GREEDY *algorithm G has a tight performance bound of*

$$\rho_{GO} = \frac{1}{2}\,.$$

Proof. Assume $A_{OO} > A_{GO}$ (otherwise $A_{OO} = A_{GO}$ and we are done). By the greedy strategy G selects the largest $j-1$ items of N_a and fails to pack the j-th largest item with weight $\tilde{a}$ in round j for some $j \geq 2$. It may continue to pack smaller items, but these can be neglected in our analysis. If $\sum_{k=1}^{j-1} a_k \geq \frac{1}{2} A_{OO}$ we are done. Otherwise, for $\sum_{k=1}^{j-1} a_k < \frac{1}{2} A_{OO}$ there is

$$\tilde{a} \leq a_{j-1} < \frac{1}{j-1} \frac{1}{2} A_{OO} \leq \frac{1}{2} A_{OO}\,. \tag{4}$$

Since $\tilde{a}$ could not be selected in round j there must be

$$\tilde{a} > c - \sum_{k=1}^{j-1} a_k - \sum_{k=1}^{j-1} b_k \iff \sum_{k=1}^{j-1} a_k > c - \sum_{k=1}^{j-1} b_k - \tilde{a} \tag{5}$$

Because of Lemma 4, $A_{OO} \leq c - \sum_{k=1}^{j-1} b_k$ holds. Putting this inequality together with (5) and plugging in (4) we get

$$A_{OO} - A_{GO} \leq c - \sum_{k=1}^{j-1} b_k - \left(c - \sum_{k=1}^{j-1} b_k - \tilde{a} \right) = \tilde{a} \leq \frac{1}{2} A_{OO}$$

and we have shown $\rho_{GO} \leq \frac{1}{2}$.

The following Example 3 gives an instance with parameter $\varepsilon > 0$ where $\lim_{\varepsilon \to 0} \rho_{GO} = \frac{1}{2}$ thus completing the proof of the theorem. □

item	1	2	3
N_a	$\frac{1}{2}$	$\frac{1}{2}-\varepsilon$	$\frac{1}{2}-\varepsilon$
N_b	2ε	ε	

Example 3. *Consider the following instance with $c = 1$.*

In the first round P_a selects the largest item 1 and P_b chooses 2ε. Now P_a cannot select another item and $A_{GO} = \frac{1}{2}$. An optimal strategy would select items 2 and 3 with a total weight of $A_{OO} = 1 - 2\varepsilon$.

Corollary 6. *Against the* Greedy *strategy G of P_b, the* Greedy *strategy of P_a has a tight performance ratio of*

$$\rho_{GG} = \frac{1}{2}.$$

Proof. The proof is analogous to the one of Theorem 5 when A_{OO} and A_{GO} are replaced by A_{OG} and A_{GG} respectively. The tightness of the bound again can be concluded from the instance used in Example 3. □

It could be expected that the optimal strategy for P_b always yields a better solution against a suboptimal strategy of P_a, such as G, than against an optimal strategy of P_a. Clearly, if P_a consumes less capacity, there is more capacity left for P_b to utilize. Surprisingly, this is not always the case as shown by the following counterexample. It may be necessary for P_b to select a less attractive item in order to block another item of P_a. However, if both agents pursue an optimal strategy, they might both benefit.

Example 4. *Consider an instance with the following data and capacity $c = 23$.*

item	1	2	3	4	5
N_a	7	7	4	4	4
N_b	10	5.5	5.5	0	0

If P_a follows the Greedy *algorithm G, it first selects $a_1 = 7$. If P_b selects an item with weight 5.5 in the first round, P_a could select the second item of weight 7 in the second round thus preventing P_b from picking a further item. Hence, P_b has to choose item 1 in the first round ending the game with $B_{GO} = 10$, while P_a can add an item of weight 4 in the second round obtaining $A_{GO} = 11$.*

In an optimal strategy, P_a would start with an item of weight 4 (otherwise the above case applies). If P_b chooses item 1 in the first round, P_a could select any item in the second round to prevent P_b from selecting any further items. Hence, P_b should select, in the first round, an item with weight 5.5. This allows P_a two choices: (1) it selects an item of weight 7, then P_b would continue with the other item of weight 5.5 and both agents are finished; (2) P_a selects another item of weight 4, then again P_b continues with the other item of weight 5.5 (since *item 1 does not fit) and reaches a total weight of $B_{OO} = 11$ while P_a can enter into a next round to submit a third item of weight 4 yielding $A_{OO} = 12$.*

Thus, we have given an instance with $A_{GO} < A_{OO}$ and $B_{GO} < B_{OO}$.

On the other hand, the following proposition holds.

Proposition 7. Greedy *for P_a always benefits if P_b applies* Greedy *rather than an optimal strategy, that is*

$$A_{GG} \geq A_{GO}.$$

Proof. We can assume $A_{GG} \neq A_{GO}$ because otherwise we are done. Let a_k resp. $\tilde{a}_k$ denote the item selected by G of agent P_a against G resp. O of P_b in round k, $k \geq 1$. Note that $a_1 = \tilde{a}_1$ holds since P_a applies the greedy strategy in both scenarios. Thus, there must be an index $i \geq 2$ such that $a_i \neq \tilde{a}_i$ and $a_j = \tilde{a}_j$ holds for $j < i$.

Case (*i*): $\tilde{a}_i > a_i$. Then, in round i agent P_a was able to pack the larger item $\tilde{a}_i$ against O of P_b but could not pack it against G of P_b. Hence, $B_{GG} > c - \sum_{k=1}^{i-1} a_k + \tilde{a}_i$. On the other hand, since in the first i rounds P_a was able to pack the items $\tilde{a}_k$, $1 \leq k \leq i$, against O of P_b, we must have

$$B_{GO} \leq c - \sum_{k=1}^{i} \tilde{a}_k = c - \sum_{k=1}^{i-1} a_k + \tilde{a}_i.$$

I.e., $B_{GG} > B_{GO}$ holds, in contradiction to the definition of O.

Case (*ii*): $\tilde{a}_i < a_i$. This means that in round i, agent P_a was able to pack the larger item a_i against G of P_b but could not pack it against O of P_b. Thus, $B_{GO} > c - \sum_{k=1}^{i-1} \tilde{a}_k + a_i$, which settles the claim because it implies

$$A_{GO} < \sum_{k=1}^{i-1} \tilde{a}_k + a_i = \sum_{k=1}^{i} a_k \leq A_{GG}.$$

□

3.2. Performance of the Look-ahead Greedy *Algorithm*

We start with a simple proposition complementing Proposition 2.

Proposition 8. *It can always be assumed that the* Look-ahead Greedy *strategy of agent P_a selects items in* nonincreasing *order of weights against the optimal strategy of agent P_b.*

Proof. Assume that for some round j there is $a_j < a_{j+1}$. Since in round j the Look-ahead Greedy computes a pair of items and selects the larger of the pair, this means that a different pair (j, k) was determined yielding $p_{\max}$. By definition of the algorithm there is $a_j \geq a_k$ and hence $a_k < a_{j+1}$. Hence, items

j and $j+1$ would have been a better pair than j and k and must have been excluded from the computation of $p_{\max}$, which means that $a_j + b_{\max} + a_{j+1} > c$.

By definition, the item with weight $b_{\max}$ would be a feasible selection for P_b after a_j. But then

$$b_{\max} > c - a_j - a_{j+1} \geq c - A_{LO} \geq B_{LO} \tag{6}$$

yields a contradiction to the optimality of the strategy of P_b. □

Note that Proposition 8 does not hold for every adversary strategy S. P_b may surprisingly select an extremely small item thus permitting P_a to select a larger item in round two and deviate from the original strategy to select a certain pair of items, each one with smaller weight.

Theorem 9. *The* Look-ahead Greedy *algorithm L has a tight performance bound of*

$$\rho_{LO} = \frac{2}{3}\,.$$

Proof. Let $Opt := A_{OO}$ and assume $A_{OO} > A_{LO}$ (otherwise there is $A_{OO} = A_{LO}$ and we are done). Denote the items selected by an optimal strategy O of P_a as $\bar{a}_j$. By Proposition 8, a_1 and a_2 are the two largest items selected by L. If $a_1 + a_2 \geq \frac{2}{3}\,Opt$ we are done. Assuming $a_1 + a_2 < \frac{2}{3}\,Opt$ it follows again from Proposition 8 that $a_j < \frac{1}{3}\,Opt$ for all $j \geq 2$.

First, we consider the special case $\bar{a}_1 > a_1$: It follows from the decision of L in the first round that $\bar{a}_1$ and a_2 can not be considered in the selection of the best pair, because this pair would be better than a_1 and a_2, but $\bar{a}_1$, as the larger of the pair, was not selected in round 1. Thus, there must exist some item $\tilde{b}$, which P_b could select in round 1, to block this pair, i.e. $\bar{a}_1 + \tilde{b} \leq c$ and $\bar{a}_1 + a_2 + \tilde{b} > c$. Clearly, this means that $A_{OO} \leq c - \tilde{b}$. Thus we have

$$A_{OO} - A_{LO} \leq c - \tilde{b} - a_1 - a_2 < \bar{a}_1 - a_1.$$

If $a_1 \geq \frac{1}{3}\,Opt$, we are done since $\bar{a}_1 < \frac{2}{3}\,Opt$ (otherwise $\bar{a}_1$ would be a sufficiently large solution for L on its own).

If $a_1 < \frac{1}{3}\,Opt$, we can use the previous inequalities to show

$$\bar{a}_1 + a_1 \geq \bar{a}_1 + a_2 > c - \tilde{b} \geq A_{OO} = Opt.$$

Since $a_1 < \frac{1}{3}\,Opt$, this implies $\bar{a}_1 > \frac{2}{3}\,Opt$ which again settles this case.

Generalizations of these arguments will be used in the following to show the statement for the general case of $\bar{a}_1 \leq a_1$.

At first we introduce two technical lemmata. Lemma 10 corresponds to the situation of P_a trying to select a pair $\tilde{a}$ and $a_{j'+1}$ in round $j'+1$ and realizing that $\tilde{a}$ could well be packed on its own, but P_b has an item $\tilde{b}$ available to block $a_{j'+1}$.

Lemma 10. *Assume that after completion of some round j', $j' < n_a$, there exist items $\tilde{a}$ resp. $\tilde{b}$ not yet selected by L resp. P_b, such that the following inequalities hold:*

$$\sum_{k=1}^{j'} a_k + \tilde{a} + \sum_{k=1}^{j'} b_k + \tilde{b} \leq c \tag{7}$$

$$\sum_{k=1}^{j'} a_k + \tilde{a} + a_{j'+1} + \sum_{k=1}^{j'} b_k + \tilde{b} > c \tag{8}$$

If $B_{OO} \geq \sum_{k=1}^{j'} b_k + \tilde{b}$ then we have

$$A_{OO} - A_{LO} < \tilde{a}.$$

Proof. Clearly, we can bound the weight obtained by L as $A_{LO} \geq \sum_{k=1}^{j'+1} a_k$. Then we get from the condition of the Lemma and by applying (8)

$$\begin{aligned} A_{OO} - A_{LO} &\leq c - B_{OO} - \sum_{k=1}^{j'+1} a_k \\ &< c - \sum_{k=1}^{j'} b_k - \tilde{b} - \left(c - \sum_{k=1}^{j'} b_k - \tilde{b} - \tilde{a} \right) \\ &= \tilde{a}. \end{aligned}$$

□

Lemma 11 states that as long as L dominates the optimal strategy in every round, P_b gains at least as much weight against the optimal strategy of P_a as against L.

Lemma 11. *If there exists an index $j' \geq 1$ such that*

$$\sum_{k=1}^{\ell} a_k \geq \sum_{k=1}^{\ell} \bar{a}_k \quad \text{for all } \ell = 1, \ldots, j',$$

then $B_{OO} \geq \sum_{k=1}^{j'} b_k$.

Proof. If $B_{OO} < \sum_{k=1}^{j'} b_k$, then the strategy of P_b can not be optimal since P_b could have chosen the items $b_1, \ldots, b_{j'}$ instead. By the condition of the lemma these items would have been a feasible choice in every round $\ell \leq j'$. □

Proceeding with the proof of Theorem 9 we now let j be the first round where the optimal strategy reaches a higher total weight than L, i.e. j is the minimal index such that

$$\sum_{k=1}^{j-1} a_k \geq \sum_{k=1}^{j-1} \bar{a}_k \text{ and } \sum_{k=1}^{j} a_k < \sum_{k=1}^{j} \bar{a}_k \,. \tag{9}$$

Note that $j \geq 2$ holds in the considered case of $\bar{a}_1 \leq a_1$.

Now we consider the item set D consisting of items selected in the first j rounds by O, but not by L, i.e. $D := \{\bar{a}_1, \dots, \bar{a}_j\} \setminus \{a_1, \dots, a_j\}$. Obviously, $D \neq \emptyset$. Let $\bar{a}_D^1$ resp. $\bar{a}_D^2$ denote the largest resp. second largest (if it exists) item in D.

Case 1: $\bar{a}_D^1 \leq \frac{1}{3}Opt$. Trivially, $\bar{a}_D^1 > a_j$ because of (9). Recall from Proposition 8 that a_j is the smallest item selected by L in the first j rounds. Considering the decision of L in round $j-1$ we distinguish two cases:

Case 1.1: $\bar{a}_D^1 > a_{j-1}$. In this case, L was able to select a pair consisting of a_{j-1} and some other item with smaller weight, possibly a_j but maybe some other item, in round $j-1$. However, the better pair $\bar{a}_D^1$ and a_{j-1} was not selected because otherwise $\bar{a}_D^1$ would have been selected by L in round $j-1$ as the larger item of the pair. This omission of $\bar{a}_D^1$ by L can have two reasons: Either $\bar{a}_D^1$ could not be added in round $j-1$ at all. But this means that

$$\sum_{k=1}^{j-2} a_k + \bar{a}_D^1 + \sum_{k=1}^{j-2} b_k > c. \tag{10}$$

From Lemma 11 we have in this case $B_{OO} \geq \sum_{k=1}^{j-2} b_k$. As in Lemma 10 it follows that

$$A_{OO} - A_{LO} \leq c - B_{OO} - \sum_{k=1}^{j-2} a_k < c - \sum_{k=1}^{j-2} b_k - \left(c - \sum_{k=1}^{j-2} b_k - \bar{a}_D^1 \right) = \bar{a}_D^1 \leq \frac{1}{3}\,Opt \tag{11}$$

and we are done.

The other reason can be that according to the definition of L, P_b has some "blocking" item $\tilde{b}$ available fulfilling (7) and (8) for $j' = j-2$ with $\tilde{a} = \bar{a}_D^1$. Note that according to (7) the items with weight $\sum_{k=1}^{j-2} b_k + \tilde{b}$ constitute a feasible solution for P_b in round $j-1$ even against a solution currently better than L and clearly better than O by definition of j (recall Lemma 11). Therefore, the condition of Lemma 10 is fulfilled and we are done since $\bar{a}_D^1 \leq \frac{1}{3}Opt$.

Case 1.2: $\bar{a}_D^1 < a_{j-1}$. In this case a_j is the only item selected by L, but not by O, which is smaller than the largest item in D. Hence, the difference between the weights of L and O after round j can be at most $\bar{a}_D^1 - a_j$, since all other items in D only diminish this difference. Formally, $\sum_{k=1}^{j} \bar{a}_k - \sum_{k=1}^{j} a_k \leq \bar{a}_D^1 - a_j$.

As in Case (ia), there are two possible reasons why L did not choose $\bar{a}_D^1$ in round j but settled for the smaller item a_j. Either $\bar{a}_D^1$ could not be added in round j. Then we can repeat the arguments of (10) and (11) verbatim exchanging $j-2$ by $j-1$ and we are done.

Or P_b has again some "blocking" item $\tilde{b}$ available to prevent the pair $\bar{a}_D^1$ and a_j, i.e. fulfilling (7) and (8) for $j' = j-1$ with $\tilde{a} = \bar{a}_D^1$. Recall that in Case (ib) we have $\sum_{k=1}^{j} \bar{a}_k \le \sum_{k=1}^{j-1} a_k + \bar{a}_D^1$. It follows from (7) that P_b could select the items $b_1, \ldots, b_{j-1}$ and $\tilde{b}$ also against O of P_a and thus the condition of Lemma 10 is satisfied which settles this case.

Case 2: $\bar{a}_D^1 \ge \frac{1}{3}Opt$. If $a_2 \ge \frac{1}{3}Opt$ then $a_1 + a_2 \ge \frac{2}{3}Opt$ and we are done. Hence, we can assume $a_2 < \frac{1}{3}Opt < \bar{a}_D^1$. Furthermore, we can also assume that a_1 was not selected by O. Assume otherwise: Then O contains both a_1 and $\bar{a}_D^1$ and this pair is also a feasible pair for L to consider in the first round. (If P_b were able to block this pair, it would do so and O could not select both). If $a_1 \ge \bar{a}_D^1$ then $a_1 + \bar{a}_D^1 \ge \frac{2}{3}Opt$ and we are done. If $a_1 < \bar{a}_D^1$ then $\bar{a}_D^1$ and a_1 are a better pair than the pair selected by L in the first round consisting of a_1 and some smaller item in contradiction to the definition of L.

Case 2.1: $|D| = 1$. In order to satisfy (9) we must have $a_1 < \bar{a}_D^1$ in this case. Therefore, $\bar{a}_D^1$ and a_2 are both contained in O and they are a better feasible pair than a_1 and a_2. Since P_b could not prevent this pair, we have again a contradiction to the definition of L.

Case 2.2: $|D| \ge 2$. We can assume $\bar{a}_D^2 < \frac{1}{3}Opt$, because otherwise L could have selected $\bar{a}_D^1$ and $\bar{a}_D^2$ in the first round and reach at least $\frac{2}{3}Opt$.

If $a_1 < \bar{a}_D^1$ then $\bar{a}_D^1$ and a_2 would be a better pair than a_1 and a_2, but they are not selected by L in the first round. This is due to the fact that P_b is able to block this pair by some item $\tilde{b}$ with $\bar{a}_D^1 + a_2 + \tilde{b} > c$ and $\bar{a}_D^1 + \tilde{b} \le c$ Thus, $A_{OO} \le c - \tilde{b} < \bar{a}_D^1 + a_2$ since $\tilde{b}$ fits also against the largest item of O. Since $\bar{a}_D^1 < \frac{2}{3}Opt$, this implies $a_2 > \frac{1}{3}Opt$ and thus $a_1 + a_2 > \frac{2}{3}Opt$ and we are done.

If $a_1 > \bar{a}_D^1$ then let a^* be the largest item from $a_2, \ldots, a_j$ not contained in O with $a^* < \bar{a}_D^2$, i.e. $a^* = \max\{a_k \mid a_k < \bar{a}_D^2, k = 2, \ldots, j\}$. Because of $a_1 > \bar{a}_D^1$ such a a^* must exist in order to satisfy (9).

If a^* was selected by L in some round $j' < j$, we can simply repeat the arguments of Case (ia) for j' instead of $j-1$ and with $\bar{a}_D^2$ replacing $\bar{a}_D^1$. Recalling that $\bar{a}_D^2 < \frac{1}{3}Opt$ we reach the desired result.

If a^* is selected in round j, i.e. $a^* = a_j$, we know from the definition of a^* that all other items selected by L, but not by O, are larger than $\bar{a}_D^2$. Since also $a_1 > \bar{a}_D^1$, the difference between the weights of L and O up to round j can be at most $\bar{a}_D^2 - a^*$, in analogy to Case (ib).

Either item $\bar{a}_D^2$ could not be added by L in round j at all, which means that

$$\sum_{k=1}^{j-1} a_k + \bar{a}_D^2 + \sum_{k=1}^{j-1} b_k > c.$$

From Lemma 11 we have $B_{OO} \ge \sum_{k=1}^{j-1} b_k$. In analogy to (11) with $j-1$ replacing $j-2$ we get $A_{OO} - A_{LO} \le \bar{a}_D^2 < \frac{1}{3}Opt$ and we are done.

Or $\bar{a}_D^2$ would fit in round j and we proceed analogous to Case (ia). Since L selected a_j in round j together with some other, smaller item, we know that the better pair $\bar{a}_D^2$ and a_j was not chosen in round j, although $\bar{a}_D^2$ would fit on its own. Thus, there must be again some blocking item $\tilde{b}$ available for P_b fulfilling (7) and (8) for $j' = j-1$ with $\tilde{a} = \bar{a}_D^2$.

It follows from (7) that

$$B_{OO} \geq \sum_{k=1}^{j-1} b_k + \tilde{b}$$

because this solution is feasible for P_b in round j even against a solution with weight $\sum_{k=1}^{j-1} a_k + \bar{a}_D^2 \geq \sum_{k=1}^{j} \bar{a}_k$, i.e., a solution at least as good as O (up to round j). Thus, we can apply Lemma 10 with $\tilde{a} = \bar{a}_D^2 \leq \frac{1}{3} Opt$.

All together we have shown $\rho_{LO} \leq \frac{2}{3}$.

The following Example 5 is a straightforward extension of Example 3. It gives an instance with parameter $\varepsilon > 0$ where $\lim_{\varepsilon \to 0} \rho_{LO} = \frac{2}{3}$ thus completing the proof of the theorem.

Example 5. *Consider an instance of our problem in which the capacity is $c = 1$ and the items weights are as follows.*

item	1	2	3	4	5
N_a	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3} - \varepsilon$	$\frac{1}{3} - \varepsilon$	$\frac{1}{3} - \varepsilon$
N_b	2ε	ε	ε		

In the first two rounds P_a by the Look-ahead Greedy *strategy selects items 1 and 2 with total weight $A_{LO} = \frac{2}{3}$ while P_b gains 3ε. After the second round P_a cannot select another item.*

An optimal strategy would select items 3, 4 and 5 with a total weight of $A_{OO} = 1 - 3\varepsilon$. □

*3.3. Performance of the k-*Look-ahead Greedy *Algorithm*

It is natural to expect that the performance of the Look-ahead Greedy heuristic should improve the more items one includes in the look ahead set, i.e. $\rho_{k\text{-LO}}$ increases in k. When moving from $k = 1$ to $k = 2$ this was shown to be true in the previous sections. The general case could be seen as being related to the construction of a polynomial time approximation scheme (PTAS) for the Subset Sum and the Knapsack problem, where subsets of a certain cardinality ℓ are enumerated and the larger ℓ, the smaller the resulting relative error ε.

Surprisingly, the following example shows that this is not the case. Instead, it can be shown that the worst case performance bound of $\frac{2}{3}$ given in Theorem 9 is also an upper bound for the k-Look-ahead Greedy algorithm for arbitrary $k \geq 3$.

Theorem 12. *The performance of the k-*Look-ahead Greedy *algorithm $k-L$ for $k \geq 3$ is bounded by*

$$\rho_{k\text{-}LO} \leq \frac{2}{3}\,.$$

Proof. The theorem can be proven by considering an instance with capacity $c = 12$, $\delta \ll \varepsilon$, $k \in \{3, \ldots, n_a\}$ and the following items weights.

item	1	2	3	4	5	6	...	n_a
N_a	$2+\varepsilon$	$2+\varepsilon$	$2-\varepsilon$	$2-\varepsilon$	$2-\varepsilon$	δ	...	δ
N_b	6	$3+\varepsilon$	$3+\varepsilon$	$2+\frac{1}{2}\varepsilon$				

P_a is able to choose any pair of items from the set $\{1, 2, 3, 4, 5\}$ together with items of size δ in the first round, since they cannot be blocked by P_b. However P_a is neither able to choose any set containing items $1, 2, 3$ (which would exceed the capacity) nor can P_a choose items 3, 4 and 5, P_b can block them with items 1 and 4. Hence the set of items considered by P_a in the first round of any k-Look-ahead Greedy contains one item of weight $2+\varepsilon$ and two items of weight $2-\varepsilon$. Thus, item 1 is selected in the first round. A residual capacity $\bar{c} = 10 - \varepsilon$ remains for P_b who has three possibilities to react. The resulting games are listed as columns in the following table:

round 1 of P_a	$2+\varepsilon$	$2+\varepsilon$	$2+\varepsilon$
round 1 of P_b	6	$3+\varepsilon$	$2+\frac{1}{2}\varepsilon$
$\bar{c}$	$4-\varepsilon$	$7-2\varepsilon$	$8-\frac{3}{2}\varepsilon$
best tuple of P_a	$2-\varepsilon, 2-\varepsilon$	$2+\varepsilon$, δ	$2+\varepsilon, 2-\varepsilon$
round 2 of P_a	$2-\varepsilon$	$2+\varepsilon$	$2+\varepsilon$
$\bar{c}$	2	$5-3\varepsilon$	$6-\frac{5}{2}\varepsilon$
round 2 of P_b		$3+\varepsilon$	$3+\varepsilon$
$\bar{c}$	2	$2-4\varepsilon$	$3-\frac{7}{2}\varepsilon$
round 3 of P_a	$2-\varepsilon$	δ	$2-\varepsilon$
round 3 of P_b			
$A_{k\text{-LO}}$	$6-\varepsilon+(n_a-3)\delta$	$4+2\varepsilon+(n_a-2)\delta$	$6+\varepsilon+(n_a-3)\delta$
$B_{k\text{-LO}}$	6	$6+2\varepsilon$	$5+\frac{3}{2}\varepsilon$

It turns out that the optimal strategy for P_b against the k-Look-ahead Greedy of P_a is given by selecting item 2 in the first round as represented in the second column of the table. Thus, we get $A_{k\text{-LO}} \approx 4$

The following table shows the optimal strategy of P_a against an optimal strategy of P_b. P_a starts the game by selecting an item of weight $2-\varepsilon$ leaving a residual capacity $\bar{c} = 10 + \varepsilon$ for P_b. For P_b three possibilities remain:

round 1 of P_a	$2-\varepsilon$	$2-\varepsilon$	$2-\varepsilon$
round 1 of P_b	6	$3+\varepsilon$	$2+\frac{1}{2}\varepsilon$
$\bar{c}$	$4+\varepsilon$	7	$8+\frac{1}{2}\varepsilon$
round 2 of P_a	$2+\varepsilon$	$2-\varepsilon$	$2+\varepsilon$
$\bar{c}$	2	$5+\varepsilon$	$6-\frac{1}{2}\varepsilon$
round 2 of P_b		$3+\varepsilon$	$3+\varepsilon$
$\bar{c}$	2	2	$3-\frac{3}{2}\varepsilon$
round 3 of P_a	$2-\varepsilon$	$2-\varepsilon$	$2+\varepsilon$
round 3 of P_b			
A_{OO}	$6-\varepsilon+(n_a-3)\delta$	$6-3\varepsilon+(n_a-3)\delta$	$6+\varepsilon+(n_a-3)\delta$
B_{OO}	6	$6+2\varepsilon$	$5+\frac{3}{2}\varepsilon$

Again, the optimal strategy for P_b is given by selecting item 2 in the first round as illustrated in the second column. We get $A_{OO} \approx 6$ which completes the proof. □

4. The Centralized Perspective

As it is often done in game theoretic settings, we can put the outcome of the game by two competing and selfish agents in perspective to a centralized view, where a single decision maker makes all selections for both item sets N_a and N_b. The goal of such a centralized decision is the maximization of the total weight obtained from both item sets. Clearly, the centralized decision has to select items from N_a and N_b in turn as in the underlying Subset Sum game.

It is easy to see that the computation of such a globally optimal solution weight W^* is an $\mathcal{NP}$-hard problem since it contains the classical Subset Sum problem (SSP). A reduction can be obtained similar to the proof of Proposition 1.

In algorithmic game theory, the Prize of Anarchy is a widely used concept to analyze the difference between a global optimum and the outcome arising from the combined solutions of selfish agents. For the Subset Sum game, we compare the optimal weight generated by the central decision to the outcome obtained by the two agents each following its own optimal strategy (cf. Section 2.1). A simple example shows that the Prize of Anarchy can be arbitrarily high.

Proposition 13. *There exist instances where*

$$\frac{W^*}{A_{OO}+B_{OO}} \to \infty\,.$$

Example 6. *Consider an instance with capacity $c = 1$ and the following sets of items.*

item	1	2
N_a	2ε	ε
N_b	$1-\varepsilon$	ε

A centralized optimal solution would select item 2 *from* N_a *and item* 1 *from* N_b *in the first round and reach* $W^* = 1$. *An optimal strategy by* P_a *would surely start with item* 1 *which leaves to* P_b *only the selection of item* 2 *and* $A_{OO} + B_{OO} = 3\varepsilon$.

The central decision maker could also consider the game as a bicriteria optimization problem where the items from each agent's set constitute one objective. The decision problem whether a certain pair of weights (W_a, W_b) can be reached is again $\mathcal{NP}$-complete from SSP.

From an approximation point of view, it is not sufficient to apply a fully polynomial approximation scheme (FPTAS) for each (SSP) associated to each agent since one has to take the rounds with alternating selections from both item sets into account. However, we can consider the *cardinality constrained subset sum problem* (kSSP), where at most k items can be selected. Following the dynamic programming approach in [4] and assuming integer weights, we can compute for each agent every reachable pair (ℓ, W) with $\ell = 1, \ldots, n_a$ resp. n_b and $W = 1, \ldots, c$. Then we search for the best combination of two solutions with equal cardinality (= number of rounds) such that agent's P_a weights reach at least W_a and agent's P_b weights reach at least W_b but their sum does not exceed c.

Now we can transform this pseudopolynomial exact dynamic programming solution procedure into an FPTAS by the usual scaling techniques (cf. [4]) and thus answer the approximation version of the above question whether a solution with weights (A, B) exists with $(1-\varepsilon)W_a \leq A \leq W_a$ and $(1-\varepsilon)W_b \leq B \leq W_b$ in time polynomial in the size of the encoded input and $1/\varepsilon$.

5. Conclusions

In this paper we have analyzed a game theoretic variant of the well known Subset Sum problem. It appears natural to extend the addressed Subset Sum game to a Knapsack Game by introducing profits for all items. In this case, the two agents would strive to maximize the total profit of their selected items while the weights still have to obey the capacity restriction.

There are two natural approaches to extend the GREEDY or k-LOOK-AHEAD GREEDY algorithms to a Knapsack Game: one may, in each step, choose the most "efficient" items (according to their profit to weight ratio), or one tries to gain as much profit as possible by simply choosing the item with largest profit.

It can easily be shown that the k-LOOK-AHEAD GREEDY ($k \geq 2$) – which stepwise lays its focus on the best k-tuple according to the *sum of the profit to weight ratios* – has an arbitrarily bad performance bound. The same example works also for $k = 1$ which corresponds to the GREEDY algorithm.

Example 7. *Consider the following instance of the problem with* $c = 1$.
Sorting by profit to weight ratios, P_a *would identify the items* $1, \ldots, k$ *as the best* k*-tuple and select one of them in the first round. This leaves* P_b *the chance to submit its only item. The residual capacity of* $(k-1)\varepsilon$ *can be used by* P_a *to*

item	1	...	k	$k+1$
N_a profit	2ε	...	2ε	1
weight	ε	...	ε	$1-\varepsilon$
N_b profit	1			
weight	$1-k\varepsilon$			

select all remaining items $2,\ldots,k$. *Hence, the game stops with a total profit of* $2k\varepsilon$ *for* P_a *against any strategy of* P_b. *An optimal strategy of* P_a *would select item* $k+1$ *in first round and item* 1 *in the second round gaining a profit of* $1+2\varepsilon$ *while* P_b *cannot select any item.*

The following example shows that also the k-Look-ahead Greedy algorithm for $k \geq 1$, i.e. including the pure Greedy algorithm, has an arbitrarily bad performance bound when focusing on the k-tuple with the *highest total profit* in each round.

Example 8. *Consider the following instance with* $n \gg k$ *and* $c = 1+\varepsilon$.

item	1	...	k	$k+1$	...	n
N_a profit	$k+\frac{k+1}{n-k}$	...	$k+\frac{k+1}{n-k}$	$1+\frac{1}{n-k}$	...	$1+\frac{1}{n-k}$
weight	$\frac{1}{k}$	...	$\frac{1}{k}$	$\frac{1}{n-k}$	...	$\frac{1}{n-k}$
N_b profit	1					
weight	ε					

Sorting by profits, similarly to Example 7, P_a *may select item* 1 *in the first round while* P_b *selects its only item. In all* $k-1$ *remaining rounds,* P_a *chooses a* k*-tuple with all remaining items from* $2,\ldots,k$ *complemented by smaller items and thus selects a large item in each round yielding a total profit of* $k^2 + \frac{k(k+1)}{n-k}$ *for any strategy* S *of* P_b. *An optimal strategy of* P_a *would select items* $k+1,\ldots,n$ *and gain a profit of* $(n-k)+1$. *For* n *tending to infinity, the performance of the profit based on* k*-Look-Ahead Greedy becomes arbitrarily bad.*

A possible topic for further research is to consider, as in [13, 14], a different game theoretic variant of the Subset Sum problem where the round-robin mechanism is replaced by a central decision mechanism which picks only one of the two items selected by the two agents in each round (cf. Section 1.1).

Another direction for further research is to study, as in the Admission Control problem, the design of mechanisms for managing agents requests in order to optimize an objective function representing a fairness criterion.

Acknowledgement

We would like to thank Christian Klamler (University of Graz) and Andrea Pacifici (Università degli Studi di Roma "Tor Vergata") for fruitful discussions and valuable comments.

Ulrich Pferschy and Joachim Schauer were supported by the Austrian Science Fund (FWF): [P 23829-N13] "Graph Problems with Knapsack Constraints".

Andreas Darmann was supported by the Austrian Science Fund (FWF): [P 23724-G11] "Fairness and Voting in Discrete Optimization".

References

[1] G. Aggarwal, J.D. Hartline. Knapsack auctions. Proceedings of the 17th annual ACM-SIAM Symposium on Discrete Algorithms, 1083–1092, (2006).

[2] M.H. Ahmed. Call admission control in wireless networks: a comprehensive survey. IEEE Communications Surveys & Tutorials, 7(1), 50–69 (2005).

[3] L. Brotcorne, S. Hanafi, R. Mansi. A dynamic programming algorithm for the bilevel knapsack problem. Operations Research Letters 37, 215–218, (2009).

[4] A. Caprara, H. Kellerer, U. Pferschy, D. Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. European Journal of Operational Research 123, 333–345, (2000).

[5] L. Epstein, E. Kleiman, J. Mestre. Parametric packing of selfish items and the subset sum algorithm. WINE'09, Lecture Notes in Computer Science 5929, 67–78 (2009).

[6] L. Epstein, E. Kleiman. Selfish bin packing. Algorithmica 60, 368–394 (2011).

[7] U. Faigle, W. Kern. On some approximately balanced combinatorial cooperative games. Mathematical Methods of Operations Research 38, 141–152 (1993).

[8] U. Faigle, W. Kern. Approximate Core Allocation for Binpacking Games. SIAM Journal of Discrete Mathematics 11(3), 387-399 (1998).

[9] M. Fujimoto, T. Yamada. An exact algorithm for the knapsack sharing problem with common items. European Journal of Operational Research 171, 693–707 (2006).

[10] M. Ghaderi, R. Boutaba. Call admission control in mobile cellular networks: a comprehensive survey. Wireless Communications and Mobile Computing 6(1), 69–93 (2006).

[11] M. Hifi, H. M'Hallab, S. Sadfi. An exact algorithm for the knapsack sharing problem. Computers and Operations Research 32, 1311–1324 (2005).

[12] V. Liberatore. Scheduling jobs before shut-down. Nordic Journal of Computing 7(3), 204–226 (2000).

[13] C. Marini, G. Nicosia, A. Pacifici, U. Pferschy, Strategies in competing subset selection. Annals of Operations Research, to appear, DOI: 10.1007/s10479-011-1057-2, (2012).

[14] G. Nicosia, A. Pacifici, U. Pferschy. Competitive subset selection with two agents. Discrete Applied Mathematics 159(16), 1865–1877 (2011).

[15] N. Nisan, T. Roughgarden, E. Tardos, V.V. Vazirani, eds. *Algorithmic Game Theory*, Cambridge University Press (2007).

[16] M.J. Osborne. *An Introduction to Game Theory*, Oxford University Press (2004).

[17] S. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. IEEE/ACM Transactions on Networking 3(6), 819–831 (1995).

[18] B. Yolken, N. Bambos. Game-based admission control for wireless systems. Proceedings of the 4th Annual International Conference on Wireless Internet (WICON 2008), 59, Maui, USA (2008).