ROMA TRE
UNIVERSITÀ DEGLI STUDI

Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79
00146 Roma, Italy

Minimum cost subset selection with two competing agents

Claudia Marini[1] Gaia Nicosia[1] Andrea Pacifici[2] Ulrich Pferschy[3]

RT-DIA-179-2010 **October, 2010**

(1) Università Roma Tre,
Via della Vasca Navale, 79
00146 Roma, Italy

(2) Università di Roma "Tor Vergata",
Via del Politecnico, 1
00133 Roma, Italy

(3) Karl-Franzens-Universität
Universitätsstrasse, 15
A-8010 Graz, Austria

ABSTRACT

We address an optimization problem in which two agents, each with a set of weighted items, compete in order to minimize the total weight of their *solution sets*. The latter are built according to a sequential game consisting in a fixed number of rounds. In every round each agent submits one item that may be included in its solution set. We study two natural rules to decide which item between the two will be included and, for both rules, we deal with the problem from different perspectives.

From a centralized point of view, we investigate (*i*) the structure and the number of efficient (i.e. Pareto optimal) solutions, (*ii*) the complexity of finding such solutions, (*iii*) the best-worst ratio, i.e. the ratio between the efficient solution with largest and smallest total weight, and (*iv*) existence of Nash equilibria.

Moreover, we address the problem from a strategic point of view, that is finding the best moves for one agent against the opponent, in two distinct scenarios. We consider *preventive* or *minimax* strategies, optimizing the objective of the agent in the worst case, and *best-response* strategies, where the items submitted by the opponent are known in advance in each round.

Keywords: Multi-agent optimization, Combinatorial game theory, Pareto optimality, Computational complexity, Minimax strategies, Online algorithms.

1 Introduction

In this paper we study a multi-agent problem motivated by certain organizational processes in clerical personnel management. We focus on the following situation: There are two agents, each of them owning one of two disjoint sets of weighted items. The agents have to select items from their set for putting them in a *joint solution set*. This process proceeds in a fixed number of rounds. In every round each of the two agents selects exactly one of its items and submits the item for possible inclusion in the solution set. A central decision mechanism chooses one of the items as "winner" of this round depending on the rule in force. The winning item is permanently included in the solution set. We consider two versions of the problem, depending on whether the losing item is permanently discarded or can be reused in the succeeding rounds. Each agent wants to minimize its total solution value which is given by the total weight of its items included in the solution set. We assume complete information, i.e. both agents know all items' weights.

This problem is motivated by the following application scenario. Each of two departments of a company has its own qualified personnel to perform its relevant tasks. From time to time, the company head management temporarily requires workers for a set of similar tasks to be executed. For each of these tasks, the two departments are obliged to propose a candidate worker from their own personnel. The company management chooses, for each task, the most suitable candidate between the two, according to different criteria. From the department point of view, it is beneficial to minimize personnel loss (i.e. the number of workers or their *total worth* in terms of their abilities, skills or relevance). In a first scenario (Rule 1), the management prefers the more skilled worker while in other cases—for instance, when tasks do not require qualified personnel, or when internal costs proportional to skills are incurred and the management wants to keep its costs low—the coopted worker is the less skilled one (Rule 2).

1.1 Related Literature

In the last decade multi-agent systems have received an increasing amount of attention in the optimization literature. In these systems two or more autonomous decision makers (agents) want to pursue their own objectives while sharing common resources. Clearly, whenever conflicts occur, the agents may want to negotiate over a suitable subset of the feasible agreements where a better solution for one agent necessarily results in a worse solution for the other. Papers dealing with multi-agent systems mainly focus on the determination of these kind of solutions, i.e. non-dominated or Pareto-optimal solutions (see [4, 6, 11] for examples of competing versions of facility location, assignment, and scheduling problems).

Another class of problems related to multi-agent systems and to our problem are those addressed in the research field of algorithmic game theory. A combinatorial game typically involves two agents, usually called *players*, alternatingly playing in well-defined moves with perfect information (i.e. all players know all information about rules and data). In this context, it is interesting to define agents strategies, i.e. how to best play the game against the opponent [5].

The problem we address in this paper can be regarded as a single-suit card game in which each of two players chooses a card from its hand. In [14] the authors study a zero-sum game in which the cards are submitted simultaneously, the highest value card wins, each card can be used only once, and the players want to maximize the total value of the won cards. In [9, 16] the so called *whistette* game is addressed. There is a totally ordered suit of $2n$ cards, distributed between the two players. The player who has the lead plays first on each trick. The player with the highest card wins a trick and obtains the lead. Players want to maximize the number of won tricks.

Another problem strictly related to ours is addressed in [10], where the authors investigate optimal strategies on how to choose a player (item) for the next match (round) in a game consisting of a sequence of matches. Two types of games are considered, given a winning probability for every pair of competing players. In the first type, after each match, the loser is eliminated from the list of remaining players while the winner remains in the list. In the second type, both players are eliminated after each match.

In addition, our problem can be viewed as a special knapsack game where two agents try to fit their items in a bounded common solution set in order to maximize their profits. In our case, we have a unit size for the items and a special mechanism to decide which items fit, i.e. are accepted in the common knapsack. Although the problem that we address in this work is relatively new, 0–1 knapsack problems (KP) in a multi-decision environment have been considered in the literature for two decades: from game-theoretic to auction applications there is a variety of papers dealing with this classical combinatorial optimization problem. Hereafter, we limit to report a few of them.

A related problem in which different players try to fit their own items in a common knapsack is the so called *knapsack sharing problem* studied by several authors (see for instance [7, 8]). A single objective function that tries to balance the profits among the players is considered in a centralized perspective. Knapsack problems are also addressed in the context of auctions. For instance, in [1], an application for selling advertisements on Internet search engines is considered. In particular, there are n agents wishing to place an item in the knapsack and each agent gives a private valuation for having an item in the knapsack, while each item has a publicly known size. In [15] a new model called the two-group knapsack game is introduced. Moreover, in [3], a two-agent knapsack problem where one agent (the leader) controls the capacity of a knapsack, and the other (follower) solves the resulting knapsack problem, is tackled by dynamic programming techniques.

Finally, a different version of the same problem we present in this paper, where agents want to *maximize* the total weight of their winning solution sets, is addressed in [12, 13].

1.2 Problem Statement

Let A and B indicate the two agents. Each agent owns a set of n items, where item i of agent A (B) has a nonnegative weight a_i (b_i). Throughout this paper we assume the items to be sorted in decreasing order of weights, i.e. $a_1 \geq a_2 \geq \ldots \geq a_n$ resp. $b_1 \geq b_2 \geq \ldots \geq b_n$. Sometimes we will identify items by their weight. All information about the input list of items is public.

We consider a game performed over c rounds: In each round both of the two agents simultaneously submit one of their items, say a_i and b_j. We consider the two most natural rules for deciding which of the two submitted items wins and is added to the solution set.

Rule 1 (R1): if $a_i \geq b_j$ then A wins;

Rule 2 (R2): if $a_i \leq b_j$ then A wins[1].

This problem can be represented by a graph model. Each agent's item is associated to a node of a weighted complete bipartite graph $G = (V^A \cup V^B, E^A \dot\cup E^B)$. An arc (i, j) belongs to E^A or to E^B depending on the winner of a comparison of a_i and b_j, which of course depends on the applied selection rule. Every pair of items (a_i, b_j) submitted simultaneously in one round can be represented by an arc (i, j) between the two corresponding nodes. Hence, any solution may be represented as a set M of c arcs on G. The total weight of items in the solution for agent A is given by $W^A(M) = \sum_{ij \in M \cap E^A} w_{ij}$ and that of B by $W^B(M) = \sum_{ij \in M \cap E^B} w_{ij}$.

Moreover, under both rules, we deal with two different scenarios.

Single-use items: each losing item is discarded and cannot be submitted a second time;

Reusable items: each losing item can be reused for submission in the succeeding rounds.

Note that, in the first scenario (single-use items), the solution M corresponds to a c-matching on the above defined bipartite graph.

In conclusion, we tackle four different versions of the minimum cost competitive subset selection problem (CSM), which we denote as $CSM(\alpha, \beta)$, with $\alpha \in \{R1, R2\}$ depending on the rule in force and $\beta \in \{\text{single}, \text{reuse}\}$ depending on the treatment of the losing items.

1.3 Our Contribution

The problem we deal with in this paper is viewed from different perspectives. First, in Section 2, we consider it from a centralized point of view as a bicriteria optimization problem and study

- the structure of efficient i.e. Pareto optimal solutions;
- bounds on the *best-worst ratio*, i.e., the ratio between the efficient solutions with largest and smallest total weight;
- the computational complexity of finding one and/or all Pareto optimal solutions.

[1] To get rid of ties, in this paper we assume that all items are different from each other.

These results are summarized in Table 1, where column "PO recogn." refers to the complexity of the problem of finding Pareto optimal solutions, column "# PO sol." reports the bounds on the number of efficient solutions, and column "Best-Worst Ratio" provides upper bounds on those ratios. However, note that the proof techniques are quite similar to those presented in [12] and [13], where the maximization version of CSM is studied.

Problem type	PO recogn.	# PO sol.	Best-Worst Ratio
$CSM(R1, single)$	NP-hard	$\Omega(2^c)$	2
$CSM(R1, reuse)$	poly	$O(c)$	unbounded
$CSM(R2, single)$	poly	$O(c)$	unbounded
$CSM(R2, reuse)$	poly	$O(c)$	unbounded

Table 1: Results for the centralized scenario.

We then address the problem of devising strategies, that is algorithms that suggests to an agent which item to submit at each round, in order to minimize the agent's weight under different information scenarios.

In Section 3, we consider two different "knowledge scenarios" for an agent, say B. In the first one, the strategy of agent A (the opponent) is completely unknown and we look for an algorithm guaranteeing the minimum weight in the worst possible case for B. In game theory literature, this is often referred to as a *minimax strategy*. We show that such a strategy always exists and it is also quite easy to find it, with the exception of the single-use case under Rule 2, for which a polynomial time algorithm based on dynamic programming is developed.

In the second scenario, agent B knows in advance the item submitted by the opponent A in each round before making its own move. We will try to answer the question how B should select its items under such an advantageous asymmetry of information. In this particular context, the strategy adopted by agent B is referred to as *best-response strategy* (see Section 4). In case of Rule 2, we are always able to devise an optimal strategy for B, while, we show that under Rule 1 this is, in general, not possible. Therefore, we provide algorithms achieving an a-priori worst-case guarantee on the quality of the solution found. The results concerning best-response strategies are summarized in Table 2.

Problem type	LB	UB
$CSM(R1, single)$	2	4-apx
$CSM(R1, reuse)$	$+\infty$	-
$CSM(R2, single)$	opt	opt
$CSM(R2, reuse)$	opt	opt

Table 2: Results on best-response strategies.

2 Efficient solutions

In this section we investigate the solution structure in an offline perspective, i.e. we consider the problem from a centralized and static point of view and look for Pareto efficient solutions.

Using the same terminology as in multicriteria optimization, a solution M is called *Pareto efficient* or simply *efficient* or *nondominated* if there exist no solution M' such that $W^A(M) \geq W^A(M')$, $W^B(M) \geq W^B(M')$ and $W^A(M) + W^B(M) > W^A(M') + W^B(M')$.

In order to better characterize the set of efficient solutions in terms of the agents objectives, hereafter we are interested in estimating the ratio between the global optimum, i.e. the best efficient solution, and the value of *any* efficient solution. This concept is somehow connected to the notion of the *price of anarchy*, studied in algorithmic game theory, which is usually defined as the worst possible ratio between the value of the global (social) optimum and the value of a solution derived by a selfish optimization. Here, we adopt a slightly different notion, called *best-worst ratio* defined as

$$\max_{M \in \mathcal{E}} \left\{ \frac{W^A(M) + W^B(M)}{W^*} \right\},$$

where $\mathcal{E}$ is the set of efficient solutions and W^* is the global optimum value.

In the following, we show that this ratio can be arbitrarily high for all CSM problems but $CSM(R1, single)$.

2.1 Problem $CSM(R1, single)$

Theorem 1 *Problem $CSM(R1, single)$ may have an exponential number of efficient solutions even when $c = n$.*

Proof. Consider the following instance for some $\varepsilon > 0$ and $i = 2, \ldots, c$.

Agent A		**Agent B**	
a_1	0	b_1	ε
a_i	2^{i-2}	b_i	$a_i + \varepsilon$

Table 3: Instance of Theorem 1

For any set $S \subseteq \{2, \ldots, c-1\}$ consider the solution M where an item a_i loses and, at the same time, b_i wins if and only if $i \in S$. It is always possible to construct such an M: clearly, a_1 always loses; then, if $i \in S$, a_i is matched to b_i, else ($i \notin S$) a_i is matched to some b_j, with $j < i$, $j \notin S$.

Neglecting the ε values, the sum of the weights of the winning items $W^A(M)+W^B(M)$ is constant and equals $2^{c-1} - 1$. Therefore, for any possible choice of S, it is possible to obtain an efficient solution since the resulting values of $W^A(M)$ are all different. Thus, there are exponentially many efficient solutions. □

We now show that, in the single-use items case under Rule 1, the problem of deciding whether a certain given objective for each agent can be achieved is $\mathcal{NP}$-complete by reduction from Partition. This implies that finding efficient solutions under Rule 1 is in general an $\mathcal{NP}$-hard task. To this purpose, we consider the following decision problem.

> Recognition $CSM(R1, single)$:
> **Instance**: positive integers a_i and b_i, $i = 1, \ldots, n$; two positive values W^A and W^B.
> **Question**: Is there a solution M of $CSM(R2, single)$ such that $W^A(M) \leq W^A$ and $W^B(M) \leq W^B$?

Theorem 2 Recognition $CSM(R1, single)$ *is $\mathcal{NP}$-complete.*

Proof. Consider an instance I of Partition with integer valued items $\{v_1, v_2, \ldots, v_n\}$ and $\sum_{i=1}^{n} v_i = V$. Assume $v_1 < v_2 < \ldots < v_n$.

For $T > v_n$ and $\varepsilon < 1/(n+1)$, build an instance I' of Recognition $CSM(R1, single)$ as follows:

$$(I') \quad \begin{array}{llll} & a_0 = \varepsilon, & b_0 = T, & \\ & a_i = v_i, & b_i = v_i - \varepsilon, & i = 1, \ldots, n, \\ & W^A = \frac{V}{2} & W^B = \frac{V}{2} + T. & \end{array}$$

Let I be a YES-instance of Partition. Then it is easy to build a solution M of I' such that $W^A(M) \leq V/2$ and $W^B(M) \leq V/2 + T$. Let $S \subseteq \{1, 2, \ldots, n\}$ be such that $\sum_{i \in S} v_i = V/2$ and $\bar{S} = \{0, 1, \ldots, n\} \setminus S$. The required solution M is given as follows: Each item i of A with $i \in S$ is matched to the corresponding item i of B, while each item j of B with $j \in \bar{S}$ is matched to

$$\arg\max\{a_h \mid h \in S, a_h < b_j\}.$$

Clearly, the two agents total weights are: $W^A(M) = \sum_{i \in S} a_i = V/2$ and $W^B(M) = \sum_{i \in \bar{S}} b_i \leq V/2 + T$.

Now, let I' be a YES-instance of Recognition $CSM(R1, single)$ and M be a solution of I' such that $\bar{S} \subseteq \{0, 1, 2, \ldots, n\}$ is the set of "winning" items of agent B. Note that $0 \in \bar{S}$, since item 0 of B always wins. So,

$$W^B(M) = \sum_{i \in \bar{S}} b_i = \sum_{i \in \bar{S}} a_i - \varepsilon \leq V/2 + T$$

and $W^A(M) \leq V/2$. Note that, since each winning item i of A is matched to a losing item of B with smaller weight, the total weight L^B of the "losing" items of agent B is such that $L^B = \sum_{i \in S} b_i \leq W^A(M) \leq V/2$. Since, $L^B + W^B(M) = V + T - (n+1)\varepsilon$, the v_i are all integers and $\varepsilon < 1/(n+1)$, then $\sum_{i \in S} v_i = \sum_{i \in \bar{S} \setminus \{0\}} v_i = V/2$. □

We now give an upper bound for the ratio between the efficient solutions with largest and smallest total weight.

Theorem 3 *The best-worst ratio of $CSM(R1, single)$ is 2.*

Proof. Let $X = \{a_{n-c+1}, a_{n-c+2}, \ldots, a_n\} \cup \{b_{n-c+1}, b_{n-c+2}, \ldots, b_n\}$. The global optimum value W^* is the solution with minimum total weight. On the other hand, the value of the efficient solution $\bar{M}$ with the *worst* global value can be bounded from above. In fact, in $\bar{M}$, the set X can be partitioned into the two sets X^V and X^L of winning and losing items. Clearly $|X^V| = |X^L| = c$. Since, under Rule 1, each winning item is matched to an item with smaller weight, then $W(X^V) \geq W(X^L)$. Therefore, $W^A(\bar{M}) + W^B(\bar{M}) = W(X^V) \geq W(X)/2 \geq W(X^L)$. This holds for any solution, therefore also the global optimum satisfies this condition. Hence, $W^* \geq W(X)/2$ and since $W(X) \geq W(X^V)$, we have $W^* \geq W(X^V)$. The statement follows. □

2.2 Problem $CSM(R1, reuse)$

Theorem 4 *Problem $CSM(R1, reuse)$ has at most $c+1$ efficient solutions. These can be computed in polynomial time.*

Proof. Suppose that a feasible solution exists such that agent A wins k rounds and therefore B wins the remaining $c-k$ rounds, for some $k \in \{0, 1, \ldots c\}$. It is not hard to see that such solution exists if there are at least k items of A greater than b_n and $c-k$ items of B greater than a_n. If this condition holds, we can construct an efficient solution as follows. W.l.o.g. we assume $a_n > b_n$ (otherwise we can exchange the roles of A and B). Let

$$a_{i^*} = \min_{i=1\ldots n} \{a_i \ : \ a_i > b_n\}.$$

In the first k rounds, b_n is matched to $a_{i^*}, a_{i^*+1}, \ldots a_{i^*+k-1}$, while in the succeeding $c-k$ rounds a_n is matched to $b_n, b_{n-1}, \ldots b_{n-c-k+1}$. □

Regarding the best-worst ratio, the following result holds.

Theorem 5 *The best-worst ratio of $CSM(R1, reuse)$ can be arbitrarily high.*

Proof. Consider the following instance with $c = n = 2$.

agent A		agent B	
a_1	2	b_1	T
a_2	ε	b_2	1

Clearly, the global optimum value $W^* = 3$, while, there is an efficient solution $\bar{M}$ of value $T+2$. The best-worst ratio can therefore be arbitrarily high for increasing values of T. □

2.3 Problem $CSM(R2, single)$

Theorem 6 *Problem $CSM(R2, single)$ has at most $c+1$ efficient solutions. These can be computed in polynomial time.*

Proof. Consider a solution M in which agent A wins k rounds, if it exists. For M to be PO, then A wins with the smallest k items, $a_n, a_{n-1}, \ldots a_{n-k+1}$ and B wins the remaining $c-k$ rounds with the smallest items $b_n, b_{n-1}, \ldots b_{n-c+k+1}$.

Then, if M exists for $k \geq 0$, the structure of the resulting solution is such that item a_{n-k+i} is matched to b_i for $i = 1, \ldots, k$ (if $k > 0$), and if $k < c$, a_i is matched to $b_{n-c+k+i}$ for $i = 1, \ldots, c-k$. See Table 4 for an illustration when $n = c = 7$ and $k = 3$.

Computing the solution values for all $c+1$ efficient solutions can be done in linear time by one scan through the list of items after sorting the items. Giving also the corresponding pairs of items explicitly as output would require $O(n^2)$ time. □

Regarding the best-worst ratio, the following result holds.

Agent A	a_7	a_6	a_5	a_1	a_2	a_3	a_4
Agent B	b_1	b_2	b_3	b_7	b_6	b_5	b_4

Table 4: Structure of a PO solution for $n = c = 7$ and $k = 3$.

Theorem 7 *The best-worst ratio of* $CSM(R2, single)$ *can be arbitrarily high.*

Proof. Consider the instance with $n = c = 2$ illustrated in Table 5.

Agent A		*Agent B*	
a_1	T	b_1	$T+1$
a_2	2	b_2	1

Table 5: Instance proving Theorem 7

Consider the two PO solutions obtained when A submits $\langle a_1,a_2\rangle$ and B answers by submitting either $\langle b_1,b_2\rangle$ or $\langle b_2,b_1\rangle$. The following values are obtained: $W^1 = T + 1$ and $W^2 = 3$, where $W^2 = W^*$.

Hence, the *Best-Worst Ratio* is at least $\frac{W^1}{W^2} = \frac{T+1}{3}$ that becomes unbounded for increasing values of T. □

2.4 Problem $CSM(R2, reuse)$

Arguments similar to those of Theorems 4 and 6 can be used to prove the following result for problem $CSM(R2, reuse)$.

Theorem 8 *Problem* $CSM(R2, reuse)$ *has at most* $c + 1$ *efficient solutions. These can be computed in polynomial time.*

Similarly, as in the previous sections, the following result holds.

Theorem 9 *The best-worst ratio of* $CSM(R2, reuse)$ *can be arbitrarily high.*

Proof. Consider the following instance with $n = c = 2$.

agent A		agent B	
a_1	$T + \varepsilon$	b_1	T
a_2	$1 + \varepsilon$	b_2	1

Clearly, the global optimum value $W^* = 2 + \varepsilon$, while, there is an efficient solution $\bar{M}$ of value $T + 1$. The best-worst ratio can therefore be arbitrarily high for increasing values of T. □

3 Minimax Strategies of Agents

In this section, we want to develop a strategy for B, in all four scenarios, such that its total gain is as small as possible under the worst possible strategy of A. More formally, let $\mathcal{S}_A$ ($\mathcal{S}_B$) be the set of strategies for agent A (B). For every pair of strategies $S_A \in \mathcal{S}_A, S_B \in \mathcal{S}_B$, $M(S_A, S_B)$ denotes the resulting solution. Then, a *minimax strategy* $MMS_B \in \mathcal{S}_B$ of agent B is defined as follows:

$$MMS_B := \arg \min_{S_B \in \mathcal{S}_B} \left(\max_{S_A \in \mathcal{S}_A} W^B(M(S_A, S_B)) \right)$$

The definition of MMS_A is analogous.

Since A is not interested in its own solution but only in worsening the solution of B as much as possible, depending on the considered scenario, we can immediately establish some useful properties.

In the single-use case, the order in which B submits its items is irrelevant. This can be easily seen observing that A can pick the worst possible answer for any c-tuple of items submitted by agent B, and execute this answer according to the ordering of B. Such an answer could be computed e.g. by solving

a maximum weight matching problem. Therefore, when $c = n$, since B cannot decide anything in this case, the question of finding a minimax strategy is irrelevant.

On these grounds, it is easy to observe that, under Rule 1, if $n > c$, the minimax strategy of B for $CSM(R1, single)$ consists of choosing its c smallest items, i.e. $S(B) = \{b_n, \ldots, b_{n-c+1}\}$. Under Rule 2, the selection of $S(B)$ is more involved and can be done by the algorithm described in Section 3.1.

For the reusable case, the problem becomes trivial under both rules: under Rule 1, i.e. $CSM(R1, reuse)$, if $b_n < a_n$, B may always lose by submitting b_n in each round. Otherwise, A is always able to lose by submitting a_n and B must win with its c smallest items. Similarly, under Rule 2, i.e. $CSM(R2, reuse)$, if $b_1 > a_1$, B may always lose by submitting b_1 in each round. Otherwise, A is always able to lose and B must win with its c smallest items.

Hereafter, we describe a minimax strategy for $CSM(R2, single)$, based on dynamic programming.

3.1 A Minimax Algorithm for $CSM(R2, single)$

Due to the above considerations, we assume $n > c$ for the remainder of this section. It is also obvious that A will always submit its c largest items to hurt B as much as possible.

Given a certain list of c submissions of B we can completely describe the outcome of the game also without an explicit matching computation. A will consider its items in decreasing order and try to lose. Therefore, A determines for each item the largest winning item of B which was not assigned in the matching before. Consider, for illustration, a_1: All items $b_i > a_1$ will lose anyway against A. So A should submit a rather small item to win against these sure losers and not waste its largest items for winning. Instead, A will attempt to match its largest item a_1 with a winning item of B. In order to keep the chances for losing alive for its smaller items, A should pick the largest possible winner of B. This is explained in detail in the following Algorithm 1.

Algorithm 1 Malicious strategy for agent A under Rule 2.

```
1: i ← 1    {counter for items of A}
2: j ← 1    {counter for items of B}
3: while j ≤ c do
4:    while b_j > a_i and j ≤ c do
5:       j ← j + 1    {find the largest available winning item of B}
6:    end while
7:    if j ≤ c then
8:       match a_i with b_j    {in this case a_i ≥ b_j and A loses}
9:       i ← i + 1
10:      j ← j + 1
11:   end if
12: end while
13: match items a_i, ..., a_c in an arbitrary way with unmatched items of B. {Any remaining items of A
    win against B in any case.}
```

Based on the structure resulting from the malicious strategy of A described in Algorithm 1 we now develop an optimal strategy for B.

For notational conveniences we define the set of items of B with a weight between two items of A as $S_i := \{b_j \mid a_{i-1} > b_j \geq a_i\}$ for $i = 2, \ldots, c$, $S_1 := \{b_j \mid b_j \geq a_1\}$ and $S_{c+1} := \{b_j \mid a_c > b_j\}$. Note that several sets S_i may be empty.

B can decide on the submission of items from some set S_{i+1} as follows: Items submitted from previous sets $S_1, \ldots, S_i$ either win or lose. In the former case they contribute to the total weight won by B, but their exact values are irrelevant at this later stage. In the latter case the weight of a losing item is again irrelevant. All together only the total weight won so far and the number of submitted items have to be taken into account to decide on the submissions from S_{i+1}. In addition, we also have to know what happened to those items of A with a possibility to force B to win with items in S_{i+1}, i.e. the items $a_1, \ldots, a_i$. In this case, their individual weight is again irrelevant, since they are all larger than items in S_{i+1}. The only interesting fact is how many of them were assigned in Algorithm 1 to be matched in one round against items already submitted by B from $S_1, \ldots, S_i$. Or the other way round, how many of them are still unmatched and would lose against any submission from S_{i+1} by B.

Based on this informal description we will consider all relevant strategies of B in a dynamic programming procedure. We will go through the items of A in decreasing order of weights and update the entries of the following dynamic programming table d for every set S_{i+1} of B.

> $d(i,j,k)$ contains the total weight won by B after considering all items with weight at least a_i, if B submits j items of weight at least a_{i+1} and wins with k of them.

Table d is defined for $i = 1,\ldots,c+1$, $j = 0,1,\ldots,c$ and $k = 0,1,\ldots,c$. Any state $d(i,j,k)$ that is not reachable will be assigned a weight ∞. From A's perspective an entry $d(i,j,k)$ indicates that k items $a_1,\ldots,a_k$ were matched to lose against items submitted by B while the remaining $i-k$ items $a_{k+1},\ldots,a_i$ remain available to lose against future submissions of B with weight less than a_i .

Note that A will follow strictly the procedure outlined above and does not have any strategic decisions left. This means that through its submissions, B also decides on k. In Algorithm 2 we compute the entries

Algorithm 2 Minimax strategy for agent B under Rule 2.

1: $d(i,j,k) \leftarrow \infty$ for $i = 1,\ldots,c+1$, $j = 0,1,\ldots,c$, $k = 0,1,\ldots,c$.
2: $d(1,\min\{|S_1|,c\},0) \leftarrow 0$
3: **for** $i = 1,\ldots,c$ **do**
4: $\quad n' \leftarrow |S_{i+1}|$
5: $\quad$ Denote the items of set S_{i+1} as $\{b'_1,\ldots,b'_{n'}\}$ in decreasing order.
6: $\quad$ **for** $j = 0,1,\ldots,i$ **do**
7: $\quad\quad j' \leftarrow \min\{j,n'\}$
8: $\quad\quad$ **for** $k = 0,1,\ldots,j$ **do**
9: $\quad\quad\quad d(i+1,j,k) \leftarrow \min\{d(i,j,k),$
$\min_{s=1,\ldots,j'}\{\min_{r=0,1,\ldots,\min\{s,k\}}\{b'_{n'-s+1}+\ldots+b'_{n'-s+r}+d(i,j-s,k-r)\}\}\}$
10: $\quad\quad$ **end for**
11: $\quad$ **end for**
12: **end for**

of d in increasing order of i and process one set S_{i+1} in each iteration. We initialize the table by ∞ except for $d(1,\min\{|S_1|,c\},0)$ which corresponds to the set S_1 (line 2). Clearly, B will submit all its items with weight larger than a_1 and lose them.

To update the table in line 9 we compute $d(i+1,j,k)$ and consider in each iteration how many items s of S_i agent B might submit and how many items r out of these s would win against A thus contributing to the total weight won by B. Clearly, B will always choose the smallest s items $b'_{n'-s+1}+\ldots+b'_{n'}$ of S_i but cannot avoid to win with the largest r of these. The remaining $j-s$ submitted items with $k-r$ winners among them are taken from the previously considered items recorded in entries $d(i,j-s,k-r)$. Since entries are only evaluated for $k \leq i$ we can be sure to have enough items of A available for losing against these k winners of B.

The optimal strategy follows by backtracking through the sequence of operations yielding the lowest total weight for c submitted items of B, i.e. $\min_{k=0,\ldots,c} d(c+1,c,k)$.

In conclusion the following theorem holds.

Theorem 10 *Algorithm 2 is a minimax strategy for $CSM(R2,single)$.*

4 Best-Response Strategies of Agents

We now consider the scenario where B only knows the submission of A in the current round and has to react immediately before both agents move on to the next round. Again we assume that $n > c$, since otherwise B could compute an optimal matching for all submissions of A beforehand. Note that we do not assume that A follows a rational strategy but may submit any available item.

In the following sections we show that for problems $CSM(R1,single)$ and $CSM(R1,reuse)$ there is no optimal strategy for B. Indeed we can show for problem $CSM(R1,single)$ that no strategy of B can have a *competitive ratio* [2] compared to the best off-line strategy better than 2, for large c. For small c our bound is slightly below 2 starting at the golden ratio for $c = 2$. However, for problem $CSM(R1,reuse)$, such a competitive ratio is unbounded. Moreover, we show that, under Rule 2, it is possible to devise algorithms yielding an optimal off-line solution for Problem $CSM(R2,single)$and $CSM(R2,reuse)$.

4.1 Best-Response for $CSM(R1, single)$

The following lower bound holds for any best-response algorithm for $CSM(R1, single)$.

Theorem 11 *For problem $CSM(R1, single)$ with arbitrary c, no on-line strategy of agent B against an arbitrary strategy of agent A can have a competitive ratio smaller than 2.*

Proof. Consider the following instance of $CSM(R1, single)$ with two parameters q and T. For convenience, items are listed in increasing order of weights, i.e. in decreasing order of indices. We list only the smallest c items of both agents. All $n - c$ remaining items of both agents have weight T. An arbitrary small constant $\varepsilon > 0$ is only used to avoid draws.

item nr.	n	$n-1$	$n-2$	$n-3$	$\dots$	$n-k+1$	$n-k$	$\dots$	$n-c+1$
agent A	$1-\varepsilon$	$q-\varepsilon$	$q^2-\varepsilon$	$q^3-\varepsilon$	$\dots$	$q^{k-1}-\varepsilon$	$q^k-\varepsilon$	$\dots$	$q^{c-1}-\varepsilon$
agent B	1	q	q^2	q^3	$\dots$	q^{k-1}	q^k	$\dots$	q^{c-1}

A submits the smallest item $a_n = 1 - \varepsilon$ in the first round and thus forces B to win in any case. We can distinguish three main cases for a reaction by B:

Case 1. B tries to win with the smallest possible item, i.e. item $b_n = 1$ in the first round. Then A continues to submit a_{n-1} in the second round which gives the same situation as before. In Case 1. we consider the strategy where B sticks to the submission of the smallest possible winning item b_{n-j+1} in the j-th round up to the final round c, where B submits $b_{n-c+1} = q^{c-1}$ thus winning all rounds.

All together in Case 1. B wins $W_1^B = \sum_{i=0}^{c-1} q^i = \frac{q^c-1}{q-1}$. An optimal strategy of B against this sequence of submissions by A would submit $b_{n-c+1} = q^{c-1} = OPT_1^B$ in the first round and manage to lose all remaining rounds.

Case 2. B starts by submitting b_n and continues as in Case 1. but in some iteration k, $k \in \{2, \dots, c-1\}$, B deviates from the previous strategy by submitting some item larger than b_{n-k+1}, i.e. with weight at least q^k. Note that $k \leq c-1$ since B will not submit a larger item in round c where only items with weight T remain. But then A immediately changes its strategy and continues to submit only items with weight T allowing B to lose all remaining rounds. Now we have

$$W_2^B \geq \sum_{i=0}^{k-2} q^i + q^k = \frac{q^{k-1}-1}{q-1} + q^k.$$

For such a strategy of A an optimal answer of B would be to submit $b_{n-k+1} = q^{k-1}$ in the first round and lose all other rounds which yields $OPT_2^B = q^{k-1}$.

Case 3. As a special case of Case 2. B might also submit an item larger than b_n already in the first round, i.e. an item with weight at least q. Then A would continue by submitting only items with weight T. Thus, B loses all remaining rounds attaining $W_3^B \geq q$, while an optimal strategy would be $OPT_3^B = b_n = 1$.

Putting the three cases together we get the following lower bound on the competitive ratio:

$$\max_q \min_{i=1,2,3} \left\{ \frac{W_i^B}{OPT_i^B} \right\} = \max_q \min \left\{ \frac{q^c-1}{q^c-q^{c-1}}, \frac{q^{k+1}-q^k+q^{k-1}-1}{q^k-q^{k-1}}, q \right\}$$

The first expression is monotonically decreasing in q (tending to 1) while the third expression is trivially increasing. The intersection point of these two cases approaches $q = 2$ from below for c tending to infinity. Hence, the best lower bound we can get from this construction is 2 attained for $q = 2$. A straightforward calculation shows that for $q = 2$ the second expression of the minimum resulting from Case 2 is always larger than the first which completes the proof. □

For small c the lower bound is slightly below 2 with a starting point given by the following corollary.

Corollary 12 *For problem $CSM(R1, single)$ with $c = 2$, no on-line strategy of agent B against an arbitrary strategy of agent A can have a competitive ratio smaller than $\phi = \frac{1}{2}(1 + \sqrt{5})$, the golden ratio.*

Algorithm 3 Strategy for agent B responding to the submissions of agent A for Problem $CSM(R1, single)$.

GEOMETRIC GREEDY RESPONSE 1

partition items of B into intervals $I_k = [b_n \cdot 2^{k-1},\ b_n \cdot 2^k)$ for $k = 1, 2, \ldots$.
$R^B \leftarrow \{b_1, b_2, ..., b_n\}$ {available items for B}
$b_{\min} \leftarrow \min\{b_i\ :\ b_i \in R^B\}$
repeat
  let a' be the item submitted by agent A
  if $b_{\min} \leq a'$ **then**
    B submits $b_{\max} \leftarrow \max\{b_i \in R^B \mid\ b_i \leq a'\}$ and loses
    remove $b_{\max}$ from R^B and update $b_{\min}$
  else
    let $b_{\min} \in I_j$ for some $j \in 1, 2, \ldots$.
    $b^j_{\max} \leftarrow \max\{b_i \in R^B \cap I_j\}$
    B submits $b^j_{\max}$ and wins
    remove $b^j_{\max}$ from R^B and update b_{min}
  end if
until all c rounds are finished

Proof. Considering the proof of Theorem 11 for $c = 2$ we can see that Case 2. does not apply. Evaluating the intersection of Cases 1. and 3. for $c = 2$ yields the result. □

In the following we will assume w.l.o.g. that agent B always submits its c smallest items. Clearly, any deviation from this assumption can only worsen the outcome of B.

Theorem 13 *Algorithm* GEOMETRIC GREEDY RESPONSE 1 *has a tight competitive ratio of* 4 *for Problem* $CSM(R1, single)$.

Proof. In the proof we will compare the items submitted by B according to Algorithm GEOMETRIC GREEDY RESPONSE 1 with the submissions of an optimal off-line response strategy OPT. Recall that both strategies will submit the identical set of the c smallest items but possibly in a different ordering. To reduce ambiguity of the optimal strategy we will introduce the following statement **(A1)**:

(A1) *There is an optimal strategy* OPT *such that whenever it is possible for* OPT *to lose a round, it will do so by using the largest possible losing item.*

Note that this property can be shown by a simple exchange argument.

We will consider a strategy OPT fulfilling **(A1)**. Furthermore, we will exclude all rounds from further consideration where B and OPT submit the same item. For ease of notation we will assume $b_n = 1$. Denote by $I(b_i)$ the interval containing some item b_i.

In a generic round A submits a', OPT submits some item b_i and B submits $b_j \neq b_i$. We will denote this outcome by an ordered pair (i, j). A label $\ell(i, j)$ taken from $\{\oplus, \ominus, 0\}$ will represent the outcome of this round regarding the total weights OPT^B resp. W^B collected so far:

$\ell(i, j) = \oplus$ represents an increase of W^B by b_j, while OPT^B remains unchanged or receives an unspecified, smaller increase. $\ell(i, j) = \ominus$ denotes the case that OPT^B gains $b_i > b_j$ while W^B is increased by only b_j in this round. If B receives an increase at most twice as large as the increase of OPT (or none at all) we will set $\ell(i, j) = 0$.

Note that the labels are not used to calculate the actual values of OPT^B and W^B but only to bound the competitive ratio. Clearly, to show a bound ≥ 2 only labels $\oplus$ and $\ominus$ will be relevant.

We generate pairs and their labels in each round according to the following cases:

Case 1. $b_i < b_j$:

Case 1.1. $a' < b_i$, OPT and B win :
We generate a new pair (i, j). If $b_j \in I(b_i)$ we set $\ell(i, j) = 0$, otherwise $\ell(i, j) = \oplus$.

Case 1.2. $a' > b_i$, OPT loses :
If $b_j < a'$ then B loses and we add a new pair (i, j) with $\ell(i, j) = 0$.
If $b_j > a'$ then B wins. We generate a new pair (i, j) with $\ell(i, j) = \oplus$.

Case 2. $b_i > b_j$:

Generate a new pair (i, j). If $a' < b_i$ then OPT wins and we set $\ell(i, j) = \ominus$. Otherwise, OPT and B lose and we set $\ell(i, j) = 0$.

The ordered pairs can always be concatenated into chains in a natural way, i.e. after (i, j), OPT might use b_j in some later round where B submits b_k which results in two consecutive pairs (i, j),(j, k) being part of a chain. Since both strategies use the same set of items, each chain will finally form a cycle. In this way, the outcome of all rounds can be represented by a collection of cycles.

In the following we will show that for the rounds in each cycle we always have $W^B \leq 4 \cdot OPT^B$ which concludes the proof.

Every cycle can be partitioned into increasing and decreasing subchains. More formally, we say that ordered pairs $(i_1, i_2), (i_2, i_3), \ldots, (i_k, i_{k+1})$ form a decreasing (increasing) subchain if $b_{i_\ell} > b_{i_{\ell+1}}$ ($b_{i_\ell} < b_{i_{\ell+1}}$) for $\ell = 1, \ldots, k$. It follows from their definition that labels $\oplus$ (labels $\ominus$) only occur for pairs in increasing (decreasing) subchains.

First, we will show several structural properties.

(i) *If* $b_i > b_j$ *and* $\ell(i, j) = 0$ *then* $I(b_i) = I(b_j)$.

Assume otherwise that $I(b_i) \neq I(b_j)$:
Recall from Case 2 above that both OPT and B lost this round. OPT lost with item b_i. By definition of the algorithm, B would rather submit b_i than b_j as a losing item. Hence, the submission of b_j means that b_i was not available anymore to B but was already used in some previous round and there exists some pair (k, i) from that round. If B had won that previous round, then by definition of the algorithm it would rather have won with b_j (from a smaller interval) which was still available for B in that round. Hence, B must have lost that round with b_i. However, in that round b_i was still available for OPT as a possibility to lose. By **(A1)** also OPT lost that round with $b_k > b_i$ and we have $\ell(k, i) = 0$.

Hence, we have the same situation for (k, i) as we had in the beginning for (i, j), and we can repeat the argument to show the existence of another pair (k', k). Again, for winning B would have had the better possibility to submit b_j and hence B must have lost with b_k. Therefore, also *Opt* must have lost with $b_{k'} > b_k$. This argument can be iterated yielding pairs of increasing weight until we finally run out of items. This yields a contradiction.

(ii) *Each increasing subchain can have at most one pair with label* $\oplus$ *in each interval.*

Assume otherwise: Consider the round were the first pair (i, j) with $\ell(i, j) = \oplus$ was generated with $b_i \in I(b_j)$, i.e. B won with b_j and OPT lost with b_i. By definition of the algorithm all items $b_r \in I(b_j)$ with $b_r > b_j$ were submitted by B before, otherwise the largest item of these would have been chosen by B for winning in this round. Also all items with weight $\leq b_i$ must have been submitted before, since otherwise B could have used them for losing. Hence, there are only items $b_r \in I(b_j)$ left with $b_i < b_r < b_j$. But these can not be part of an increasing subchain containing (i, j).

(iii) *If two increasing subchains are concatenated by a decreasing subchain consisting only of pairs with label* 0*, at most one of the two increasing subchains can have a label* $\oplus$ *in each interval.*

Let (i, j) be the first pair with $\ell(i, j) = \oplus$ generated in a certain interval. As shown above only items $b_r \in I(b_j)$ with $b_i < b_r < b_j$ remain available for submission by B. If a second pair (s, r) with $\ell(s, r) = \oplus$ is generated in a later round as part of a second increasing subchain, then no decreasing subchain can concatenate the two increasing subchains:

Assume otherwise: Then there must exist two pairs (u, v) and (x, y) in the decreasing subchain both with label 0, i.e. all four items are lost, such that $b_v \leq b_s < b_u < b_y < b_r < b_x$, possibly $b_v = b_s$. (The simpler case where $b_u > b_r$ and (x, y) does not appear, follows by a similar argument.) Since B always loses with the largest available item, (x, y) must have been generated before (u, v). (This argument can be applied iteratively over all pairs possibly existing between (x, y) and (u, v) in the decreasing subchain.) If (x, y) was generated before (s, r) then B could have used b_r instead of b_y for losing with a larger item. If (s, r) was generated before (u, v) then B could have used b_v for losing that round. Thus the order of generation of these three pairs yields a contradiction.

Each increasing subchain contains at most one pair with label $\oplus$ in some interval I_k by (ii). It either continues into the next interval I_{k+1} or it ends in I_k. In the latter case it can either be followed by a decreasing subchain reaching into some smaller interval, or it is followed by a series of decreasing and increasing subchains in I_k. By (iii) these subchains can not include another pair with label $\oplus$ in I_k.

Recall from Case 2 and (i) that interval borders can only be crossed (crossing an interval border means that a pair in a subchain consists of two items belonging to neighboring intervals) in a decreasing subchain by pairs with label $\ominus$. We will sum up the contribution of these pairs for a decreasing subchain starting in interval I_s and ending in interval I_t with $s > t$. There is at least one pair with label $\ominus$ starting in I_s and contributing at least 2^{s-1} to the weight of Opt. In each intermediate interval I' between I_s and I_t a pair with label $\ominus$ may end and a new pair start. The contribution to W^B of the pair ending in I' is at most twice as large as the contribution to OPT^B by the starting pair. Hence, these can be omitted in bounding the competitive ratio by 4. Finally, W^B may gain at most 2^t in the last interval.

Because of the cycle property, for each upper interval bound 2^k of I_k for $k = t, \ldots, s-1$ that is crossed by the decreasing subchain, there must be a pair crossing the same interval bound in an increasing subchain. In the worst case, there is a pair with a label $\oplus$ for each such interval crossing, contributing a weight of at most 2^k. The final pair of the increasing subchain reaching into I_s may yield a weight of 2^s. Bounding the competitive ratio of the rounds belonging to one decreasing subchain and the induced increasing subchains yields:

$$\frac{2^t + \sum_{i=t}^{s} 2^i}{2^{s-1}} = \frac{2^t + 2^{s+1} - 2^t}{2^{s-1}} = 4 \tag{1}$$

Again because of the cycle property, there can be no increasing subchain crossing an interval bound without a complementing decreasing subchain. Hence, the above construction includes all pairs with label $\oplus$ for all increasing subchains crossing interval bounds.

If a decreasing subchain does not cross any interval bounds it may also be part of a cycle containing only items from the same interval.

If at least one pair of the decreasing subchain is labeled $\ominus$ (which can be seen as the above case with $s = t$) it is easy to see that in the worst case both Opt and B win an item with weight (roughly) 2^{s-1} for the pair labeled $\ominus$. The single pair of the cycle (recall (ii) and (iii)) labeled $\oplus$ contributes at most 2^s yielding a ratio of 2 for this special case.

If all pairs of the decreasing subchain are labeled 0, then B and Opt use the items contained in this cycle in the same rounds. If it is possible to lose, both will lose with the largest possible item. If it is unavoidable to win, then B will win with the largest possible item in the interval thus keeping the best possible chances to lose further rounds. In this way, B maximizes the number of lost rounds. Hence, Opt will win at least as often as B, while B may win with items at most twice as large as the winning submissions of Opt.

To show that the bound of 4 is tight consider the following instance of $CSM(R1, single)$ with $c = k+2$. For convenience, items are listed again in increasing order of weights, i.e. in decreasing order of indices. We list only the smallest c items of both agents. All $n - c$ remaining items of both agents have weight T. An arbitrary small constant $\varepsilon > 0$ is for technical reasons.

item nr.	n	$n-1$	$n-2$	$n-3$	$\ldots$	$n-k+1$	$n-k$	$n-k-1$
agent A	1	2	4	8	$\ldots$	2^{k-1}	2^k	2^{k+2}
agent B	$2-\varepsilon$	$4-\varepsilon$	$8-\varepsilon$	$16-\varepsilon$	$\ldots$	$2^k-\varepsilon$	$2^k+\varepsilon$	$2^{k+1}-\varepsilon$

In each interval I_ℓ for $\ell = 1, \ldots, k$ there is an item of A at the lower interval end and an item of B at the upper interval end. I_{k+1} is different containing two items of B and one of A, while I_{k+2} contains only an item of A.

A submits its items in strictly increasing order of weights starting with $b_n = 1$. Opt reacts by submitting item b_{n-k} with weight $2^k + \varepsilon$ and wins. In all remaining rounds ℓ, Opt can lose by submitting its smallest remaining item with $2^\ell - \varepsilon < 2^\ell$. Also the final round is lost with $2^{k+1} - \varepsilon$ and we have $OPT^B = 2^k + \varepsilon$.

Algorithm Geometric Greedy Response 1 wins the first round with the largest item in I_1, i.e. $b_n = 2 - \varepsilon$. It continues to win every round ℓ until round $k+1$ where the largest item in I_{k+1} has weight $2^{k+1} - \varepsilon$. The final round is lost with $2^k + \varepsilon$. This yields quite similar to (1)

$$W^B = \sum_{i=1}^{k} (2^i - \varepsilon) + 2^{k+1} - \varepsilon = 2^{k+2} - 1 - (k+1)\varepsilon.$$

For ε tending to 0 the ratio W^B/OPT^B tends to 4 as k (and c) tend to infinity. □

4.2 Best-Response for $CSM(R1, reuse)$

We can show that any best-response algorithm for $CSM(R1, reuse)$ may perform as bad as possible compared to an optimal strategy.

Theorem 14 *For problem $CSM(R1, reuse)$ with $c \geq 3$, no on-line strategy of agent B against an arbitrary strategy of agent A can have a bounded competitive ratio.*

Proof. Consider the following instance of $CSM(R1, reuse)$ with a parameter T. For convenience, items are listed in increasing order of weights, i.e. in decreasing order of indices. An arbitrary small constant $\varepsilon > 0$ is only used to avoid draws[2].

item nr.	n	$n-1$	$n-2, \ldots, n-c+1$	$n-c, \ldots, 1$
agent A	$1-\varepsilon$	$T-\varepsilon$	$T-\varepsilon$	$T^2-\varepsilon$
agent B	1	T	T^2	T^2

A submits the smallest item a_n in the first round and thus forces B to win in any case. Since B has only items of three different weights, there are three main cases to consider for B:

Case 1. At the first round B responds by playing b_n. Now A submits b_{n-1} in the second round. In Case 1.1 B submits b_{n-1} and wins T. A continues to submit all remaining $c-2$ items with weight $T-\varepsilon$. B cannot avoid to win all these rounds each with an item of weight T^2 yielding a total gain of $W_1^B = (c-2)T^2 + T + 1$.

An optimal answer of B against this strategy of A submits b_{n-1} in the first round and loses all subsequent rounds with item b_n. Thus we have $OPT_1^B = T$.

Case 1.2, where B submits an item of weight T^2 in the second round, works in a completely analogous way and yields either the same total weight or even worse $(c-1)T^2 + 1$.

Case 2. At the first round B responds by playing b_{n-1}. In this case A continues to submit all $c-1$ items with weight T^2. B can lose all these rounds e.g. with item b_n yielding $W_2^B = T$.

An optimal strategy would submit b_n in the first round and b_{n-1} in all subsequent rounds which gives $OPT_2^B = 1$.

Case 3. At the first round B responds by playing an item with weight T^2. This case can be handled in the same way as Case 2.

In all cases we get a competitive ratio of at least T for $c \geq 3$ which concludes the proof. □

For the special case $c = 2$ the same weaker lower bound holds as for $CSM(R1, single)$(see Corollary 12).

Theorem 15 *For problem $CSM(R1, reuse)$ with $c = 2$, no on-line strategy of agent B against an arbitrary strategy of agent A can have a competitive ratio smaller than $\phi = \frac{1}{2}(1+\sqrt{5})$, the golden ratio.*

Proof. Consider the following instance of $CSM(R1, reuse)$ with $c = 2$ and $n = 3$, where $k > 1$ is a suitable parameter.

item nr.	1	2	3
agent A	$2k$	$k-\varepsilon$	$1-\varepsilon$
agent B	k	k	1

A submits the smallest item $a_3 = 1-\varepsilon$ in the first round and thus forces B to win in any case. Since B has only items of two different weights, there are only two cases to consider for B:

Case 1. At the first round B responds by playing $b_3 = 1$. Then A submits $a_2 = k-\varepsilon$ and B cannot avoid to win with an item of weight k yielding a total gain of $W_1^B = k+1$.

An optimal answer of B against this strategy of A submits an item of weight k in the first round and loses the other round with b_3. Thus we have $OPT_1^B = k$.

[2]We assume in this paper that all weights are different. To simplify the proof, we use $b_{n-2} = \ldots = b_1$ and $a_{n-1} = \ldots = a_{n-c+1}$ in this example, but different ε values would work just as well.

Case 2. At the first round B responds by playing $b_2 = k$. In this case A submits $a_1 = 2k$ in the second round that B can lose.

An optimal strategy would submit b_3 in the first round and b_2 in all subsequent rounds which gives $OPT_2^B = 1$.

Altogether we get a lower bound for the worst-case competitive ratio of

$$LB = \min\left\{\frac{k+1}{k}, \frac{k}{1}\right\} \tag{2}$$

The maximum of this lower bound is attained if both expressions of the minimum are equal. An elementary calculation yields the solution of this equation as the golden ratio

$$k = \frac{1}{2} \cdot (1 + \sqrt{5}\,). \tag{3}$$

□

Observe that, when $c = 2$, it is possible to design a trivial best-response algorithm attaining a competitive ratio equal to the golden ratio, by using the arguments in the above proof.

4.3 Best-Response for $CSM(R2, reuse)$

In this section we provide an optimal best-response algorithm for problem $CSM(R2, single)$. Before describing the algorithm, simple exchange arguments prove the following lemma.

Lemma 16 *For problem $CSM(R2, single)$, there is an optimal best-response algorithm for agent B such that*

- *when B is forced to win, it uses the smallest available item;*
- *whenever possible, B loses (i.e. there is no convenience to win a round).*

So, for problem $CSM(R2, single)$, the main issue for a best-response algorithm is to decide which item should be used to lose at each round. In fact, submitting a large item may reduce the chances of losing against large items of A in the future. On the other hand, small items can be useful when B is forced to win in a succeeding round.

Therefore, the proposed Algorithm 4 takes into account the above issues, determining at each round the *best losing item*. In particular, when B may lose, the procedure BEST LOSER (Algorithm 5) returns the losing item for agent B as the largest available item $\tilde{b}$ such that:

- B loses the round;
- the number of potential losing items of B larger than $\tilde{b}$ is maximized.

Let us briefly describe Algorithm 4 through a simple example.

Example 17 *Consider the instance depicted in Figure 1 in which, at a certain round (current round) the items of A are $\{19, 15, 12, 10, 5, 4\}$ and those of B are $\{16, 13, 9, 3, 2, 1\}$. Assume A submits item $a' = 5$. Then, B must choose, among the items greater than a', the one to submit to lose the round. According to*

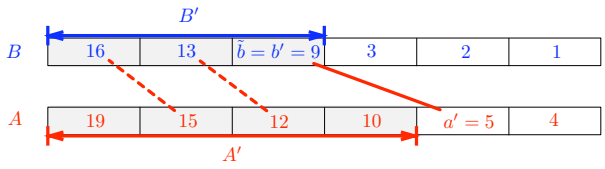

Figure 1: BEST LOSER returns $\tilde{b} = b'$

procedure BEST LOSER, *B decides to submit the item of value 9, since the two larger items of values 16 and 13 can be used to lose against A's items of value 15 and 12 in succeeding rounds. (Note that in this case the procedure* BEST LOSER *terminates because, in the repeat-loop, the variable* `found` *becomes true.)*

Algorithm 4 Best-response algorithm for agent B under Rule 2.

BEST-RESPONSE RULE 2

$R^A \leftarrow \{a_1, \ldots, a_n\}$ {set of remaining items for A}
$R^B \leftarrow \{b_1, \ldots, b_n\}$ {set of remaining items for B}
repeat
  let a' be the item submitted by agent A
  $b_{\min} \leftarrow \min\{b_i \mid b_i \in R^B\}$
  $b_{\max} \leftarrow \max\{b_i \mid b_i \in R^B\}$
  if $b_{\max} > a'$ **then**
    $\tilde{b} \leftarrow$ BEST LOSER(a', R^A, R^B)
    B submits $\tilde{b}$ and loses
  else
    B submits $b_{\min}$ and wins
  end if
  update R^A and R^B
until all c rounds are finished

Algorithm 5 Procedure BEST LOSER returns the item that should be used to lose

BEST LOSER(a', R^A, R^B)

$B' \leftarrow \{b_i \in R^B \mid b_i > a'\}$
$A' \leftarrow A^> = \{a_i \in R^A \mid a_i > a'\}$
`found` $\leftarrow$ false
while (`found` = false) and ($B' \neq \emptyset$) **do**
  $b_i \leftarrow \max\{b_j \in B'\}$
  if $\exists a_j \in A' : a_j < b_i$ **then**
    $a_k \leftarrow \max\{a_j \in A' \mid a_j < b_i\}$ {Potentially matching a_k with b_i.}
    $A' \leftarrow A' \setminus \{a_k\}$
    $B' \leftarrow B' \setminus \{b_i\}$
  else
    `found` $\leftarrow$ true
  end if
end while
return b_i

If item $a_4 = 8$ (instead of $a_4 = 10$), the procedure BEST LOSER *would have "matched" all the items of B' to smaller items of A' (thus terminating with $B' = \emptyset$).*

In both cases the procedure BEST LOSER *returns $\tilde{b} = b_3 = 9$.*

A different situation for agent B is depicted in Figure 2 in which, at the current round the items of A have values $\{19, 15, 14, 7, 5\}$ and those of B $\{16, 13, 9, 4, 3\}$. If A submits item $a' = 7$, B responds with item $b_2 = 13$, since only one larger item ($b_1 = 16$) can be used to lose against larger items of A in succeeding rounds.

We may now prove the main result of this section.

Theorem 18 *Algorithm* BEST-RESPONSE RULE 2 *yields the optimal off-line solution of Problem $CSM(R2, single)$.*

Proof. Hereafter, we prove that Algorithm 4 produces an off-line optimal solution for Agent B by showing that, at each round, B responds in an optimal way to the submission of A.

As already observed, when B is forced to win, the choice of Algorithm 4 consisting in the smallest available item is optimal. We therefore focus our analysis on the rounds in which B can lose.

Let a' be the item submitted by A in the current round. With respect to the sets of available items R^A and R^B in the current round, let $b' = \min\{b_i \in R^B \mid b_i > a'\}$, $B^> = \{b_i \in R^B \mid b_i > b'\}$, $B^< = \{b_i \in R^B \mid b_i < b'\}$, $A^> = \{a_i \in R^A \mid a_i > a'\}$, and $A^< = \{a_i \in R^A \mid a_i < a'\}$.

Depending on the output of Procedure BEST LOSER in the current round, we may distinguish two cases, namely, $\tilde{b} > b'$ (Case 1) and $\tilde{b} = b'$ (Case 2).

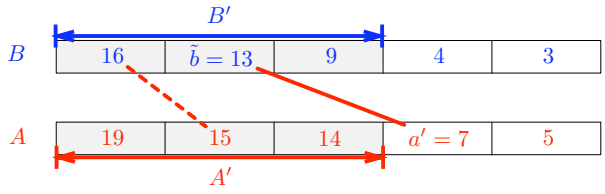

Figure 2: BEST LOSER returns $\tilde{b} > b'$

Case 1. When $\tilde{b} > b'$, we consider two subcases depending on whether $\tilde{b}$ coincides with $b_{\max}$ or not.

If $\tilde{b} = b_{\max}$, then all the items in $A^>$ are larger than those in $B^>$. Observe that, there is no convenience in keeping item $b_{\max}$ for losing or winning in one of the succeeding rounds. In fact, it can be used for losing only when agent A submits items in $A^<$ (and in this case any item not smaller than b' can be used for losing and it does not matter which of them we submit now). On the other hand, if in the succeeding rounds, agent A submits items in $A^>$, then B will be forced to win (therefore $b_{\max}$ is the worst choice).

If $b' < \tilde{b} < b_{\max}$, then the items of $B^>$ strictly greater than $\tilde{b}$ can be used in the succeeding rounds to lose with the items of $A^>$. Note that, as in the previous case, there is no convenience in keeping item $\tilde{b}$ for losing or winning in one of the succeeding rounds. In fact, if agent A submits items in $A^<$, any item between b' and $\tilde{b}$ is still available to lose (and, again, it does not matter which of them we submit now). On the other hand, if agent A submits items in $A^>$ and agent B can lose, then B might as well use one of the items of $B^>$ greater than $\tilde{b}$. If agent A submits items in $A^>$ and agent B must win, it is convenient to keep a smaller item to win, for instance b'.

Case 2. If $\tilde{b} = b'$, there exists a matching between *all* the items of $B^>$ and items of $A^>$ such that all the items of $B^>$ lose, so every item of $B^>$ may be useful to lose against the items of $A^>$ in the succeeding rounds.

Since agent B according to Algorithm 4 plays item b', we must show that this choice is optimal, that is, there is no convenience in keeping b' for using it in a successive round, say the i-th round. Therefore, suppose that an optimal off-line algorithm for B (OPT, in the following) keeps b' for losing (*Case 2.a*) or winning (*Case 2.b*) in round i. We show that in *Case 2.a*, OPT cannot obtain a better solution than Algorithm 4, while *Case 2.b* may never occur.

In *Case 2.a*, let us assume that during round i, A submits an item $\bar{a} < b'$, and B is forced to win (since b' and all items of $B^>$ have been used to lose, they are not available anymore). Since OPT kept b' for losing in round i, then it must win at least one extra round (with respect to Algorithm 4) before i. In conclusion, the number of winning rounds of B is not greater than in OPT and therefore losing with b' against a' is optimal (since B always wins with the smallest available item).

In *Case 2.b*, at round i, agent B according to Algorithm 4 wins with $\bar{b}$ against $\bar{a}$ and $\bar{a} > \bar{b} > b'$ (otherwise, if $\bar{b} < b'$ the choice of B in the current round is obviously optimal). Moreover, since $b' > a'$ one has that $\bar{a} \in A^>$. In addition, we have that $\bar{b}$ is the smallest item available in round i (Line 11 of Algorithm 4). Let $\alpha = |A^>| - |B^>|$. Since $\tilde{b} = b'$, we have that $\alpha \geq 0$. Note that α items of $A^>$ cannot be matched with larger items of $B^>$ (see Figure 3).

At round i all items of $B^<$ have been previously submitted and therefore matched to items of A: Let γ indicate the number of items in $B^<$ that lost and β the number of remaining items in $B^<$ that won between the current and the i-th rounds. Therefore:

$$|B^<| = \gamma + \beta \tag{4}$$

Note that these γ items lost against items in $A^<$, while the other β items won against items in $A^>$. (In fact, by contradiction, if an item of $B^<$ is forced to win against an item of $A^<$ in a round preceding the i-th, then B cannot lose against it and hence $B^> = \emptyset$. However, this contradicts the fact that at round i, B has at least one item available in $B^>$, namely $\bar{b}$.)

During the rounds preceding i, the submitted items of $B^>$ were all used to lose (note that these items could not be used to win). Let δ indicate the number of items in $B^>$ that lost against items in $A^>$ and ξ those that lost against items in $A^<$ between the current and the i-th rounds.

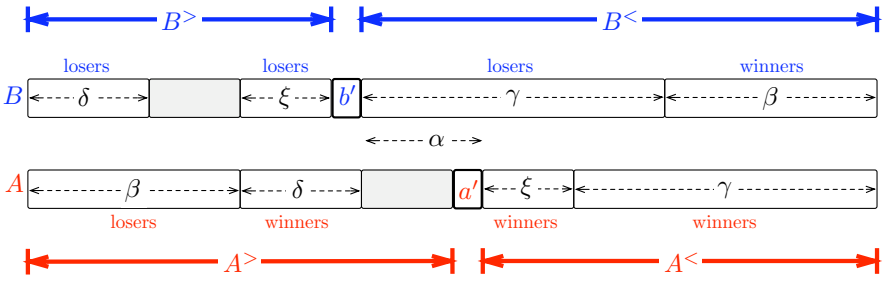

Figure 3: Illustration of Case 2.b with $\beta > \alpha$

If $\alpha = 0$, i.e. $|A^>| = |B^>|$, for each item in $A^>$ there is a corresponding item in $B^>$ that can be used to lose. Therefore, b' cannot be used to win in round i (i.e. $\alpha = 0$, is not consistent with *Case 2.b*).

If α is strictly positive, i.e. $|A^>| > |B^>|$, we distinguish three possible cases depending on the value of α with respect to that of β.

First, let $\beta = \alpha$. Observe that $|A^>| = |B^>| + \alpha$ and $|A^>| + |A^<| = |B^>| + |B^<| = n - 1$ which implies that $|B^<| = |A^<| + \alpha$. Hence, recalling equation (4), we have $\gamma = |A^<|$. Therefore, all losing items of $B^<$ lose against *all* items of $A^<$. Moreover, agent B wins against exactly α items of $A^>$. As a consequence, all remaining available items of $B^>$ may lose against all remaining items of $A^>$, and therefore b' cannot be used to win in round i (i.e. $\beta = \alpha$ is not consistent with *Case 2.b*).

Secondly, let $\beta > \alpha$. Clearly, between the current and the i-th round, B wins against some items of $A^>$ which the procedure BEST LOSER, in the current round, matches to items of $B^>$. This is due to the fact that B must have used some of these "matched" items in rounds preceeding the i-th necessarily for losing against items in $A^<$ and therefore $\xi > 0$. In fact, these ξ items in $B^>$ could not have been used:

- to lose against items in $A^>$, nor
- to win since, from the above considerations (equation (4) and the fact that the submitted items of $B^>$ were all used to lose), the number of winning items of B between current and i-th round is equal to β. Now, at round i, *all* items of $B^<$ have been previously submitted and in particular, β items of B have already won and, as a consequence, all the other items (and the ξ in particular) are losing.

This implies, together with the definition of ξ, that $\xi = \beta - \alpha$ and therefore $|A^<| = \xi + \gamma$ (see Figure 3). Similarly as in the previous case, all remaining available items of $B^>$ may lose against all remaining items of A. Therefore b' cannot be used to win in round i (i.e. $\beta > \alpha$ is not consistent with *Case 2.b*).

Thirdly, let $\beta < \alpha$. This implies that $\gamma > |A^<|$. However, this cannot occur since no item in $B^<$ can lose against items larger than a'. Again, this is not consistent with *Case 2.b*.

From the above case analysis we can conclude that Algorithm 4 behaves optimally in each round. □

4.4 Best-Response for $CSM(R2, reuse)$

Designing a best-response algorithm for problem $CSM(R2, reuse)$ is trivial. Differently from the single-use case where the main issue concerns which item to lose with, here losing items can be reused making the decision much easier.

Hence, a trivial optimal algorithm can be summarized as follows:

In a generic round, A plays a'. Let $b_{\max}$ ($b_{\min}$) be largest (smallest) available B item. If $b_{\max} > a'$, B loses with any losing available item (e.g., $b_{\max}$); otherwise B wins with $b_{\min}$.

5 Conclusions

In this paper we addressed a multi-agent optimization problem where two agents compete to add items in a solution set. We consider four different versions of the problem and, for all versions, we give results concerning efficient solutions and the definition of strategies for optimizing the objective of a single agent against the other.

This work puts several directions forward for future research. Of course, whether the competitive ratio of best-response strategies for problem $CSM(R1, single)$ (Section 4.1) could be improved is still an interesting question.

In addition, investigating new rules for deciding which of the two submitted items wins (e.g., the larger item wins, unless it is more than twice as large as the smaller item), or studying strategies under different knowledge scenarios (e.g., each agents, in alternating rounds, knows in advance the item submitted by the opponent before making its own move) are indeed challenging open issues.

References

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

[2] A. Borodin, R. El-Yaniv. Online Computation and Competitive Analysis, Cambridge University Press, UK, (1998)

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

[4] J. Cardinal, M. Hoefer. Non-cooperative facility location and covering games. Theoretical Computer Science 411, pp. 1855–1876 (2010)

[5] E.D. Demaine. Playing games with algorithms: Algorithmic combinatorial game theory, Lecture Notes in Computer Science 2136, pp. 18–33 (2001)

[6] G. Felici, M. Mecoli, P.B. Mirchandani, A. Pacifici. Equilibrium in a two-agent assignment problem, International Journal of Operational Research 6, pp. 4–26 (2008)

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

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

[9] J. Kahn, J.C. Lagarias, H.S. Witsenhausen. Single-suit two-person card play, International Journal of Game Theory 16, pp. 291–320 (1987)

[10] N. Katoh, J. Koyanagi, M. Ohnishi, T. Ibaraki. Optimal strategies for some team games, Discrete Applied Mathematics 35, pp. 275–291 (1992)

[11] J. Y.-T. Leung, M. Pinedo, G. Wan. Competitive two-agent scheduling and its applications, Operations Research 58, pp. 458469 (2010)

[12] G. Nicosia, A. Pacifici, U. Pferschy. Subset weight maximization with two competing agents, Lecture Notes in Computer Science 5783, pp. 74-85 (2009)

[13] G. Nicosia, A. Pacifici, U. Pferschy. Competitive subset selection with two agents, accepted for publication on Discrete Applied Mathematics, (2010)

[14] K.H. Schlag, A. Sela. You play (an action) only once, Economic Letters 59, pp. 299–303 (1998)

[15] Z. Wang, W. Xing, S.-C. Fang, Two-group knapsack game, Theoretical Computer Science 411, pp. 1094–1103 (2010)

[16] J. Wästlund. A solution of two-person single-suit whist, The Electronic Journal of Combinatorics 12, R43 (2005)