On regular and approximately fair allocations of indivisible goods

Diodato Ferraioli, Laurent Gourvès, Jérôme Monnot

On Regular and Approximately Fair Allocations of Indivisible Goods

Diodato Ferraioli*
DIAG
Sapienza Università di Roma
00185 Rome, Italy
ferraioli@dis.uniroma1.it

Laurent Gourvès
Jérôme Monnot
CNRS, LAMSADE
Université Paris-Dauphine
75775 Paris Cedex, France
{laurent.gourves,jerome.monnot}@dauphine.fr

ABSTRACT

An active stream of research is devoted to the design of polynomial approximation algorithms for the fair allocation of indivisible goods. Central to this field is the MaxMin Allocation problem, for which there is a significant gap between known approximation and inapproximability results. Closing this gap is a stimulating challenge.

To this end, we consider a *regular* version of MaxMin Allocation where each agent must receive exactly k goods, for a given integer k. We call this problem k-division. The analysis of this problem allows us to highlight two interesting features of the classical MaxMin Allocation problem. First, we show a close connection of the problem with matroid theory. This connection provides us an exact algorithm for a special case of k-division and a $1/k$-approximation algorithm for general inputs. Moreover, we show that the difficulty of the MaxMin Allocation may be caught by an apparently simpler problem, namely the k-division problem in which an agent's utility for a single good can only take one value out of three.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences – Economics; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms, Economics, Theory

Keywords

Computational social choice, Fairness, Approximation

1. INTRODUCTION

The problem of fairly allocating a given set of items to a given set of agents is very old, and it is treated in many fields including political science, economics (social science in particular) and, more recently, theoretical computer science. Numerous variants of the problem exist. Are the items goods or chores? Are they divisible (e.g. lands) or indivisible (e.g. jobs)? Is it possible to make monetary compensations? Even for the notion of fairness, the context leads to distinct, and still significant, formulations. For example, one can require that no agent finds her share less valuable than the share of another one; this is commonly known as *envy-freeness*. However, it is sometimes possible to satisfy a criterion of fairness and make a poor utilization of the resources at the same time. Hence, efficiency comes naturally into play.

Several concepts which capture efficiency and fairness at the same time have been defined. Here we focus on one of these, known as *maxmin*, whose goal is to maximize the welfare of the least happy agent. The typical problem concerning the maxmin concept has been named MaxMin allocation [4, 8, 2]. Here, we are given a set A of n agents, a set R of m indivisible resources and a binary relation $E \subseteq A \times R$ where $(i, r) \in E$ means that resource r is available to agent i. There are non-negative utilities $u_i(r)$ for each $i \in A$ and $r \in R$ such that $(i, r) \in E$. A feasible solution is an allocation of the resources $\mathcal{R} = (R_1, \ldots, R_n)$ where R_i are the resources received by agent i, such that $R_i \cap R_j = \emptyset$ and $(i, R_i) = \{(i, r) : r \in R_i\} \subseteq E$ for each $1 \leq i, j \leq n$. The goal is to maximize $u(\mathcal{R}) = \min_{i \in A} u_i(R_i)$ where $u_i(R_i)$ is defined as $\sum_{r \in R_i} u_i(r)$ (some variants in which $u_i(R_i)$ is not additive have been also studied; see e.g. [8]). A lot of work has been devoted to the MaxMin allocation problem (see the related work section). It is indeed known that the problem is hard to solve and several approximation algorithms have been proposed. Nevertheless, a large gap exists between the best inapproximability bound and the approximation ratio of the best known approximation algorithm.

In order to gain insight on the nature of this gap, we consider a special restriction of the MaxMin allocation problem. Specifically, we consider a set A of n agents, a set R of kn indivisible resources and a binary relation $E \subseteq A \times R$. Moreover, each agent $i \in A$ has a non negative utility $u_i(r)$ for resource $r \in R$ which is available to her. We want to find an allocation of the resources $\mathcal{R} = (R_1, \ldots, R_n)$ such that $|R_i| = k$, $R_i \cap R_j = \emptyset$, $(i, R_i) \subseteq E$ for each $1 \leq i, j \leq n$. The goal is to maximize $u(\mathcal{R}) = \min_{i \in A} u_i(R_i)$ where $u_i(R_i)$ is defined as above. Hence, this problem, that we name as the k-division problem, is just constraining MaxMin Allocation to assign exactly k items per agent.[1]

*Supported by the EU FET project MULTIPLEX 317532.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

[1]Note that this constraint has a severe effect on the role of the binary relation E. Indeed, for the MaxMin Allocation problem, any instance with $E \subset A \times R$ can be reduced to an equivalent instance $(A, R, E', (u)_{i \in A})$ for which the relation E' is complete (that is each resource is

Even if this restricted problem has been introduced for getting an insight in the original problem, it turns out to be interesting on its own. Indeed, giving to each agent a fixed number of items is a basic additional notion of fairness. One can think of n kids making a joint celebration of their birthday. Their presents are gift-wrapped and it is preferable that the number of parcels, which is a kid's first estimation, is exactly the same for everyone. Another example comes from program committees: here reviewers are usually asked to express preferences about the papers they prefer to review and there are constraints on the possible assignments (e.g., a reviewer cannot have a close connection with the assigned paper's authors). Nevertheless, the committee must assign the same number of papers to any reviewer and this assignment should be fair with respect to the expressed preferences. This application also motivates a limitation of our problem to the case in which utility may assume only few values: indeed, preferences are usually expressed through forms allowing only a limited set of inputs.

Related work. The literature on social welfare optimization in multi-agent resource allocation with fairness constraints is vast and we refer the interested reader to the survey by Nguyen et al. [11]. Here we focus on the previous work about approximation algorithms for the MaxMin Allocation problem, where an algorithm is ρ-approximate if $APX(I)/OPT(I) \geq \rho$ holds for any instance I ($APX(I)$ and $OPT(I)$ being the values returned by the approximation and the optimal algorithms, respectively).

Bezáková and Dani consider the MaxMin allocation problem in which m indivisible goods are distributed to n agents [4]. They give two deterministic approximation algorithms with *a priori* incomparable ratios. The first one is matching-based and it has a ratio of $1/(m-n+1)$. The second consists of modeling the problem with an integer linear program (ILP). An appropriate rounding of the optimum of the relaxed ILP provides an approximate solution. The approximation ratio is the optimal value of the relaxed ILP minus the largest value an agent can have for a single item.

On the negative side, Bezáková and Dani show that, following a reduction of 3-Matching, no ρ-approximation algorithm with $\rho > \frac{1}{2}$ exists, unless **P**=**NP** [4]. This means that the problem is computationally hard, and even a polynomial algorithm that produces a solution arbitrarily close to the optimum is unlikely. A similar result, with a different reduction, is given independently by Golovin [8]. Two other $\frac{1}{2}$-inapproximability results for a restricted case where each item has a nonzero value for at most two agents exist [6, 3].

Golovin [8] proposes an algorithm that builds an allocation for which a $1-1/p$ fraction of the agents has utility at least a p^{-1} fraction of the optimum. Bansal and Sviridenko [2] study a restricted case in which $u_i(r) \in \{0, p_r\}$. They provide an LP relaxation, referred to as *configuration LP*, from which they derive an approximation algorithm with factor $\Omega(\log\log\log n/\log\log n)$. An even more restricted case, called "Big goods/Small goods", is shown $\Omega(1/\sqrt{n})$-approximable in [8]. Khot and Ponnuswami [10] and Asadpour and Saberi [1] provide approximation algorithms for the general MaxMin Allocation problem with ratios $(2n-1)^{-1}$ and $\Omega\left(1/\left(\sqrt{n}\log^3 n\right)\right)$, respectively. Note that these algorithms improve on the original $\frac{1}{m-n+1}$ bound only for m sufficiently large. The last developments are due to Chakrabarty et al. [6] who give a quasi-polynomial $\tilde{\Omega}(1/n^{\epsilon})$-approximation algorithm, for any $\epsilon \leq \frac{\log\log m}{\log m}$.

Our contribution. The analysis of k-division highlights interesting features of the MaxMin allocation problem.

First, we show there exists a close connection between the problem and matroid theory (also in [9], matroid theory is linked to the allocation of indivisible goods with the aim to build solutions with worst case guarantees for agents). Indeed, we depart from previous techniques (e.g. linear programming) and, drawing on matroid theory, we propose an approximation algorithm with ratio $1/k$.

Next, we investigate instances in which the agents' utilities for the items can take a limited number of values. In particular, we provide an exact algorithm for the k-division problem in which an agent's utility for a resource can only take two values. This algorithm will be useful also for improving the approximation ratio for instances in which the agent's utility can take more than two values. These improved ratios turns out to be far from being tight, even for only three utility values. However, we highlight that closing the approximation gap for this apparently simple problem may be a breakthrough shading a completely new light on the long-standing gap in the approximation ratio of the MaxMin allocation problem.

2. PRELIMINARY RESULTS

Checking if an instance $(A, R, E, (u_i)_{i\in A})$ of the *k-division* problem contains a feasible solution can be done in polynomial time. Indeed, the problem is equivalent to finding a flow of value kn in the network $G = (V, E)$ such that the set of vertices V contains a source vertex s, a target vertex t, a vertex v_i for each agent $i \in A$ and a vertex w_r for each resource $r \in R$, and the set of edges E contains an edge (s, v_i) for each $i \in A$, an edge (v_i, w_r) for each $(i, r) \in E$, and an edge (w_r, t) for each $r \in R$, each of capacity one.

Moreover, finding the max-min allocation among the feasible ones can be done in polynomial time if $k = 1$. Indeed, the problem is equivalent to finding the bottleneck weighted matching in a bipartite graph $B = ((A, R), E)$ where the edge (i, r) weights $u_i(r)$. Unfortunately, next theorem shows that it is not possible to extend this result to higher k.

Theorem 1. *For any $k \geq 2$, the k-division problem is strongly **NP**-complete even if E is complete and there is $B \geq 0$ such that for every resource $r \in R$, $\sum_{i\in A} u_i(r) = B$.*

Proof. For $k = 2$, we reduce the 2-numerical matching with target sums (2NMTS in short) problem. The inputs of 2NMTS is a sequence $a_1, \ldots, a_n$ of n positive integers with $\sum_{i=1}^{n} a_i = n(n+1)$ and $1 \leq a_i \leq 2n$ for $i = 1, \ldots, n$. We want to decide if there are two permutations f and g of the integers $\{1, \ldots, n\}$ such that $f(i) + g(i) = a_i$ for $i = 1, \ldots, n$. 2NMTS is strongly **NP**-complete [13].

available to each agent) simply by setting utility zero for each $(i, r) \in E' \setminus E$. This equivalence does not hold in the k-division problem. To see this, consider two agents $A = \{1, 2\}$, four resources $r_1, \ldots, r_4$ (so, $k = 2$) and $E = \{(1, r_1), (1, r_2), (2, r_2), (2, r_3), (2, r_4)\}$. Let $u_1(r_1) = u_2(r_2) = M > 2$ while $u_1(r_2) = u_2(r_3) = u_2(r_4) = 1$. There is a unique feasible allocation, namely $\mathcal{R} = (R_1, R_2)$ with $R_1 = \{r_1, r_2\}$ and $R_2 = \{r_3, r_4\}$, of value $u(\mathcal{R}) = 2$. If we complete this instance as described above, then the instance $\mathcal{R}^* = (R_1^*, R_2^*)$ with $R_1^* = \{r_1, r_3\}$ and $R_2^* = \{r_2, r_4\}$ becomes feasible. But $u(\mathcal{R}^*) = M > u(\mathcal{R})$.

Consider an instance $(a_i)_i$ of 2NMTS. We build an instance $(A, R, (u_i)_{i\in A})$ of the 2-DIVISION problem as follows: there are n agents $A = \{1, \ldots, n\}$ and $2n$ resources $R = \{r_j, r'_j : j = 1, \ldots, n\}$. For $j = 1, \ldots, n$, the utility of agent $i \in A$ for resources $r_j, r'_j \in R$ is given by $u_i(r_j) = u_i(r'_j) = n + j - \frac{a_i}{2}$. This construction can be done within polynomial time. In particular, the utilities are polynomially bounded, since $0 < u_i(r) < 2n$ for every $r \in R$ and every $i \in A$.

We claim that $a_1, \ldots, a_n$ is a `yes`-instance for the 2NMTS problem if and only if there is an allocation of the resources $\mathcal{R} = (R_1, \ldots, R_n)$ with $|R_i| = 2$ such that $u(\mathcal{R}) \geq 2n$.

Clearly, if there are two permutations f and g of $\{1, \ldots, n\}$ such that $f(i) + g(i) = a_i$ for $i = 1, \ldots, n$, then by setting $R_i = \{r_{f(i)}, r'_{g(i)}\}$ for $i = 1, \ldots, n$, we obtain an allocation $\mathcal{R} = (R_1, \ldots, R_n)$, with $|R_i| = 2$, such that $u_i(R_i) = 2n + f(i) + g(i) - a_i = 2n$ for any agent i. Hence, $u(\mathcal{R}) \geq 2n$.

Conversely, let $\mathcal{R} = (R_1, \ldots, R_n)$ with $|R_i| = 2$ be an allocation of the resources such that $u(\mathcal{R}) \geq 2n$. Since $u_i(r_j) = u_i(r'_j)$ for any $i, j \in \{1, \ldots, n\}$ we can assume without loss of generality that in any allocation $\mathcal{R}' = (R'_1, \ldots, R'_n)$, any agent takes a resource from $\{r_1, \ldots, r_n\}$ and a resource from $\{r'_1, \ldots, r'_n\}$, i.e., for any $i \in \{1, \ldots, n\}$, $|R'_i \cap \{r_1, \ldots, r_n\}| = 1$ and $|R'_i \cap \{r'_1, \ldots, r'_n\}| = 1$. Thus, given $\mathcal{R}$, we define $f(i)$ as the index of the unique element in $R_i \cap \{r_1, \ldots, r_n\}$ and $g(i)$ as the index of the unique element in $R_i \cap \{r'_1, \ldots, r'_n\}$, i.e. $f(i) = j$ for j such that $r_j \in R_i \cap \{r_1, \ldots, r_n\}$ and $g(i) = j$ for j such that $r'_j \in R_i \cap \{r'_1, \ldots, r'_n\}$. Observe that, since a resource is assigned to exactly one agent, the functions f and g are partitions. Moreover, we can rewrite the utility of agent i for $\mathcal{R}$ as $u_i(R_i) = 2n + f(i) + g(i) - a_i$. Since $\sum_{i=1}^{n} a_i = n(n+1)$, it follows that $\sum_{i\in A} u_i(R_i) = \sum_{i\in A} (2n + f(i) + g(i) - a_i) = n^2$. Thus, $2n \leq u(\mathcal{R}) \leq \frac{1}{n}\sum_{i=1}^{n} u_i(R_i) = 2n$, that is $u(\mathcal{R}) = 2n$ and, consequently, $u_i(R_i) = 2n$ for each $i \in A$. It follows that $f(i) + g(i) = a_i$ for $i = 1, \ldots, n$, and thus $a_1, \ldots, a_n$ is a `yes`-instance.

For $k \geq 3$, the proof is similar. In fact, we add a set R' of $(k-2)n$ new resources and we set $u_i(r') = 3n$ for each agent $i \in A$ and for each new resources $r' \in R'$. It is then easy to see that $a_1, \ldots, a_n$ is a `yes`-instance for 2NMTS if and only if there exists an allocation $\mathcal{R} = (R_1, \ldots, R_n)$ of $R \cup R'$ such that $|R_i| = k$ and $u(\mathcal{R}) \geq 2n + 3(k-2)n$. □

3. MATROIDS & k-DIVISION

In this section we show an interesting relationship existing between the k-DIVISION problem and matroid theory.

Matroid theory. A *matroid* $\mathcal{M} = (X, \mathcal{F})$ consists of a finite set of n elements X and a collection $\mathcal{F}$ of subsets of X such that: (i) $\emptyset \in \mathcal{F}$; (ii) if $F_2 \subseteq F_1$ and $F_1 \in \mathcal{F}$, then $F_2 \in \mathcal{F}$; (iii) for every pair $F_1, F_2 \in \mathcal{F}$ such that $|F_1| < |F_2|$, there exists $e \in F_2 \setminus F_1$ such that $F_1 \cup \{e\} \in \mathcal{F}$. The elements of $\mathcal{F}$ are called *independent* sets. We refer the interested reader to [12] for more details on matroid theory.

For an example of matroid, consider an undirected graph $G = (V, E)$ and consider $\mathcal{M} = (X, \mathcal{F})$ such that $X = E$ and the independent sets represent all possible forests (acyclic sets of edges) of G. Observe that the graph $G_\emptyset = (V, \emptyset)$ is a forest and, hence, $\emptyset \in \mathcal{F}$. Moreover, if the graph $G_1 = (V, F_1)$ is a forest, then for each $F_2 \subseteq F_1$ the graph $G_2 = (V, F_2)$ is also a forest and hence $F_2 \in \mathcal{F}$. Finally, if both $G_1 = (V, F_1)$ and $G_2 = (V, F_2)$ are forests and $|F_1| < |F_2|$, then G_2 has fewer connected components; hence we have that there is a component $C \subseteq V$ in G_2 that corresponds to two or more components of G_1; then, along any path in C such that the endpoints belong to different components of G_1, there must be an edge $e \in F_2$ whose endpoints are in different components; thus adding this edge to F_1 produces a forest with more edges and hence also condition (iii) above is satisfied. Then, we can conclude that $\mathcal{M}$ is a matroid, that is usually called the *graphic matroid* of G.

Other useful examples are the partition and the laminar matroids. $(E, \mathcal{F})$ is a *partition matroid* if given ℓ non negative integers $b_1, \ldots, b_\ell$ and ℓ disjoint sets $E_1, \ldots, E_\ell$, we have $E = \cup_{i=1}^{\ell} E_i$ and $\mathcal{F} = \{F \subseteq E : |F \cap E_i| \leq b_i, i = 1, \ldots, \ell\}$. $(E, \mathcal{L})$ is a *laminar matroid* if, given ℓ non negative integers $b_1, \ldots, b_\ell$ and ℓ sets $E_1, \ldots, E_\ell$ such that for any pair i, j, one of the following three cases occurs: $E_i \subseteq E_j$, $E_j \subseteq E_i$ or $E_i \cap E_j = \emptyset$, we have $E = \cup_{i=1}^{\ell} E_i$ and $\mathcal{L} = \{L \subseteq E : |L \cap E_i| \leq b_i, i = 1, \ldots, \ell\}$. Note that the partition matroid is a special case of the laminar matroid.

Several optimization problems have been considered for matroids. A classical problem consists of computing, given an assignment of weights $w(e) \in \mathbb{R}^+$ to each element $e \in X$, a base $B \in \mathcal{F}$ that maximizes $\sum_{e\in B} w(e)$. It is known that this problem can be solved by an algorithm polynomial in $|E|$ [12]. Another optimization problem considers matroids $\mathcal{M}_1 = (X, \mathcal{F}_1)$ and $\mathcal{M}_2 = (X, \mathcal{F}_2)$ defined over the same set of elements X and an assignment of weights $w(e) \in \mathbb{R}^+$ to each element $e \in X$, and consists in finding the set F of maximum weight in $\mathcal{F}_1 \cap \mathcal{F}_2$. Even this problem is known to be solvable in time polynomial in $|E|$ [12].

k-DIVISION as the intersection of two matroids. Now we model k-DIVISION as an optimization problem on matroids.

For an instance $(A, R, E, (u_i)_{i\in A})$ of the k-DIVISION problem, we can denote by $A_i \subseteq R$ the relations that state which resources are available to agent i, i.e. $A_i = \{(i, r) \in E\}$ for every $i \in A$. Similarly, we denote by R_j the relations that state to which agents the resource j is available, i.e. $R_j = \{(a, j) \in E\}$ for every $j \in R$. It is not difficult to see that both $(A_1, \ldots, A_n)$ and $(R_1, \ldots, R_{kn})$ form a partition of E. We can then consider two partition matroids $\mathcal{M}_1 = (E, \mathcal{F}_1)$ and $\mathcal{M}_2 = (E, \mathcal{F}_2)$ such that $\mathcal{F}_1 = \{F \subseteq E : |F \cap A_i| \leq k, i \in A\}$ and $\mathcal{F}_2 = \{F \subseteq E : |F \cap R_j| \leq 1, j \in R\}$. Then it is straightforward to see that $(A, R, E, (u)_{i\in A})$ admits a feasible solution (i.e. exactly k resources are allocated to every agent) if there exists a set F of size kn in the intersection of these matroids, i.e. $F \in \mathcal{F}_1 \cap \mathcal{F}_2$. Indeed, from the first matroid we know that an agent gets at most k resources; from the second matroid we know that a resource is allocated to at most one agent.

Note that, since it is possible to compute the maximal intersection of two matroids in polynomial time, this gives an alternative way for checking in polynomial time if an instance of k-DIVISION admits at least one feasible allocation. We next show that modeling the problem with matroids allows us to achieve new interesting results. For the sake of presentation, from now on, we always suppose that the given instances admit at least one feasible allocation.

k-DIVISION with two utilities. Consider the restriction of the k-DIVISION problem in which an agent's utility for a resource can only take two values v_1 or v_2, with $0 \leq v_1 < v_2$. Matroid theory allows us to solve this problem.

THEOREM 2. *There exists a polynomial time algorithm that solves the k-*DIVISION *problem when the agent's utility assumes only values v_1 or v_2, with $0 \leq v_1 < v_2$.*

PROOF. Let $(A, R, E, (u_i)_{i\in A})$ be an instance of the problem and let us denote by u^* the utility of the least happy agent. We have that there exists a unique integer b such that $u^* = bv_1 + (k-b)v_2$. Thus, in order to guarantee that the utility of the least happy agent is at least u^*, it is sufficient to guarantee that every agent receives k resources, and for *at most* b of them the utility is v_1.

Remember the definition of A_i given above. For every agent i, we consider $A_i^1 \subseteq A_i$ that contains any element $(i,r) \in A_i$ such that $u_i(r) = v_1$. Then, for each $b = 0, \ldots, k$ we can define a laminar matroid $\mathcal{M}_1^b = (E, \mathcal{F'}_1)$ where

$$\mathcal{F'}_1 = \{F \subseteq E\colon |F \cap A_i^1| \leq b, |F \cap A_i| \leq k, i \in A\}.$$

Then any agent receives k resources, and for *at most* b of them the utility is v_1 if and only if the intersection between $\mathcal{M}_1^b$ and the matroid $\mathcal{M}_2$ defined above is not empty (and this can be checked in polynomial time). By iterating on $b \in \{0, \ldots, k\}$, we then find an allocation where the utility of the least happy agent is u^*. □

Note that the k-DIVISION problem cannot be solved, even for only two utility values, by the approach of allocating every item to the agent who likes it the most, that instead works in similar but not equivalent settings [5].

Approximating the* k-DIVISION *problem. Matroid theory turns out to help to design also approximate algorithms.

THEOREM 3. *For any $k \geq 1$, there exists a polynomial time algorithm that $\frac{1}{k}$-approximates the* k-DIVISION *problem.*

PROOF. Let $(A, R, E, (u_i)_{i\in A})$ be an instance of the problem. We denote by $\mathcal{U}$ the set of all possible utilities an agent can have for a single resource. Note that there are at most n^2k such distinct values, thus $|\mathcal{U}| \leq n^2k$.

Remember the definition of A_i for each agent $i \in A$ and of R_j for each resource $j \in R$. Moreover, for every agent i and $w \in \mathcal{U}$, let $A_i^w = \{(i,r) \in A_i\colon u_i(r) < w\}$. We clearly have $A_i^w \subseteq A_i$. Then, given $w \in \mathcal{U}$, we can define the laminar matroid $\mathcal{M}_1^w = (E, \mathcal{F}_1^w)$ where

$$\mathcal{F}_1^w = \{F \subseteq E\colon |F \cap A_i^w| \leq k-1, |F \cap A_i| \leq k, i \in A\}.$$

As previously stated, we can find in polynomial time the independent set I_w of maximum cardinality in the intersection of $\mathcal{M}_1^w$ and $\mathcal{M}_2$ defined above. Observe that if $|I_w| = kn$ then the instance of k-DIVISION admits a feasible allocation, i.e. each agent gets k resources, and for at least one of these, her utility is at least w.

Let $w^\sharp$ be the largest value in $\mathcal{U}$ such that $|I_{w^\sharp}| = kn$. One can find $w^\sharp$ and the corresponding independent set $I_{w^\sharp}$ by an exhaustive search in $\mathcal{U}$. Since every step is polynomial and the number of steps is polynomial, so is the whole algorithm.

Thus, we are only left with proving that $I_{w^\sharp}$ is a $1/k$-approximate solution. Let $\mathcal{R}^* = (R_1^*, \ldots, R_n^*)$ be an optimal allocation. Denote by p the least happy agent, i.e. $u_p(R_p^*) \leq u_i(R_i^*)$ for all i. Let $w' = u_p(R_p^*)/k$. Every agent i has at least one resource r in R_i^* such that $u_i(r) \geq w'$, otherwise $u_i(R_i^*) < kw' = u_p(R_p^*)$. Since w' may not belong to $\mathcal{U}$, let us define w'' as $\min\{w \in \mathcal{U}\colon w \geq w'\}$. We have that every agent i has at least one resource r in R_i^* such that $u_i(r) \geq w'' \geq w'$ and $w'' \in \mathcal{U}$. Thus in $\mathcal{F}_1^{w''} \cap \mathcal{F}_2$ there is an independent set $I_{w''}$ of cardinality $|kn|$. Then the returned solution has value at least $w^\sharp \geq w'' \geq w' \geq u_p(R_p^*)/k = u(\mathcal{R}^*)/k$ where $\mathcal{R}^*$ is the optimal allocation. □

We now show how the above algorithm works.

EXAMPLE 1. *Let $(A, R, E, (u)_{i\in A})$ be an instance of 2-*DIVISION *such that $|A| = n$, $|R| = 2n$ and E is complete, i.e. $(i,r) \in E$ for each agent $i \in A$ and resource $r \in R$. We assume n is even. We partition the set of resources R in quadruples of resources $(r_i^1, r_i^2, r_i^3, r_i^4)$ for $i = 1, \ldots, \frac{n}{2}$. As for the utility functions, for agents $i = 1, \ldots, \frac{n}{2}$, we set $u_i(r_i^1) = 1$, $u_i(r_i^2) = 1$ and $u_i(r) = 0$ for any other resource $r \in R$; for agents $i = \frac{n}{2}+1, \ldots, n$, we set $u_i(r_{i-n/2}^3) = 2$, $u_i(r_{i-n/2}^2) = 1$ and $u_i(r) = 0$ for any other resource $r \in R$.*

The optimal assignment is then $\mathcal{R}^* = (R_i^*)_{i=1,\ldots,n}$ where $R_i^* = (r_i^1, r_i^2)$ for each $i = 1, \ldots, \frac{n}{2}$, whereas for $i = \frac{n}{2}+1, \ldots, n$, we have $R_i^* = (r_{i-n/2}^3, r_{i-n/2}^4)$. Note that each agent obtains at least utility 2, so $u(\mathcal{R}^*) = 2$.

As for the approximation algorithm described above, $\mathcal{U} = \{0, 1, 2\}$. Since a feasible solution exists, it costs at least 0.

The algorithm then checks if there is an assignment in which each agent receives exactly two resources and for at least one of these her utility is at least 1. Since such an assignment exists (e.g., $\mathcal{R}^*$), then the test must be successful.

Next, the same work is done with the last element in $\mathcal{U}$. Unluckily, no assignment such that each agent receives at least one resource for which her utility is at least 2 exists (for agents $i = 1, \ldots, \frac{n}{2}$ no resource has utility at least 2).

Thus the largest value in $\mathcal{U}$ for which the test has been successful is 1 and hence the approximation algorithm returns an allocation with at least this value, that is a $\frac{1}{k}$-approximation of the optimal solution.

4. THE THREE UTILITY CASE

One can wonder if the algorithms presented in the previous section and based on the relation between the k-DIVISION problem and matroid theory are optimal. In this section, we try to answer this question. It is interesting to note that this "tentative" answer highlights an interesting relation between a long-standing problem about the MAXMIN ALLOCATION problem and the apparently more simple problem of k-DIVISION in which the agent's utility for an item can only take one value out of three. These values are denoted by v_1, v_2 and v_3 and they satisfy $0 \leq v_1 < v_2 < v_3$.

Inapproximability with three utilities. We start by showing that Theorem 2 is tight. That is, the restriction to two distinct utilities is necessary if one seeks a polynomial algorithm (unless $P = PN$). Actually, we show an even stronger result: there is no polynomial algorithm that ρ-approximate the k-DIVISION problem in which the agents' utilities take only three values, for some constant ρ which depends on (v_1, v_2, v_3). The proof of this result resembles a similar one given in [4]. However, we give here a full exposition because it involves peculiarities that are not present in [4]. Moreover, the insight provided by this proof will be useful later.

THEOREM 4. *For any $k \geq 2$, $\varepsilon > 0$ and three values, $0 \leq v_1 < v_2 < v_3$, unless **P=NP**, there is no polynomial time algorithm that $(\rho + \varepsilon)$-approximate the* k-DIVISION *problem*

with three distinct utility values, where

$$\rho = \frac{\upsilon_2 + (k-1)\upsilon_1}{\min\{2\upsilon_2, \upsilon_1 + \upsilon_3\} + (k-2)\upsilon_1}.$$

The theorem holds even if E is complete.

PROOF. We give a gap-introducing reduction of the 3-DIMENSIONAL MATCHING (3DM in short) problem. An instance of 3DM consists of a subset $\mathcal{C} = \{e_1, \ldots, e_n\} \subseteq X \times Y \times Z$ of n triples, where X, Y, Z are 3 pairwise disjoint sets of size ℓ with $X = \{x_1, \ldots, x_\ell\}$, $Y = \{y_1, \ldots, y_\ell\}$ and $Z = \{z_1, \ldots, z_\ell\}$. A matching is a subset $M \subseteq \mathcal{C}$ such that no two elements in M agree in any coordinate. The goal of 3DM is to decide if there exists a perfect matching M on $\mathcal{C}$, that is, a matching of size ℓ. This problem is known to be **NP**-complete (problem [SP1] in [7]).

Let $(\mathcal{C}, X, Y, Z)$ be an instance of 3DM. For $i = 1, \ldots, \ell$, let M_i be the indices of the triples for which the third coordinate corresponds to z_i, i.e. M_i represents the set

$$\{j \in \{1, \ldots, n\} \colon \exists x \in X, y \in Y \text{ s.t. } e_j = (x, y, z_i) \in \mathcal{C}\}.$$

Note that $(M_i)_{i \in \{1,\ldots,\ell\}}$ is a partition of $\{1, \ldots, n\}$.

We build an instance of the k-DIVISION problem as follows. There are n agents $A = \{1, \ldots, n\}$ (one per triple in $\mathcal{C}$) and kn resources partitioned in three blocks BR_1, BR_2 and BR_3. $BR_1 = BR_1^X \cup BR_1^Y$ with $BR_1^X = \{r_1^X, \ldots, r_\ell^X\}$ and $BR_1^Y = \{r_1^Y, \ldots, r_\ell^Y\}$ corresponding to the elements of X and Y, respectively. $BR_2 = \cup_{i=1}^{\ell} BR_2^i$ where BR_2^i is a set of $|M_i| - 1$ new resources. BR_3 consists of $(k-1)n - \ell$ dummy resources. Observe that $|BR_1| = 2\ell$, $|BR_2| = \sum_{i=1}^{\ell} |BR_2^i| = n - \ell$. Hence, the instance has kn resources, as desired. We suppose that E is complete, and for any agent $i \in A$, we set: for $r_j^X \in BR_1^X$, $u_i(r_j^X) = \upsilon_2$ if $e_i = (x_j, y, z) \in \mathcal{C}$ for some $y \in Y$, $z \in Z$, and $u_i(r_j^X) = \upsilon_1$ otherwise; for $r_j^Y \in BR_1^Y$, $u_i(r_j^Y) = \upsilon_2$ if $e_i = (x, y_j, z) \in \mathcal{C}$ for some $x \in X$, $z \in Z$, and $u_i(r_j^Y) = \upsilon_1$ otherwise; for $r \in BR_2$, $u_i(r) = \upsilon_3$ if $r \in BR_2^j$ and $i \in M_j$ for some $j \in \{1, \ldots, \ell\}$, and $u_i(r) = \upsilon_1$ otherwise; for $r \in BR_3$, $u_i(r) = \upsilon_1$. Thus, the utilities can only take values in $\{\upsilon_1, \upsilon_2, \upsilon_3\}$ as desired.

We claim that there exists a subset $J \subseteq \{1, \ldots, n\}$ with $|J| = \ell$ such that $M_J = \{e_j \colon j \in J\}$ is a perfect matching if and only if there is an allocation $\mathcal{R} = (R_1, \ldots, R_n)$ such that $|R_i| = k$ and $u_i(R_i) \geq \min\{2\upsilon_2, \upsilon_1 + \upsilon_3\} + (k-2)\upsilon_1$ for each $i = 1, \ldots, n$.

Suppose there is $J \subseteq \{1, \ldots, n\}$ with$|J| = \ell$ such that $M_J = \{e_j \colon j \in J\}$ is a perfect matching (thus $(\mathcal{C}, X, Y, Z)$ is a `yes`-instance of 3DM). We build $\mathcal{R} = (R_1, \ldots, R_n)$ as follows. For $j \in J$, R_j contains resources r_p^X and r_q^Y such that $e_j = (x_p, y_q, z)$ for some $z \in Z$ and $k-2$ dummy resources. The set R_j of resources assigned to an agent $j \notin J$ contains one resource from BR_2^i where i is such that $j \in M_i$ and $k-1$ resources from BR_3. It is not hard to check that this assignment is feasible and that each agent takes either two resources that she values υ_2 plus $k-2$ resources valued υ_1, or one resource valued υ_3 plus $k-1$ resources valued υ_1. Thus, $u(\mathcal{R}) = \min\{2\upsilon_2, \upsilon_1 + \upsilon_3\} + (k-2)\upsilon_1$.

Conversely, consider an allocation $\mathcal{R} = (R_1, \ldots, R_n)$ with $|R_i| = k$, $i = 1, \ldots, n$, such that $u(\mathcal{R}) \geq \min\{2\upsilon_2, \upsilon_1 + \upsilon_3\} + (k-2)\upsilon_1$. Observe that $\mathcal{R}$ must allocate at most $k-1$ dummy resources per agent, since they are valued υ_1. Similarly, if $\mathcal{R}$ allocates exactly $k-1$ dummy resources to an agent i, then agent i must have utility υ_3 for her remaining resource. Since the only resources valued υ_3 are the resources in BR_2, it follows that at most $|BR_2| = n - \ell$ agents receive $k-1$ dummy resources. Moreover, note that $\mathcal{R}$ must allocate to an agent i at least $k-2$ dummy resources. Indeed, since there are $|BR_3| = (k-1)n - \ell$ dummy resources, if an agent receives at most $k-3$ of these resources, it must be necessary either to assign $k-1$ dummy resources to more than $n - \ell$ agents, or to assign k dummy resources to some agent, in both cases a contradiction. A similar contradiction is reached if the number of agents that can receive $k-2$ dummy resources is less than ℓ.

Thus, we showed that there are at least ℓ agents with $k-2$ dummy resources but at most $n - \ell$ agents with $k-1$ of these resources and no agent with a different number of these resources. Hence there exists a set $J \subset \{1, \ldots, n\}$, with $|J| = \ell$, such that R_i for $i \notin J$ contains $k-1$ resources that she values υ_1 and one resource valued υ_3. For $i \in J$, R_i contains $k-2$ resources valued υ_1 and two resources valued at least υ_2. Since all resources in BR_2 are used by the agents not in J, each agent $i \in J$ must take her two non-dummy resources from BR_1. Let us say that these resources are r_p^X and r_q^Y. Since agent i values both of them υ_2, it must be the case that $e_i = (x_p, y_q, z)$ for some $z \in Z$.

We finally show that given two triples $e'_j = (x', y', z')$ and $e_{j''} = (x'', y'', z'')$ with $j', j'' \in J$, $j' \neq j''$, it must be $z' \neq z''$. For the sake of contradiction, suppose $z' = z''$ for some pair of triples. Then, there exists $z_{j^\star} \in Z$ such that $z_{j^\star} \notin Z_J$, where $Z_J = \{z \in Z \colon \exists x \in X, y \in Y, j \in J \text{ s.t. } e_j = (x, y, z)\}$. Now let us focus on the set of agents in $M_{j^\star}$: for all but one of these agents we can assign a resource from $BR_2^{j^\star}$. For the remaining agent, no resource from $BR_2^{j^\star}$ is available (since $|BR_2^{j^\star}| = |M_{j^\star}| - 1$), and this is the case also for the resources of BR_1 (by assumption). However, this agent has value υ_1 for any other resource. Then, her utility must be $k\upsilon_1$, a contradiction.

Thus, $M_J = \{e_j \colon j \in J\}$ is a perfect matching and then $(\mathcal{C}, X, Y, Z)$ is a `yes`-instance of 3DM. This proves that it is **NP**-hard to distinguish if $u(\mathcal{R}) \geq \min\{2\upsilon_2, \upsilon_1 + \upsilon_3\} + (k-2)\upsilon_1$ or $u(\mathcal{R}) < \min\{2\upsilon_2, \upsilon_1 + \upsilon_3\} + (k-2)\upsilon_1$. By observing that the last condition is equivalent to $u(\mathcal{R}) \leq \upsilon_2 + (k-1)\upsilon_1$, the desired inapproximability result is obtained. □

Observe that the approximation ratio tends, as expected, to one when υ_1 is close to υ_2 or when υ_2 is close to υ_3. On the other hand, the above approximation ratio is always lower-bounded by $\frac{1}{2}$, with the worst-case obtained when $\upsilon_1 = 0$ and $2\upsilon_2 \leq \upsilon_3$. Hence, we have the following corollary.

COROLLARY 1. *For any $k \geq 2$ and $\varepsilon > 0$, unless **P=NP**, there is no polynomial time algorithm that $\frac{1}{2}+\varepsilon$-approximates the k-DIVISION problem, even if E is complete and there are only three distinct utility values.*

More accurate approximation with three utilities.

Theorem 4 highlights that the approximation algorithm given in Theorem 3 may not be tight even if the agents' utilities can only take three values. For this reason, we propose a new algorithm that exploits this restriction to possibly achieve better approximations. This algorithm repeatedly uses the existence of a polynomial algorithm for the k-DIVISION problem when the agents' utilities can only take two values.

THEOREM 5. *For any $k \geq 2$, there exists a polynomial time algorithm that ρ-approximates to the k-DIVISION prob-*

lem when the agents' utilities assume only values v_1, v_2 and v_3, with $0 \le v_1 < v_2 < v_3$, where

$$\rho = \begin{cases} \frac{v_2+(k-1)v_1}{kv_2}, & \text{if } 2v_2 \le v_1 + v_3; \\ \frac{v_1+(k-1)v_2}{v_1+(k-1)v_3}, & \text{otherwise.} \end{cases}$$

PROOF. Consider an instance $I = (A, R, E, (u)_{i\in A})$ of the problem. Our approximation algorithm mainly merges two consecutive utilities in different ways and solves the resulting instances using the algorithm described in Theorem 2.

Specifically, we build three instances of the k-DIVISION problem in which the agents' utilities just take two values, namely $I' = (A, R, E, (u')_{i\in A})$, $I'' = (A, R, E, (u'')_{i\in A})$ and $I''' = (A, R, E''', (u)_{i\in A})$, where u', u'' and E''' are defined as follows: concerning u', for any agent $i \in A$ and any resource $r \in R$ such that $(i, r) \in E$, we set $u'_i(r) = v_1$ if $u_i(r) = v_1$ and $u'_i(r) = v_2$ otherwise; as for u'', for any agent $i \in A$ and any resource $r \in R$ such that $(i, r) \in E$, we set $u''_i(r) = v_3$ if $u_i(r) = v_3$ and $u''_i(r) = v_1$ otherwise; finally, we have that E''' discards for any agent the resources she values v_1, i.e., $E''' = \{(i, r) \in E\colon u_i(r) > v_1\}$. Note that instance I''' may not contain any feasible solution (i.e., some agent may receive less than k resources) but we have previously highlighted that this case can be checked in polynomial time. Our algorithm is then the following:

Algorithm 1 Approximation algorithm for 3 utilities

Require: $A, R, E, (u)_{i\in A}$ with $|A| = n$ and $|R| = kn$.
1: **if** I''' admits feasible solutions **then**
2: Solve I''' and let $\mathcal{R}'''$ be the returned solution.
3: **end if**
4: Solve I' and evaluate the returned solution $\mathcal{R}'$ in I.
5: Solve I'' and evaluate the returned solution $\mathcal{R}''$ in I.
6: **return** The best solution among $\mathcal{R}'$, $\mathcal{R}''$ and $\mathcal{R}'''$.

Now we evaluate the approximation ratio of this algorithm. Let $\mathcal{R}^* = (R^*_1, \ldots, R^*_n)$ be an optimal allocation for instance I with value OPT. The utility of agent i in the assignment $\mathcal{R}^*$ is $u_i(R^*_i) = a_i v_1 + b_i v_2 + c_i v_3$, for $a_i, b_i, c_i \in \{0, \ldots, k\}$ such that $a_i + b_i + c_i = k$. Observe that if $\text{OPT} = kv_1$, then any feasible solution is optimal and so will be also the solution returned by Algorithm 1. Thus, from now on we consider $\text{OPT} > kv_1$.

Let us first focus on the case $2v_2 \le v_1 + v_3$. If $\text{OPT} \ge (k-1)v_3 + v_2$, then we must have $a_i = 0$ for each agent $i \in A$. Hence, $\mathcal{R}^*$ is feasible and optimal also for instance I'''. Thus, Algorithm 1 finds the optimal solution.

Since $2v_2 \le v_1 + v_3$, the largest value that OPT can take below $(k-1)v_3 + v_2$ is $\text{OPT} = (k-1)v_3 + v_1$. If $2v_2 \ne v_1 + v_3$ (ie, $2v_2 < v_1 + v_3$) we must have $c_i \ge k-1$ for each agent $i \in A$. Indeed, $\text{OPT} = (k-1)v_3 + v_1 > (k-2)v_3 + 2v_2$. Hence, if by contradiction $c_{i'} \le k-2$ for some agent $i' \in A$, then $\text{OPT} \le u_{i'}(R^*) \le (k-2)v_3 + 2v_2$. Thus, $u''_i(R^*_i) = c_i v_3 + (k-c_i)v_1 = c_i(v_3 - v_1) + kv_1 \ge (k-1)v_3 + v_1 = \text{OPT}$ for each $i \in A$. Then $u''(\mathcal{R}^*) = u(\mathcal{R}^*)$ and thus the optimal assignment for I'' returned by our algorithm is exactly $\mathcal{R}^*$.

Thus, we are interested in the behavior of our algorithm when $kv_1 < \text{OPT} \le (k-2)v_3 + 2v_2$, that is equivalent to say that $\text{OPT} \in [(k-1)v_1 + v_2, (k-2)v_3 + 2v_2]$. We split this interval in the subintervals $S_1 = [(k-1)v_1 + v_2, kv_2]$ and $S_j = [(j-1)v_3 + (k+1-j)v_1, (j-1)v_3 + (k+1-j)v_2]$ for $j = 2, \ldots, k-1$. Note that the subintervals S_j are not decreasing. They may not be disjoint, but $\text{OPT} \in \cup_{j=1}^{k-1} S_j$, since $v_3 + (k-1)v_1$ is less than or equal to the smallest value above kv_2 and, similarly, $jv_3 + (k-j)v_1$ is less than or equal to the smallest value above $(j-1)v_3 + (k+1-j)v_2$ for $j = 2, \ldots, k-2$.

Now, suppose that $\text{OPT} \in S_1$. Consider the solution $\mathcal{R}'$ produced on instance I'. Since $u'_i(r) \le u_i(r)$, we have $u(\mathcal{R}') \ge u'(\mathcal{R}')$. Since $u'(\mathcal{R}')$ is optimal for I', then $u'(\mathcal{R}') \ge u'(\mathcal{R}^*)$. Now, we evaluate $\mathcal{R}^*$ in instance I'. Since $\text{OPT} \ge (k-1)v_1 + v_2$, we have that $a_i \le k-1$ for any $i \in A$. Thus, $u'_i(R^*_i) = a_i v_1 + (k-a_i)v_2 = kv_1 + (k-a_i)(v_2 - v_1) \ge (k-1)v_1 + v_2$ for any $i \in A$. Thus, $u'(\mathcal{R}^*) \ge (k-1)v_1 + v_2$. From $\text{OPT} \le kv_2$, we then get

$$u(\mathcal{R}') \ge u'(\mathcal{R}') \ge u'(\mathcal{R}^*) \ge (k-1)v_1 + v_2 \ge \frac{(k-1)v_1 + v_2}{kv_2}\text{OPT}.$$

Then our algorithm $\frac{(k-1)v_1+v_2}{kv_2}$-approximates the optimum.

Suppose instead that $\text{OPT} \in S_j$ for some $j = 2, \ldots, k-1$ and $\text{OPT} \notin S_{j-1}$ (this is possible since the sets S_j are not decreasing). Consider the solution $\mathcal{R}''$ produced on the instance I''. Since $u''_i(r) \le u_i(r)$, we deduce that $u(\mathcal{R}'') \ge u''(\mathcal{R}'')$. By construction, $u''(\mathcal{R}'')$ is optimal for I'', thus $u''(\mathcal{R}'') \ge u''(\mathcal{R}^*)$. Now, we evaluate solution $\mathcal{R}^*$ in the instance I''. Since $\text{OPT} \ge (j-1)v_3 + (k+1-j)v_1$ (by $\text{OPT} \in S_j$) and $\text{OPT} > (j-2)v_3 + (k+2-j)v_2$ (by $\text{OPT} \notin S_{j-1}$), we deduce that $c_i \ge j-1$ for any agent $i \in A$. Hence, $u''_i(R^*_i) = (k-c_i)v_1 + c_i v_3 = c_i(v_3 - v_1) + kv_1 \ge (j-1)v_3 + (k+1-j)v_1$ for every $i \in A$. Thus, $u''(\mathcal{R}^*) \ge (j-1)v_3 + (k+1-j)v_1$. Then, since $\text{OPT} \le (j-1)v_3 + (k+1-j)v_2$, we have

$$\begin{aligned} u(\mathcal{R}'') \ge u''(\mathcal{R}'') \ge u''(\mathcal{R}^*) &\ge (j-1)v_3 + (k+1-j)v_1 \\ &\ge \frac{(j-1)v_3 + (k+1-j)v_1}{(j-1)v_3 + (k+1-j)v_2}\text{OPT} \\ &\ge \frac{(j-1)v_2 + (k+1-j)v_1}{kv_2}\text{OPT}, \end{aligned}$$

where the last inequality follows from the mapping being increasing in v_3 and $v_3 > v_2$. Now, since $(j-1)v_2 \ge v_2 + (j-2)v_1$ (since $j \ge 2$) we obtain that our algorithm returns at least an $\frac{(k-1)v_1+v_2}{kv_2}$-approximation to the optimum.

Now consider the case $2v_2 > v_1 + v_3$. If $\text{OPT} \ge (k-2)v_3 + 2v_2$, then we must have $a_i = 0$ for each agent $i \in A$. Hence, $\mathcal{R}^*$ is feasible and optimal also for instance I'''. Thus, Algorithm 1 finds the optimal solution.

As a consequence, we are interested in the behavior of our algorithm when $kv_1 < \text{OPT} < (k-2)v_3 + 2v_2$, that is equivalent to say that $\text{OPT} \in [(k-1)v_1 + v_2, (k-1)v_3 + v_1]$ because $(k-2)v_3 + 2v_2 > (k-1)v_3 + v_1 > (k-2)v_3 + v_2 + v_1$. We split this interval in the subintervals $S_j = [(k-j)v_1 + jv_2, (k-j)v_1 + jv_3]$ for $j = 1, \ldots, k-1$. As previously, the sets S_j are not decreasing and $\text{OPT} \in \cup_{j=1}^{k-1} S_j$.

Suppose that $\text{OPT} \in S_j$ for some $j = 1, \ldots, k-1$ and $\text{OPT} \notin S_{j-1}$ (this is possible since the sets S_j are not decreasing). Consider solution $\mathcal{R}'$ produced on instance I'. Since $u'_i(r) \le u_i(r)$, we deduce that $u(\mathcal{R}') \ge u'(\mathcal{R}')$. By construction, $u'(\mathcal{R}')$ is optimal for I', thus $u'(\mathcal{R}') \ge u'(\mathcal{R}^*)$. Now, we evaluate solution $\mathcal{R}^*$ in the instance I'. Since $\text{OPT} \ge (k-j)v_1 + jv_2$ (by $\text{OPT} \in S_j$) and $\text{OPT} > (k+1-j)v_1 + (j-1)v_3$ (by $\text{OPT} \notin S_{j-1}$ if $j > 1$ and by hypothesis otherwise), we deduce that $a_i \le k-j$ for each agent $i \in A$. Hence, $u'_i(R^*_i) = a_i v_1 + (k-a_i)v_2 = kv_1 + (k-a_i)(v_2 - v_1) \ge (k-j)v_1 + jv_1$ for any $i \in A$. Thus, $u'(\mathcal{R}^*) \ge (k-j)v_1 + jv_2$. Then, since

$\text{OPT} \leq (k-j)v_1 + jv_3$, we have

$$u(\mathcal{R}') \geq u'(\mathcal{R}') \geq u'(\mathcal{R}^*) \geq (k-j)v_1 + jv_2$$
$$\geq \frac{(k-j)v_1 + jv_2}{(k-j)v_1 + jv_3}\text{OPT} \geq \frac{v_1 + (k-1)v_2}{v_1 + (k-1)v_3}\text{OPT},$$

where the last inequality follows from the mapping being decreasing in j and $j \leq k-1$. That is, our algorithm $\frac{v_1+(k-1)v_2}{v_1+(k-1)v_3}$-approximates the optimum. □

Let us give an example of how the algorithm works.

EXAMPLE 2. *Consider the instance of the problem described in Example 1. This instance uses exactly three distinct utilities: 0, 1 and 2.*

By applying Algorithm 1 to this instance, we first observe that there is no feasible solution in I''', i.e. no solution in which each agent receives exactly two resources that she values 1 or 2. Indeed, there are only $2\ell+(n-\ell) = n+\ell < 2n$ of these resources.

As for I', we observe that allocation $\mathcal{R}' = (R'_i)_{i=1,\ldots,n}$ with $R'_i = (r_i^1, r_i^3)$ for $i = 1, \ldots, \frac{n}{2}$ and $R'_i = (r_{i-n/2}^2, r_{i-n/2}^4)$ for $i = \frac{n}{2}+1, \ldots, n$ has value $u'(\mathcal{R}') = 1$ and it is optimal for I'. Indeed, as stated above, for any feasible allocation $\mathcal{R} = (R_i)_{i=1,\ldots,n}$ there is at least one agent i who gets at least one resource valued 0. Hence, $u'(\mathcal{R}) \leq u'(\mathcal{R}')$. Thus, allocation $\mathcal{R}'$ can be returned at step 4 of Algorithm 1 and its value in I is $u(\mathcal{R}') = 1$.

Let us finally consider instance I''. We observe that allocation $\mathcal{R}'' = (R''_i)_{i=1,\ldots,n}$ such that $R''_i = (r_i^2, r_i^3)$ for $i = 1, \ldots, \frac{n}{2}$ and $R''_i = (r_{i-n/2}^1, r_{i-n/2}^4)$ for $i = \frac{n}{2}+1, \ldots, n$ has value $u''(\mathcal{R}'') = 0$ and it is optimal for I''. Indeed, for each feasible allocation $\mathcal{R} = (R_i)_{i=1,\ldots,n}$ there is at least one agent i who does not get any resource valued 2 (e.g., each agent $j = 1, \ldots, \frac{n}{2}$ values a resource at most 1). Thus, for any allocation there is at least one agent i such that $u''_i(R_i) = 0$. and, hence, $u''(\mathcal{R}) = 0 = u''(\mathcal{R}'')$. Then, allocation $\mathcal{R}''$, whose valuation in I is $u(\mathcal{R}'') = 0$, can be returned at step 5 of Algorithm 1.

Since our algorithm returns the allocation that has the best valuation in I among $\mathcal{R}'$, $\mathcal{R}''$ and, if defined, $\mathcal{R}'''$, in our example it will return $\mathcal{R}'$ of value 1, that is an $\frac{1}{2} = \frac{v_2+(k-1)v_1}{kv_2}$-approximation of the optimum.

At this point some remarks should be made. Observe that when $k = 2$, the approximation ratio of Algorithm 1 matches the bounds given in Theorem 4. For any $k \geq 2$ and $2v_2 \leq v_1+v_3$, the approximation ratio is larger than $\frac{1}{k}$, with the worst case achieved when $v_1 = 0$. Note that $1/k$ is the approximation ratio given in Theorem 3. On the other side, if $2v_2 > v_1 + v_3$, then $\frac{v_1+(k-1)v_2}{v_1+(k-1)v_3}$ is strictly larger than $\frac{1}{2}$. This is noteworthy because Corollary 1 states that it is not possible to provide such a guarantee for all instances.

It is also interesting to note that the approximation ratio of Algorithm 1 tends to 1 when v_2 is close to v_1 or to v_3. This is expected, since in this case we are close to the two-utility case which can be solved in polynomial time (Theorem 2).

However, it may look surprising that the approximation ratio of Algorithm 1 does not depend on v_3 when $2v_2 \leq v_1 + v_3$. Similarly, we have that in the opposite case, the approximation ratio in practice depends only on v_2 and v_3. However, these observations can be easily explained: if $2v_2 \leq v_1 + v_3$, then $v_3 - v_2 \geq v_2 - v_1$, i.e. v_2 cannot be close to v_3 more than how much it is close to v_1. Thus it is the distance between v_1 and v_2 that measures the gap from having only two utility values. Similarly, if $2v_2 > v_1 + v_3$, then $v_3 - v_2 < v_2 - v_1$, so now the gap from having only two utility values should be measured by the distance between v_2 and v_3. Note that in the lower bound given in Theorem 4 we observe the same dichotomy: if $2v_2 \leq v_1 + v_3$, then the bound does not depend on v_3, whereas if $2v_2 > v_1+v_3$, then the bound depends in practice only on v_2 and v_3.

We also observe that the idea exploited in Algorithm 1, that is merging two utilities and solve the corresponding "rounded" instances, can be extended to the case of more than three different utilities (at the cost of making the analysis of the algorithm more complex). However we skip this extension since the three-utility case has already interesting connections with long-standing problems in the area.

Closing the gap? The MaxMin Allocation obstacle.

Unfortunately, the algorithm described in Theorem 5 does not match the inapproximability result given in Theorem 4. Indeed, if for $2v_2 > v_1 + v_3$ the upper and the lower bounds almost match, for the opposite case they diverge since the lower bound is never worse than one half, whereas the upper bound tends to $\frac{1}{k}$. For this reason, we assume from now on that $2v_2 \leq v_1 + v_3$ and we wonder if we can improve the corresponding approximation ratio. The next theorem suggests that this may not be possible.

THEOREM 6. *For any $k \geq 2$ and any v satisfying $2v_2 + (k-2)v_1 \leq v \leq kv_2$, unless **P=NP**, there is no polynomial time algorithm that can decide if the k-DIVISION problem with three distinct utility values admits a feasible allocation $\mathcal{R}$ such that $u(\mathcal{R}) \geq v$.*

PROOF. We assume without loss of generality that $v = av_1 + bv_2 + cv_3$ for $a, b, c \in \{0, \ldots, k\}$ such that $a + b + c = k$. We distinguish two cases depending on whether $v = 2v_2 + L$ or not, where $L = a_L v_1 + b_L v_2 + c_L v_3$ for $a_L, b_L, c_L \in \{0, \ldots, k-2\}$ such that $a_L + b_L + c_L = k-2$.

If $v = 2v_2 + L$, then we apply exactly the same reduction as in the proof of Theorem 4, except that now we partition the set BR_3 of dummy items in $n+1$ subsets BR_3^i, for $i = 0, \ldots, n$. Each set BR_3^i for $i = 1, \ldots, n$ contains $b_L + c_L$ resources, whereas the set BR_3^0 contains any remaining resource. Moreover, for each agent $i \in A$ we set utilities for resources in BR_3^i so that for b_L of these resources we have $u_i(r) = v_2$ and for c_L of these resources we have $u_i(r) = v_3$. The utility of agent i for any other resource remains the same as stated in the reduction of Theorem 4.

By mimicking the proof of Theorem 4, we then have that an instance $(\mathcal{C}, X, Y, Z)$ of 3DM has a perfect matching if and only if the instance the k-DIVISION problem with three distinct utility values resulting from the reduction admits a feasible allocation $\mathcal{R}$ such that $u(\mathcal{R}) \geq v$.

If v cannot be written as $2v_2 + L$, then we let $v^\star$ be the smallest value above v such that $v^\star = 2v_2 + L^\star$. We now consider exactly the same reduction as above with $v^\star$ in place of v. This proves that an instance $(\mathcal{C}, X, Y, Z)$ of 3DM has a perfect matching if and only if the instance the k-DIVISION problem with three distinct utility values resulting from the reduction admits a feasible allocation $\mathcal{R}$ with $u(\mathcal{R}) \geq v^\star$.

Therefore the theorem follows by showing that $u(\mathcal{R}) \geq v$ implies $u(\mathcal{R}) \geq v^\star$. Indeed, suppose for the sake of contradiction that $u(\mathcal{R}) \geq v$ but there is an agent i such that $u_i(R_i) = v$. It is not hard to check that $v^\star = v - 2v_1 - v_3 +$

$3\upsilon_2$. Hence, $\mathcal{R}$ has to allocate to agent i each resource in BR_3^i except a resource r that she values υ_2. Note that this resource r is valued υ_1 by any other agent. Thus, any agent receiving this resource should compensate it with a resource that she values υ_3. It follows then, by a pigeon-hole-principle argument, that there will be an agent j unable to compensate and, consequently, $u_j(R_j) < v$, a contradiction. □

We say that an approximation algorithm for k-DIVISION is a *meta-search* algorithm if: (i) a subset V of size polynomial in the input consisting of possible allocation values is considered; (ii) for each $v \in V$ one tests if a feasible allocation of value at least v exists; (iii) the allocation corresponding to the largest v for which the test is successful is returned. Note that the two approximation algorithms presented in this work are meta-search algorithms. This is immediate for the algorithm given in Theorem 3; for Algorithm 1, any test corresponds to running the algorithm described in Theorem 2 and evaluating the returned solution for I.

Then, we obtain the following corollary to Theorem 6.

COROLLARY 2. *For any $k \geq 2$ and $\varepsilon > 0$, unless **P=NP**, there is no meta-search algorithm that $(\rho+\varepsilon)$-approximates the k-DIVISION problem with three distinct utilities, where*

$$\rho = \frac{\upsilon_2 + (k-1)\upsilon_1}{k\upsilon_2}.$$

PROOF. A meta-search algorithm with a better approximation ratio contradicts Theorem 6. Indeed, there must exist $\upsilon_2 + (k-1)\upsilon_1 < v \leq k\upsilon_2$ for which it is able to check if there exists a feasible allocation of cost at least v. Otherwise, given an instance of k-DIVISION with OPT $= k\upsilon_2$, the algorithm can only return an instance with cost $\upsilon_2 + (k-1)\upsilon_1$, failing to improve the approximation ratio. □

Thus, Algorithm 1 is optimal in its class of algorithms, that is it gives the best possible approximation ratio among the ones allowed for a meta-search algorithm. But can we do better? Which kind of algorithms we should consider for trying to improve the approximation ratio?

Note that Theorem 6 states that no algorithm is able to decide if there exists an allocation of value v for some $2\upsilon_2 + (k-2)\upsilon_1 \leq v \leq k\upsilon_2$ for *any input*. However, there may exist a polynomial algorithm A that takes this decision for only *many input*. For example, an algorithm A may be able to decide if there exists an allocation of value v when the input is an instance whose optimal value is greater than v but unable to decide this for any instance whose optimal value is v. Such an algorithm would allow to improve the approximation ratio to $\min\left\{\frac{\upsilon_2+(k-1)\upsilon_1}{v}, \frac{v}{k\upsilon_2}\right\} > 1/k$.

So the question becomes: does Algorithm A exist? Unfortunately, the answer to this question appears related to the long standing gap between $\frac{1}{2}$ and $\frac{1}{m-n+1}$ regarding the approximation ratio of the MAXMIN ALLOCATION problem for sufficiently small values of m. Indeed, the latter problem can be easily reduced to k-DIVISION for $k = m-n+1$ by adding sufficiently many dummy resources. Thus, by showing that A does not exist it follows that, even with only three utility values, MAXMIN ALLOCATION cannot be approximated within a better ratio than $\frac{\upsilon_2+(m-n)\upsilon_1}{(m-n+1)\upsilon_2} \geq \frac{1}{m-n+1}$, matching the best known approximation algorithm for this problem. On the other side, if Algorithm A exists, this will shade a new and probably definitive light on the MAXMIN ALLOCATION problem, since also for this problem any known algorithm is a meta-search algorithm. Thus, this apparently simpler problem, namely k-DIVISION with only three utility values, may catch the entire difficulty in closing this long-standing problem about MAXMIN ALLOCATION.

5. CONCLUSION & OPEN PROBLEMS

The analysis of the k-DIVISION problem provided us with several insights in the problem of MAXMIN ALLOCATION.

First, we devised an interesting connection between the maxmin objective and matroid theory. We showed how this connection allows to design exact or approximate algorithms for the k-DIVISION problem. It would be interesting to understand at which extent matroids can help in the agenda of tightening the approximation ratios for these problems.

Then, we showed that the difficulty in closing the approximability gap of MAXMIN ALLOCATION may be caught by an apparently simpler problem. Can this simplicity help us to close or at least to tighten this gap?

We also take a step in this direction, by ruling out the possibility to improve the approximation ratio through a meta-search algorithm. Thus, it is natural to ask whether there exists an algorithm that does not belong to this class.

6. REFERENCES

[1] A. Asadpour and A. Saberi. An Approximation Algorithm for Max-Min Fair Allocation of Indivisible Goods. *SIAM J. Comput.*, 39(7): 2970-2989, 2010.

[2] N. Bansal and M. Sviridenko. The Santa Claus problem. STOC 2006.

[3] M. Bateni, M. Charikar, and V. Guruswami. MaxMin allocation via degree lower-bounded arborescences. STOC 2009.

[4] I. Bezáková and V. Dani. Allocating indivisible goods. *SIGecom Exchanges*, 5(3): 11-18, 2005.

[5] S. Bouveret, M. Lemaître, H. Fargier, and J. Lang. Allocation of indivisible goods: a general model and some complexity results. AAMAS 2005.

[6] D. Chakrabarty, J. Chuzhoy, and S. Khanna. On Allocating Goods to Maximize Fairness. FOCS 2009.

[7] M. Garey and D. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. Freeman, 1979.

[8] D. Golovin. Max-min fair allocation of indivisible goods. *Technical report 2348*, Computer Science Department, Carnegie Mellon University, 2005.

[9] L. Gourvès, J. Monnot, and L. Tlilane. A Matroid Approach to the Worst Case Allocation of Indivisible Goods. IJCAI 2013.

[10] S. Khot and A. K. Ponnuswami. Approximation Algorithms for the Max-Min Allocation Problem. APPROX-RANDOM 2007.

[11] T. Nguyen, M. Roos, and J. Rothe. A survey of approximability and inapproximability results for social welfare optimization in multiagent resource allocation. *Annals of Math. and AI*, 68(1-3): 65-90, 2013.

[12] J.G. Oxley. *Matroid Theory*. Oxford University Press, 1992.

[13] W. Yu, H. Hoogeveen, and J. K. Lenstra. Minimizing Makespan in a Two-Machine Flow Shop with Delays and Unit-Time Operations is NP-Hard. *J. Scheduling*, 7(5):333-348, 2004.