MinCovTarget: a new standard for fair allocation

Giovanni Puccetti,* Ludger Rüschendorf,† and Steven Vanduffel.‡

May 21, 2023

Abstract

The problem of the fair allocation of indivisible items is a relevant and challenging economic problem with several applications. For small dimensional frameworks, the problem can be solved exactly by full enumeration of all the possible allocations of the items. For higher dimensional scenarios, a numerical algorithm is called for.

In this paper, we compare two state-of-the-art algorithms for the fair allocation of a number of indivisible goods amongst several agents: the Spliddit and MinCovTarget algorithms. In particular, we investigate MinCovTarget with respect to the possible choices of the target value and we propose based on our experiments two versions of the algorithm: MinCovTarget+ and MinCovTarget*.

MinCovTarget+ uses different target values and selects, amongst all the minimal envy allocations found, the ones providing maximum social welfare. Our numerical analysis shows that, in case one aims to minimize envy, MinCovTarget+ is at least as good as Spliddit and, in larger dimensional frameworks, MinCovTarget+ provides no-envy allocations with an higher value of social welfare than Spliddit and at a fraction of the computation time. Moreover, MinCovTarget+ is able to rapidly obtain envy-free allocations also in the high dimensional scenarios for which Spliddit cannot be used.

MinCovTarget* is an even faster version of the algorithm that uses a single target value set equal to the total value of goods. MinCovTarget* can handle huge dimensional scenarios (hundreds of agents and goods) within seconds, finding allocations with negligible envy that approximate maximum social welfare.

Our numerical experiments have been run with a random uniform integral valuation of the goods to be allocated and with a novel design of the value matrix which takes into account dependent valuations.

All the numerical estimates in this paper have been obtained on a Mac mini (3,2 GHz 6-Core Intel Core i7, 16 GB RAM). The corresponding MATLAB code is available at:

`https://github.com/giovannipuccetti/MinCovTarget`

*Giovanni Puccetti (corresponding author, `giovanni.puccetti@unimi.it`). University of Milan, Italy.
†Ludger Rüschendorf (`ruschen@stochastik.uni-freiburg.de`). University of Freiburg, Germany.
‡Steven Vanduffel (`steven.vanduffel@vub.ac.be`). Vrije Universiteit Brussel, Belgium.

1 Introduction and motivation

The problem of the fair allocation of indivisible items is a relevant and challenging economic problem which has several applications in everyday life. In fact, our social interactions often call for a method to allocate a number of indivisible goods or resources amongst a number of agents in an optimal way. The head of a firm might be in the need of allocating a limited number of tablets, computers, and private offices amongst the firm's employees. A group of designated heirs needs a fair way to divide the real estates and the jewellery coming from a inheritance, or a couple might need to settle these goods for a divorce. A group of friends needs to compute the rent shares for the different rooms in a flat. These are all practical situations calling for a fair allocation of goods/resources which are indivisible in nature: they lose their value when broken, divided into pieces or shared with someone else. What rule should one follow to allocate them?

A most intuitive and from a global perspective arguably fair allocation rule consists in the maximization of *social welfare*: one maximizes the sum of the values received by each of the agents involved. However, the grander scope of maximizing social utility leaves room for individual unhappiness. If some individuals value some items a lot, they will probably have it at the cost of leaving some other people envying them (see the example given in Section 5). One would rather prefer an allocation that makes everyone happy, that is under which each agent believes to have received the best bundle, with no need of envying the other players. Such an allocation is called *envy-free*, and even if envy-freeness is not always achievable in practice (a single diamond will not make everyone smile), one can always try to reach the minimum level of envy amongst the agents. Finding a *minimum envy* allocation is a mathematically and computationally difficult task, where even long negotiated solutions cannot compare to efficient allocations found by rigorous and robust algorithms.

The majority of disputes amongst friends, heirs or couples can be rapidly solved by the Spliddit algorithm, introduced in [3]. The Spliddit algorithm was originally designed to maximize so-called *Nash welfare*, the sum of the *logarithmic*-utilities gained by each agent, but has been claimed to provide provably fair solution with no-envy, making everybody happy [10]. At the website `spliddit.org` private individuals could find a ready-to-use, cool tool to successfully fix their disputes. At the time of writing the Spliddit website has been discontinued, and we could not find any detailed numerical study showing the performance of the algorithm and proving its efficiency.

Besides some minor implicit assumptions, no major drawback of the Spliddit algorithm is visible unless one tackles a high-dimensional allocation problem, where, say, 300 goods are be allocated to 30 agents. For this case, Spliddit was able to provide an allocation within 10 minutes only 7 times over 50 simulations of the goods valuations provided by the agents. The competitor algorithm that we propose below, MinCovTarget+, handles the same scenario always finding a no-envy allocation in about 16 seconds.

Real life examples where high dimensional instances arise include for instance the cases when a number of medical supplies or vaccines are to be distributed among hospitals in a given country, or when a union of nations is willing to distribute resources amongst its country members, or a large firm is going to distribute goods/indivisible bonuses amongst its employees. High dimensional instances might also include the fair allocation of a mix of indivisible and divisible goods, as for instance the case in which an inheritance includes some cash.

Motivated by the lack of a numerical study of Spliddit in the literature, in this paper we provide a detailed comparison between the Spliddit and MinCovTarget algorithms. Based on our analysis, we propose the MinCovTarget+ and MinCovTarget* algorithms as new standard solutions for the fair allocation problem.

The MinCovTarget algorithm originates from the MinCov algorithm which was introduced in [4] to minimize *social inequality*, that is to find allocations under which each agent believes to have received (more or less) the same value as everybody else. Via the introduction of a *target* parameter, the MinCovTarget algorithm makes it possible to also deal with various other fair allocation criteria such as maximum social welfare and minimum envy. The MinCovTarget+ algorithm uses different target values and selects, amongst all the minimal envy allocations found, (the) one that provides maximum social welfare.

Our numerical analysis shows that the MinCovTarget+ performance in finding allocation with no envy is at least as good as Spliddit (but at a fraction of Spliddit time cost) in all scenarios, and strictly improves Spliddit in low-dimensional scenarios. Moreover, in large dimensional frameworks, the allocations found by MinCovTarget+ have an higher value of social welfare. Finally, MinCovTarget+ is able to rapidly obtain envy-free allocations also in the high dimensional scenarios for which Spliddit cannot be readily implemented.

The MinCovTarget* algorithm is a version of MinCovTarget that uses a single target value set equal to the total value of goods. MinCovTarget* is extremely fast and can handle huge dimensional problems within seconds. It is able to provide allocations approximating maximum social welfare and with levels of envy that are fairly close to zero.

The aim of this paper is to compare the performance of Spliddit and MinCovTarget against the four fairness criteria introduced above and formally defined below: maximum social welfare (1), minimum envy (2), maximum Nash welfare (3), and minimum social inequality (4). It is important to stress that, for all these criteria, the problem of fair allocation of indivisible goods is computationally $\mathcal{NP}$-hard (see [2], [7], [9], and references therein). Roughly speaking, this means that, unless one is able for very small dimensional instances to solve the problem exactly by full enumeration of the finitely possible allocations or by linear programming, finding the global optimum is computationally intractable. As a result, algorithms for fair allocation (including the two studied here) are typically approximate, in the sense that they approximate the problem (as done by Spliddit) or they do not find the global solution, but rather a local one (as done by MinCovTarget).

In [4], a preliminary comparison between these algorithms against Nash welfare, social inequality, and envy has been carried out, but using MinCovTarget without proper consideration for the selection of target values as we do in this paper. Moreover, the Spliddit algorithm was not used there, whereas in this paper we provide a full numerical investigation (which we could not find in the literature) of the original algorithm. To broaden the scope of applications of our study, we also introduce a novel way of designing agents' valuations, taking into account dependence. Finally, we note that there exists a variety of practical methods for allocating indivisible goods amongst *two* agents [6]. These methods are not considered in this study.

2 Mathematical framework, social welfare and envy

The purpose of fair allocation is to allocate a finite set of $d \geqslant 1$ indivisible goods to a finite number of $n \geqslant 2$ agents so that the final allocation satisfies an a-priori fixed fairness criterion. We refer to [1], [3], [4], and references therein, for a full background on the classic economic problem of fair allocation and the many fairness criteria existing in the literature.

The starting point of any fair division problem is the so called value matrix $\mathbf{V} = (v_{ij})$, collecting the values $v_{ij} \geqslant 0$ given by agent i to item j, for $i \in \mathcal{N} := \{1, \ldots, n\}$ and $j \in \mathcal{D} := \{1, \ldots, d\}$. Examples of value matrices are illustrated in Figures 1–3. The binary matrix $\mathbf{X} = (x_{ij})$ represents

an allocation amongst the players, with $x_{ij} = 1$ if agent i receives item j, $x_{ij} = 0$ otherwise. We impose that

$$\sum_{i \in \mathcal{N}} x_{ij} = 1, j \in \mathcal{D},$$

that is each item is allocated to exactly one agent (and nothing is thrown away). Under an allocation $\mathbf{X}$, the total value gained by agent i is therefore given by

$$U_i(\mathbf{X}) = \sum_{j \in \mathcal{D}} x_{ij} v_{ij}.$$

The maximization of the sum of the utilities gained by each agent is called *maximum social welfare*, that is

$$\max_{\mathbf{X} \in \mathcal{A}} \sum_{i \in \mathcal{N}} U_i(\mathbf{X}), \tag{1}$$

where $\mathcal{A}$ is the set of all possible allocations of d objects to n agents. Notice that, being $\mathcal{A}$ finite (there are n^d possible allocations), the maximum in (1) is attained (there always exists at least one maximum social welfare allocation).

As anticipated in the introduction, *the gold standard of fairness* [3] is the search for minimum envy. An allocation $\mathbf{X}$ is said to be *envy-free* if no agent prefers the bundle of goods received by any other agent to the own one. Since an envy-free allocation might not exist (if only one item is to be allocated, someone is going to be unhappy), it makes sense to minimize the maximal envy among any pair of agents. Following [7], we define the envy $e(\mathbf{X})$ of an allocation $\mathbf{X}$ as the maximal envy among any pair of agents, i.e.,

$$e(\mathbf{X}) = \max\{e_{ik}(\mathbf{X}); i, k \in \mathcal{N}\}, \text{ where } e_{ik}(\mathbf{X}) = \max\{0, U_i^k(\mathbf{X}) - U_i^i(\mathbf{X})\}.$$

If $e_{ik}(\mathbf{X}) = 0$, then agent i does not envy agent k. We say that $\mathbf{X}$ is a *minimum envy* allocation if it solves

$$\min_{\mathbf{X} \in \mathcal{A}} e(\mathbf{X}). \tag{2}$$

Similarly to social welfare, a minimum envy allocation always exists.

3 Spliddit algorithm and Nash welfare

The Spliddit algorithm pursues *maximum Nash welfare* (MNW), i.e. it aims at maximizing the product (rather than the sum) of the total values gained by each agent. By applying a logarithm, this problem is equivalent to

$$\max_{\mathbf{X} \in \mathcal{A}} \sum_{i \in \mathcal{N}} \log U_i(\mathbf{X}), \tag{3}$$

i.e. to the maximization the sum of the log-utilities of each agent. Again notice that the maximum in (3) is attained. Spliddit does not solve directly (3), but an approximation which is given by the following mixed integer linear programming (MILP). With $\mathcal{M} = \{1, 3, \ldots, M-1\}$, Spliddit solves:

$$\max \sum_{i\in\mathcal{N}} W_i,\ s.t. \quad \begin{cases} W_i \leqslant \log(k) + (\log(k+1) - \log(k))(\sum_{j\in\mathcal{D}} x_{ij}v_{ij} - k), i \in \mathcal{N}, k \in \mathcal{M} \text{ (utility proxi)}; \\ \sum_{j\in\mathcal{D}} x_{ij}v_{ij} \geqslant 1, i \in \mathcal{N} \text{ (at least 1-utility to each agent)}; \\ \sum_{j\in\mathcal{N}} x_{ij} = 1, j \in \mathcal{D} \text{ (all goods are allocated to exactly one agent)}; \\ x_{ij} \in \{0,1\}, i \in \mathcal{N}, j \in \mathcal{D} \ (x_{ij} = 1 \text{ iff agent } i \text{ receives item } j). \end{cases} \tag{MILP}$$

There are a few assumptions under which the above MILP turns out to be a good approximation of the maximization of Nash welfare (3).

First, the logarithm is approximated by the minimum of a finite family of affine functions (this is the first constraint), which is equal to the logarithm at integral points. We notice that this can be done if the log function is replaced by any concave utility function f, and in general works by replacing the term $(f(k+1)-f(k))$ by $f'(k)$. Here, it is worth remarking that maximizing the sum of logarithmic utilities implicitly assumes that the agents are strictly risk averse. Agents should be aware of this when declaring their values to a regulator running the algorithm.

The utility approximation described above is possible since Spliddit requires integral valuations of the goods, that is each agent is asked to distribute M points (with $M = 1000$ in the original version) amongst the d items. This means that each row of the value matrix $\mathbf{V}$ adds up exactly to M, i.e.

$$\sum_{j\in\mathcal{D}} v_{ij} = M, i \in \mathcal{N}.$$

The fact that agents' utilities are integral and can at most attain the value M is an essential feature of Spliddit, which is necessary to make it work. The value of M gives an implicit upper bound on the dimensionality of the problems that Spliddit can handle: valuations provided by agents for more than $M = 1000$ items cease to be fully meaningful, and the value of M cannot be increased without heavily affecting the cost of computation of the final solution.

The second constraint of the above MILP forces a solution where each agent receives at least 1-utility, i.e. at least one object. This prevents each agent from receiving nothing and hence the corresponding logarithm from being minus infinity. In the original version, a preprocessing steps finds the largest set of players for which this is feasible. In any case, the fulfilment of this constraint implies that Spliddit can only be run when $d \geqslant n$.

The third and fourth constraints in the MILP represent the standard requirements in the fair allocation of indivisible items.

These assumptions appear to be reasonable and make it possible to obtain in principle allocations that exhibit maximum Nash welfare. Spliddit has its Achilles' heel, however, in the limited dimensionality of the instances it can handle. On our machine, Spliddit can generally handle up to a few dozens of agents within a minute, thus is able to deal with most requests coming from private customers (e.g. for inheritance or divorce disputes). However, already with $d \geqslant 100$ goods to be allocated to $n = 30$ agents, the above MILP becomes intractable.

4 MinCovTarget algorithm and social inequality

The MinCovTarget+ algorithm that we propose below is an efficient version of the so-called MinCovTarget and MinCov algorithms. The MinCov algorithm aims to obtain an allocation with *minimum social inequality*, i.e. under which the utilities gained by each agent are made as similar as possible, and this under each agent's valuation. Formally, denoting by

$$U_i^k(\mathbf{X}) = \sum_{j \in \mathcal{D}} x_{kj} v_{ij}$$

the utility gained by agent k under agent i valuations (note that $U_i^i(\mathbf{X}) = U_i(\mathbf{X})$), the MinCov algorithm tries to minimize the social inequality $I(\mathbf{X})$ of the system defined by

$$I(\mathbf{X}) = \frac{\sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{N}} \left(U_i^k(\mathbf{X}) - M/n\right)^2}{n^2}.$$

The MinCovTarget algorithm introduces a malus $\tau_i > 0$ (called the target value) for agent i in order to prioritize the own utility with respect to that of the others. The MinCovTarget algorithm aims at solving

$$\min_{\mathbf{X} \in \mathcal{A}} \; I_\tau(\mathbf{X}), \tag{4}$$

where

$$I_\tau(\mathbf{X}) = \frac{\sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{N}} \left(U_i^k(\mathbf{X}) - \mathbf{1}_{\{k=i\}} \tau_i - (M - \tau_i)/n\right)^2}{n^2}.$$

Since Spliddit forces the total value of the goods to be the same for each agent, in the following we assume that the target value is homogeneous, i.e. $\tau_i = \tau, i \in \mathcal{N}$. The target allows one to interpolate between the search for social equality (for $\tau = 0$, leading to MinCov) and the rule of maximum social welfare (for $\tau \to \infty$).

The MinCovTarget algorithm is based on a series of efficient local swaps: starting from a trivial allocation in which the first agent is given all goods, it reallocates one item at the time so that $I_\tau(\mathbf{X})$ is maximally reduced on the single swap. MinCovTarget does not rely an any particular assumption besides the fact all the components of $\mathbf{V}$ have to be positive, something that can always be reached by adding a (small) constant to all v_{ij}. Actually, this assumption can be weakened to require that at least one agent – say agent 1 – gives positive utilities to all objects. Agents' utilities are evaluated linearly and the initial value matrix does not need to be standardized.

The MinCovTarget+ algorithm uses an entire set $\mathcal{T}$ of different target values, and hence finds an allocation for each target value used. Amongst all the allocations found, it selects the ones having minimal envy, and, amongst the ones with minimal envy, it outputs the one(s) with maximal social welfare.

MinCovTarget+ algorithm

Let $\mathcal{T}$ be a non-empty set of target values. For all $\tau \in \mathcal{T}$, repeat the following procedure:

1. give all items to the first agent;
2. choose one item at random and, if possible, re-allocate it to another agent so that the inequality $I_\tau(\mathbf{X})$ of the new allocation is maximally reduced;
3. repeat 2. until the inequality of the allocation found has not been reduced for d iterations of 2., and denote by $\mathbf{X}_\tau$ the final allocation found.

Amongst all the allocations $\mathbf{X}_\tau, \tau \in \mathcal{T}$, having minimal envy, output the one(s) with maximal social welfare.

The exact way to choose the agent to which to reallocate the item chosen in 2. is described in [4] and is based on a matrix reformulation of the problem. In practice, step 2. prescribes to reallocate the chosen item to the poorest agent, where wealth is evaluated through a particular weighted average of the remaining goods. This allows, at each step, to obtain the maximal reduction of the inequality in the economic system.

Notice that the MinCovTarget+ is an heuristic algorithm that compares local solution, i.e. allocations with an inequality $I_\tau(\mathbf{X})$ that cannot be reduced by performing further local swaps. The final allocation found, however, is not guaranteed to be the global solution, that is the allocation with maximal social welfare amongst all the ones having minimal envy.

5 An illustrative example

We give a pedagogical example which illustrates a typical low-dimensional framework of the fair allocation problem. We choose a setting with $d = 10$ items to be allocated to $n = 4$ agents. For these small dimensions, it is possible to perform a brute force search and solve the fair allocation problem exactly by full enumeration of all the possible allocations, which are roughly one million ($n^d = 4^{10} = 1048576$).

Suppose that a regulator asks each agent to privately disclose the own valuations of each item to be allocated, and that all such valuations are collected in the matrix $\mathbf{V}$ given in Figure 1.

In Table 1 we compare the allocations found by performing a Brute Force (A) search for maximum social welfare, and by running the Spliddit (B), MinCovTarget+ (C) and MinCov (D) algorithms. In the table we report the items received by each agent under the allocations found by the algorithms, and a matrix representing each agent's valuations of the own bundle (in bold) and of the bundles received by the others.

From Table 1A it is evident why the search for maximum social welfare is not a good fairness rule to be prioritized in general. The (only) maximum social welfare allocation gives nothing to agent 1, who envies all the other agents. In particular agent 1 envies most the bundle received by agent 2, which evaluates at 426. So this allocation, though maximizing the sum of utilities of all agents (the sum of the diagonal elements in the matrix shown) produces an envy which is almost half of the total value of the goods. We notice that this allocation can be more rapidly found by using Spliddit with a linear function f in the MILP.

Spliddit (1B) works perfectly as it finds the unique allocation maximizing Nash welfare. However, this allocation has a level of envy of 21 (the fourth player gets 305 and is envying the second

agent's bundle which evaluates 326) and a relatively good level of social welfare (1506).

MinCovTarget+ (1C) is run with a set of 51 different target values $\mathcal{T} = \{0, 40, \ldots, 1960, 2000\}$. The advantage of using MinCovTarget+ is that, even if the average level of envy (85.3) of the 51 allocations found is larger than the envy of the allocation found by Spliddit (21), the algorithm selects the two target levels attaining the minimum (zero) level of envy. Between the two no-envy allocations, MinCovTarget+ outputs the one having the largest level of social welfare. In the end, MinCovTarget+ finds a no-envy allocation with a relatively small reduction of social welfare (1482) as compared to Spliddit.

In Table 1D we also report the final allocation found by MinCov. Even if MinCov gives the best result for social inequality, it fails to find the global solution, and by far misses good results for the other rules. We remark here that there are 24 best allocations for inequality (found by Brute Force) and the minimal envy one has an envy of 8 and a social welfare of 1228. From these figures, the rule of minimal inequality does not appear to produce good allocations.

In a framework where a full enumeration of all possible allocations is feasible in a reasonable time, one should prefer a solution with no envy (everybody is happy), maximizing a secondary objective function amongst all possible no-envy allocations. As a secondary rule to choose amongst no-envy allocations, we select social welfare (but one can set whatever fairness rule at this point). Following this methodology, in this example one should select the allocation with an envy of 0 and a social welfare of 1498 (these are the bundles received by the agents: $a1 = \{2, 3, 4\}$, $a2 = \{5, 7\}$, $a3 = \{8, 10\}$, $a4 = \{1, 6, 9\}$). We notice that the level of social welfare of this allocation (1498) is very close to the one (1482) of the no-envy allocation found by MinCovTarget+ (at a fraction of the time spent by Spliddit and Brute Force).

We remark that the maximum level of social welfare is not attained by a no-envy allocation, or, equivalently, that there does not exist in general a no-envy solution maximizing social welfare. This shows how the various fairness rules introduced here and in the literature are in general conflicting: prioritising a no-envy solution might imply a reduction of social welfare.

A brute force search in principle provides the optimal solution for any fairness rule, but enumerating the possible n^d allocations quickly becomes intractable with increasing dimensions. For $d = 100$ goods to be allocated to $n = 10$ agents, one of the scenarios treated in Section 7, there likely are more possible allocations (10^{100}) than atoms in our universe. Higher dimensional scenarios then call for an algorithm to allocate goods.

The example shown in this section is pedagogical in nature and its results cannot be representative of the performance of the algorithms in general (also because MinCovTarget+ contains randomness in the selection of items to be subsequently re-allocated). To obtain a proper comparison, in what follows we will simulate 1000 value matrices and count the number of success of the MinCovTarget+ and Spliddit algorithms. It seems senseful to compare the performance of the algorithms against minimal envy, maximum Nash and maximum social welfare.

```
V =

   121    94   141   142    63    97   101    97    41   103
   193    21   205   103   195     8   161    36    10    68
    23    51    29   145   144   154   135   128    18   173
   159    95   169    25   167   162    68     6   143     6
```

Figure 1: Value matrix for $d = 10$ items to be allocated to $n = 4$ players.

1A. Brute force (maximum social welfare) allocation, 17 secs

items	agent	utilities				fairness rule	value	optimum
		1	2	3	4			
–	1	**0**	426	342	232	envy	426	0
1, 3, 5, 7	2	0	**754**	207	39	social welfare	**1600**	1600
4, 8, 10	3	0	331	**446**	223	Nash welfare	-Inf	10.2890
2, 6, 9	4	0	563	37	**400**	social inequality	50106	1972

1B. Spliddit (maximum Nash welfare) allocation, 1 sec

items	agent	utilities				fairness rule	value	optimum
		1	2	3	4			
2, 3, 4	1	**377**	184	301	138	envy	21	0
1, 5	2	329	**388**	265	18	social welfare	1506	1600
7, 8, 10	3	225	167	**436**	172	Nash welfare	**10.2890**	10.2890
6, 9	4	289	326	80	**305**	social inequality	12649	1972

1C. MinCovTarget+ (minimal envy) allocation, 0.032 sec

items	agent	utilities				fairness rule	value	optimum
		1	2	3	4			
3, 4	1	**283**	222	263	232	envy	**0**	0
1, 7	2	308	**354**	299	39	social welfare	1482	1600
5, 8, 10	3	174	158	**445**	223	Nash welfare	10.2512	10.2890
2, 6 ,9	4	194	227	179	**400**	social inequality	9229	1972

1D. MinCov (minimal inequality) allocation, 0.010 sec

items	agent	utilities				fairness rule	value	optimum
		1	2	3	4			
2, 3	1	**338**	160	280	222	envy	151	0
5, 6	2	294	**203**	149	354	social welfare	1059	1600
4, 8, 9	3	253	298	**291**	158	Nash welfare	9.6563	10.2890
1, 7	4	270	329	174	**227**	social inequality	4256	1972

Table 1: Allocations found by various algorithms for the value matrix **V** in Figure 1. We give the logarithm of Nash welfare.

6 Design of the value matrix

In order to perform a numerical comparison of algorithms for fair allocation (against different allocation rules), one needs a way of simulating random value matrices, to replicate different real life situations.

In [3], random *uniform* valuations are used. In the case of Spliddit, this means that each agent is assumed to randomly assign the M available points to the d objects in a uniform way. Following this approach, the values x_{ij} are independently drawn from a uniform distribution between 0 and M and hence standardized and rounded so that all valuations of a single agent are integral and add to M. To run a comparison with MinCovTarget, we possibly adjust each generated matrix so that the first agent gives positive utility to all objects (again notice that this can always be reached by adding $\epsilon > 0$ in the cases the Spliddit constraint on agents' utilities is not binding). An example of a simulated value matrix $\mathbf{V}$ for $n = 4, d = 10$, is shown in Figure 2.

```
V =

    82    30    78    40    82   158   173    33   189   135
   104    15    78   127    81   140   130   127    77   121
     0    94   212    14    71   158    43    50   349     9
    68    78   155   152    45   157     9    95    71   170
```

Figure 2: An example of simulated value $\mathbf{V}$ matrix with random uniform valuations for $n = 4, d = 10$. According to Spliddit rules, each agent assign $M = 1000$ points amongst the different goods (each row adds to 1000).

In this study we introduce a new, more general, design for the simulation of value matrices, that we believe to be more realistic than random uniform valuations. In practice, valuations of the same object by different agents should be dependent, in particular positively correlated. If, as part of an inheritance, we have to allocate a gold and a plastic watch, it is more realistic to assume that all agents will be likely to give higher valuations for the first and, conversely, lower valuations for the latter.

Thus, we introduce a second way of generating value matrices, where we assume that the valuations (uniformly distributed between 0 and M) of the same object are dependent with the dependence modelled by a Gaussian copula (see for instance [8], [5]) having equicorrelation matrix with correlation parameter equal to $\rho \in [0, 1]$. The correlation parameter ρ allows one to interpolate between the previously introduced case of uniform independent valuations ($\rho = 0$) and the perfect correlation case ($\rho = 1$) in which all agents give identical valuations of the same object. In the simulations to follow, we find it suitable to assume $\rho = 0.5$. An example of simulated value matrix $\mathbf{V}$ with the novel procedure, for $n = 4, d = 10$, and $\rho = 0.5$ is shown in Figure 3. The code for matrix design and simulations can be downloaded from the link provided in the abstract and can be of interest for other applications.

```
V =

   202   151    55   152    62    92   153    31    76    26
   152   212    13   105    95    22   173     3   123   102
    90   139    97   114    10    78   179    36   146   111
   205    79     5   170    40    16   200    15   166   104
```

Figure 3: An example of a simulated value $\mathbf{V}$ matrix with dependent valuations for $n = 4, d = 10$, and $\rho = 0.5$.

7 Numerical results

In this section, we compare the MinCovTarget+ and the Spliddit algorithms under different scenarios with increasing dimensionality of the fair allocation problem. For each choice of n and d (that we call a scenario), we simulate a number of 1000 value matrices (with uniform and dependent design) and evaluate the % of times one algorithm is no worse than the other (ties are counted as success so cumulative percentages in the following figures exceed 100%) against the rules of minimum envy, maximum Nash welfare, and maximum social welfare. For each simulated value matrix we run MinCovTarget on 51 different target values $\mathcal{T} = \{0, 40, \ldots, 1960, 2000\}$.

When running our experiments we noted that Spliddit suffers from the curse of dimensionality and cannot be really applied when the number n of agents or the number d of goods is relatively large. Indeed, when selecting $d = 300$ goods to be allocated to $n = 30$ agents, our code was able to compute a Spliddit solution within 10 minutes in only 7 cases over 50 simulations of the value matrix. By contrast, MinCovTarget+ can handle the same case in about 16 seconds, always finding a solution with no envy. Therefore, in order to have a meaningful comparison of the different algorithms we first deal with low to medium dimensional set-ups.

7.1 Comparison in low to medium dimensions

In Figure 4 we show percentages of success of each algorithm in three different scenarios with increasing dimensions, and uniform (left figures) and dependent (right figures) valuations of the goods.

For mid-dimensional scenarios with $n = 10$ and $n = 20$, Spliddit and MinCovTarget+ *always* find a no-envy solution. By the design of the two algorithms, the no-envy allocations found by Spliddit guarantees a higher level of Nash welfare, whereas MinCovTarget+ produce better allocations with respect to social welfare. We notice that the MinCovTarget+ algorithm (even including the selection of the optimal target value) is much faster than Spliddit, and always succeed in finding a solution within seconds. On the contrary Spliddit, for $n = 20, d = 200$, is not able to find a solution within 2 minutes for the 9.5% (10.9%) of the uniform (dependent) value matrices simulated in the experiment.

In the low-dimensional scenario ($n = 4, d = 10$), MinCovTarget+ turns out to be much better than Spliddit with respect to envy, as a result of the possibility of selecting the best target level. However, as noticed in the illustrative example given above, the selection of no-envy allocations comes at a slight reduction of social welfare, so in this low-dimensional scenarios Spliddit has higher success rates with respect both Nash and social welfare.

7.2 Performance of MinCovTarget+ in high dimensions

The MinCovTarget+ algorithm can also handle higher dimensional scenarios, with envy performance increasing in the number of goods. In Figure 6, left, we show the computation time of MinCovTarget+ for different number of agents with increasing number of objects. In these dimensions we cannot compare the performance of MinCovTarget+ with any competitor. However, in the right part of the figure, we show the level of envy attained as a percentage of M, the total value of goods. The performance of MinCovTarget+ with respect to minimum envy is excellent and the allocations found are expected to provide a very good approximation of maximum social welfare. As an example, MinCovTarget+ finds a no-envy allocation for a fair allocation problem with $n = 50$ agents and $d = 250$ goods in less than 20 seconds.

7.3 MinCovTarget*

The choice of the set of target values is pivotal to the MinCovTarget+ algorithm. Our choice of 51 target values (from a null target value to twice the total value of goods) provides a good trade off between the search for no-envy and high social welfare allocations, and computation time. Even if the MinCovTarget+ can also handle higher dimensional scenarios, for really huge framework one could be willing to save computational time by reducing the set of target values used by the algorithm.

A good option for huge dimensional scenarios is to use a single target level set equal to the total value M of the goods to be allocated, that is to set $\mathcal{T} = \{M\}$. In this case we call the algorithm MinCovTarget*.

The choice of this particular target level comes from observing Figure 5, which shows the average values of envy, Nash welfare and social welfare attained by the MinCovTarget+ algorithm with a single, fixed choice of the target. The figure shows how a target value equal to M is able to approximate the Nash welfare and slightly improve the social welfare attained by Spliddit by keeping the level of envy reasonably close to zero. Smaller values of the target would decrease Nash and social welfare, higher values would increase envy.

The MinCovTarget* algorithm can handle huge scenarios within seconds, with envy performance increasing in the number of goods. In Figure 7, left, we show the computation time of MinCovTarget* for different number of agents with increasing number of objects. In the right part of the figure, we show the level of envy attained as a percentage of M, the total value of goods. The performance of MinCovTarget* with respect to minimum envy is excellent and the allocations found are expected to provide a very good approximation of maximum social welfare. As an example, MinCovTarget* is able to handle a fair allocation problem with $n = 200$ agents and $d = 400$ goods in about 26 seconds, keeping the level of envy under 0.5% of the total value of goods; see Figure 7.

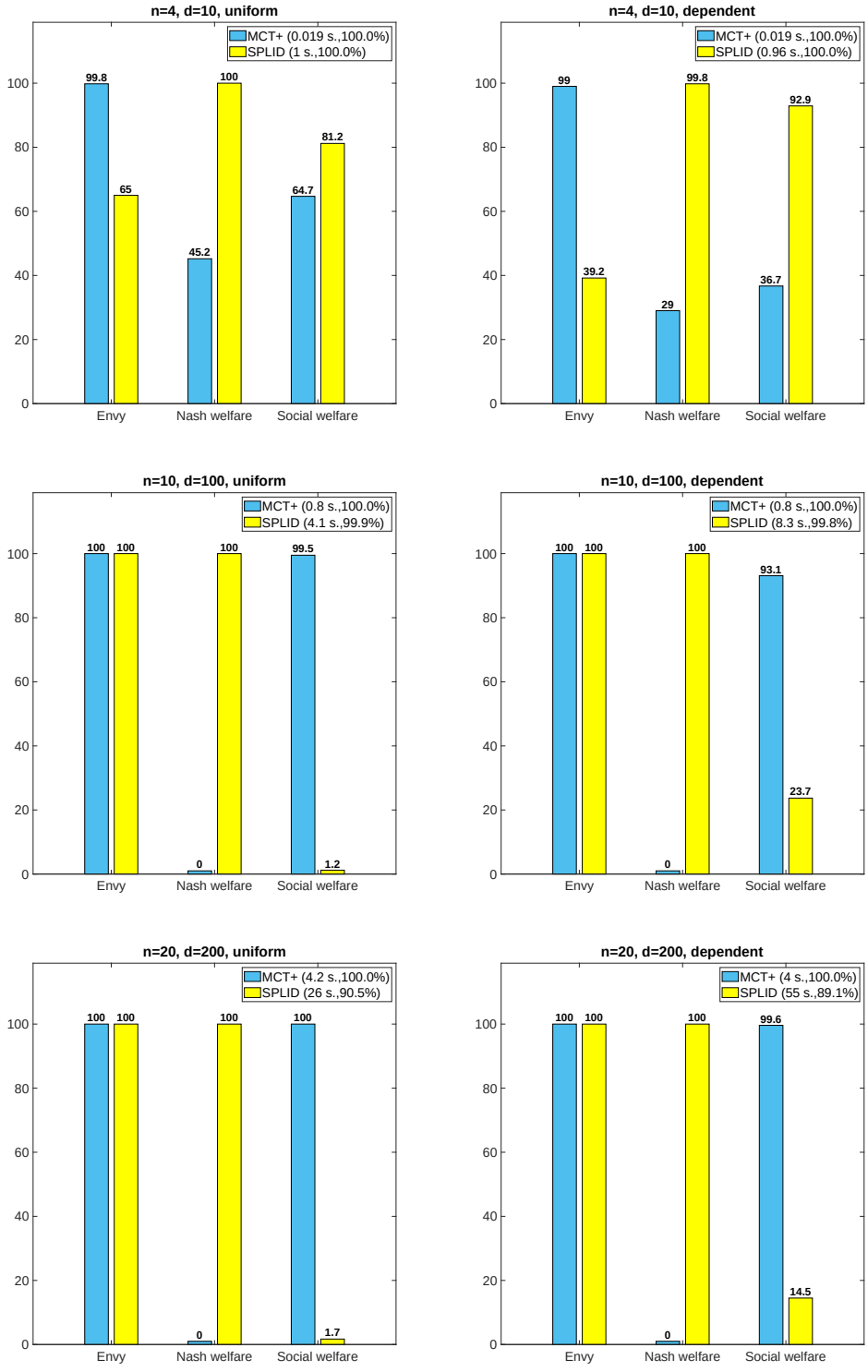

Figure 4: For three different scenarios, we show the percentage of times that each algorithm performs best (ties are counted as success) with respect to three different fairness criteria. Average computation times are shown in the legend with the indication of the algorithm success rate, that is the % of times that an algorithm was able to find a solution in less than 2 minutes (in case of failure, the simulated matrix is not taken into account). Figures are computed over a total number of 1000 random *uniform* matrices (left figures) and *dependent* matrices (right figures). We remark that for $n \geqslant 10$ both algorithms always find *no-envy* solutions.

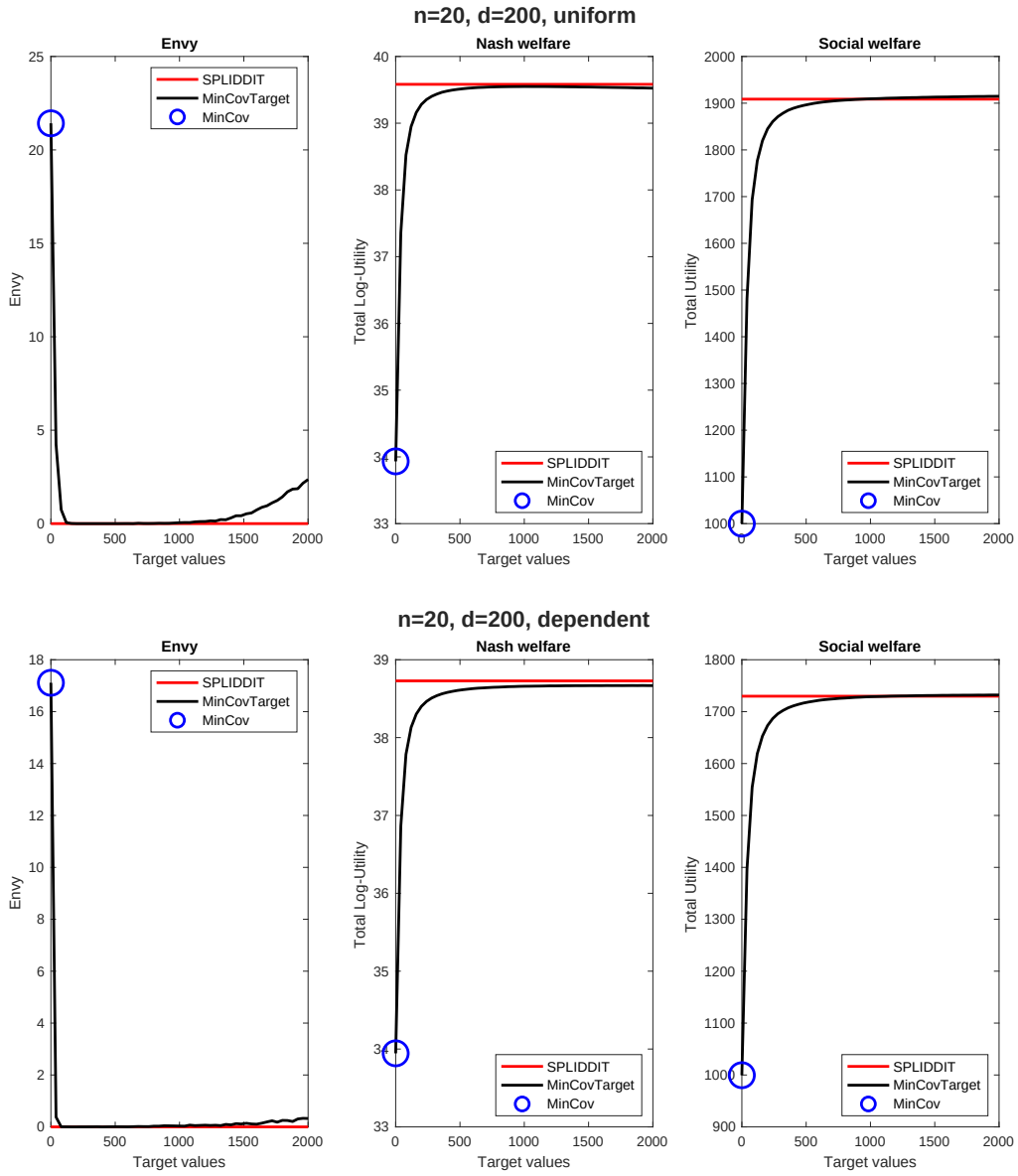

Figure 5: For $d = 200$ goods to be allocated amongst $n = 20$ agents, we show how the value of three different objective functions vary against the choice of the fixed target value for MinCovTarget+ used with a single target value. The red line indicates the value attained by Spliddit while the blue circle corresponds to MinCov ($\tau = 0$). For a fixed target value, MinCovTarget+ figures are averages over 1000 simulations of the value matrix. Only simulations of the value matrix for which Spliddit was able to find a solution in less than 2 minutes are taken into account. These corresponds to 90.5% of the cases for uniform value matrices (top plot) and 89.1% of the cases for dependent value matrices (bottom plot).

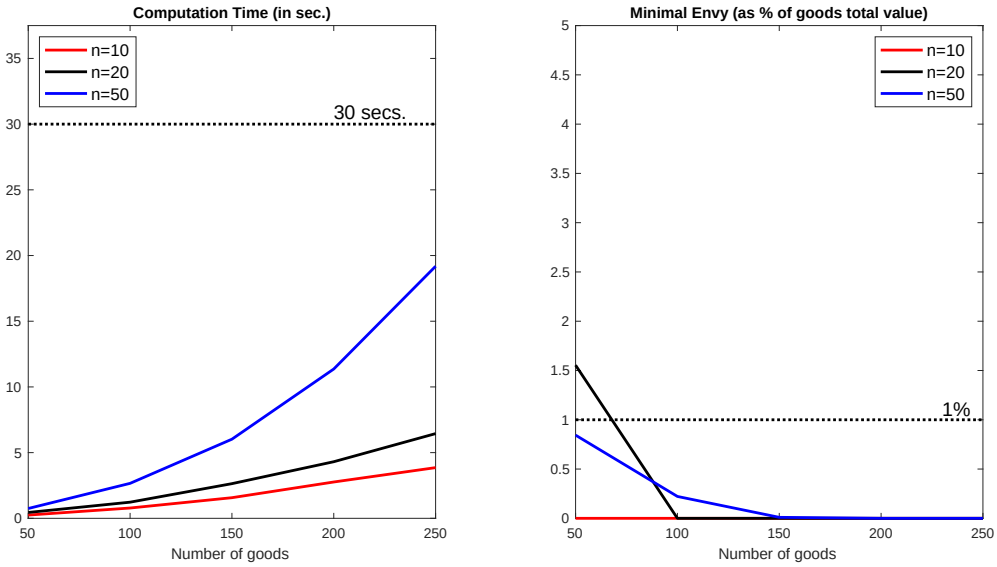

Figure 6: Computation time in seconds and minimum envy attained (as a % of the total value M of goods) by the MinCovTarget+ algorithm. Figures are averages over 50 simulations of a uniform value matrix.

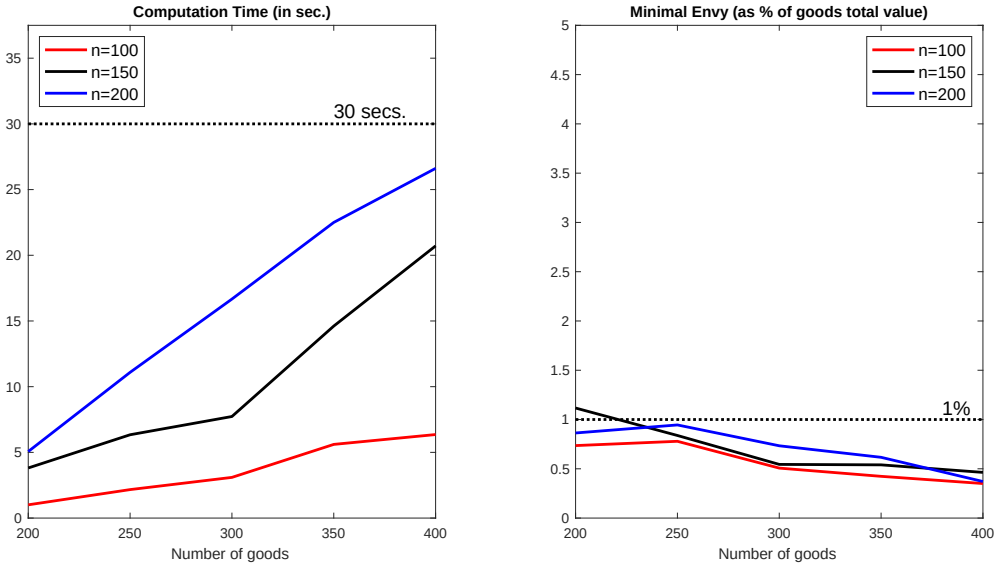

Figure 7: Computation time in seconds and minimum envy attained (as a % of the total value M of goods) by the MinCovTarget* algorithm. Figures are averages over 20 simulations of a uniform value matrix. To allow for meaningful agents' valuations, for these scenarios the value of all goods has been increased to $M = 10000$.

8 Conclusions

In this paper, we compare two state-of-the-art algorithms for the fair allocation of a number of indivisible goods amongst several agents: the Spliddit and MinCovTarget algorithms. Based on our numerical analysis, we propose two versions of the MinCovTarget algorithm, which we call MinCovTarget+ and MinCovTarget*.

If one agrees that, in a fair allocation problem, no-envy allocations should be preferred and, in the case many no-envy allocations exist, one should select the one attaining maximum social welfare, MinCovTarget+ is shown to be a new solution for the fair allocation problem.

With respect to minimal envy, MinCovTarget+ is better than or compares to Spliddit in all scenarios, but in larger dimensional frameworks provides allocations with a higher value of social welfare and at a fraction of the computation time spent by Spliddit. Moreover, in high dimensions, Spliddit cannot be readily implemented whereas MinCovTarget+ is able to rapidly obtain envy-free allocations.

Finally, if one is willing to save computation time at the cost of a reasonable extra level of envy, the so called MinCovTarget* algorithm, that is MinCovTarget+ used with a target value equal to the total value of goods, is able to handle huge dimensional scenarios (hundreds of agents and goods) within seconds. The interested reader is invited to contribute to this project by downloading our code, replicating our experiments, and possibly suggesting extensions or further analytics.

References

[1] Amanatidis, G., G. Birmpas, A. Filos-Ratsikas, and A. Voudouris (2022). Fair division of indivisible goods: A survey. In L. Raedt (Ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence - Survey Track (IJCAI-22)*, pp. 5385–5393. International Joint Conferences on Artificial Intelligence Organization.

[2] Bouveret, S., M. Lemaître, H. Fargier, and J. Lang (2005). Allocation of indivisible goods: a general model and some complexity results. In *Proceedings of the 4th international joint conference on Autonomous Agents and Multiagent Systems*, pp. 1309–1310.

[3] Caragiannis, I., D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang (2019). The unreasonable fairness of maximum nash welfare. *ACM Transactions on Economics and Computation (TEAC) 7*(3), 1–32.

[4] Cornilly, D., G. Puccetti, L. Rüschendorf, and S. Vanduffel (2022). Fair allocation of indivisible goods with minimum inequality or minimum envy. *European Journal of Operational Research 297*, 741–752.

[5] Durante, F. and C. Sempi (2016). *Principles of Copula Theory*. CRC Press, Boca Raton FL.

[6] Kilgour, D. M. and R. Vetschera (2018). Two-player fair division of indivisible items: Comparison of algorithms. *European Journal of Operational Research 271*(2), 620–631.

[7] Lipton, R., E. Markakis, E. Mossel, and A. Saberi (2004). On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM conference on Electronic Commerce*, pp. 125–131.

[8] Nelsen, R. B. (2006). *An Introduction to Copulas* (2nd ed.). Berlin: Springer.

[9] Nguyen, T., M. Roos, and J. Rothe (2013). A survey of approximability and inapproximability results for social welfare optimization in multiagent resource allocation. *Ann. Math. Artif. Intell. 68*, 65–90.

[10] Shah, N. (2017). Spliddit: two years of making the world fairer. *XRDS: Crossroads, The ACM Magazine for Students 24*(1), 24–28.