Università degli Studi "G. D'Annunzio"
Dipartimento di Scienze

Filling knapsacks with candies: Integer Linear Programming in Fair Division

Marco Dall'Aglio Raffaele Mosca

First draft: July 2004
This version: November 2005

Technical Report no. R-2006-001 *Research Series*

Filling knapsacks with candies: Integer Linear Programming in Fair Division

Marco Dall'Aglio [1] Raffaele Mosca [1]

[1] *Dipartimento di Scienze*
Università d'Annunzio
Viale Pindaro, 42
65127 — Pescara, Italy
maglio@sci.unich.it, r.mosca@unich.it

First draft: July 2004
This version: November 2005

Abstract. We consider the problem of allocating a finite number of indivisible items to two players. The search for a minimax allocation can be formulated as an Integer Linear Programming (ILP) problem, carrying some similarities with the 0-1 knapsack problem. Typical tools of integer programming, such as dynamic programming or the branch and bound algorithm can be successfully adapted to design new procedures in fair division.

Keywords: *Fair division, Adjusted Winner procedure, Integer programming*

Contents

1 Introduction **3**

2 The problem **4**

3 A dynamic programming approach **5**
3.1 A method from Optimization 0-1 Knapsack 5

4 Relaxing Integer Fair Division: The Adjusted Winner procedure **9**
4.1 The Adjusted Winner (AW) algorithm 9
4.2 Computational efficiency of AW 15
4.3 The maximin problem with initial endowments 17

5 A branch and bound algorithm **19**
5.1 A variable elimination test 20
5.2 The algorithm 21

A Appendix: The examples in detail **23**
A.1 Example 3.1 23
A.2 Example 3.2 25

B Acknowledgements **27**

1 Introduction

This paper presents two different algorithms for allocating a set of indivisible items between two players with subjective preferences over the items.

While most of the literature in fair division theory deals with with one or more completely divisible goods (such as cakes or pieces of land), a series of papers by Brams with several coauthors (see [3] and [4]) drew attention on the problem of allocating several indivisible items.

When it comes to the design of specific procedures, however, it turns out that most of the proposals devise some technique to treat some, or all, of the contended items as divisible. This is the case of randomization, where items are given according to a probability distribution, or of side payments to compensate the giving up of some item. As noted in [8], there are situations where these methods are impractical or impossible to implement.

If we focus on methods that deal exclusively with the allocation of indivisible items, with no side actions to mitigate the discontent of some players, we find, quite surprisingly, a very narrow choice. A recent addition to the classical methods of strict and balanced alternation, described in [5] and [6], was given by Herreiner and Puppe in [8], with their descending demand procedure. Each player, in turn, declare their most preferred bundle (i.e. a collectio0n of items) until a feasible arrangement is met.

Equally short is the list of papers where programming techniques are used in the design of fair division procedures. In [9] Kuhn defines a linear program that has the Knaster rule for the efficient allocation of items with side payments as its solution. Demko and Hill [7] define a maximin optimization problem. They show that this problem is computationally intractable and provide a lower bound for optimal value. The second half of the paper deals with randomized solutions for the same problem and how these can be computes through linear programming and duality techniques.

We adopt the same framework used in [7], focusing on the case of two players. Each player assigns a non-negative value to each item. The evaluations are additive, but no normalization is required, so the total value of the items may differ for the two players. As in [7] we investigate the close relationship between fair division theory and operations research, but we take a different direction. We note that this fair division problems bears some formal analogy with the classical 0-1 knapsack problem. We study this resemblance with the aim of providing new procedures in fair division. We come up with two algorithms, based respectively on dynamic programming and branch-and-bound techniques. The former is computationally fast and partially mitigates the negative results stated in [7], while the latter keeps the procedural appeal of a widely used procedure in fair division: the Adjusted Winner algorithm.

2 The problem

We consider the following problem.

Let $M = \{1, \ldots, m\}$ be a set of items which have to be divided between two players. Let $a_1, a_2, \ldots, a_m$ ($b_1, b_2, \ldots, b_m$ resp.) be the non-negative evaluations of the various objects by Player 1 (Player 2, resp.).

An *integer allocation* for the m items is described by a vector $x = (x_1, \ldots, x_m) \in \{0,1\}^m$. If $x_i = 1$ (resp. $x_i = 0$), then item i goes to Player 1 (resp. Player 2).

The satisfaction of the two players is given by, respectively,

$$v_1(x) = \sum_{i \in M} a_i x_i \qquad v_2(x) = \sum_{i \in M} b_i (1 - x_i) \tag{1}$$

A popular interpretation of this model is the following: two children, Alice and Bob, are given a set of m hard candies to be shared between themselves. Candies are indivisible and each of them is assigned to one of the children. Children value the sweets according to their personal taste. An allocation is sought that is optimal according to the social welfare criterion. Here, the maximin criterion is considered.

The 2-player Integer Fair Division (IFD) problem is the following:

Find an integer allocation $x = (x_1, \ldots, x_m)$ that achieves

$$z^* = \max_{x \in \{0,1\}^m} \min\{v_1(x), v_2(x)\} \tag{IFD}$$

This problem can be written as an integer linear program

$$\begin{array}{lll} \max & z & \\ \text{s.t.} & \sum_{i \in M} a_i x_i \geq z & \\ & \sum_{i \in M} b_i (1 - x_i) \geq z & \\ & x_i \in \{0,1\} & i = 1, \ldots, m \end{array} \tag{2}$$

As noted in [7], (IFD) is NP-hard. In fact, assume that $a_i = b_i$ for every $i \in M$: then solving IFD gives an answer to the problem of finding a partition of a set of positive integers in two subsets of equal sum, which is NP-complete (see for instance [14]).

A solution for (IFD) exists, but may not be unique. Moreover, while a maximin solution is always equitable in the divisible case, the players' value may differ in the present situation. In particular, we may single out the maximin allocation that maximizes the gap between the preferences, $|v_1(x) - v_2(x)|$. This is referred to as the *equimax* (or Rawls, or Dubins-Spanier) allocation. This also corresponds to the balanced solution in [8].

In what follows, we are primarily concerned with finding a maximin solution. Within the framework of each method we report whether we are able to find all the maximin solutions.

3 A dynamic programming approach

3.1 A method from Optimization 0-1 Knapsack

We refer to a method introduced in [13], p. 420, for Optimization 0-1 Knapsack problem. For $k = 1, ..., m$, define $M(k) = \{1, ..., k\} \subseteq M$. Let $A(m) = \sum_{i \in M} a_i$ and $B(m) = \sum_{i \in M} b_i$. Consider

$$z_k(d) = \max_{x \in B^k} \{ \sum_{i \in M(k)} a_i x_i : \sum_{i \in M(k)} b_i(1 - x_i) \geq d \} \quad \text{for } d = 0, 1, ..., B(m) \tag{3}$$

Let z^* be the value of an optimal solution of IFD. Then one has that

$$z^* = \max \{ \min\{z_m(d), d\} : d = 0, 1, ..., B(m) \} \ . \tag{4}$$

Therefore z^* can be computed once the values $z_m(d)$ for $d = 0, 1, ..., B(m)$ are known.

We proceed recursively, initializing the recursion with

$$z_1(d) = \begin{cases} a_1 & \text{if } d \leq 0 \\ 0 & \text{if } 0 < d \leq b_1 \\ -\infty & \text{otherwise (i.e., the problem is meaningless)} \ . \end{cases} \tag{5}$$

Note that if $x_k = 1$ in an optimal solution of (3) then

$$z_k(d) = a_k + \max_{x \in B^k} \left\{ \sum_{i \in M(k-1)} a_i x_i : \sum_{i \in M(k-1)} b_i(1 - x_i) \geq d \right\} = a_k + z_{k-1}(d)$$

On the other hand, if $x_k = 0$ in an optimal solution of (3) then

$$z_k(d) = \max_{x \in B^k} \left\{ \sum_{i \in M(k-1)} a_i x_i : \sum_{i \in M(k-1)} b_i(1 - x_i) \geq d - b_k \right\} = z_{k-1}(d - b_k)$$

Hence for $k = 2, ..., m$ and $d = 0, 1, ..., B(m)$, we obtain

$$z_k(d) = \max\{a_k + z_{k-1}(d), z_{k-1}(d - b_k)\} \tag{6}$$

A backtracking procedure yields the optimal solution. The basic idea is to start with m and d^*, an integer that attains optimality in (4) and plug $z_m(d^*)$ in (6). Set $x_m = 1$ or $x_m = 0$ depending on which member attains the maximum in the recursion formula. Continue recursively, starting with $z_{m-1}(d^*)$ or $z_{m-1}(d^* - b_m)$, depending on the value of x_m.

Example 3.1. Suppose Alice and Bob share 8 hard candies with the following preferences

candy	1	2	3	4	5	6	7	8
Alice	12	18	50	40	20	20	10	5
Bob	5	10	35	30	15	22	30	28

The dynamic procedure yields the optimal value

$$z^* = \max\{\min\{z_8(d), d\}, d = 0, 1, \ldots, 175\} = 102$$

The backtracking procedure returns the optimal solution

$$x_1^* = x_3^* = x_4^* = 1 \qquad x_2^* = x_5^* = x_6^* = x_7^* = x_8^* = 0$$

It will be shown later that this optimal solution is also unique.

In order to obtain all the maximin solutions, the symmetric problem with the roles of the two players reversed, should be considered as well.

Example 3.2. Suppose that 4 candies are to be shared, with values,

candy	1	2	3	4
Alice	32	28	22	18
Bob	25	25	25	25

The dynamic programming approach to (IFD) returns the following solution

$$x_1^* = x_2^* = 1 \qquad x_3^* = x_4^* = 0 \tag{7}$$

with value $z^* = 50$. This solution is not unique. In fact, if we swap the objective function and the constraint in (3) (and let d range within 0 and $A(m)$), we obtain a whole set of new solutions

$$\begin{aligned} x_1^* = x_3^* = 1 \qquad & x_2^* = x_4^* = 0 \\ x_2^* = x_3^* = 1 \qquad & x_1^* = x_4^* = 0 \\ x_1^* = x_4^* = 1 \qquad & x_2^* = x_3^* = 0 \end{aligned}$$

It is easy to check the the solution (7) is the unique equimax solution.

The method based on dynamic programming does not seem to lend itself easily to a procedural interpretation. Instead, it offers good results in terms of computational efficiency. Assume for the rest of the section that all the evaluation scores a_i and b_i ($i \in M$) are non-negative integers. In a fashion similar to [13], it easy to show that each $z_m(d)$ can be computed in time $O(mB(m))$, and z^* can be computed in the same order of time. Equivalently, we may define

$$a_{\max} = \max\{a_i, i \in M\} \qquad b_{\max} = \max\{b_i, i \in M\}\ .$$

Since $B(m) \leq m b_{\max}$, the computational time needed to solve (IFD) is of order $O(m^2 b_{\max})$. Consequently, the computational effort needed for a large number m of items, depends on the growth rate of $b_{\max}$, or, equivalently, of $B(m)$. If the growth of $b_{\max}$ is bounded by a polynomial, dynamic programming works efficiently, even for a large m.

As already noted in the examples, the roles of the players in the optimization problem may be reversed. Thus, if $b_{\max}$ grows too fast, one may swap the roles of the two players in (3) in the hope that the growth of $a_{\max}$ is slower. The time needed for completing the whole procedure can be more properly stated as $O(m^2 \min\{a_{\max}, b_{\max}\})$.

What happens if both $a_{\max}$ and $b_{\max}$ grow fast with m? Under an additional assumption little is lost: we can still use a polynomial time dynamic programming procedure that approximates the solution of the original problem to any fixed degree of precision. The additional assumption requires the existence of a $p \in (0,1)$ such that, eventually in m,

$$z^* \geq p \ \min\{a_{\max}, b_{\max}\} \ . \tag{8}$$

A sufficient condition for (8) to hold is that $a_{\max}$ and $b_{\max}$ always refer to different items. Alternatively, we may require that for some $p' \in (0,1)$ and eventually in m

$$\text{either } a_{\max} \leq p' A(m) \text{ or } b_{\max} \leq p' B(m) \ . \tag{9}$$

In fact, suppose that the first inequality holds and $a_{\max}$ and $b_{\max}$ are both associated to the same item. That item is assigned to Player 2, while Player 1 gets the rest, scoring $A(m) - a_{\max} \geq (\frac{1}{p'} - 1) a_{\max}$. Thus

$$z^* \geq \min\left\{\frac{1-p'}{p'}, 1\right\} \min\{a_{\max}, b_{\max}\} \ .$$

The same is true when the second inequality in (9) holds.

For an overview on results in approximation we refer to [2], [11] and [13]. The following result, in particular, is based upon a truncation technique: the least significant digits of each (integer) coefficient a_i and b_i are replaced by zeroes. This result is an adaptation of Theorem 11.5 in [2].

Proposition 3.3. *Suppose* (8) *holds for some* $p \in (0,1)$. *For any* $\varepsilon > 0$, *there exists an algorithm that runs in time* $O\left(\frac{m^3}{\varepsilon\, p}\right)$ *and returns a feasible solution with value* $\bar{z}^*$ *such that*

$$\bar{z}^* \geq (1-\varepsilon) z^*$$

Proof. For a fixed nonnegative integer t we proceed with the truncation of the t least significant digits of each coefficient. With $i \in M$ let $\hat{a}_i = \lfloor a_i / 10^t \rfloor$ and $\bar{a}_i = 10^t \hat{a}_i$. Similarly, define $\hat{b}_i = \lfloor b_i / 10^t \rfloor$ and $\bar{b}_i = 10^t \hat{b}_i$. Next, we obtain $\bar{B}(m)$, $\bar{z}_m(d)$, $\bar{z}^*$ and

$\bar{x}^*$ by replacing a_i and b_i with $\bar{a}_i$ and $\bar{b}_i$ on the original definitions of, respectively, $B(m)$, $z_m(d)$, z^* and x^*.

The following inequalities are trivial:

$$\sum_{i\in M} a_i x_i^* \leq \sum_{i\in M} \bar{a}_i x_i^* + m10^t \tag{10}$$

$$\sum_{i\in M} b_i(1-x_i^*) \leq \sum_{i\in M} \bar{b}_i(1-x_i^*) + m10^t \tag{11}$$

Let $\bar{x}'$ be the solution corresponding to $\bar{z}_m(\sum_{i\in M} \bar{b}_i(1-x_i^*))$. By the optimality of $\bar{x}'$ it follows that

$$\sum_{i\in M} \bar{a}_i x_i^* \leq \sum_{i\in M} \bar{a}_i \bar{x}_i' \tag{12}$$

and by the feasibility of the same solution

$$\sum_{i\in M} \bar{b}_i(1-x_i^*) \leq \sum_{i\in M} \bar{b}_i(1-\bar{x}_i') \tag{13}$$

Also, since $\bar{z}^*$ is the solution of (4) (with the appropriate coefficients), we have

$$\min\left\{\sum_{i\in M} \bar{a}_i \bar{x}_i', \sum_{i\in M} \bar{b}_i(1-\bar{x}_i')\right\} \leq \bar{z}^* \tag{14}$$

The inequalities (10), (11), (12), (13) and (14) merge into

$$z^* = \min\left\{\sum_{i\in M} a_i x_i^*, \sum_{i\in M} b_i(1-x_i^*)\right\} \leq \min\left\{\sum_{i\in M} \bar{a}_i x_i^*, \sum_{i\in M} \bar{b}_i(1-x_i^*)\right\} +$$
$$m10^t \leq \min\left\{\sum_{i\in M} \bar{a}_i \bar{x}_i', \sum_{i\in M} \bar{b}_i(1-\bar{x}_i')\right\} + m10^t \leq \bar{z}^* + m10^t$$

Therefore, $z^* \leq \bar{z}^* + m10^t$.

Since (8) holds,

$$\frac{z^* - \bar{z}^*}{z^*} \leq \frac{m\ 10^t}{p\ \min\{a_{\max}, b_{\max}\}}\ .$$

Two cases may occur:

1. $m/(p\ \min\{a_{\max}, b_{\max}\}) > \varepsilon$. Then,

$$\min\{a_{\max}, b_{\max}\} < \frac{m}{\varepsilon\ p}$$

and we can solve (IFD) with the exact algorithm, in time

$$O\left(m^2 \min\{a_{\max}, b_{\max}\}\right) = O\left(\frac{m^3}{\varepsilon\ p}\right)$$

2. $m/(p\ \min\{a_{\max}, b_{\max}\}) \leq \varepsilon$. We find a nonnegative integer t such that

$$\frac{\varepsilon}{10} < \frac{m\ 10^t}{p\ \min\{a_{\max}, b_{\max}\}} \leq \varepsilon$$

Here we apply the exact algorithm to the instance where a_i and b_i are replaced by $\hat{a}_i$ and $\hat{b}_i$, respectively. This is called the *scaled* instance. Since

$$\min\{\hat{a}_{max}, \hat{b}_{max}\} = 10^{-t}\ \min\{\bar{a}_{max}, \bar{b}_{max}\} \leq 10^{-t}\ \min\{a_{max}, b_{max}\} < \frac{10m}{\varepsilon\ p}\ ,$$

we conclude that

$$O\left(m^2 \min\{\hat{a}_{\max}, \hat{b}_{\max}\}\right) = O\left(\frac{m^3}{\varepsilon\ p}\right)$$

□

4 Relaxing Integer Fair Division: The Adjusted Winner procedure

Suppose now that children are given muffins (with different flavors), instead of hard candies. Each muffin can be given in its entirety to one of the children – or it can be split (not necessarily in equal parts). We are now dealing with the division of m divisible items between two players. We introduce a linear program which is a relaxation of (IFD). No general method – such as the simplex – is needed to solve this linear program. Instead, we will show that a popular procedure, known as *Adjusted Winner*, (**AW**) does the job.

4.1 The Adjusted Winner (AW) algorithm

It is assumed that all items $i \in M$ are completely divisible and homogeneous. Thus player 1 can receive a part $x_i \in [0, 1]$ of item i, while player 2 gets the rest. The two players will benefit, respectively, by $x_i a_i$ and $(1 - x_i)b_i$ from the splitting. The overall satisfaction of each player is still given by (1).

We look for an allocation $x \in [0, 1]^m$ that achieves

$$z^+ = \max_{x \in [0,1]^m} \min\{v_1(x), v_2(x)\} \tag{DFD}$$

Problem (DFD) can be written as a linear program

$$\begin{array}{lll} \max & z & \\ \text{s.t.} & \sum_{i \in M} a_i x_i \geq z & \\ & \sum_{i \in M} b_i (1 - x_i) \geq z & \\ & 0 \leq x_i \leq 1 & i = 1, \ldots, m \end{array} \tag{15}$$

An allocation $x \in [0,1]^m$ is *equitable* if

$$v_1(x) = v_2(x) \ ,$$

and is *(strong) Pareto optimal* (or efficient) if there is no other allocation that weakly dominates x, i.e., there is no other allocation $\tilde{x}$ such that $v_i(\tilde{x}) \geq v_i(x)$, $i = 1, 2$, with a strict inequality for at least one of the players. Equivalently, x is Pareto optimal if, whenever $\tilde{x}$ is any other allocation for which

$$v_i(\tilde{x}) \geq v_i(x) \qquad i = 1, 2 \ ,$$

then

$$v_i(\tilde{x}) = v_i(x) \qquad i = 1, 2 \ .$$

The Adjusted Winner (AW) algorithm was introduced by Brams and Taylor in [5] (with many applications analyzed in [6]). Their aim was to provide a step-by-step procedure returning a partition that is equitable, Pareto optimal and envy-free (in the sense that none of the player feels that the other player has received more than him/herself). Here we show that the very same solution solves (DFD). A brief sketch of the algorithm follows – for a more detailed account we refer to [5] and [6]. There are two phases:

the "winning" phase. each player temporarily receives the items that he/she values more than the other player. The total score of each player is computed

the "adjusting" phase. Items are transferred, one at a time from the "richer" player to the "poorer" one, starting with the items with ratio a_i/b_i closer to 1. To reach equitability one item may be split into two parts, the fraction assigned to each player being determined by an equation (see p.70 in [5] or p.74 in [6])

In order to show how the AW algorithm solves (DFD) we restate it. The aim is to enhance its mathematical structure. The price we pay with this translation is the loss of the procedural appeal of AW. We keep in mind, however, that the solution for (DFD) can always be implemented as a step-by-step procedure that does not require external referees, nor obscure computer programs.

an alternative version of the Adjusted Winner

Labelling Re-label the items according to the decreasing order of the preferences' ratio. The m items are numbered 1 to m so that

$$\frac{a_1}{b_1} \geq \frac{a_2}{b_2} \geq \ldots \geq \frac{a_m}{b_m} \tag{16}$$

Splitting Look for the index $r \in M$ such that

$$\sum_{i=1}^{r-1} a_i \leq \sum_{i=r}^{m} b_i$$

and

$$\sum_{i=1}^{r} a_i > \sum_{i=r+1}^{m} b_i$$

with the assumptions

$$\sum_{i=1}^{0} a_i = \sum_{i=m+1}^{m} b_i = 0\,. \tag{17}$$

Pick the solution $x^* = (x_1^*, \ldots, x_m^*)$

$$\begin{aligned} x_1^* = \cdots = x_{r-1}^* &= 1 \\ x_{r+1}^* = \cdots = x_m^* &= 0 \\ x_r^* &= \frac{\sum_{i=r}^{m} b_i - \sum_{i=1}^{r-1} a_i}{a_r + b_r} \end{aligned} \tag{18}$$

The value of the procedure is

$$z^+ = \sum_{i=1}^{r-1} a_i + x_r a_r$$

Proposition 4.1. *The AW algorithm solves* (DFD) *solves (15). Therefore, the AW solution is also minimax.*

We will show that the allocation (18) solves (DFD).

Some preliminary results are required. First of all consider the *allocation range*.

$$\mathcal{D} = \{(v_1(x), v_2(x)) : x \in [0,1]^m\}$$

Lemma 4.2. *$\mathcal{D}$ is a convex and compact set in $\mathbb{R}^2$.*

Proof. Pick $x, y \in [0,1]^m$ and $\gamma \in [0,1]$. Then, for $i = 1, 2$

$$v_i(\gamma x + (1-\gamma)y) = \gamma v_i(x) + (1-\gamma) v_i(y)$$

and $\mathcal{D}$ is convex. Compactness is a consequence of the compactness of $[0,1]^m$ and the continuity of the v_i's. More in detail, $\mathcal{D} \subset [0, A(m)] \times [0, B(m)]$, so $\mathcal{D}$ is bounded. Consider now a sequence $\{x_n\}$ in $[0,1]^m$ for which $(v_1(x_n), v_2(x_n))$ converges. Since $[0,1]^m$ is compact, there exists a subsequence $\{x_{n'}\}$ converging to some $x^* \in [0,1]^m$. Since v_1 and v_2 are continuous, we have

$$(v_1(x_{n'}), v_2(x_{n'})) \to (v_1(x^*), v_2(x^*)) \in \mathcal{D}$$

and $\mathcal{D}$ is closed. □

Next we characterize the maximin solutions.

Lemma 4.3. *A minimax solution always exists. An allocation is maximin if and only if it is Pareto optimal and equitable.*

Proof. The proof relies partly on graphical arguments. We draw the set $\mathcal{D}$ of all the allocations' values. An allocation x is Pareto if there is no other point of $\mathcal{D}$ in the upper quadrant pointed on $(v_1(x), v_2(x))$ (with the exception of x itself). The allocation is equitable if $(v_1(x), v_2(x))$ lies on the bisector of the positive quadrant.

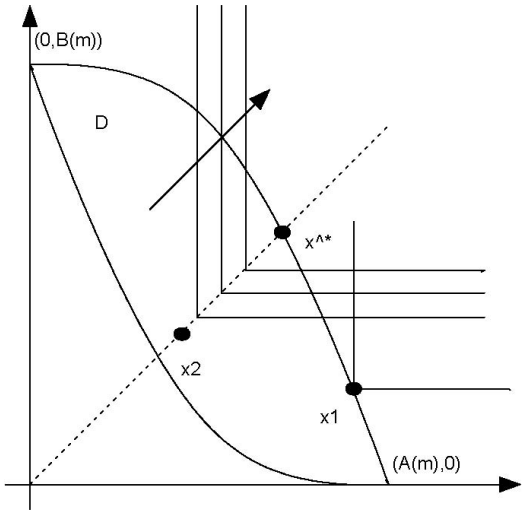

Figure 1. *Maximin (x^*), Pareto (x_1), Equitable (x_2) allocations*

Let $\mathcal{Q}$ be the family of upper quadrants pointed on the equitable allocations. A maximin solution is obtained by considering the supremum of the quadrants in $\mathcal{Q}$ that intersects $\mathcal{D}$ (See Figure 1). Since $\mathcal{D}$ is compact, the supremum is attained, and a maximin solution x^* exists. This solution is also equitable. Argue by contradiction and suppose, without loss of generalilty, that $v_1(x^*) < v_2(x^*)$. Since $\mathcal{D}$ is convex, it contains the segment with endpoints $(v_1(x^*), v_2(x^*))$ and $(A(m), 0)$. This segment intersects the bisector of the positive quadrant at some point (t, t) with $t > v_1(x^*)$ (see Figure 2). Therefore x^* cannot be maximin.

Also, a maximin, equitable solution must be Pareto as well. In fact, the upper quadrant pointed on this solution is a member of $\mathcal{Q}$ (see Figure 3). No other point of $\mathcal{D}$ lies on the border (for the above argument on equitability) nor on the interior of the quadrant (for the allocation is maximin). To prove the converse implication, suppose x is Pareto and equitable. No other point of $\mathcal{D}$ lies on the upper quadrant pointed on $(v_1(x), v_2(x))$ (see Figure 3). Since this quadrant is in $\mathcal{Q}$, x is maximin.

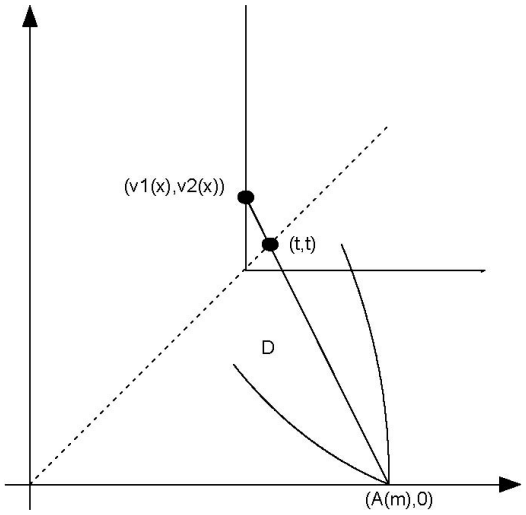

Figure 2. *A maximin allocation is equitable*

□

Following Akin [1], we give an operational description of the Pareto optimal allocations. For any $w \in [0,1]$ and $i = 1, \ldots, m$ define

$$q_i(w) = \max\{w\ a_i, (1-w)\ b_i\}$$

Lemma 4.4. *For any allocation* $x = (x_1, \ldots, x_m) \in [0,1]^m$

$$w\ a_i\ x_i + (1-w)\ b_i\ (1-x_i) \leq q_i(w) \quad \textit{for all } i = 1, \ldots, m \tag{19}$$

and

$$w\ v_1(x) + (1-w)\ v_2(x) \leq \sum_{i \in M} q_i(w). \tag{20}$$

Moreover

$$w\ v_1(x) + (1-w)\ v_2(x) = \sum_{i \in M} q_i(w) \tag{21}$$

if and only if

$$\begin{array}{llll} x_i = 1 & \textit{whenever} & w\ a_i > (1-w)\ b_i \\ x_i = 0 & \textit{whenever} & w\ a_i < (1-w)\ b_i \end{array} \tag{22}$$

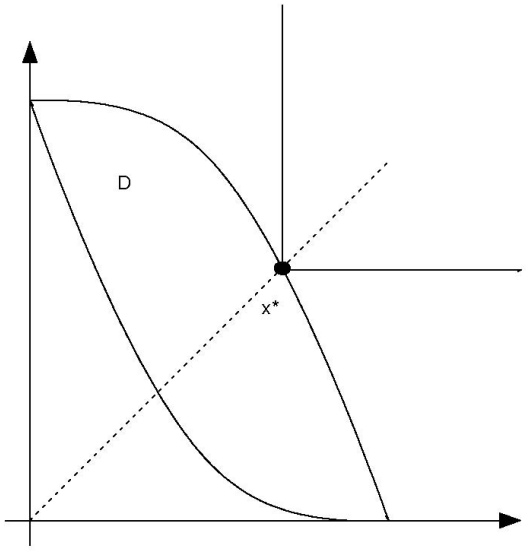

Figure 3. *A maximin allocation is equitable*

Proof. Since $x_i, 1 - x_i \geq 0$, then $w\ a_i\ x_i \leq q_i(w)\ x_i$ and $(1-w)\ b_i\ (1-x_i) \leq q_i(w)\ (1-x_i)$ for all $i \in M$. Add the two inequalities to obtain (19). Sum over all i's to obtain (20). If (22) holds, then (19) holds with a strict equality sign for each i and, therefore (21) holds true. Conversely, suppose that (21) holds and, without loss of generality, that for some j, $w\ a_j > (1-w)\ b_j$ but $x_j < 1$. Then $w\ a_j\ x_j + (1-w)\ b_j(1-x_j) < q_j(w)$ and $w\ v_1(x) + (1-w)\ v_2(x) < \sum_{i \in M} q_i(w)$. A contradiction. □

Lemma 4.5. *If (21) holds for some $w \in (0,1)$, then x is Pareto optimal.*

Proof. Consider another allocation $\hat{x}$ that dominates x. Since $w, 1-w > 0$ this implies

$$w\ v_1(\hat{x}) \geq w\ v_1(x) \qquad \text{and} \qquad (1-w)\ v_2(\hat{x}) \geq (1-w)\ v_2(x) \tag{23}$$

If we apply (20) to $\hat{x}$ and (21) to x, we obtain

$$w\ v_1(\hat{x}) + (1-w)\ v_2(\hat{x}) \leq \sum_{i \in M} q_i(w) = w\ v_1(x) + (1-w)\ v_2(x)$$

A comparison with (23) yields

$$v_1(\hat{x}) = v_1(x) \qquad \text{and} \qquad v_2(\hat{x}) = v_2(x)$$

Thus x is Pareto optimal. □

Lemma 4.6. *The AW procedure yields an equitable allocation*

Proof.

$$v_1(x^*) = \sum_{i=1}^{r-1} a_i + a_r \left(\frac{\sum_{i=r}^{m} b_i - \sum_{i=1}^{r-1} a_i}{a_r + b_r} \right) = \frac{b_r \sum_{i=1}^{r-1} a_i + a_r \sum_{i=r}^{m} b_i}{a_r + b_r}$$

and

$$v_2(x^*) = \sum_{i=r+1}^{m} b_i + b_r \left(1 - \frac{\sum_{i=r}^{m} b_i - \sum_{i=1}^{r-1} a_i}{a_r + b_r} \right) =$$

$$\sum_{i=r+1}^{m} b_i + b_r \left(\frac{\sum_{i=1}^{r} a_i - \sum_{i=r+1}^{m} b_i}{a_r + b_r} \right) =$$

$$\frac{a_r \sum_{i=r+1}^{m} b_i + b_r \sum_{i=1}^{r} a_i}{a_r + b_r} = \frac{a_r \sum_{i=r}^{m} b_i + b_r \sum_{i=1}^{r-1} a_i}{a_r + b_r} = v_1(x^*)$$

□

Proof of Proposition 4.1. By Lemma 4.6, the AW procedure yields an equitable procedure. By virtue of Lemma 4.3 it only remains to show that x is also Pareto optimal. The AW procedure finds a $\lambda > 0$ such that

$$\begin{aligned} x_i &= 1 \qquad \text{whenever} \qquad \frac{a_i}{b_i} > \lambda \\ x_i &= 0 \qquad \text{whenever} \qquad \frac{a_i}{b_i} < \lambda \end{aligned} \tag{24}$$

Since there exists a unique $w \in (0,1)$ such that

$$\lambda = \frac{1-w}{w}$$

then (24) is equivalent to (22), and, by Lemma 4.5, x is Pareto optimal. □

4.2 Computational efficiency of AW

The first step of the alternative version of AW a sorting of the m items is required. Since this is the most time-consuming operation in the algorithm and since sorting m items can be done in time $O(m \log m)$, the whole algorithm requires the same order of time. It is easy to verify, however, that the essence of the AW procedure is to define a $\lambda^* > 0$ such that if

$$\begin{aligned} &\text{if } \tfrac{a_i}{b_i} > \lambda^* \quad \text{then } x_i = 1 \\ &\text{if } \tfrac{a_i}{b_i} < \lambda^* \quad \text{then } x_i = 0 \\ &\text{if } \tfrac{a_i}{b_i} = \lambda^* \quad \text{then } x_i \text{ is split} \end{aligned} \tag{25}$$

and the splitting occurs so that the resulting partitions are equitable. Therefore, if λ^* is known, the linear programming relaxation can be solved in linear time. We now give an algorithm that finds this λ^* and solves the linear programming relaxation in $O(m)$ time.

An efficient version of the Adjusted Winner Let M^1 and M^0 denote the variables fixed to 1 and 0 respectively, and let M^f be the free variables. Given a candidate value λ, let $M^> = \{j \in M^f : a_j/b_j > \lambda\}$, $M^= = \{j \in M^f : a_j/b_j = \lambda\}$, and $M^< = \{j \in M^f : a_j/b_j < \lambda\}$. Also let

$$S_1(\lambda) = \sum_{j \in M^1 \cup M^>} a_j \qquad T_1(\lambda) = \sum_{j \in M \setminus (M^1 \cup M^>)} b_j$$
$$S_2(\lambda) = \sum_{j \in M^1 \cup M^> \cup M^=} a_j \qquad T_2(\lambda) = \sum_{j \in M \setminus (M^1 \cup M^> \cup M^=)} b_j \, .$$

Inizialization: $M^1 = M^0 = \emptyset$; $M^f = M$.

Step 1: Let λ be the median of $\{a_j/b_j : j \in M^f\}$.

Step 2: Construct the sets $M^>$, $M^=$, $M^<$, and calculate $S_1(\lambda)$, $T_1(\lambda)$, $S_2(\lambda)$, $T_2(\lambda)$.

- **i.** $S_1(\lambda) > T_1(\lambda)$ implies that λ is too small. Set $M^0 := M^0 \cup M^= \cup M^<$ and $M^f := M^>$. Return to Step 1.
- **ii.** $S_2(\lambda) < T_2(\lambda)$ implies that λ is too large. Set $M^1 := M^1 \cup M^> \cup M^=$ and $M^f := M^<$. Return to Step 1.
- **iii.** If $S_1(\lambda) = T_1(\lambda)$ or $S_2(\lambda) = T_2(\lambda)$, then one immediately obtains an optimal integer solution (for example, if $S_1(\lambda) = T_1(\lambda)$ then an optimal solution is obtained by setting $M^1 := M^1 \cup M^>$ and $M^0 := M^0 \cup M^= \cup M^<$). Otherwise, if $S_1(\lambda) < T_1(\lambda)$ and $S_2(\lambda) > T_2(\lambda)$ take the elements of $M^=$ in arbitrary order. If $M^= = \{j(1), ..., j(p)\}$, find q such that:

$$S_1(\lambda) + \sum_{i=1}^{q-1} a_{j(i)} \leq T_2(\lambda) + \sum_{i=q}^{p} b_{j(i)}$$

and

$$S_1(\lambda) + \sum_{i=1}^{q} a_{j(i)} > T_2(\lambda) + \sum_{i=q+1}^{p} b_{j(i)} \, .$$

Set $M^1 := M^1 \cup \{j(1), ..., j(q-1)\}$, $r = j(q)$, and $M^0 := M^0 \cup \{j(q+1), ..., j(p)\}$.

The algorithm terminates with an optimal solution to AW with $x_j = 1$ for $j \in M^1$, $x_j = 0$ for $j \in M^0$, and $x_r = (\sum_{j \in M^0 \cup \{r\}} b_j - \sum_{j \in M^1} a_j)/(a_r + b_r)$. To verify that the algorithm has $O(m)$ running time, one can use the corresponding argument introduced in [13], based on the fact that the median of k numbers can be found in $O(k)$ time.

4.3 The maximin problem with initial endowments

The AW procedure is flexible enough to cover the situation where the two players own initial endowments. This variation is interesting in its own rights. An optimal allocation is sought when the utility of each player is the sum of the initial endowment and the values of the items (or fractions thereof) received. Our interest in this problem, however, is mainly instrumental. In order to implement a branch-and-bound method for the indivisible items' case we need to solve several instances of the linear relaxed problem in which certain items are forcedly assigned to the players. These items represent their initial wealth. Let $\alpha > 0$ (resp. $\beta > 0$) the initial endowment of player 1 (pl.2, resp.) The utility is now given by

$$w_1(x) = \alpha + \sum_{i \in M} x_i a_i = \alpha + v_1(x)$$
$$w_2(x) = \beta + \sum_{i \in M} (1 - x_i) b_i = \beta + v_2(x)$$

The LP problem of interest is now:

$$\begin{array}{lll} \max & z & \\ \text{s.t.} & \alpha + \sum_{i \in M} a_i x_i \geq z & \\ & \beta + \sum_{i \in M} b_i (1 - x_i) \geq z & \\ & 0 \leq x_i \leq 1 & i = 1, \ldots, m \end{array} \qquad \text{(DFD-ie)}$$

Once again the maximin solution coincides with the Pareto and equitable solution, but only when the value of the assignable items according to the poorer player is larger or equal to the difference between the initial endowments. We propose the following:

The Adjusted Winner procedure with initial endowments (AW-ie)

1. Label the items according to the preferences ratios as in (16)
2. If
$$\alpha + \sum_{i \in M} a_i \leq \beta$$
then $x_1^* = x_2^* = \ldots = x_m^* = 1$ and $z^+ = \alpha + \sum_{i \in M} a_i$

3. If
$$\beta + \sum_{i \in M} b_i \leq \alpha$$
then $x_1^* = x_2^* = \ldots = x_m^* = 0$ and $z^+ = \beta + \sum_{i \in M} b_i$

4. Otherwise

   - look for the index $r \in M$ such that
   $$\alpha + \sum_{i=1}^{r-1} a_i \leq \beta + \sum_{i=r}^{m} b_i$$
   and
   $$\alpha + \sum_{i=1}^{r} a_i > \beta + \sum_{i=r+1}^{m} b_i$$
   with the usual assumptions (17) when $r = 1$ or $r = m$.
   - The solution in this case will be
   $$\begin{aligned} x_1^* = \cdots = x_{r-1}^* &= 1 \\ x_{r+1}^* = \cdots = x_m^* &= 0 \\ x_r^* &= \frac{\beta - \alpha + \sum_{i=r}^{m} b_i - \sum_{i=1}^{r-1} a_i}{a_r + b_r} \end{aligned} \tag{26}$$
   and
   $$z^+ = \alpha + \sum_{i=1}^{r-1} a_i + x_r a_r$$

Proposition 4.7. *The allocation (26) solves (DFD-ie).*

In this case the utility of the two player is given, respectively, by $w_1(x) = \alpha + v_1(x)$ and $w_2(x) = \beta + v_2(x)$ and the allocation range $\tilde{\mathcal{D}}$ is simply a translation of the allocation range $\mathcal{D}$ by (α, β), and is thus convex and compact by Lemma 4.2.

Proof. Case 1. Since $\alpha + \sum_{i \in M} a_1 x_i \leq \beta$, then $x \leq y$ for any $(x, y) \in \tilde{\mathcal{D}}$. Therefore, $(\alpha + \sum_{i \in M} a_1 x_i, \beta)$ obtained when $x \equiv 1$, maximizes the minimax criterion. A similar reasoning applies for case 2.

For case 3 we only need to show that the AW-ie algorithm yields an equitable, Pareto-optimal allocation. The initial endowment does not change the characterization of Pareto-optimality . As for the AW procedure, the AW-ie algorithm fixes a $\lambda > 0$ such that (24) holds. Therefore, by Lemma 4.5 the allocation returned by AW-ie is Pareto-optimal.

Equitability of the allocation is obtained in a manner similar to Lemma 4.6. In fact,

$$w_1(x^*) = \alpha + \sum_{i=1}^{r-1} a_i + a_r \left(\frac{\beta - \alpha + \sum_{i=r}^{m} b_i - \sum_{i=1}^{r-1} a_i}{a_r + b_r} \right) = \frac{\alpha b_r + b_r \sum_{i=1}^{r-1} a_i + \beta a_r + a_r \sum_{i=r}^{m} b_i}{a_r + b_r}$$

and

$$\begin{aligned} w_2(x^*) &= \beta + \sum_{i=r+1}^{m} b_i + b_r \left(1 - \frac{\beta - \alpha + \sum_{i=r}^{m} b_i - \sum_{i=1}^{r-1} a_i}{a_r + b_r} \right) = \\ &\beta + \sum_{i=r+1}^{m} b_i + b_r \left(\frac{\alpha - \beta + \sum_{i=1}^{r} a_i - \sum_{i=r+1}^{m} b_i}{a_r + b_r} \right) = \\ &\frac{\alpha b_r + \beta a_r + a_r \sum_{i=r+1}^{m} b_i + b_r \sum_{i=1}^{r} a_i}{a_r + b_r} = \\ &\frac{\alpha b_r + \beta a_r + a_r \sum_{i=r}^{m} b_i + b_r \sum_{i=1}^{r-1} a_i}{a_r + b_r} = w_1(x^*) \end{aligned}$$

□

5 A branch and bound algorithm

When solving the maximin allocation problem (IFD) there is a finite number of possible candidates to choose from. In principle the solution can be obtained in finite time by computing the value of each allocation for the two players. This process can be considerably speeded up if we consider a branch-and-bound technique that splits the original problem into smaller subproblems and uses upper bounds to avoid exploring certain parts of the set of feasible integer solutions. This approach may be not as fast as the one based on dynamic programming, but it makes repeated use of the Adjusted Winner procedure with initial endowment and keeps the procedural character of the latter.

In what follows, we will consider a series of constrained subproblems in which some of the items have already been assigned to the players. Let $A, B \subset M$, with $A \cap B = \emptyset$. Let $S(A, B)$ be the constrained problem in which the items in A (B, resp.) are assigned to player 1 (pl.2, resp.), i.e., $x_i = 1$ for each $i \in A$ ($x_i = 0$ for each $i \in B$). $S(\emptyset, \emptyset)$ denotes the original (unconstrained) problem.

For a given couple of disjoint index sets, A, B in M, let $\bar{x}(A, B)$ denote a feasible allocation for the constrained problem and let $\bar{z}(A, B)$ denote the corresponding

value. Moreover, let $x^*(A,B)$ and $z^*(A,B)$ denote the solution and the value of $S(A,B)$. Finally let $x^+(A,B)$ and $z^+(A,B)$ be, respectively, the solution and value for the linear relaxation of $S(A,B)$, i.e. for the case where splitting of the contended items is allowed. Clearly, the following holds for each couple of A and B:

$$\bar{z}(A,B) \leq z^*(A,B) \leq z^+(A,B) \tag{27}$$

The results in Section 4 can be used to compute $x^+(A,B)$ and $z^+(A,B)$. In particular we set $\alpha = \sum_{i\in A} v_1(x_i)$ and $\beta = \sum_{i\in B} v_2(x_i)$, and divide the remaining $m' = |M'|$ items according to the AW-ie procedure. Since $x^+(A,B)$ contains at most one fractional component, $\bar{x}(A,B)$ may be obtained by approximating the fractional coordinate to the nearest integer, 0 or 1.

5.1 A variable elimination test

The branch-and-bound procedure defines a series of subproblems in which an increasing numbers are forcedly assigned to one player or the other. Since the procedure becomes simpler as the number of pre-assigned items increases,and following [13], p.452, we consider a variable elimination test that, for any given subproblem, checks whether additional items can be assigned priori to any further analysis.

Let $A, B \subset M$ be a couple of disjoint sets of items and take $i \in M' = M \setminus (A \cup B)$.

Proposition 5.1. (*a*) *If*

$$z^+(A \cup \{i\}, B) < \bar{z}(A,B) \tag{28}$$

then $x^*(A, B \cup \{i\})$ *solves* $S(A,B)$*, while* $x^*(A \cup \{i\}, B)$ *does not.*

(*b*) *If*

$$z^+(A, B \cup \{i\}) < \bar{z}(A,B) \tag{29}$$

then $x^*(A \cup \{i\}, B)$ *solves* $S(A,B)$*, while* $x^*(A, B \cup \{i\})$ *does not.*

Proof. By assumption and (27) we have

$$z^*(A \cup \{i\}, B) \leq z^+(A \cup \{i\}, B) < \bar{z}(A,B) \leq z^*(A,B)$$

So $x^*(A \cup \{i\}, B)$ cannot be a solution for $S(A,B)$. If this is the case, then $x^*(A, B \cup \{i\})$ must be a solution for the same problem. Part (b) is established symmetrically. □

The result simply states that whenever condition (28) ((29), resp.) occurs, then $S(A,B)$ can be replaced by $S(A, B \cup \{i\})$ ($S(A \cup \{i\}, B)$, resp.). When the two sides of (28), or (29), attain equality, there is a partial extension of the previous result:

Proposition 5.2. (*a*) *If* $z^+(A \cup \{i\}, B) \leq \bar{z}(A,B)$*, then either* $x^*(A, B \cup \{i\})$ *or* $\bar{x}(A,B)$ *solve* $S(A,B)$*.*

(*b*) *If* $z^+(A, B\cup\{i\}) \le \bar{z}(A, B)$*, then either* $x^*(A\cup\{i\}, B)$ *or* $\bar{x}(A, B)$ *solve* $S(A, B)$*.*

(*c*) *If* $z^+(A \cup \{i\}, B) \le \bar{z}(A, B)$ *and* $z^+(A, B \cup \{i\}) \le \bar{z}(A, B)$*, then* $\bar{x}(A, B)$ *solves* $S(A, B)$*.*

Proof. (a) By assumption

$$z^+(A \cup \{i\}, B) \le \bar{z}(A, B) \le z^*(A, B)$$

Assume now that $x^*(A, B \cup \{i\})$ does not solve $S(A, B)$. Then $x^*(A \cup \{i\}, B)$ will work instead, and thus

$$z^*(A, B) \le z^*(A \cup \{i\}, B) \le z^+(A \cup \{i\}, B)$$

Comparing the two inequalities, we conclude that $\bar{z}(A, B) = z^*(A, B)$ and $\bar{x}(A, B)$ solves $S(A, B)$. Part (b) is proved with a symmetrical argument.

(c) By definition

$$\bar{z}(A, B) \le z^*(A, B) \le z^+(A, B) \le \max\{z^+(A \cup \{i\}, B), z^+(A, B \cup \{i\})\}$$

while the hypotheses reads

$$\max\{z^+(A \cup \{i\}, B), z^+(A, B \cup \{i\})\} \le \bar{z}(A, B)$$

Thus $\bar{x}(A, B)$ solves $S(A, B)$. □

The use of Proposition 5.2 is more subtle: when situation (a) occurs than we replace $S(A, B)$ with $S(A, B\cup\{i\})$ and continue with the sub-partitioning to obtain a solution $\tilde{x}$. This solution is then compared with $\bar{x}(A, B)$. The one with the higher value is the solution for (IFD).

At first sight, Proposition 5.2 is more powerful than Proposition 5.1 since it binds more items to the players, thus making the problem simpler. Using this result, however, may result in the loss of some solutions. Part (a) of the statement does not prevent $x^*(A \cup \{i\})$ from being a possible solution for $S(A, B)$ (and a symmetrical conclusion holds for part (b). So, if the goal is to capture all the solutions for (IFD), Proposition 5.1 is the one to choose.

The problem remaining after the elimination test has been carried out is called the *reduced problem*. Note that λ^* for the reduced problem is the same as that for the original problem.

5.2 The algorithm

All the elements are set to formulate a branch-and-bound algorithm for the maximin problem with indivisible items (IFD). The algorithm follows the general scheme for

branch-and-bound, where the original problem $S(\emptyset, \emptyset)$ is recursively split into a series of constrained problems with some of the items assigned in advance to one player or the other. As usual for this kind of algorithms, it is convenient to represent the splitting process with a tree graph. When a subproblem cannot yield any more candidates for the solution of the original problem, the branch corresponding to that subproblem is cut (or pruned) and no other branch generates from that node of the tree.

The general framework is adapted to the peculiar features of the problem in question. For instance, the linear relaxation of each subproblem has a twofold purpose: on one hand it gives an upper bound for the value of the integer solution, but when the solution for the linear relaxation is not integer, it also suggests how to operate the splitting, by assigning the item corresponding to the unique fractional component to one player or the other.

In building the tree, several integer solutions are met and the best of them (in terms of objective function) are recorded. Here we are interested in finding all the solutions to (IFD). Therefore $\bar{X}$ will denote the set of best solutions met so far, while $\bar{z}$ is their common value.

Each subproblem $S(A, B)$ may have three different labels attached to it: "new", "open" or "close": a subproblem is new when its linear relaxation has not been computed yet; once the computation occurs, the problem is open or close depending on whether the solution for the relaxation is integer or not. Furthermore, a subproblem may also be closed when its upper bound is smaller than the best current admissible solution. Open problems are split according to the above mentioned rule. The algorithm ends when all the subproblem are closed.

The algorithm runs as follows:

Initialization. Set $\bar{X} = \emptyset$ and $\bar{z} = -\infty$. Label $S(\emptyset, \emptyset)$ as new.

The generic cycle is made of the following steps

Compute bounds. For any new subproblem $S(A, B)$ perform the variable elimination test derived from Proposition 5.1 and denote with $S(A', B')$ the resulting subproblem with (possibly) more items preassigned to the players.

- Compute $x^+(A', B')$ and $z^+(A', B')$ using the AW-ie algorithm.
- Examine $x^+(A', B')$.
  - If $x^+(A', B')$ is integer then set $\bar{x}(A', B') = x^+(A', B')$ and $\bar{z}(A', B') = z^+(A', B')$. Label $S(A', B')$ as close.
  - If $x^+(A', B')$ has a fractional component then set $\bar{x}(A', B') = \mathrm{rnd}(x^+(A', B'))$ with corresponding value $\bar{z}(A', B')$. Label $S(A', B')$ as close.

- Update the optimal set
  - If $\bar{z}(A', B') > \bar{z}$ then set $\bar{z} = \bar{z}(A', B')$ and $\bar{X} = \{\bar{x}(A', B')\}$.
  - If $\bar{z}(A', B') = \bar{z}$ and $\bar{x}(A', B') \notin \bar{X}$ then add this solution to $\bar{X}$.

List and close List the open subproblems. Close all the $S(A, B)$ such that

$$z^+(A, B) < \bar{z} \,. \tag{30}$$

If there is no open subproblem left than exit the algorithm and return $\bar{X}$ as the optimal solution set with value $\bar{z}$.

Choose and split Choose the open problem $S(A, B)$ with higher upper bound $z^+(A, B)$. The relaxed solution $x^+(A, B)$ has one fractional component $i \in M \setminus (A \cup B)$. Replace $S(A, B)$ (labelled close) with two subproblems $S(A \cup \{i\}, B)$ and $S(A, B \cup \{i\})$, labelling them as new. Continue with the next cycle.

Some of the rules in the algorithm may be changed. For instance another criterion may be selected to pick an open problem. A naive motivation for the chosen rule is that the higher the bound, the more likely is the subproblem to deliver an optimal solution.

As noted previously, we may use a variable elimination test based on Proposition 5.2. The algorithm will be quicker, but some solutions may be left off of the solution set $\bar{X}$.

We go back to the examples in Section 2. The same data can be used as the input for the branch-and-bound algorithm. The graph trees for these instances are shown in Fig. 4 and 5.

If we try to replace Proposition 5.1 with the stronger Proposition 5.2 in the elimination test for the data in Example 3.2, the graph tree in Fig. 6 shows that only 3 subproblems are examined (in place of 7), but the algorithm fails to capture 2 out of the 4 solutions of the example.

A Appendix: The examples in detail

A.1 Example 3.1

The branch-and-bound algorithm begins with

Initialization. Set $\bar{X} = \emptyset$ and $\bar{z} = -\infty$. Label $S(\emptyset, \emptyset)$ as new.

Cycle 1: Compute bounds. The elimination test on the new subproblem $S(\emptyset, \emptyset)$ yields a reduced subproblem $S(\emptyset, \{7, 8\})$ with upper and lower bounds:

$$x^+(\emptyset, \{7,8\}) = (1, 1, 1, 0.6429, 0, 0, 0, 0) \qquad z^+(\emptyset, \{7,8\}) = 105.714$$
$$\bar{x}(\emptyset, \{7,8\}) = (1, 1, 1, 1, 0, 0, 0, 0) \qquad \bar{z}(\emptyset, \{7,8\}) = 95$$

Therefore, the solution set is updated: $\bar{X} = \{(1, 1, 1, 1, 0, 0, 0, 0)\}$ and $\bar{z} = 95$.

List and close. The list of open problems is $\mathcal{S} = \{S(\emptyset, \{7, 8\}\}$. Since the only subproblem fails test (30), the algorithm continues.

Choose and split. The unique problem is split into the following new subproblems: $S(\{4\}, \{7, 8\})$ and $S(\emptyset, \{4, 7, 8\})$.

Cycle 2: compute bounds. After the elimination tests for the two new subproblem, we obtain the following subproblems:

- Subproblem $S(\{4\}, \{7, 8\})$ with bounds:

  $$x^+(\{4\}, \{7,8\}) = (1, 1, 0.7059, 1, 0, 0, 0, 0) \qquad z^+(\{4\}, \{7,8\}) = 105.294$$
  $$\bar{x}(\{4\}, \{7,8\}) = (1, 1, 1, 1, 0, 0, 0, 0) \qquad \bar{z}(\{4\}, \{7,8\}) = 95$$

  so no update occurs.

- Subproblem $S(\{3\}, \{4, 7, 8\})$ with bounds:

  $$x^+(\{3\}, \{4,7,8\}) = (1, 1, 1, 0, 1, 0.2381, 0, 0) \qquad z^+(\{3\}, \{4,7,8\}) = 104.762$$
  $$\bar{x}(\{3\}, \{4,7,8\}) = (1, 1, 1, 0, 1, 0, 0, 0) \qquad \bar{z}(\{3\}, \{4,7,8\}) = 100\ .$$

  Therefore $\bar{X} = \{(1, 1, 1, 0, 1, 0, 0, 0)\}$ and $\bar{z} = 100$

List and close. The open problems are $\mathcal{S} = \{S(\{4\}, \{7, 8\}), S(\{3\}, \{4, 7, 8\})\}$ and none of them is closed. The algorithm continues.

Choose and split. Subproblem $S(\{4\}, \{7, 8\})$ is picked and split into the new subproblems $S(\{3, 4\}, \{7, 8\})$ and $S(\{4\}, \{3, 7, 8\})$.

Cycle 3: compute bounds. The two new subproblems enter the elimination test.

- Subproblem $S(\{3, 4\}, \{7, 8\})$ becomes $S(\{3, 4\}, \{2, 5, 6, 7, 8\})$ and

  $$x^+(\{3,4\}, \{2,5,6,7,8\}) = (1, 0, 1, 1, 0, 0, 0, 0) \qquad z^+(\{3,4\}, \{2,5,6,7,8\}) = 102$$

  so $\bar{X} = \{(1, 0, 1, 1, 0, 0, 0, 0)\}$ and $\bar{z} = 102$ and the subproblem is closed.

- Subproblem $S(\{4\},\{3,7,8\})$ becomes $S(\{2,4,5,6\},\{3,7,8\})$ and

  $x^+(\{2,4,5,6\},\{3,7,8\}) = (0,1,0,1,1,1,0,0) \qquad z^+(\{2,4,5,6\},\{3,7,8\}) = 98$

  No update occurs and the subproblem is closed.

List and close. The list of open problems is now $\mathcal{S} = \{S(\{3\},\{4,7,8\})\}$ and the algorithm continues.

Choose and split. Subproblem $S(\{3\},\{4,7,8\})\}$ is picked and split into the new subproblems $S(\{3,6\},\{4,7,8\})$ and $S(\{3\},\{4,6,7,8\})$.

Cycle 4: compute bounds. The elimination tests is performed and the two new subproblem with the following results

- Subproblem $S(\{1,2,3,6\},\{4,5,7,8\})$ with bounds:

  $x^+(\{1,2,3,6\},\{4,5,7,8\}) = (1,1,1,0,0,1,0,0) \qquad z^+(\{1,2,3,6\},\{4,5,7,8\}) = 100$

  Since $z^+(\{1,2,3,6\},\{4,5,7,8\}) < \bar{z}$, no update occurs and the problem is closed.

- Subproblem $S(\{1,2,3,5\},\{4,6,7,8\})$ with bounds:

  $x^+(\{1,2,3,5\},\{4,6,7,8\}) = (1,1,1,0,1,0,0,0) \qquad z^+(\{1,2,3,5\},\{4,6,7,8\}) = 100$

  No update occurs and the problem is closed

List and close. The list of open problems $\mathcal{S}$ is empty and the algorithm returns the unique optimal solution $(1,0,1,1,0,0,0,0)$ with value 102.

A.2 Example 3.2

Here we show that the branch-and-bound algorithm captures all the solutions to this instance.

Initialization. Set $\bar{X} = \emptyset$ and $\bar{z} = -\infty$. Label $S(\emptyset,\emptyset)$ as new.

Cycle 1: compute bounds. The elimination test does not add any constraint, so the bounds for $S(\emptyset,\emptyset)$ are:

$$x^+(\emptyset,\emptyset) = (1, 0.8132, 0, 0) \qquad z^+(\emptyset,\emptyset) = 54.717$$
$$\bar{x}(\emptyset,\emptyset) = (1,1,0,0) \qquad \bar{z}(\emptyset,\emptyset) = 50$$

and the solution is updated: $\bar{X} = \{(1,1,0,0)\}$ and $\bar{z} = 50$.

List and close. The list of open problems is $\mathcal{S} = \{S(\emptyset, \emptyset)\}$. Since the only available problem fails test (30) the algorithm continues.

Choose and split. Problem $S(\emptyset, \emptyset)$ is split as $S(\{2\}, \emptyset)$ and $S(\emptyset, \{2\})$.

Cycle 2: compute bounds The two new subproblems undergo the elimination test and become, respectively

- $S(\{2\}, \{4\})$ with bounds:

$$x^+(\{2\}, \{4\}) = (0.8246, 1, 0, 0) \qquad z^+(\{2\}, \{4\}) = 54.386$$
$$\bar{x}(\{2\}, \{4\}) = (1, 1, 0, 0)$$

  with no update.

- $S(\{1\}, \{2\})$ with bounds:

$$x^+(\{1\}, \{2\}) = (1, 0, 0.9149, 0) \qquad z^+(\{1\}, \{2\}) = 52.1277$$
$$\bar{x}(\{1\}, \{2\}) = (1, 0, 1, 0) \qquad \bar{z}(\{1\}, \{2\}) = 50$$

  Here $(1, 0, 1, 0)$ is appended to the set of solutions $\bar{X}$.

List and close. The list of open problems becomes $\mathcal{S} = \{S(\{2\}, \{4\}), S(\{1\}, \{2\})\}$ and none of them is closed.

Choose and split. Subproblem $S(\{2\}, \{4\})$ is split as $S(\{1, 2\}, \{4\})$ and $S(\{2\}, \{1, 4\})$.

Cycle 3: compute bounds. The new subproblem enter the variable elimination test which yield:

- problem $S(\{1, 2\}, \{3, 4\})$ with bound

$$x^+(\{1, 2\}, \{3, 4\}) = (1, 1, 0, 0)$$

  which is already in $\bar{X}$, while the problem is closed.

- problem $S(\{2, 3\}, \{1, 4\})$ with bound

$$x^+(\{2, 3\}, \{1, 4\}) = (0, 1, 1, 0) \qquad z^+(\{2, 3\}, \{1, 4\}) = 50$$

  this solution is appended to $\bar{X}$ and the subproblem is closed.

List and close. The list $\mathcal{S}$ contains only $S(\{1\}, \{2\})$, but it is not closed.

Choose and split. Problem $S(\{1\},\{2\})$ is split into $S(\{1,3\},\{2\})$ and $S(\{1\},\{2,3\})$.

Cycle 4: compute bounds. After the elimination test we have

- Problem $S(\{1,3\},\{2,4\})$ with

$$x^+(\{1,3\},\{2,4\}) = (1,0,1,0)$$

  which is already in $\bar{X}$.

- Problem $S(\{1,4\},\{2,3\})$ with

$$x^+(\{1,4\},\{2,3\}) = (1,0,0,1) \qquad \bar{z}(\{1,4\},\{2,3\}) = 50$$

  and this solution is appended to $\bar{X}$.

Both problems are closed

List and close. The list of open problems is empty, so the algorithm ends with solution set

$$\bar{X} = \{(1,1,0,0),(1,0,1,0),(0,1,1,0),(1,0,0,1)\} \text{ and } \bar{z} = 50$$

B Acknowledgements

The authors wish to thank Erio Castagnoli. The work was presented in a preliminary version at the 20th EURO conference held in Rhodes (Greece), July 4-7, 2004.

References

[1] Akin, E., 1995, Vilfredo Pareto cuts the cake, Journal of Mathematical Economics, 24, 23–44.

[2] Bertsimas, D., and J.N. Tsitsiklis, 1997, Introduction to linear optimization, Athena Scientific, Belmon, Massachusetts, U.S.A.

[3] Brams, S.J., P.H. Edelman and P.C. Fishburn, 2003, Fair division of indivisble items, Theory and Decision, 55(2), 147–180.

[4] Brams, S.J. and P.C. Fishburn, 2000, Fair division of indivisble items between two people with identical preferences: Envy-freeness, Pareto-optimality, and equity, Social Choice and Welfare, 17, 247–267.

[5] Brams, S.J., and A.D. Taylor, 1996, Fair division: from cake-cutting to dispute resolution, Cambridge University Press

[6] Brams, S.J., and A.D. Taylor, 1999, The win-win solution, guaranteeing fair shares to everybody, W.W.Norton.

[7] Demko,S., and T.P. Hill, 1988, Equitable distribution of indivisible objects, Mathematical Social Sciences, 16, 145–158.

[8] Herreiner, D., and C. Puppe, 2002, A simple procedure for finding equitable allocations of indivisible goods, Social Choice and Welfare, 19, 415–430.

[9] Kuhn, H.W. 1967, On games of fair division, In M. Shubik (ed.), Essays in Mathematical Economics in Honor of Oskar Morgenstern. Princeton University Press.

[10] Legut, J., and M. Wilczyński, 1988, Optimal partitioning of a measurable space, Proceedings of the American Mathematical Society, 104, 262–264.

[11] Papadimitriou,C.H., and K. Steiglitz, 1982, Combinatorial optimization: algorithms and complexity, Prentice & Hall.

[12] Sassano, A., 1999, Modelli e algoritmi della ricerca operativa, Franco Angeli

[13] Wolsey, L.A., and G.L. Nemhauser, 1988, Integer and combinatorial optimization, Wiley-Interscience.

[14] Vazirani, V.V., 2001, Approximation algorithms, Springer-Verlag.

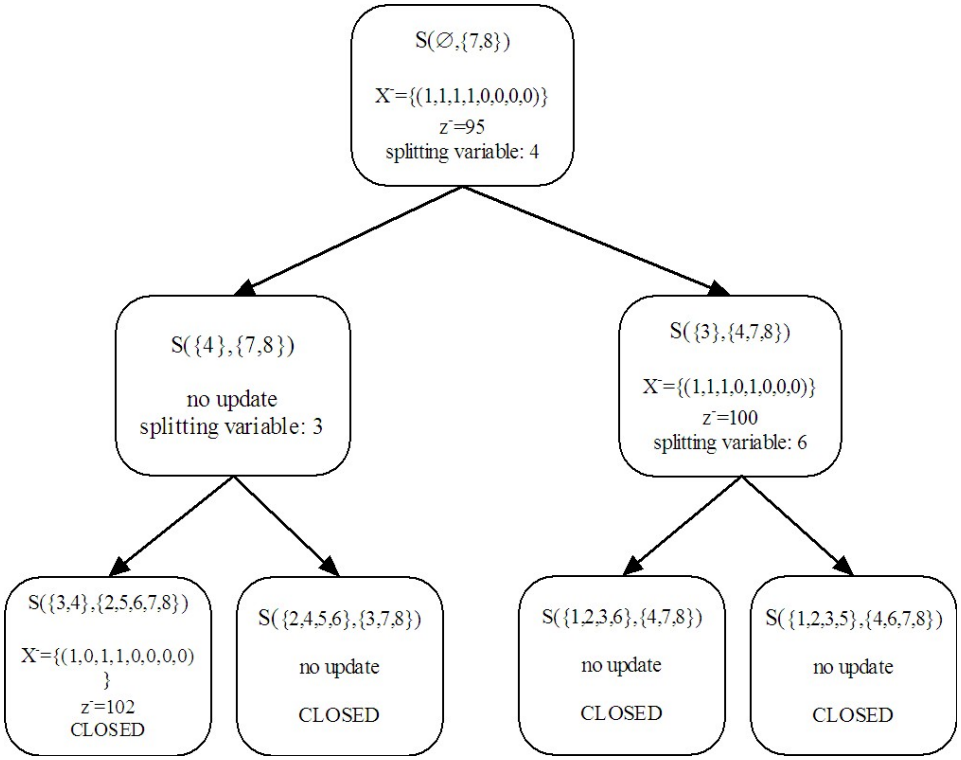

Figure 4. *The branch-and-bound algorithm for Example 2.1*

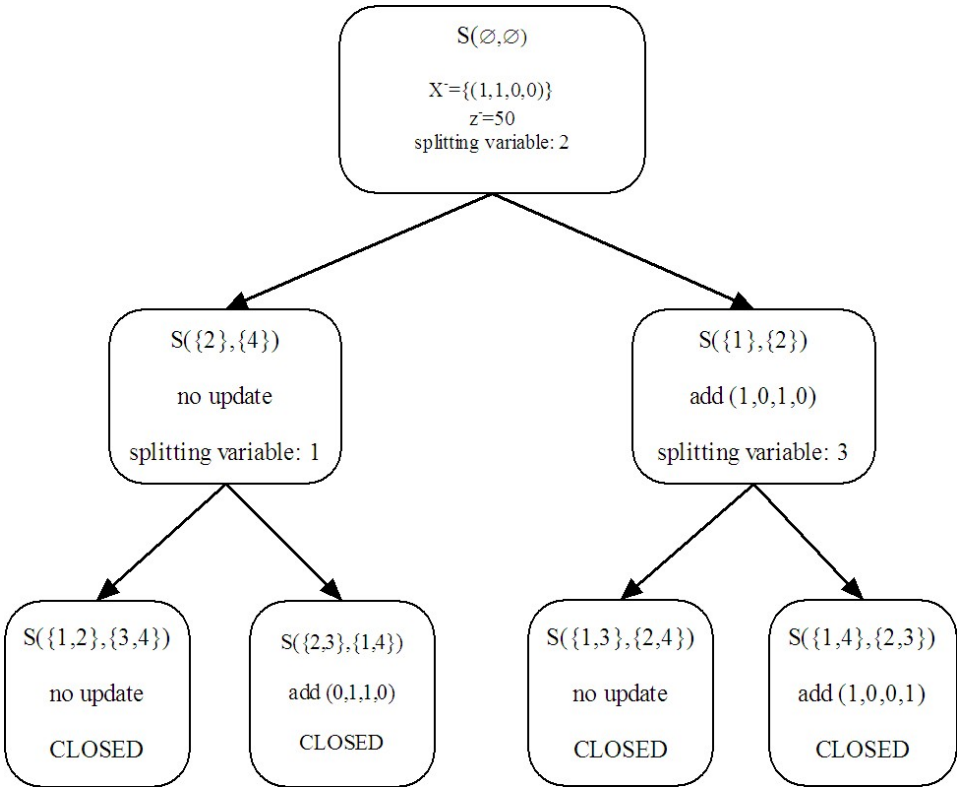

Figure 5. *The branch-and-bound algorithm for Example 2.2*

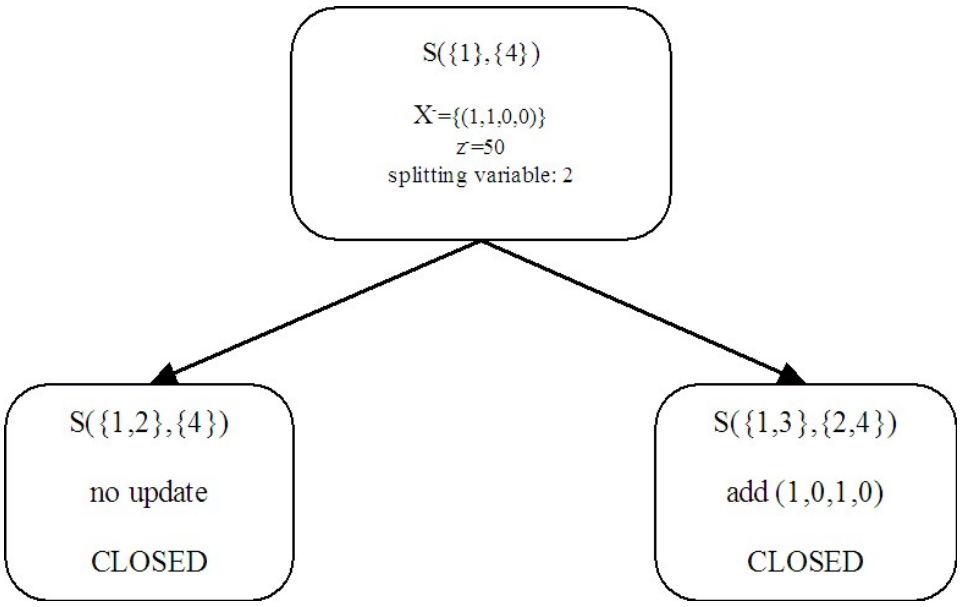

Figure 6. *Example 2.2 with the elimination test derived from Proposition 5.2*