How to allocate hard candies
Adjusted Winner and indivisible items

M. Dall'Aglio R. Mosca

Università G. d'Annunzio
Pescara — Italy

1st Comsoc
Amsterdam, December 2006

What is fair division theory?

Fair Division: How to give
some sweet food
to two or more children
with personal likes and dislikes

Different situations

- One item — completely divisible (a cake)
- Several goods — completely divisible (muffins)
- Several goods — indivisible (hard candies)

What is fair division theory?

Fair Division: How to give (assign, allocate)
some sweet food (economic resources, rights)
to two or more children (agents, players)
with personal likes and dislikes (subjective preferences)

Different situations

- One item — completely divisible (a cake)
- Several goods — completely divisible (muffins)
- Several goods — indivisible (hard candies)

The mathematical setting

Partition of indivisible goods between two persons.
Table of evaluation

item	1	2	$\cdots$	m
Alice	a_1	a_2	$\cdots$	a_m
Bob	b_1	b_2	$\cdots$	b_m

Assumptions:

$$(a)\ a_i, b_i \geq 0 \qquad (b)\ \sum_i a_i = \sum_i b_i$$

Definitions

$$v_A = \text{ total value (sum) of the items given to Alice}$$
$$v_B = \text{ total value (sum) of the items given to Bob}$$

A procedure with divisible items

Assumption: linearity

If Alice (Bob, resp.) gets fraction $t \in (0, 1)$ of item i she values it ta_i (tb_i, resp.)

The "Adjusted Winner" (AW) procedure

1 – The "winning" phase Each player receives the items that he/she values more than the other player. The total score of both is computed.

2 – The "adjusting" phase Items are transferred, one at a time, from the richer player to the poorer one, starting with the items of the richer player with ratio a_i/b_i closer to 1. The process continues until both have the same score. To reach perfect equality, one item may be split.

Properties of AW

Theorem (Brams and Taylor, 1996)

The AW allocation is

- **envy-free** Each player does not wish to swap bundles
- **efficient** There is no other allocation that is better for both
- **equitable** Both players are treated equally: $v_A = v_B$

Properties of AW

Theorem (Brams and Taylor, 1996)

The AW allocation is

- **envy-free** Each player does not wish to swap bundles
- **efficient** There is no other allocation that is better for both
- **equitable** Both players are treated equally: $v_A = v_B$

Proposition

The AW allocation is maximin. It solves

$$z^+ = \max_{\text{allocations}} \min\{v_A, v_B\}$$

Alice and Bob can split items

Proof: An allocation is maximin iff it is efficient and equitable.

Goals of the present work

Compute

$$z^* = \max_{\text{allocations}} \min\{v_A, v_B\}$$
Alice and Bob cannot split items

by means of

Goals of the present work

Compute

$$z^* = \max_{\text{allocations}} \min\{v_A, v_B\}$$
Alice and Bob cannot split items

by means of

- A step-by step procedure in the same spirit of AW. A set of rules that if followed by the players bring them to the optimal solution. Such a procedure should be:

intuitive	Each step must be easy to understand
plausible	Each step must be simple to argue
manageable	Each step must be straightforward to compute

Goals of the present work

Compute

$$z^* = \max_{\text{allocations}} \min\{v_A, v_B\}$$
Alice and Bob cannot split items

by means of

- A step-by step procedure in the same spirit of AW. A set of rules that if followed by the players bring them to the optimal solution. Such a procedure should be:

intuitive	Each step must be easy to understand
plausible	Each step must be simple to argue
manageable	Each step must be straightforward to compute

- A (faster) computer routine

A side result

An extension of AW to the case with endowments

The Adjusted Winner with endowments

We modify the AW procedure to cover the case where some of the items have been assigned in advance to the children. We consider

$$\begin{array}{rcl} z^{+}(\mathcal{A},\mathcal{B}) & = & \max_{\text{allocations}} \min\{v_A, v_B\} \\ \text{s.t.} & & \text{Alice takes all items in } \mathcal{A} \\ & & \text{Bob takes all items in } \mathcal{B} \\ & & \text{Alice and Bob can split items} \end{array}$$

with $\mathcal{A}, \mathcal{B}$ disjoint subsets of items.
The procedure takes care of the disputable items (not in $\mathcal{A}$ nor $\mathcal{B}$)

The Adjusted Winner with endoments (AW-e)

0 – A preliminary phase Alice (Bob, resp.) receives the items in $\mathcal{A}$ ($\mathcal{B}$, resp.)

1 – The "winning" phase Each player receives the disputable items that he/she values more than the other player. The total score (of disputable and constrained items) of both is computed.

2 – The "adjusting" phase Disputable items are transferred, one at a time, from the richer player to the poorer one, starting with the items of the richer player with ratio a_i/b_i closer to 1. The process continues until...

- both have the same score, or
- until possible if equality is not reached

AW-e and equitable allocations

The value of the items forcedly assigned to the children is

$$\alpha = \sum_{i \in \mathcal{A}} a_i \qquad \beta = \sum_{i \in \mathcal{B}} b_i$$

An equitable allocation is reached only in the case

$$-\sum_{i \in D} a_i \leq \alpha - \beta \leq \sum_{i \in D} b_i$$

where $D = (A \cup B)^c$ is the set of disputable items.

AW-e and equitable allocations

The value of the items forcedly assigned to the children is

$$\alpha = \sum_{i \in \mathcal{A}} a_i \qquad \beta = \sum_{i \in \mathcal{B}} b_i$$

An equitable allocation is reached only in the case

$$-\sum_{i \in D} a_i \leq \alpha - \beta \leq \sum_{i \in D} b_i$$

where $D = (A \cup B)^c$ is the set of disputable items.
Otherwise,

- If $\alpha + \sum_{i \in D} a_i < \beta$ all items given to Alice, who remains poorer than Bob
- If $\beta + \sum_{i \in D} b_i < \alpha$ all items given to Bob, who remains poorer than Alice

Basic ideas for the maxmin allocation with intact items

- Based on integer linear programming techniques (branch-and-bound)
- The original problem is divided into smaller subproblems, where items in $\mathcal{A}$ ($\mathcal{B}$, resp.) are assigned to Alice (Bob, resp.)

$$
\begin{array}{rcl}
z^*(\mathcal{A},\mathcal{B}) & = & \max_{\text{allocations}} \min\{v_A, v_B\} \\
\text{s.t.} & & \text{Alice takes all items in } \mathcal{A} \\
& & \text{Bob takes all items in } \mathcal{B} \\
& & \text{Alice and Bob cannot split items}
\end{array}
\qquad (S(\mathcal{A},\mathcal{B}))
$$

Basic ideas — continued

The corresponding AW procedure with endowments:

- gives an upper bound for the problem without splitting

$$z^*(\mathcal{A}, \mathcal{B}) \leq z^+(\mathcal{A}, \mathcal{B})$$

Basic ideas — continued

The corresponding AW procedure with endowments:

- gives an upper bound for the problem without splitting

$$z^*(\mathcal{A}, \mathcal{B}) \leq z^+(\mathcal{A}, \mathcal{B})$$

- in case the solution of AW-e is intact, it solves the corresponding problem with indivisible items

Basic ideas — continued

The corresponding AW procedure with endowments:

- gives an upper bound for the problem without splitting

$$z^*(\mathcal{A}, \mathcal{B}) \leq z^+(\mathcal{A}, \mathcal{B})$$

- in case the solution of AW-e is intact, it solves the corresponding problem with indivisible items
- in case of splitting it shows how to add new constraints: if item i is split, two new subproblems are considered:

$$\begin{array}{ccc} & S(\mathcal{A}, \mathcal{B}) & \\ & \downarrow \qquad \downarrow & \\ S(\mathcal{A} \cup \{i\}, \mathcal{B}) & & S(\mathcal{A}, \mathcal{B} \cup \{i\}) \end{array}$$

Basic ideas — continued

The corresponding AW procedure with endowments:

- gives an upper bound for the problem without splitting

$$z^*(\mathcal{A}, \mathcal{B}) \leq z^+(\mathcal{A}, \mathcal{B})$$

- in case the solution of AW-e is intact, it solves the corresponding problem with indivisible items
- in case of splitting it shows how to add new constraints: if item i is split, two new subproblems are considered:

$$\begin{array}{ccc} & S(\mathcal{A}, \mathcal{B}) & \\ \swarrow & & \searrow \\ S(\mathcal{A} \cup \{i\}, \mathcal{B}) & & S(\mathcal{A}, \mathcal{B} \cup \{i\}) \end{array}$$

- In case we find an admissible allocation which performs better than the upper bound: we are on a wrong way — better change the constraints.

A sketch of the step-by-step procedure

At each step

- $\bar{z}$ is the value of the best intact solution recorded so far
- $\mathcal{A}, \mathcal{B}$ items currently assigned to the players

The AW-e procedure is run:

1. If the value of AW-e is not greater than $\bar{z}$, the last item added to the constraints is

A sketch of the step-by-step procedure

At each step

- $\bar{z}$ is the value of the best intact solution recorded so far
- $\mathcal{A}, \mathcal{B}$ items currently assigned to the players

The AW-e procedure is run:

1. If the value of AW-e is not greater than $\bar{z}$, the last item added to the constraints is
   - given to the other player, or, if this was already done,

A sketch of the step-by-step procedure

At each step

- $\bar{z}$ is the value of the best intact solution recorded so far
- $\mathcal{A}, \mathcal{B}$ items currently assigned to the players

The AW-e procedure is run:

1. If the value of AW-e is not greater than $\bar{z}$, the last item added to the constraints is
   - given to the other player, or, if this was already done,
   - put back in the pile of disputed items.

A sketch of the step-by-step procedure

At each step

- $\bar{z}$ is the value of the best intact solution recorded so far
- $\mathcal{A}, \mathcal{B}$ items currently assigned to the players

The AW-e procedure is run:

1. If the value of AW-e is not greater than $\bar{z}$, the last item added to the constraints is
   - given to the other player, or, if this was already done,
   - put back in the pile of disputed items.
2. if the solution of AW-e is intact, its value replaces $\bar{z}$. The last item added to the constraints is given to other or put back in the disputed items.

A sketch of the step-by-step procedure

At each step

- $\bar{z}$ is the value of the best intact solution recorded so far
- $\mathcal{A}, \mathcal{B}$ items currently assigned to the players

The AW-e procedure is run:

1. If the value of AW-e is not greater than $\bar{z}$, the last item added to the constraints is
   - given to the other player, or, if this was already done,
   - put back in the pile of disputed items.
2. if the solution of AW-e is intact, its value replaces $\bar{z}$. The last item added to the constraints is given to other or put back in the disputed items.
3. if the solution of AW-e divides an item, this is forcedly given to the player who benefits more from it (and added to the constraints $\mathcal{A}$ or $\mathcal{B}$)

A sketch of the step-by-step procedure

At each step

- $\bar{z}$ is the value of the best intact solution recorded so far
- $\mathcal{A}, \mathcal{B}$ items currently assigned to the players

The AW-e procedure is run:

1. If the value of AW-e is not greater than $\bar{z}$, the last item added to the constraints is
   - given to the other player, or, if this was already done,
   - put back in the pile of disputed items.
2. if the solution of AW-e is intact, its value replaces $\bar{z}$. The last item added to the constraints is given to other or put back in the disputed items.
3. if the solution of AW-e divides an item, this is forcedly given to the player who benefits more from it (and added to the constraints $\mathcal{A}$ or $\mathcal{B}$)

The process ends when no item can be added or removed from the constraints

A shortcut for AW-e

When items are added to/removed from the constraints, players do not need to rerun AW-e from scratch at each step

- the solution from the previous run of AW-e is recorded.
- if needed, items in the constraints are moved from one player to the other. The score of both is computed.
- Disputed items (according to the new constraints are transferred from the richer player to the poorer one. The transfer begins with the items of the richer player with
  - smallest ratio a_i/b_i if Alice is richer
  - highest ratio a_i/b_i if Bob is richer

A computer routine

Works on the same principles, plus
A larger set of constraints $\Rightarrow$ Fewer subproblems to analyze
At each step:

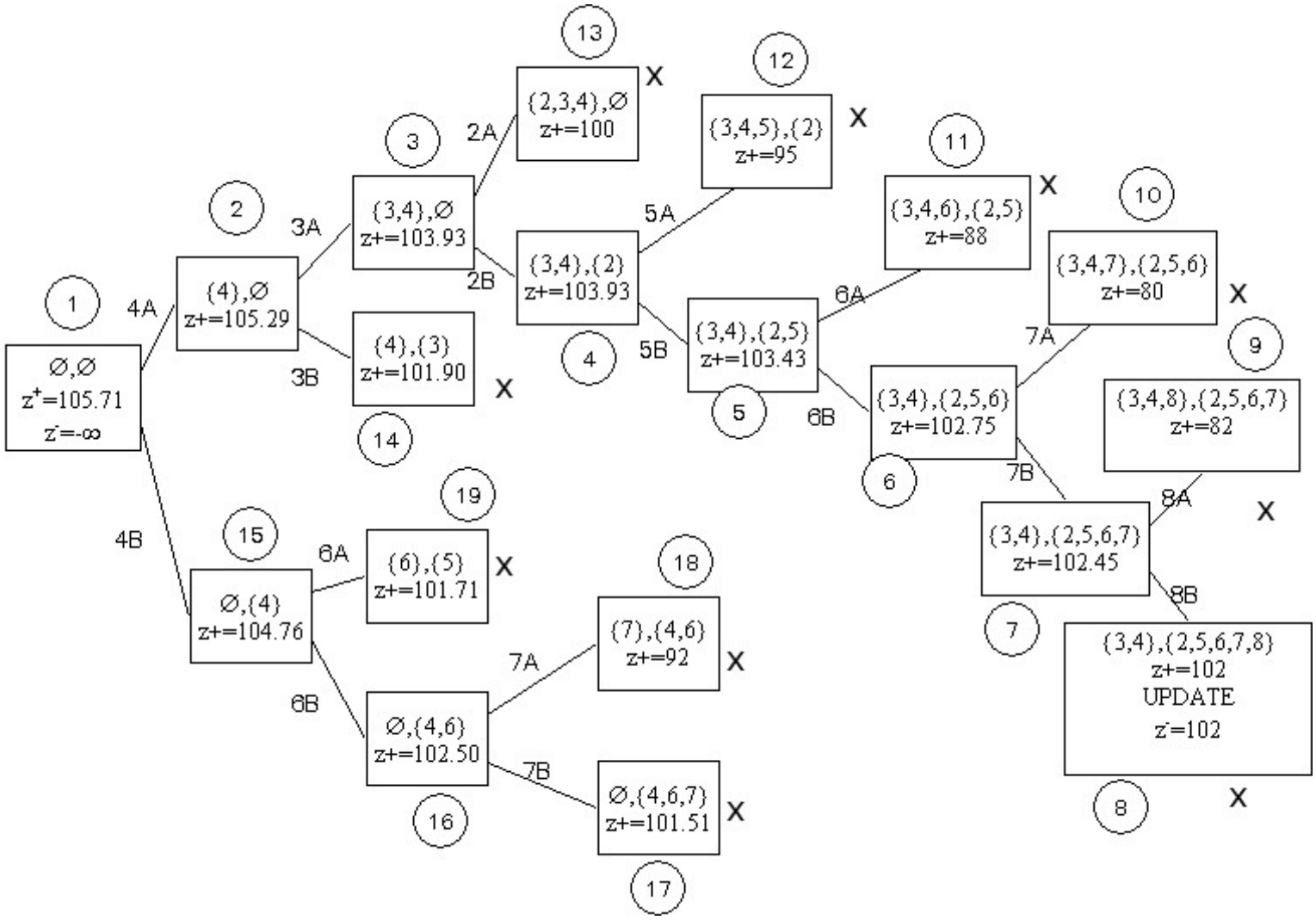
- More than one item can be added to the constraints — with a variable elimination test
- A new admissible solution is computed and several subproblems can be closed simultaneously

Example

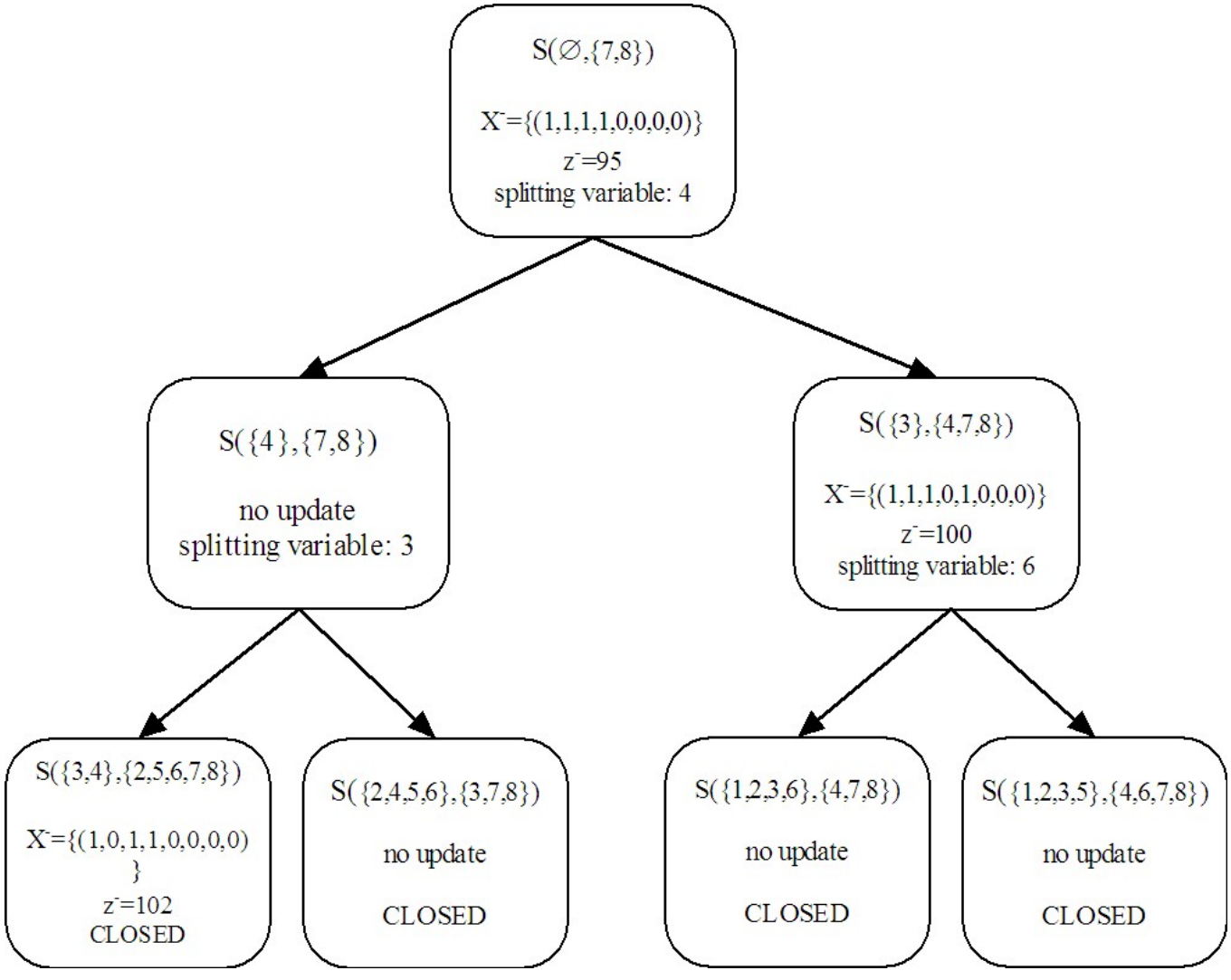
Muffin	Alice	Bob	w/ split	w/out sp.
1	10	30	B	B
2	20	15	B	B
3	18	10	A	B
4	12	5	A	A
5	50	35	A	A
6	40	30	split	A
7	20	22	B	B
8	5	28	B	B
Total	175	175		

w/ split $(v_A, v_B) = (105.71, 105.71)$
w/out split $(v_A, v_B) = (102, 105)$

Example — step-by-step procedure



Example — computer routine



Example 2: The Panama canal treaty negotiation

A consulting firm interviews the two negotiating teams

Issue	USA	Pan.	w/ split	w/out sp.
1 US defense rights	22	9	USA	USA
2 Use rights	22	15	USA	USA
3 Land and water	15	15	split	Pan.
4 Expansion rights	14	3	USA	USA
5 Duration	11	15	Pan.	Pan.
6 Expansion routes	6	5	USA	USA
7 Compensation	4	11	Pan.	Pan.
8 Jurisdiction	2	7	Pan.	Pan.
9 US military rights	2	7	Pan.	Pan.
10 Defense role of Pan.	2	13	Pan.	Pan.
Total	100	100		

w/ split $(V_{USA}, v_{Pan}) = (64, 68)$
w/out split $(V_{USA}, v_{Pan}) = (66, 66)$

Open problems

- Dynamic programming works faster than branch-and-bound

  $\Rightarrow$ A procedure based on dynamic programming
- Extension to more than 2 players
- (extension of AW to more than 2 players?)

APPENDIX – The AW procedure

Suppose $v_A \geq v_B$. Then Alice begins transferring items to Bob, one at a time, starting with the item with ratio a_i/b_i closer to 1 (and greater than or equal to 1). The handover continues until perfect equitability is achieved, or the roles of the "richer" and "poorer" player are reversed. In the last case, suppose that after the handover of, say, item r we have $v_A < v_B$. Item r is then split, with Alice getting a fraction given by

$$x_r = \frac{b_r + v_B^{-r} - v_A^{-r}}{a_r + b_r}$$

where v_A^{-r} and v_B^{-r} are the scores obtained by the two players so far in the process *without* considering item r. Bob gets the remaining fraction.

APPENDIX — Example

Muffin	Alice	Bob	ratio	
1	10	30		
2	20	15		
3	18	10		
4	12	5		
5	50	35		
6	40	30		
7	20	22		
8	5	28		
Total	175	175		

Alice and Bob state their preferences

.

APPENDIX — Example — The "winning" phase

Muffin	Alice	Bob	ratio	
1	10	30		$\Rightarrow$ Bob
2	20	15		$\Rightarrow$ Alice
3	18	10		$\Rightarrow$ Alice
4	12	5		$\Rightarrow$ Alice
5	50	35		$\Rightarrow$ Alice
6	40	30		$\Rightarrow$ Alice
7	20	22		$\Rightarrow$ Bob
8	5	28		$\Rightarrow$ Bob
Total	175	175		

Each child takes the muffin he/she values more than the other child

$$v_A = 140 \qquad v_B = 80$$

APPENDIX — Example — The "adjusting" phase — 1

Muffin	Alice	Bob	ratio	
1	10	30		
2	20	15	1.33	A.⇒ B.
3	18	10	1.8	
4	12	5	2.4	
5	50	35	1.43	
6	40	30	1.33	(or this)
7	20	22		
8	5	28		
Total	175	175		

Alice is richer

She gives the muffin with ratio of preferences closer to 1

$$v_A = 120 \qquad v_B = 95$$

APPENDIX — Example — The "adjusting" phase — 2

Muffin	Alice	Bob	ratio	
1	10	30		
2	20	15		
3	18	10	1.8	
4	12	5	2.4	
5	50	35	1.43	
6	40	30	1.33	A.$\Rightarrow$ B.
7	20	22		
8	5	28		
Total	175	175		

Alice is still richer

But if she gives the whole muffin, she becomes poorer than Bob

$$v_A = 80 \qquad v_B = 125$$

APPENDIX — Example — The "adjusting" phase — 3

Muffin	Alice	Bob	ratio	
1	10	30		
2	20	15		
3	18	10	1.8	
4	12	5	2.4	
5	50	35	1.43	
6	25.71	10.71	1.33	split
7	20	22		
8	5	28		
Total	175	175		

If the muffin is split
Alice gets 64% of muffin 6, Bob gets 36%. They obtain the same score

$$v_A = 105.71 \quad v_B = 105.71$$

APPENDIX — Example — Remark

Muffin	Alice	Bob	ratio	
1	10	30		
2	20	15	1.33	
3	18	10	1.8	
4	12	5	2.4	
5	50	35	1.43	
6	5.71	25.71	1.33	split
7	20	22		
8	5	28		
Total	175	175		

The solution is not unique. They now split muffin 6, so that Alice gets 14% and Bob 86% of it. We still have

$$v_A = 105.71 \quad v_B = 105.71$$

(they get the same value)

APPENDIX: step-by-step

For each set of constraints, $\mathcal{A}$ and $\mathcal{B}$, the following info is required

- the last item added to the constraints (and who got it)
- the state of the subproblem

  open new constraints may be appended
  closed items in the constraints may be freed

$\bar{X}, \bar{z}$ best current solution(s) and value

Step 0: Initialization

All items are free and the best current solution is empty

APPENDIX: step-by-step

Step 1: forward step

For a given set of constraints, $\mathcal{A}$ and $\mathcal{B}$:

- Compute AW-e, $x^+ = x^+(\mathcal{A}, \mathcal{B})$, and its value $z^+ = z^+(\mathcal{A}, \mathcal{B})$
- If $z^+ < \bar{z}$ the subproblem is closed. Go to step 2.
- If x^+ is intact, compare z^+ with the best current solution $\bar{z}$ and, in case, update.
- Else, j is the split item and t is the fraction of it assigned to Alice.
  - If $ta_j \geq (1-t)b_j$ assign the item to Alice
  - Else, assign it to Bob

  In both cases denote the new subproblem as open. Repeat this step with the new constraints

APPENDIX: step-by-step

Step 2: Backward step

Remove the most recently added item, say j, from the constraints

- If the new subproblem is open, mark it close, and give item j to the other child. Make a step forward.
- If the new subproblem has no constraints and is closed, exit the procedure (Step 3)
- If the new subproblem contains at least a constraint and is closed, repeat this step.

Step 3: Exit

The best current solution is optimal

APPENDIX — A shortcut for the computer routine

The set of constraints can be augmented via a preliminary test
A larger set of constraints $\Rightarrow$ a simpler problem $\Rightarrow$ a shorter tree

A variable elimination test (VET)

- An admissible solution with no splitting for $S(\mathcal{A}, \mathcal{B})$ is given by

$$\bar{z}(\mathcal{A}, \mathcal{B}) = \text{rnd}(z^+(\mathcal{A}, \mathcal{B}))$$

- For each "free" item i
  - If $z^+(\mathcal{A} \cup \{i\}, \mathcal{B}) < \bar{z}(\mathcal{A}, \mathcal{B})$, then item i is added to $\mathcal{B}$
  - If $z^+(\mathcal{A}, \mathcal{B} \cup \{i\}) < \bar{z}(\mathcal{A}, \mathcal{B})$, then item i is added to $\mathcal{A}$

APPENDIX — The computer routine in detail

$\bar{X}$ temporary optimal solution, $\bar{z}$ temporary optimal value
$S(\mathcal{A}, \mathcal{B})$ can have three labels: new, open, close

Initalization

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

A cycle

1. For each new $S(\mathcal{A}, \mathcal{B})$
   - 1.1 do the VET $\rightarrow S(\mathcal{A}', \mathcal{B}')$
   - 1.2 solve $S^+(\mathcal{A}', \mathcal{B}')$
   - 1.3 if x^+ integer: update $\bar{X}, \bar{z}$ if necessary, label $S(\mathcal{A}', \mathcal{B}')$ close
   - 1.4 if x^+ not integer, label $S(\mathcal{A}', \mathcal{B}')$ open
2. Close all open $S(\mathcal{A}, \mathcal{B})$ such that $z^+(\mathcal{A}, \mathcal{B}) < \bar{z}$
3. if no open subpbm. left, $\bar{X}, \bar{z}$ optimal, exit the procedure
4. Pick $S(\mathcal{A}, \mathcal{B})$ with higher $z^+(\mathcal{A}, \mathcal{B})$ and replace with $S(\mathcal{A} \cup \{i\}, \mathcal{B})$ and $S(\mathcal{A}, \mathcal{B} \cup \{i\})$. Label them as new.
5. Continue with the next step