games

Article

Efficient Solutions to Multidimensional Claims Problem

Sijia Xu [1] and Peng Liu [2,*]

[1] East China University of Science and Technology, Shanghai 200237, China; xusijia@ecust.edu.cn
[2] East China Normal University, Shanghai 200062, China
* Correspondence: pliu@fem.ecnu.edu.cn

Abstract

In this paper, we study the multidimensional claims problem and introduce the eating algorithm to this problem. It is shown that a solution is efficient if and only if it is the outcome of the eating algorithm with a profile of specific eating functions. Moreover, we adapt three classical solutions, i.e., the constrained equal awards rule, constrained equal losses rule, and proportional rule, to the current setting and show that they are special cases of the eating algorithm with specific eating functions.

Keywords: multidimensional claims problem; efficiency; eating algorithm

JEL Classification: D63; D74

1. Introduction

A claims problem refers to the task of allocating a given amount among a group of claimants. A rule needs to select an allocation for each problem, no matter what the amount to be allocated is, how many claimants there are, and what their claims are. A fundamental paper in the literature in this area is by O'Neill (1982). Two excellent surveys include those by Thomson (2003, 2015).

We extend the classical claims problem to a multidimensional setting. Specifically, instead of one amount, there are finitely many goods to be allocated, each with a fixed capacity. A package refers to a combination of various goods. It is worth noting that, for specific applications, not all combinations are feasible. Consider for instance a passenger train with two intermediate stops. There are three different goods, each with a capacity equal to the number of seats on the train. A package hence refers to a ticket, which is a combination of different goods. It is evident that there are in total six feasible packages.[1] Each claimant requests an amount of a package, and the task of the mechanism designer is to allocate packages to them. For the aforementioned example, the task is to determine the number of different tickets for sale regarding a predicted demand. Another example can be found in Acosta et al. (2022), where a government aims to reduce various pollutants, but each pollutant may affect multiple environmental issues (like climate change and air quality). The government sets emission limits for each issue. Each pollutant "claims" a share of the allowed emissions based on the amount being emitted. The challenge is to allocate limited emissions across pollutants, considering that the same pollutant affects multiple issues.

We introduce the eating algorithm for the current multidimensional problem and show that an allocation is efficient if and only if it is the outcome of the eating algorithm with some specified eating functions (Theorem 1). The eating algorithm can be viewed

check for updates

Academic Editor: Richard McLean

Received: 16 August 2025
Revised: 26 September 2025
Accepted: 28 September 2025
Published: 9 October 2025

Citation: Xu, S., & Liu, P. (2025). Efficient Solutions to Multidimensional Claims Problem. *Games*, *16*(5), 51. https://doi.org/10.3390/g16050051

as a concrete embodiment of the greedy idea in computer science. One can refer to chapter 15 of Cormen et al. (2022) for applications of the greedy idea to various problems. Regarding economic studies, it has been used to solve the random allocation problem (Bogomolnaia & Moulin, 2001). Moreover, it has been shown that the outcomes of the eating algorithm with specific eating functions characterize efficient random allocations. From this perspective, our Theorem 1 can be seen as another success in line with the ideas of Bogomolnaia and Moulin (2001).

Next, we adapt the classical rules in the literature regarding the claims problem to the current multidimensional setting. In particular, we provide formal definitions of the constrained equal awards rule, constrained equal losses rule, and constrained proportional rule. Moreover, we show that all these rules are special cases of the eating algorithm with specific eating functions. Put otherwise, the outcomes of these rules can be generated by the eating algorithm with carefully chosen eating functions.

2. Model

Let G be a set of finitely many goods, with $|G| \geq 2$. Each good $a \in G$ has a fixed **capacity**, denoted $c_a \in \mathbb{R}_{++}$.[2] We collect capacities in a vector, denote it $c \in \mathbb{R}^G_{++}$, and call it the capacity vector. A **package** consists of certain amounts of various goods. Specifically, a package is denoted by a vector $P \in \mathbb{R}^G_+$ such that $\sum_{a \in G} P_a > 0$.[3] The collection of feasible packages is denoted by $\mathcal{P}$. The **demand** on package $P \in \mathcal{P}$ is denoted $d_P \in \mathbb{R}_+$. The vector $d \in \mathbb{R}^{\mathcal{P}}_+$ collects these demands and is called a demand vector.

An allocation **problem** is described by a couple of one capacity vector and one demand vector, i.e., $(c, d) \in \mathbb{R}^G_{++} \times \mathbb{R}^{\mathcal{P}}_+$. To avoid triviality, we assume that, for each problem (c, d), there exists at least one good $a \in G$ such that $\sum_{P \in \mathcal{P}} d_P \cdot P_a > c_a$. This implies that scarcity exists and that the allocation problem is not trivial. Faced with a problem, the mechanism designer needs to specify an **allocation**, i.e., a vector $x \in \mathbb{R}^{\mathcal{P}}_+$, where x_P denotes the size of package P that is allocated. A **rule** is a function $r : \mathbb{R}^G_{++} \times \mathbb{R}^{\mathcal{P}}_+ \to \mathbb{R}^{\mathcal{P}}_+$ that selects one allocation for each problem.

We provide a specific problem below.

Example 1. *Consider a passenger train traveling from Shanghai to Beijing, with intermediate stops at Nanjing and Jinan. In our language, there are three different goods $G = \{SN, NJ, JB\}$, where SN represents a seat from Shanghai to Nanjing, NJ a seat from Nanjing to Jinan, and JB a seat from Jinan to Beijing. Suppose there are in total 1000 seats on the train. The capacity vector is hence $c = (1000, 1000, 1000) \in \mathbb{R}^G_{++}$. The set $\mathcal{P}$ consists of 6 feasible packages as follows.*

$$\begin{array}{cccccc} & & SN & NJ & JB & \\ P^1 = & (& 1 & 0 & 0 &) \\ P^2 = & (& 1 & 1 & 0 &) \\ P^3 = & (& 1 & 1 & 1 &) \\ P^4 = & (& 0 & 1 & 0 &) \\ P^5 = & (& 0 & 1 & 1 &) \\ P^6 = & (& 0 & 0 & 1 &) \end{array}$$

Consider for instance that the mechanism designer has predicted, according to historical data, a demand vector as follows. The problem is hence described by (c, d), and the task of the mechanism designer is to determine an allocation vector.

$$\begin{array}{cccccccc} & & P^1 & P^2 & P^3 & P^4 & P^5 & P^6 & \\ d = & (& 450 & 200 & 450 & 100 & 800 & 500 &) \end{array}$$

Throughout the paper, we impose two properties on desirable rules. The first is feasibility, requiring (i) for each package, the amount allocated cannot exceed its demand; and (ii) for each good, the amount allocated cannot exceed its capacity.[4] We assume throughout the paper that a rule must be feasible.

Definition 1. *An allocation vector $x \in \mathbb{R}^{\mathcal{P}}_{+}$ is feasible at problem (c, d) if $x \leq d$ and, for each $a \in G$, $\sum_{P \in \mathcal{P}} x_P \cdot P_a \leq c_a$. A rule is feasible if it selects at each problem a feasible allocation vector.*

The second property is efficiency, requiring that no package's allocation can be increased without decreasing another's.

Definition 2. *An allocation vector $x \in \mathbb{R}^{\mathcal{P}}_{+}$ is efficient at problem (c, d) if there is no allocation vector $y \in \mathbb{R}^{\mathcal{P}}_{+}$ such that y is feasible at (c, d) and $y > x$. A rule is efficient if it selects at each problem an efficient allocation vector.*

The multidimensional claims problems defined in the current paper can be seen as the classical claims problem extended in two directions. First, there are multiple feasibility constraints, each of which corresponds to each good. Second, the definition of feasibility is not that the summation of allocations is less than or equal to a fixed amount but concerns a structured combination.

3. Three Classical Allocation Rules

We introduce three classical rules to the current setting. We first provide formal definitions of them and then apply them to the problem in Example 1 as an illustration.

The constrained equal awards rule below follows the idea of equal division.

Definition 3. *The constrained equal awards rule (CEA) selects an allocation for each problem (c, d) by the algorithm as follows.*

- ***Initialization***: $d^0_P \equiv d_P, x^0_P \equiv 0, \forall P \in \mathcal{P}$.
- ***Step*** $k = 1, \ldots$:

$$\lambda^k \equiv sup\left\{\lambda \in \mathbb{R}_+ : \sum_{P \in \mathcal{P}} x^{k-1}_P \cdot P_a + \sum_{P \in \mathcal{P}} min\{\lambda, d^{k-1}_P\} \cdot P_a \leq c_a, \forall a \in G\right\}$$

$$x^k_P \equiv x^{k-1}_P + min\{\lambda^k, d^{k-1}_P\}$$

$$d^k_P \equiv \begin{cases} 0, & \text{if } \exists a \in G \text{ s.t. } P_a > 0 \text{ and } \sum_{P \in \mathcal{P}} x^k_P \cdot P_a = c_a \\ d^{k-1}_P - min\{\lambda^k, d^{k-1}_P\}, & \text{otherwise} \end{cases}$$

- ***Termination condition***: $d^k_P = 0, \forall P \in \mathcal{P}$.

Instead of equally allocating the awards, the constrained equal losses rule below follows the idea of equalizing losses.

Definition 4. *The constrained equal losses rule (CEL) selects an allocation for each problem (c, d) by the algorithm as follows.*

- ***Initialization***: $d^0_P \equiv d_P, x^0_P \equiv 0, \forall P \in \mathcal{P}$.
- ***Step*** $k = 1, \ldots$:

$$\lambda^k \equiv inf\left\{\lambda \in \mathbb{R}_+ : \sum_{P\in\mathcal{P}} x_P^{k-1} \cdot P_a + \sum_{P\in\mathcal{P}} max\{0, d_P^{k-1} - \lambda\} \cdot P_a \le c_a, \forall a \in G\right\}$$
$$x_P^k \equiv x_P^{k-1} + max\{0, d_P^{k-1} - \lambda^k\}$$
$$d_P^k \equiv \begin{cases} 0, & if\ \exists a \in G\ s.t.\ P_a > 0\ and\ \sum_{P\in\mathcal{P}} x_P^k \cdot P_a = c_a \\ d_P^{k-1} - max\{0, d_P^{k-1} - \lambda^k\}, & otherwise \end{cases}$$

- ***Termination condition***: $d_P^k = 0, \forall\ P \in \mathcal{P}$.

The constrained proportional rule below allocates the goods proportional to demands.

Definition 5. *The constrained proportional rule (CP) selects an allocation x for each problem (c,d) by the algorithm as follows.*

- ***Initialization***: $d_P^0 \equiv d_P, x_P^0 \equiv 0, \forall P \in \mathcal{P}$.
- ***Step*** $k = 1, \ldots$:

$$\lambda^k \equiv sup\left\{\lambda \in \mathbb{R}_+ : \sum_{P\in\mathcal{P}} x_P^{k-1} \cdot P_a + min\{\frac{d_P^{k-1}}{\sum_{P\in\mathcal{P}} d_P^{k-1}} \cdot \lambda, d_P^{k-1}\} \cdot P_a \le c_a, \forall a \in G\right\}$$
$$x_P^k \equiv x_P^{k-1} + min\{\frac{d_P^{k-1}}{\sum_{P\in\mathcal{P}} d_P^{k-1}} \cdot \lambda^k, d_P^{k-1}\}$$
$$d_P^k \equiv \begin{cases} 0, & if\ \exists a\ s.t.\ P_a > 0\ and\ \sum_{P\in\mathcal{P}} x_P^k \cdot P_a = c_a \\ d_P^{k-1} - min\{\frac{d_P^{k-1}}{\sum_{P\in\mathcal{P}} d_P^{k-1}} \cdot \lambda^k, d_P^{k-1}\}, & otherwise \end{cases}$$

- ***Termination condition***: $d_P^k = 0, \forall\ P \in \mathcal{P}$.

Example 2. *Consider the problem described in Example 1. The allocation vectors chosen by three classical rules are as follows.*

$$\begin{array}{lcccccc} & P^1 & P^2 & P^3 & P^4 & P^5 & P^6 \\ CEA(c,d) = & (\ 450 & 200 & 333\frac{1}{3} & 100 & 333\frac{1}{3} & 333\frac{1}{3}\ ) \\ CEL(c,d) = & (\ 450 & 175 & 200 & 75 & 550 & 250\ ) \\ CP(c,d) = & (\ 450 & 190\frac{10}{21} & 257\frac{1}{7} & 95\frac{5}{21} & 457\frac{1}{7} & 285\frac{5}{7}\ ) \end{array}$$

For realistic applications, the above allocations can be rounded to integers. For illustration, we present in Appendix A the calculations of CEA allocation.

The model in the current paper is continuous in the sense that the capacities, demands, and allocations can take any real numbers, whereas Liu and Xu (2024) provide a model in which discrete objects need to be packaged to be bundles for allocation. The similarity between two models is that both of them study the allocation of combinations of different goods. The differences include two points. The first is that the model in the current paper is continuous, as mentioned, while the model in Liu and Xu (2024) is discrete, where the number of goods can take only integers. The second is that the current paper focuses on efficient allocations, while Liu and Xu (2024) focuses on how to package different objects in a consistent way.

4. Eating Rule and the Results

We first provide the formal definition of the eating algorithm and then the relevant results.

Definition 6. *Given a problem (c,d) and, for each package $P \in \mathcal{P}$, a speed function $s_P : [0,1] \to \mathbb{R}_+$ such that $\int_0^1 s_P(v)dv = d_P$, the* ***eating rule*** *determines an allocation, denoted $E^s(c,d)$, by the algorithm as follows.*

- ***Initialization:*** $t^0 \equiv 0$; $\mathcal{P}^0 \equiv \mathcal{P}$; $G^0 \equiv G$; $x_P^0 = 0, \forall P \in \mathcal{P}$.
- ***Step*** $k = 1, \ldots$:

$$\begin{aligned}
t_a^k &\equiv sup\left\{ t \in [0,1] \middle| \sum_{P \in \mathcal{P}} x_P^{k-1} \cdot P_a + \sum_{P \in \mathcal{P}^{k-1}} \int_{t^{k-1}}^{t} s_P(v) \cdot P_a dv \le c_a \right\}, \forall a \in G^{k-1} \\
t^k &\equiv min_{a \in G^{k-1}} t_a^k \\
x_P^k &\equiv \begin{cases} x_P^{k-1} + \int_{t^{k-1}}^{t^k} s_P(v) dv, & if\ P \in \mathcal{P}^{k-1} \\ x_P^{k-1}, & otherwise \end{cases} \\
G^k &\equiv G^{k-1} \backslash \{a \in G^{k-1} | t_a^k = t^k\} \\
\mathcal{P}^k &\equiv \{P \in \mathcal{P} | a \in G^k, \forall a\ s.t.\ P_a > 0\}
\end{aligned}$$

- ***Termination condition:*** $\mathcal{P}^k = \emptyset$.

Theorem 1. *Given any problem (c,d), an allocation x is efficient if and only if there is a profile of speed functions $(s_P)_{P \in \mathcal{P}}$ such that $E^s(c,d) = x$.*

Proof. We prove sufficiency by contradiction. Suppose $x = E^s(c,d)$ is not efficient at (c,d). Then, there is another allocation $x' \neq x$ and a package $P^* \in \mathcal{P}$ such that $d_{P^*} \ge x'_{P^*} > x_{P^*}$ and $x'_P \ge x_P$ for all $P \neq P^*$. By definition of allocations, $\forall a \in G$ such that $P_a^* > 0$, we have $\sum_{P \in \mathcal{P}} x_P \cdot P_a < \sum_{P \in \mathcal{P}} x'_P \cdot P_a \le c_a$. Let k be the step in eating algorithm where $P^* \in \mathcal{P}^{k-1} \backslash \mathcal{P}^k$. It is evident that $\int_0^{t^k} s_P(v)dv \le x_P$ for all $P \in \mathcal{P}$. Hence, $t_a^k < t^k$ for all $a \in G$ such that $P_a^* > 0$. Given this, $d_{P^*} > x_{P^*}$ and $\int_0^1 s_{P^*}(v)dv = d_{P^*}$ imply $t^k < 1$, $\int_{t^k}^1 s_{P^*}(v)dv > 0$, and, hence, $t_a^k < t^k$ for all $a \in G$ such that $P_a^* > 0$. Consequently, we have $P^* \in \mathcal{P}^k$: contradiction!

We now prove necessity. Thus, let x be efficient at (c,d). Define $\overline{\mathcal{P}} = \{P \in \mathcal{P} : \exists a \in G$ s.t. $P_a > 0$, and $\sum_{P \in \mathcal{P}} x_P \cdot P_a = c_a\}$. Consider the profile of speed functions as follows.

$$\begin{aligned}
&\forall P \in \overline{\mathcal{P}} \quad && s_P(v) = \begin{cases} 2x_P, & v \in [0, 1/2) \\ 2(d_P - x_P), & v \in [1/2, 1] \end{cases} \\
&\forall P \in \mathcal{P} \backslash \overline{\mathcal{P}} \quad && s_P(v) = d_P, v \in [0,1]
\end{aligned}$$

By definition of the problems, $\overline{\mathcal{P}} \neq \emptyset$. Hence, there are two steps in the eating algorithm with $t^1 = 1/2$ and $t^2 = 1$. By definition of $\overline{\mathcal{P}}$, $\forall P \in \overline{\mathcal{P}}$, $\int_0^{1/2} s_P(v)dv = x_P$. Consequently, $E_P^s(c,d) = x_P$ for all $P \in \overline{\mathcal{P}}$. For $P \in \mathcal{P} \backslash \overline{\mathcal{P}}$, it is evident that $E_P^s(c,d) = s_P$, which completes the proof. □

Theorem 2. *Given any problem (c,d), we have three equalities as follows.*[5]

1. *$CEA(c,d) = E^s(c,d)$, where $\forall P \in \mathcal{P}$,*

$$s_P(v) = \begin{cases} max\ d, & v \in [0, d_P / max\ d] \\ 0, & v \in (d_P / max\ d, 1] \end{cases}.$$

2. *$CEL(c,d) = E^s(c,d)$, where $\forall P \in \mathcal{P}$,*

$$s_P(v) = \begin{cases} 0, & v \in [0, 1 - d_P / max\ d) \\ max\ d, & v \in [1 - d_P / max\ d, 1] \end{cases}.$$

3. *$CP(c,d) = E^s(c,d)$, where $\forall P \in \mathcal{P}$, $s_P(v) = d_P, v \in [0,1]$.*

In the above eating speed functions, $max\ d = max_{P \in \mathcal{P}} d_P$.

The proof is direct and purely mechanical. We hence omit the formal proof but provide only an explanation as follows. For CEA, the idea is equal division, subject to the constraint that the allocation of no package exceeds its demand. Hence, to represent it as a specific eating procedure, we need only to equalize the eating speed of all packages, subject to the constraint that the eating procedure of each specific package ends when the amount eaten reaches d_P. For CEL, the eating procedure is symmetric as its idea is to equalize losses. Finally, for CP, the idea is to divide proportionally to demands. Hence, it is evident that we ought to let the eating speed equal the demand.

Instead, we consider the problem in Example 1 and provide in Appendix B the verification of the equivalence with respect to CEA.

5. Conclusions

We model the multidimensional claims problem and show that the eating algorithm characterizes efficiency. Moreover, three classical rules, adapted to the current setting, are evident special cases of the eating algorithm. However, compared to the literature on the classical claims problem, an interesting problem not addressed here involves characterizations of these rules.[6]

Author Contributions: Conceptualization, S.X. and P.L.; Methodology, P.L.; Formal analysis, S.X. and P.L.; Writing—original draft, S.X. and P.L.; Writing—review & editing, S.X. and P.L.; Funding acquisition, S.X. All authors have read and agreed to the published version of the manuscript.

Funding: Sijia Xu's study was supported by National Natural Science Foundation of China: 72274062.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Appendix A. Calculations Related to Example 2

We calculate the CEA allocation for illustration.

To determine λ^1, one needs to find the largest λ with which all the following three inequalities hold.

$$\begin{aligned} SN: \quad & 0 + min\{\lambda, 450\} \cdot 1 + min\{\lambda, 200\} \cdot 1 + min\{\lambda, 450\} \cdot 1 \\ & + min\{\lambda, 100\} \cdot 0 + min\{\lambda, 800\} \cdot 0 + min\{\lambda, 500\} \cdot 0 \leq 1000 \\ NJ: \quad & 0 + min\{\lambda, 450\} \cdot 0 + min\{\lambda, 200\} \cdot 1 + min\{\lambda, 450\} \cdot 1 \\ & + min\{\lambda, 100\} \cdot 1 + min\{\lambda, 800\} \cdot 1 + min\{\lambda, 500\} \cdot 0 \leq 1000 \\ JB: \quad & 0 + min\{\lambda, 450\} \cdot 0 + min\{\lambda, 200\} \cdot 0 + min\{\lambda, 450\} \cdot 1 \\ & + min\{\lambda, 100\} \cdot 0 + min\{\lambda, 800\} \cdot 1 + min\{\lambda, 500\} \cdot 1 \leq 1000 \end{aligned}$$

One can find that $\lambda^1 = 333\frac{1}{3}$, with which the inequality with respect to JB holds with equality and the other two hold with strict inequality. This λ^1 provides x^1 and d^1 in the table as follows.

$$\begin{array}{ccccccccc} & & P^1 & P^2 & P^3 & P^4 & P^5 & P^6 & \\ x^1 = & (& 333\frac{1}{3} & 200 & 333\frac{1}{3} & 100 & 333\frac{1}{3} & 333\frac{1}{3} &) \\ d^1 = & (& 116\frac{2}{3} & 0 & 0 & 0 & 0 & 0 &) \end{array}$$

To determine λ^2, one needs to find the largest λ with which all the following three inequalities hold.

$$\begin{aligned} SN: \quad & 866\tfrac{2}{3} + min\{\lambda, 116\tfrac{2}{3}\} \cdot 1 + min\{\lambda, 0\} \cdot 1 + min\{\lambda, 0\} \cdot 1 \\ & + min\{\lambda, 0\} \cdot 0 + min\{\lambda, 0\} \cdot 0 + min\{\lambda, 0\} \cdot 0 \leq 1000 \\ NJ: \quad & 966\tfrac{2}{3} + min\{\lambda, 116\tfrac{2}{3}\} \cdot 0 + min\{\lambda, 0\} \cdot 1 + min\{\lambda, 0\} \cdot 1 \\ & + min\{\lambda, 0\} \cdot 1 + min\{\lambda, 0\} \cdot 1 + min\{\lambda, 0\} \cdot 0 \leq 1000 \\ JB: \quad & 1000 + min\{\lambda, 116\tfrac{2}{3}\} \cdot 0 + min\{\lambda, 0\} \cdot 0 + min\{\lambda, 0\} \cdot 1 \\ & + min\{\lambda, 0\} \cdot 0 + min\{\lambda, 0\} \cdot 1 + min\{\lambda, 0\} \cdot 1 \leq 1000 \end{aligned}$$

It can be verified that $\lambda^2 = \infty$, yielding x^2 and d^2 as follows. To determine $\lambda^2 = \infty$, note that JB has been exhausted in the first step; hence, the allocations of P^3, P^5, and P^6 cannot increase. Note also that the allocations of P^2 and P^4 have reached their demand. Hence, only P^1 can be increased in the second step. However, since there is no binding constraint specific to it, λ can be arbitrarily large.

$$\begin{array}{rcccccccc} & & P^1 & P^2 & P^3 & P^4 & P^5 & P^6 & \\ x^2 = & (& 450 & 200 & 333\tfrac{1}{3} & 100 & 333\tfrac{1}{3} & 333\tfrac{1}{3} &) \\ d^2 = & (& 0 & 0 & 0 & 0 & 0 & 0 &) \end{array}$$

Since $d^2_P = 0$ for all packages P, the termination condition is invoked and hence the algorithm terminates. The final outcome is $CEA(c,d) = x^2$ above.

For CEL and CP, we present only tables as follows. Detailed calculations can be provided upon request.

CEL calculations: $CEL(c,d) = x^3$

		P^1	P^2	P^3	P^4	P^5	P^6	
$\lambda^1 = 250$								
$x^1 =$	(	200	0	200	0	550	250	)
$d^1 =$	(	250	200	0	100	0	0	)
$\lambda^2 = 25$								
$x^2 =$	(	400	175	200	75	550	250	)
$d^2 =$	(	50	0	0	0	0	0	)
$\lambda^3 = 0$								
$x^3 =$	(	450	175	200	75	550	250	)
$d^3 =$	(	0	0	0	0	0	0	)

CP calculations: $CP(c,d) = x^3$

		P^1	P^2	P^3	P^4	P^5	P^6	
$\lambda^1 = \frac{1000}{7}$								
$x^1 =$	(	$\frac{1800}{7}$	$\frac{800}{7}$	$\frac{1800}{7}$	$\frac{400}{7}$	$\frac{3200}{7}$	$\frac{2000}{7}$	)
$d^1 =$	(	$\frac{1350}{7}$	$\frac{600}{7}$	0	$\frac{300}{7}$	0	0	)
$\lambda^2 = \frac{800}{7}$								
$x^2 =$	(	$\frac{3000}{7}$	$\frac{4000}{21}$	$\frac{1800}{7}$	$\frac{2000}{21}$	$\frac{3200}{7}$	$\frac{2000}{7}$	)
$d^2 =$	(	$\frac{150}{7}$	0	0	0	0	0	)
$\lambda^3 = \frac{150}{7}$								
$x^3 =$	(	450	$\frac{4000}{21}$	$\frac{1800}{7}$	$\frac{2000}{21}$	$\frac{3200}{7}$	$\frac{2000}{7}$	)
$d^3 =$	(	0	0	0	0	0	0	)

Appendix B. Verification with Respect to Theorem 2

Consider the problem in Example 1 and the speed functions in the first bullet of Theorem 2. We create as follows 6 eating functions, one for each package.

$$s_{P^1}(v)=\begin{cases}800, & v\in[0,450/800]\\ 0, & v\in(450/800,1]\end{cases} \quad s_{P^2}(v)=\begin{cases}800, & v\in[0,200/800]\\ 0, & v\in(200/800,1]\end{cases}$$
$$s_{P^3}(v)=\begin{cases}800, & v\in[0,450/800]\\ 0, & v\in(450/800,1]\end{cases} \quad s_{P^4}(v)=\begin{cases}800, & v\in[0,100/800]\\ 0, & v\in(100/800,1]\end{cases}$$
$$s_{P^5}(v)=\begin{cases}800, & v\in[0,800/800]\\ 0, & v\in(800/800,1]\end{cases} \quad s_{P^6}(v)=\begin{cases}800, & v\in[0,500/800]\\ 0, & v\in(500/800,1]\end{cases}$$

To determine the end of the first step t^1, one needs to find the following three.

$$\begin{aligned} t^1_{SN} \equiv sup\Big\{t\in[0,1]\Big|\,0+\int_0^t s_{P^1}(v)\cdot 1dv+\int_0^t s_{P^2}(v)\cdot 1dv+\int_0^t s_{P^3}(v)\cdot 1dv\\ +\int_0^t s_{P^4}(v)\cdot 0dv+\int_0^t s_{P^5}(v)\cdot 0dv+\int_0^t s_{P^6}(v)\cdot 0dv\le 1000\Big\}\\ t^1_{NJ} \equiv sup\Big\{t\in[0,1]\Big|\,0+\int_0^t s_{P^1}(v)\cdot 0dv+\int_0^t s_{P^2}(v)\cdot 1dv+\int_0^t s_{P^3}(v)\cdot 1dv\\ +\int_0^t s_{P^4}(v)\cdot 1dv+\int_0^t s_{P^5}(v)\cdot 1dv+\int_0^t s_{P^6}(v)\cdot 0dv\le 1000\Big\}\\ t^1_{JB} \equiv sup\Big\{t\in[0,1]\Big|\,0+\int_0^t s_{P^1}(v)\cdot 0dv+\int_0^t s_{P^2}(v)\cdot 0dv+\int_0^t s_{P^3}(v)\cdot 1dv\\ +\int_0^t s_{P^4}(v)\cdot 0dv+\int_0^t s_{P^5}(v)\cdot 1dv+\int_0^t s_{P^6}(v)\cdot 1dv\le 1000\Big\}\end{aligned}$$

It is easy to find $t^1_{SN}=400/800$, $t^1_{NJ}=350/800=0.4375$, and $t^1_{JB}=333\frac{1}{3}/800\approx 0.4167$. Hence, $t^1=333\frac{1}{3}/800$, with which we have

$$\begin{aligned} &x^1_{P^1}=\textstyle\int_0^{t^1} s_{P^1}(v)\cdot 1dv=333\frac{1}{3} \quad && x^1_{P^2}=\textstyle\int_0^{t^1} s_{P^2}(v)\cdot 1dv=200\\ &x^1_{P^3}=\textstyle\int_0^{t^1} s_{P^3}(v)\cdot 1dv=333\frac{1}{3} && x^1_{P^4}=\textstyle\int_0^{t^1} s_{P^4}(v)\cdot 1dv=100\\ &x^1_{P^5}=\textstyle\int_0^{t^1} s_{P^5}(v)\cdot 1dv=333\frac{1}{3} && x^1_{P^6}=\textstyle\int_0^{t^1} s_{P^6}(v)\cdot 1dv=333\frac{1}{3}\\ &G^1\equiv\{SN,NJ\}\\ &\mathcal{P}^1=\{P^1,P^2,P^4\}\end{aligned}$$

It is evident that the first step of the eating algorithm is the same as the first step of CEA. We omit the following steps to save space.

Notes

1. For more details, please refer to Example 1.
2. By convention, $\mathbb{R}$ denotes the set of real numbers, $\mathbb{R}_+$ the set of non-negative real numbers, and $\mathbb{R}_{++}$ the set of positive real numbers.
3. Throughout the paper, goods are denoted by lowercase English letters and packages uppercase letters.
4. By convention, for two vectors of same size, $x=(x_i)_{i=1}^n$ and $y=(y_i)_{i=1}^n$, $x\ge y$ means $x_i\ge y_i$ for all i; $x>y$ means $x_i\ge y_i$ for all i and $x_i>y_i$ for some i; $x\gg y$ means $x_i>y_i$ for all i.
5. For the demand vector d, $\max d\equiv\max\{d_P : P\in\mathcal{P}\}$.
6. Relevant characterizations in the classical setting can be found in the surveys Thomson (2003, 2015) and the papers cited therein.

References

Acosta, R. K., Algaba, E., & Sánchez-Soriano, J. (2022). Multi-issue bankruptcy problems with crossed claims. *Annals of Operations Research*, *318*(2), 749–772. [CrossRef]

Bogomolnaia, A., & Moulin, H. (2001). A new solution to the random assignment problem. *Journal of Economic Theory*, *100*(2), 295–328. [CrossRef]

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms* (4th ed.). MIT Press.

Liu, P., & Xu, S. (2024). Packaging for allocation. *Economics Letters*, *243*, 111932. [CrossRef]

O'Neill, B. (1982). A problem of rights arbitration from the Talmud. *Mathematical Social Sciences*, *2*(4), 345–371. [CrossRef]

Thomson, W. (2003). Axiomatic and game-theoretic analysis of bankruptcy and taxation problems: A survey. *Mathematical Social Sciences*, *45*(3), 249–297. [CrossRef]

Thomson, W. (2015). Axiomatic and game-theoretic analysis of bankruptcy and taxation problems: An update. *Mathematical Social Sciences*, *74*, 41–59. [CrossRef]