CHAPTER 15

Cost Sharing

Kamal Jain and Mohammad Mahdian

Abstract

The objective of cooperative game theory is to study ways to enforce and sustain cooperation among agents willing to cooperate. A central question in this field is how the benefits (or costs) of a joint effort can be divided among participants, taking into account individual and group incentives, as well as various fairness properties.

In this chapter, we define basic concepts and review some of the classical results in the cooperative game theory literature. Our focus is on games that are based on combinatorial optimization problems such as facility location. We define the notion of cost sharing, and explore various incentive and fairness properties cost-sharing methods are often expected to satisfy. We show how cost-sharing methods satisfying a certain property termed cross-monotonicity can be used to design mechanisms that are robust against collusion, and study the algorithmic question of designing cross-monotonic cost-sharing schemes for combinatorial optimization games. Interestingly, this problem is closely related to linear-programming-based techniques developed in the field of approximation algorithms. We explore this connection, and explain a general method for designing cross-monotonic cost-sharing schemes, as well as a technique for proving impossibility bounds on such schemes. We will also discuss an *axiomatic* approach to characterize two widely applicable solution concepts: the Shapley value for cooperative games, and the Nash bargaining solution for a more restricted framework for surplus sharing.

15.1 Cooperative Games and Cost Sharing

Consider a setting where a set $\mathcal{A}$ of n agents seek to cooperate in order to generate value. The value generated depends on the coalition S of agents cooperating. In general, the set of possible outcomes of cooperation among agents in $S \subseteq \mathcal{A}$ is denoted by $V(S)$, where each outcome is given by a vector in $\mathbb{R}^S$, whose i'th component specifies the utility that the agent $i \in S$ derives in this outcome. The set $\mathcal{A}$ of agents along with the function V defines what is called a *cooperative game* (also known as a *coalitional game*) *with nontransferable utilities* (abbreviated as an *NTU game*). A special case, called a *cooperative game with transferable utilities* (abbreviated as a *TU game*), is when the

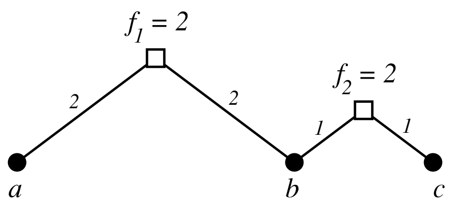

Figure 15.1. An example of the facility location game.

value generated by a coalition can be divided in an arbitrary way among the agents in S. In other words, a TU game is defined by specifying a function v: $2^{\mathcal{A}} \mapsto \mathbb{R}$, which gives the value $v(S) \in \mathbb{R}$ generated by each coalition S. We assume $v(\emptyset) = 0$. The set of all possible outcomes in such a game is defined as $V(S) = \{x \in \mathbb{R}^S \colon \sum_{i \in S} x_i \leq v(S)\}$.

The notion of a cooperative game was first proposed by von Neumann and Morgenstern. This notion seeks to abstract away all other aspects of the game except the combinatorial aspect of the coalitions that can form. This is in contrast with noncooperative games, where the focus is on the set of choices (moves) available to each agent.

Note that in the definition of a cooperative game, we did not restrict the values to be nonnegative.[1] In fact, the case that all values are nonpositive is the focus of this chapter, as it corresponds to the problem of sharing the cost of a service among those who receive the service (this is by taking the value to be the negative of the cost). Again, the cost-sharing problem can be studied in both the TU and the NTU models. The TU model applies to settings where, for example, a service provider incurs some (monetary) cost $c(S)$ in building a network that connects a set S of customers to the Internet, and needs to divide this cost among customers in S. In practice, the cost function c is often defined by solving a combinatorial optimization problem. One example, which we will use throughout the chapter, is the facility location game defined below.

Definition 15.1 In the *facility location game*, we are given a set $\mathcal{A}$ of agents (also known as cities, clients, or demand points), a set $\mathcal{F}$ of facilities, a facility opening cost f_i for every facility $i \in \mathcal{F}$, and a distance d_{ij} between every pair (i, j) of points in $\mathcal{A} \cup \mathcal{F}$ indicating the cost of connecting j to i. We assume that the distances come from a metric space; i.e., they are symmetric and obey the triangle inequality. For a set $S \subseteq \mathcal{A}$ of agents, the cost of this set is defined as the minimum cost of opening a set of facilities and connecting every agent in S to an open facility. More precisely, the cost function c is defined by $c(S) = \min_{\mathcal{F}' \subseteq \mathcal{F}} \{\sum_{i \in \mathcal{F}'} f_i + \sum_{j \in S} \min_{i \in \mathcal{F}'} d_{ij}\}$.

Example 15.2 Figure 15.1 shows an instance of the facility location game with 3 agents $\{a, b, c\}$ and 2 facilities $\{1, 2\}$. The distances between some pairs are marked in the figure, and other distances can be calculated using the triangle

[1] If all values are nonnegative, the problem is called a *surplus sharing* problem.

inequality (e.g., the distance between facility 1 and client c is $2 + 1 + 1 = 4$). The cost function defined by this instance is the following:

$$c(\{a\}) = 4, \quad c(\{b\}) = 3, \quad c(\{c\}) = 3,$$
$$c(\{a, b\}) = 6, \quad c(\{b, c\}) = 4, \quad c(\{a, c\}) = 7, \quad c(\{a, b, c\}) = 8.$$

Since the monetary cost may be distributed arbitrarily among the agents, it is natural to model the above example as a TU game. An example of a case where the NTU model is more applicable is a network design problem where the cost incurred by an agent i in the set of agents S receiving the service corresponds to the delay this agent suffers. There are multiple designs for the network connecting the customers in S, and each design corresponds to a profile of delays that these agents will suffer. The set of possible outcomes for the coalition S is defined as the collection of all such profiles, and is denoted by $C(S)$. As delays are nontransferrable, this setting is best modeled as an NTU cost-sharing game. For another example of an NTU game, see the housing allocation problem in Section 10.3 of this book.

As most of the work on cost sharing in the algorithmic game theory literature has so far focused on TU games, this chapter is mainly devoted to such games; henceforth, by a cost-sharing game we mean a TU game, unless otherwise stated.

15.2 Core of Cost-Sharing Games

A central notion in cooperative game theory is the notion of *core*. Roughly speaking, the core of a cooperative game is an outcome of cooperation among all agents where no coalition of agents can all benefit by breaking away from the grand coalition. Intuitively, the core of a game corresponds to situations where it is possible to sustain cooperation among all agents in an economically stable manner.

In this section, we define the notion of core for cost-sharing games, and present two classical results on conditions for nonemptiness of the core. We show how the notion of core for TU games can be relaxed to an approximate version suitable for hard combinatorial optimization games, and observe a connection between this notion and the integrality gap of a linear programming relaxation of such problems.

15.2.1 Core of TU Games

Formally, the core of a TU cost-sharing game is defined as follows.

Definition 15.3 Let $(\mathcal{A}, c)$ be a TU cost-sharing game. A vector $\boldsymbol{\alpha} \in \mathbb{R}^{\mathcal{A}}$ (sometimes called a *cost allocation*) is in the *core* of this game if it satisfies the following two conditions:

- **Budget balance:** $\sum_{j \in \mathcal{A}} \alpha_j = c(\mathcal{A})$.
- **Core property:** for every $S \subseteq \mathcal{A}$, $\sum_{j \in S} \alpha_j \leq c(S)$.

Example 15.4 As an example, consider the facility location game of Example 15.2 (Figure 15.1). It is not hard to verify that the vector $(4, 2, 2)$ lies in the core of this game. In fact, this is not the only cost allocation in the core of this game; for example, (4, 1, 3) is also in the core. On the other hand, if a third facility with opening cost 3 and distance 1 to agents a and c is added to this game, the resulting game will have an empty core. To see this, note that after adding the third facility, we have $c(\{a, c\}) = 5$. Now, if there is a vector $\boldsymbol{\alpha}$ in the core of this game, we must have

$$\begin{aligned}\alpha_a + \alpha_b &\leq c(\{a, b\}) = 6\\ \alpha_b + \alpha_c &\leq c(\{b, c\}) = 4\\ \alpha_a + \alpha_c &\leq c(\{a, c\}) = 5\end{aligned}$$

By adding the above three inequalities and dividing both sides by 2, we obtain $\alpha_a + \alpha_b + \alpha_c \leq 7.5 < c(\{a, b, c\})$. Therefore $\boldsymbol{\alpha}$ cannot be budget balanced.

A classical result in cooperative game theory, known as the Bondareva–Shapley theorem, gives a necessary and sufficient condition for a game to have nonempty core. To state this theorem, we need the following definition.

Definition 15.5 A vector $\boldsymbol{\lambda}$ that assigns a nonnegative weight λ_S to each subset $S \subseteq \mathcal{A}$ is called a *balanced collection of weights* if for every $j \in \mathcal{A}$, $\sum_{S:j\in S} \lambda_S = 1$.

Theorem 15.6 *A cost-sharing game* $(\mathcal{A}, c)$ *with transferable utilities has a nonempty core if and only if for every balanced collection of weights* $\boldsymbol{\lambda}$*, we have* $\sum_{S\subseteq\mathcal{A}} \lambda_S c(S) \geq c(\mathcal{A})$.

PROOF By the definition of the core, the game $(\mathcal{A}, c)$ has a nonempty core if and only if the solution of the following linear program (LP) is precisely $c(\mathcal{A})$. (Note that this solution can never be larger than $c(\mathcal{A})$.)

$$\begin{aligned}&\text{Maximize} && \textstyle\sum_{j\in\mathcal{A}} \alpha_j\\ &\text{Subject to} && \forall S \subseteq \mathcal{A} : \textstyle\sum_{j\in S} \alpha_j \leq c(S).\end{aligned} \tag{15.1}$$

By strong LP duality, the solution of the above LP is equal to the solution of the following dual program:

$$\begin{aligned}&\text{Minimize} && \sum_{S\subseteq\mathcal{A}} \lambda_S c(S)\\ &\text{Subject to} && \forall j \in \mathcal{A} : \sum_{S:j\in S} \lambda_S = 1\\ & && \forall S \subseteq \mathcal{A} : \lambda_S \geq 0.\end{aligned} \tag{15.2}$$

Therefore, the core of the game is nonempty if and only if the solution of the LP (15.2) is equal to $c(\mathcal{A})$. By definition, feasible solutions of this program are balanced collections of weights. Therefore, the core of the game $(\mathcal{A}, c)$ is nonempty if and only if for every balanced collection of weights (λ_S), $\sum_{S\subseteq\mathcal{A}} \lambda_S c(S) \geq c(\mathcal{A})$. □

As an example, the proof of emptiness of the core given in Example 15.4 can be restated by defining a vector $\boldsymbol{\lambda}$ as follows: $\lambda_{\{a,b\}} = \lambda_{\{b,c\}} = \lambda_{\{a,c\}} = \frac{1}{2}$ and $\lambda_S = 0$ for every other set S. It is easy to verify that $\boldsymbol{\lambda}$ is a balanced collection of weights and $\sum_{S \subseteq \mathcal{A}} \lambda_S c(S) < c(\mathcal{A})$.

15.2.2 Approximate Core

As we saw in Example 15.4, a difficulty with the notion of core is that the core of many cost-sharing games, including most combinatorial optimization games based on computationally hard problems, is often empty. Furthermore, when the underlying cost function is hard to compute (e.g., in the facility location game), even deciding whether the core of the game is empty is often computationally intractable. This motivates the following definition.

Definition 15.7 A vector $\boldsymbol{\alpha} \in \mathbb{R}^{\mathcal{A}}$ is in the γ-approximate core (or γ-core, for short) of the game $(\mathcal{A}, c)$ if it satisfies the following two conditions:

- **γ-Budget balance:** $\gamma c(\mathcal{A}) \leq \sum_{j \in \mathcal{A}} \alpha_j \leq c(\mathcal{A})$.
- **Core property:** for every $S \subseteq \mathcal{A}$, $\sum_{j \in S} \alpha_j \leq c(S)$.

For example, in the facility location game given in Example 15.4, the vector $(3.5, 2.5, 1.5)$ is in the $\frac{7.5}{8}$-core of the game. Note that the argument given to show the emptiness of the core of this game actually proves that for every $\gamma > \frac{7.5}{8}$, the γ-core of this game is empty.

The Bondareva–Shapley theorem can be easily generalized to the following approximate version.

Theorem 15.8 *For every $\gamma \leq 1$, a cost-sharing game $(\mathcal{A}, c)$ with transferable utilities has a nonempty γ-core if and only if for every balanced collection of weights $\boldsymbol{\lambda}$, we have $\sum_{S \subseteq \mathcal{A}} \lambda_S c(S) \geq \gamma c(\mathcal{A})$.*

The proof is similar to the proof of the Bondareva–Shapley theorem and is based on the observation that by LP duality, the γ-core of the game is nonempty if and only if the solution of the LP (15.2) is at least $\gamma c(S)$. Note that if the cost function c is subadditive (i.e., $c(S_1 \cup S_2) \leq c(S_1) + c(S_2)$ for any two disjoint sets S_1 and S_2), then the optimal *integral* solution of the LP (15.2) is precisely $c(\mathcal{A})$. Therefore,

Corollary 15.9 *For any cost-sharing game $(\mathcal{A}, c)$ with a subadditive cost function, the largest value γ for which the γ-core of this game is nonempty is equal to the integrality gap of the LP (15.2).*

As it turns out, for many combinatorial optimization games such as set cover, vertex cover, and facility location, the LP formulation (15.2) is in fact equivalent to the standard (polynomial-size) LP formulation of the problem, and hence Corollary 15.9 translates into a statement about the integrality gap of the standard LP formulation of the problem. Here, we show this connection for the facility location game.

We start by formulating the facility location problem as an integer program. In this formulation, x_i and y_{ij} are variables indicating whether facility i is open, and whether agent j is connected to facility i.

$$\begin{aligned} &\text{Minimize} && \textstyle\sum_{i\in\mathcal{F}} f_i x_i + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{A}} d_{ij} y_{ij} \\ &\text{Subject to} && \forall j \in \mathcal{A}:\ \textstyle\sum_{i\in\mathcal{F}} y_{ij} \geq 1 \\ & && \forall i \in \mathcal{F}, j \in \mathcal{A}:\ x_i \geq y_{ij} \\ & && \forall i \in \mathcal{F}, j \in \mathcal{A}:\ x_i, y_{ij} \in \{0,1\}. \end{aligned} \tag{15.3}$$

By relaxing the second constraint to $x_i, y_{ij} \geq 0$ we obtain an LP whose dual can be written as follows:

$$\begin{aligned} &\text{Maximize} && \textstyle\sum_{j\in\mathcal{A}} \alpha_j \\ &\text{Subject to} && \forall i \in \mathcal{F}, j \in \mathcal{A}:\ \beta_{ij} \geq \alpha_j - d_{ij} \\ & && \forall i \in \mathcal{F}:\ \textstyle\sum_{j\in\mathcal{A}} \beta_{ij} \leq f_i \\ & && \forall i \in \mathcal{F}, j \in \mathcal{A}:\ \alpha_j, \beta_{ij} \geq 0 \end{aligned} \tag{15.4}$$

Note that we may assume without loss of generality that in a feasible solution of the above LP, $\beta_{ij} = \max(0, \alpha_j - d_{ij})$. Thus, to specify a dual solution it is enough to give $\boldsymbol{\alpha}$. We now observe that the above dual LP captures the core constraint of the facility location game; i.e., it is equivalent to the LP (15.1).

Proposition 15.10 *For any feasible solution $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of the LP (15.4), $\boldsymbol{\alpha}$ satisfies the core property of the facility location game.*

PROOF We need to show that for every set $S \subseteq \mathcal{A}$, $\sum_{j\in S} \alpha_j \leq c(S)$, where $c(S)$ is the cost of the facility location problem for agents in S. First we note that for any facility i and set of agents $R \subseteq \mathcal{A}$, by adding the first and the second inequalities of the LP (15.4) for facility i and every $j \in R$ we obtain

$$\sum_{j\in R} \alpha_j \leq f_i + \sum_{j\in R} d_{ij}. \tag{15.5}$$

Now, consider an optimal solution for the set of agents S, and assume $i_1, \ldots, i_k$ are facilities that are open and R_ℓ is the set of agents served by facility i_ℓ in this solution. Summing Inequality (15.5) for every (i_ℓ, R_ℓ) will yield the result. □

By the above proposition, the solution of the dual LP (15.4) (which, by LP duality, is the same as the LP relaxation of (15.3)) is equal to the solution of the LPs (15.1) and (15.2). Furthermore, the optimal integral solution of (15.3) is $c(\mathcal{A})$. Therefore, the integrality gap of (15.3) is the same as that of (15.2) and gives the best budget balance factor that a cost allocation satisfying the core property can achieve. The best known results in the field of approximation algorithms show that this gap (in the worst case) is between $\frac{1}{1.52}$ and $\frac{1}{1.463}$.

15.2.3 Core of NTU Games

We conclude this section with a classical theorem due to Scarf, which gives a sufficient condition for the nonemptiness of the core of NTU games similar to the one given in Theorem 15.6 for TU games. However, unlike in the case of TU games, the condition given in Scarf's theorem is not necessary for the nonemptiness of the core.

Formally, the core of an NTU cost-sharing game $(\mathcal{A}, C)$ is the collection of all cost allocations $\boldsymbol{\alpha} \in C(\mathcal{A})$ such that there is no nonempty coalition $S \subseteq \mathcal{A}$ and cost allocation $\mathbf{x} \in C(S)$ for which $x_j < \alpha_j$ for all $j \in S$. Note that this definition coincides with Definition 15.3 in the case of a TU game. In the following theorem the *support* of a balanced collection of weights $\boldsymbol{\lambda}$ denotes the collection of all sets S with $\lambda_S > 0$.

Theorem 15.11 *Let $(\mathcal{A}, C)$ be a cost-sharing game with nontransferable utilities. Assume for every balanced collection of weights $\boldsymbol{\lambda}$ and every vector $\mathbf{x} \in \mathbb{R}^{\mathcal{A}}$ the following property holds: if for every set S in the support of $\boldsymbol{\lambda}$, the restriction of $\mathbf{x}$ to the coordinates in S is in $C(S)$, then $\mathbf{x} \in C(\mathcal{A})$. Then $(\mathcal{A}, C)$ has a nonempty core.*

The proof of the above theorem, which is beyond the scope of this chapter, uses an adaptation of the Lemke–Howson algorithm for computing Nash equilibria (described in Section 3.4 of this book), and is considered an early and important contribution of the algorithmic viewpoint in game theory. However, the worst case running time of this algorithm (like the Lemke–Howson algorithm) is exponential in $|\mathcal{A}|$. This is in contrast to the proof of the Bondareva–Shapley theorem, which gives a polynomial-time algorithm[2] for computing a point in the core of the game, if the core is nonempty.

15.3 Group-Strategyproof Mechanisms and Cross-Monotonic Cost-Sharing Schemes

The cost-sharing problem defined in this chapter models the pricing problem for a service provider with a given set of customers. In settings where the demand is sensitive to the price, an alternative choice for the service provider is to conduct an auction between the potential customers to select the set of customers who can receive the service based on their willingness to pay and the cost structure of the problem. The goal is to design an auction mechanism that provides incentives for individuals as well as groups of agents to bid truthfully. In this section, we study this problem and exhibit its connection to cost sharing.

We start with the definition of the setting. Let $\mathcal{A}$ be a set of n agents interested in receiving a service. The cost of providing service is given by a cost function c: $2^{\mathcal{A}} \mapsto \mathbb{R}^+ \cup \{0\}$, where $c(S)$ specifies the cost of providing service for agents in S. Each agent i has a value $u_i \in \mathbb{R}$ for receiving the service; that is, she is willing to pay at most u_i to get the service. We further assume that the utility of agent i is given by $u_i q_i - x_i$,

[2] This is assuming a suitable representation for the cost function c, e.g., by a separation oracle for (15.1), or in combinatorial optimization games where statements like Proposition 15.10 hold.

where q_i is an indicator variable that indicates whether she has received the service or not, and x_i is the amount she has to pay. A *cost-sharing mechanism* is an algorithm that elicits a bid $b_i \in \mathbb{R}$ from each agent, and based on these bids, decides which agents should receive the service and how much each of them has to pay. More formally, a cost-sharing mechanism is a function that associates to each vector $\mathbf{b}$ of bids a set $Q(\mathbf{b}) \subseteq \mathcal{A}$ of agents to be serviced, and a vector $\mathbf{p}(\mathbf{b}) \in \mathbb{R}^n$ of payments. When there is no ambiguity, we write Q and $\mathbf{p}$ instead of $Q(\mathbf{b})$ and $\mathbf{p}(\mathbf{b})$, respectively. We assume that a mechanism satisfies the following conditions:

- *No Positive Transfer (NPT)*: The payments are nonnegative (i.e., $p_i \geq 0$ for all i).
- *Voluntary Participation (VP)*: An agent who does not receive the service is not charged (i.e., $p_i = 0$ for $i \notin Q$), and an agent who receives the service is not charged more than her bid (i.e., $p_i \leq b_i$ for $i \in Q$)
- *Consumer Sovereignty (CS)*: For each agent i, there is some bid b_i^* such that if i bids b_i^*, she will get the service, no matter what others bid.

Furthermore, we want the mechanisms to be approximately budget-balanced. We call a mechanism γ-budget-balanced with respect to the cost function c if the total amount the mechanism charges the agents is between $\gamma c(Q)$ and $c(Q)$ (i.e., $\gamma c(Q) \leq \sum_{i \in Q} x_i \leq c(Q)$).

We look for mechanisms, called *group strategyproof mechanisms*, which satisfy the following property in addition to NPT, VP, and CS. Let $S \subseteq \mathcal{A}$ be a coalition of agents, and $\mathbf{u}, \mathbf{u}'$ be two vectors of bids satisfying $u_i = u_i'$ for every $i \notin S$ (we think of $\mathbf{u}$ as the values of agents, and $\mathbf{u}'$ as a vector of strategically chosen bids). Let $(Q, \mathbf{p})$ and $(Q', \mathbf{p}')$ denote the outputs of the mechanism when the bids are $\mathbf{u}$ and $\mathbf{u}'$, respectively. A mechanism is *group strategyproof* if for every coalition S of agents, if the inequality $u_i q_i' - p_i' \geq u_i q_i - p_i$ holds for every $i \in S$, then it holds with equality for every $i \in S$. In other words, there should not be any coalition S and vector $\mathbf{u}'$ of bids such that if members of S announce $\mathbf{u}'$ instead of $\mathbf{u}$ (their true value) as their bids, then every member of the coalition S is at least as happy as in the truthful scenario, and at least one person is happier.

Moulin showed that cost-sharing methods satisfying an additional property termed *cross-monotonicity* can be used to design group-strategyproof cost-sharing mechanisms. The cross-monotonicity property captures the notion that agents should not be penalized as the serviced set grows. To define this property, we first need to define the notion of a cost-sharing *scheme*.

Definition 15.12 Let $(\mathcal{A}, c)$ denote a cost-sharing game. A *cost-sharing scheme* is a function that for each set $S \subseteq \mathcal{A}$, assigns a cost allocation for S. More formally, a cost-sharing scheme is a function $\xi \colon \mathcal{A} \times 2^{\mathcal{A}} \mapsto \mathbb{R}$ such that, for every $S \subseteq \mathcal{A}$ and every $i \notin S$, $\xi(i, S) = 0$. We say that a cost-sharing scheme ξ is γ-budget balanced if for every set $S \subseteq \mathcal{A}$, we have $\gamma c(S) \leq \sum_{i \in S} \xi(i, S) \leq c(S)$.

Definition 15.13 A cost-sharing scheme ξ is *cross-monotone* if for all $S, T \subseteq \mathcal{A}$ and $i \in S$, $\xi(i, S) \geq \xi(i, S \cup T)$.

```
Mechanism $\mathcal{M}_\xi$
    Initialize $S \leftarrow \mathcal{A}$.
    Repeat
        Let $S \leftarrow \{i \in S : b_i \geq \xi(i, S)\}$.
    Until for all $i \in S, b_i \geq \xi(i, S)$.
    Return $Q = S$ and $p_i = \xi(i, S)$ for all $i$.
```

Figure 15.2. Moulin's group-strategyproof mechanism.

The following proposition shows that cross-monotonicity is a stronger property than core.

Proposition 15.14 *Let ξ be an γ-budget-balanced cross-monotonic cost sharing scheme for the cost-sharing game $(\mathcal{A}, c)$. Then $\xi(., \mathcal{A})$ is in the γ-core of this game.*

PROOF We need to verify only that $\xi(., \mathcal{A})$ satisfies the core property, i.e., for every set $S \subseteq \mathcal{A}$, $\sum_{i \in S} \xi(i, \mathcal{A}) \leq c(S)$. By the cross-monotonicity property, for every $i \in S$, $\xi(i, \mathcal{A}) \leq \xi(i, S)$. Therefore, $\sum_{i \in S} \xi(i, \mathcal{A}) \leq \sum_{i \in S} \xi(i, S) \leq c(S)$, where the last inequality follows from the γ-budget balance property of ξ. □

Given a cross-monotonic cost-sharing scheme ξ for the cost-sharing game $(\mathcal{A}, c)$, we define a cost-sharing mechanism $\mathcal{M}_\xi$ as presented in Figure 15.2.

The following proposition provides an alternative way to view the mechanism $\mathcal{M}_\xi$.

Proposition 15.15 *Assume ξ is a cross-monotonic cost sharing scheme for the cost-sharing game $(\mathcal{A}, c)$, and $b_i \in \mathbb{R}^+ \cup \{0\}$ for every $i \in \mathcal{A}$. Then there is a unique maximal set $S \subseteq \mathcal{A}$ satisfying the property that for every $i \in S$, $b_i \geq \xi(i, S)$. The mechanism $\mathcal{M}_\xi$ returns this set.*

PROOF Assume that two different maximal sets S_1 and S_2 satisfy the stated property, i.e., $b_i \geq \xi(i, S_1)$ for every $i \in S_1$ and $b_i \geq \xi(i, S_2)$ for every $i \in S_2$. Then for every $i \in S_1$, $b_i \geq \xi(i, S_1) \geq \xi(i, S_1 \cup S_2)$, where the last inequality follows from cross-monotonicity of ξ. Similarly, for every $i \in S_2$, $b_i \geq \xi(i, S_1 \cup S_2)$. Therefore, the set $S_1 \cup S_2$ also satisfies this property. This contradicts with the maximality of S_1 and S_2.

Let S^* denote the unique maximal set satisfying $b_i \geq \xi(i, S^*)$ for all $i \in S^*$. We claim that $\mathcal{M}_\xi$ never eliminates any of the agents in S^* from the serviced set S. Consider, for contradiction, the first step where it eliminates an agent $i \in S^*$ from the serviced set S. This means that we must have $b_i < \xi(i, S)$. However, since $S^* \subseteq S$, by cross-monotonicity we have $\xi(i, S) \leq \xi(i, S^*)$, and hence $b_i < \xi(i, S^*)$, contradicting the definition of S^*. Therefore, the set Q returned by $\mathcal{M}_\xi$ contains S^*. By maximality of S^*, it cannot contain any other agent, that is, $Q = S^*$. □

We are now ready to prove the following theorem of Moulin.

Theorem 15.16 *If ξ is an γ-budget-balanced cross-monotonic cost-sharing scheme, then $\mathcal{M}_\xi$ is group-strategyproof and γ-budget balanced.*

PROOF Assume, for contradiction, that there is a coalition T of agents that benefits from bidding according to the vector $\mathbf{u}'$ instead of their true values $\mathbf{u}$. Agents in T can be partitioned into two sets T^+ and T^- based on whether their bid in $\mathbf{u}'$ is greater than their bid in $\mathbf{u}$, or not. First, we claim that it can be assumed, without loss of generality, that T^+ is empty, i.e., agents cannot benefit from overbidding. To see this, we start from a bid vector where every agent in T bids according to $\mathbf{u}'$ (and others bid truthfully), and reduce the bids of the agents in T^+ to their true value one by one. If at any step, e.g., when the bid of agent $i \in T^+$ is reduced from u'_i to u_i, the outcome of the auction changes, then by Proposition 15.15, i must be a winner when she bids according to $\mathbf{u}'$, and not a winner when she bids according to $\mathbf{u}$. This means that $u'_i \geq \xi(i, S_i) > u_i$, where S_i is the set of winners when i bids according to $\mathbf{u}'$. However, this means that the agent i must pay an amount greater than her true value in the scenario where every agent in T bids according to $\mathbf{u}'$. This is in contradiction with the assumption that agents in T all benefit from the collusion. By this argument, the bid of every agent in T^+ can be lowered to her true value without changing the outcome of the auction. Therefore, we assume without loss of generality that T^+ is empty.

Now, let S' and S denote the set of winners in the untruthful and the truthful scenarios (i.e., when agents bid according to $\mathbf{u}'$ and $\mathbf{u}$), respectively. As the bid of each agent in $\mathbf{u}'$ is less than or equal to her bid in $\mathbf{u}$, by Proposition 15.15, $S' \subseteq S$. By cross-monotonicity, this implies that the payment of each agent in the untruthful scenario is at least as much as her payment in the truthful scenario. Therefore, no agent can be strictly happier in the untruthful scenario than in the truthful scenario. □

Moulin's theorem shows that cross-monotonic cost-sharing schemes give rise to group-strategyproof mechanisms. An interesting question is whether the converse also holds, i.e., is there a way to construct a cross-monotonic cost-sharing scheme given a group-strategyproof mechanism? The answer to this question is negative (unless the cost function is assumed to be submodular), as there are examples where a cost function has a group-strategyproof mechanism but no cross-monotonic cost-sharing scheme. A partial characterization of the cost-sharing schemes that correspond to group-strategyproof mechanisms in terms of a property called semi-cross-monotonicity is known; however, finding a complete characterization of cost-sharing schemes arising from group-strategyproof mechanisms remains an open question.

15.4 Cost Sharing via the Primal-Dual Schema

In Section 15.2.2, we discussed how a cost allocation in the approximate core of a game can be computed by solving an LP, and noted that for many combinatorial optimization games, this LP is equivalent to the dual of the standard LP relaxation of the problem and the cost shares correspond to the dual variables. In this section, we explain how a technique called the *primal-dual schema* can be used to compute cost shares that not only are in the approximate core of the game, but also satisfy the cross-monotonicity property, and hence can be used in the mechanism described in the previous section. The primal-dual schema is a standard technique in the field of approximation algorithms, where the focus is on computing an approximately optimal primal solution, and the dual variables (cost shares) are merely a by-product of the algorithm.

The idea of the primal-dual schema, which is often used to solve cost minimization problems, is to write the optimization problem as a mathematical program that can be relaxed into an LP (we refer to this LP as the primal LP). The dual of this LP gives a lower bound on the value of the optimal solution for the problem. Primal-dual algorithms simultaneously construct a solution to the primal problem and its dual. This is generally done by initially setting all dual variables to zero, and then gradually increasing these variables until some constraint in the dual program goes tight. This constraint hints at an object that can be *paid for* by the dual to be included in the primal solution. After this, the dual variables involved in the tight constraint are *frozen*, and the algorithm continues by increasing other dual variables. The algorithm ends when a complete solution for the primal problem is constructed. The analysis is based on proving that the values of the constructed primal and dual solutions are close to each other, and therefore they are both close to optimal.[3]

We will elaborate on two examples in this section: submodular games, where a simple primal-dual algorithm with no modification yields cross-monotonic cost-shares, and the facility location game, where extra care needs to be taken to obtain a cross-monotonic cost-sharing scheme. In the latter case, we introduce a rather general technique of using "ghost duals" to turn the standard primal-dual algorithm for the problem into an algorithm that returns a cross-monotonic cost-sharing scheme.

15.4.1 Submodular Games

Let us start with a definition of submodular games.

Definition 15.17 A cost-sharing game $(\mathcal{A}, c)$ is called a *submodular game* if the cost function c satisfies

$$\forall S, T \subseteq \mathcal{A}, \quad c(S) + c(T) \geq c(S \cup T) + c(S \cap T).$$

The above condition is equivalent to the condition of decreasing marginal cost, which says that for every two agents i and j and every set of agents $S \subset \mathcal{A} \setminus \{i, j\}$, the marginal cost of adding i to S (i.e., $c(S \cup \{i\}) - c(S)$) is no less than the marginal cost of adding i to $S \cup \{j\}$ (i.e., $c(S \cup \{i, j\}) - c(S \cup \{j\})$). Recall that we always assume $c(\emptyset) = 0$.

[3] In many primal-dual algorithms, a postprocessing step is required to bring the cost of the primal solution down. However, this step often does not change the cost shares.