Marginal Contribution Nets: A Compact Representation Scheme for Coalitional Games*

Samuel Ieong†
Computer Science Department
Stanford University
Stanford, CA 94305
sieong@stanford.edu

Yoav Shoham
Computer Science Department
Stanford University
Stanford, CA 94305
shoham@stanford.edu

ABSTRACT

We present a new approach to representing coalitional games based on rules that describe the marginal contributions of the agents. This representation scheme captures characteristics of the interactions among the agents in a natural and concise manner. We also develop efficient algorithms for two of the most important solution concepts, the Shapley value and the core, under this representation. The Shapley value can be computed in time linear in the size of the input. The emptiness of the core can be determined in time exponential only in the treewidth of a graphical interpretation of our representation.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; J.4 [**Social and Behavioral Sciences**]: Economics; F.2 [**Analysis of Algorithms and Problem Complexity**]

General Terms

Algorithms, Economics

Keywords

Coalitional game theory, Representation, Treewidth

1. INTRODUCTION

Agents can often benefit by coordinating their actions. Coalitional games capture these opportunities of coordination by explicitly modeling the ability of the agents to take joint actions as primitives. As an abstraction, coalitional games assign a payoff to each group of agents in the game. This payoff is intended to reflect the payoff the group of agents can secure for themselves regardless of the actions of the agents not in the group. These choices of primitives are in contrast to those of non-cooperative games, of which agents are modeled independently, and their payoffs depend critically on the actions chosen by the other agents.

*This research is supported by NSF grant ITR0205633.

†Samuel Ieong is supported by a Richard and Naomi Horowitz Stanford Graduate Fellowship.

EC'05, June 5–8, 2005, Vancouver, British Columbia, Canada.

1.1 Coalitional Games and E-Commerce

Coalitional games have appeared in the context of e-commerce. In [7], Kleinberg *et al.* use coalitional games to study recommendation systems. In their model, each individual knows about a certain set of items, is interested in learning about all items, and benefits from finding out about them. The payoffs to groups of agents are the total number of distinct items known by its members. Given this coalitional game setting, Kleinberg *et al.* compute the value of the private information of the agents is worth to the system using the solution concept of the Shapley value (definition can be found in section 2). These values can then be used to determine how much each agent should receive for participating in the system.

As another example, consider the economics behind supply chain formation. The increased use of the Internet as a medium for conducting business has decreased the costs for companies to coordinate their actions, and therefore coalitional game is a good model for studying the supply chain problem. Suppose that each manufacturer purchases his raw materials from some set of suppliers, and that the suppliers offer higher discount with more purchases. The decrease in communication costs will let manufacturers find others interested in the same set of suppliers cheaper, and facilitates formation of coalitions to bargain with the suppliers. Depending on the set of suppliers and how much from each supplier each coalition purchases, we can assign payoffs to the coalitions depending on the discount it receives. The resulting game can be analyzed using coalitional game theory, and we can answer questions such as the stability of coalitions, and how to fairly divide the benefits among the participating manufacturers. A similar problem, combinatorial coalition formation, has previously been studied in [8].

1.2 Evaluation Criteria for Coalitional Game Representation

To capture the coalitional games described above and perform computations on them, we must first find a representation for these games. The naïve solution is to enumerate the payoffs to each set of agents, therefore requiring space

exponential in the number of agents in the game. For the two applications described, the number of agents in the system can easily exceed a hundred; this naïve approach will not be scalable to such problems. Therefore, it is critical to find good representation schemes for coalitional games.

We believe that the quality of a representation scheme should be evaluated by four criteria.

Expressivity: the breadth of the class of coalitional games covered by the representation.

Conciseness: the space requirement of the representation.

Efficiency: the efficiency of the algorithms we can develop for the representation.

Simplicity: the ease of use of the representation by users of the system.

The ideal representation should be fully expressive, i.e., it should be able to represent any coalitional games, use as little space as possible, have efficient algorithms for computation, and be easy to use. The goal of this paper is to develop a representation scheme that has properties close to the ideal representation.

Unfortunately, given that the number of degrees of freedom of coalitional games is $O(2^n)$, not all games can be represented concisely using a single scheme due to information theoretic constraints. For any given class of games, one may be able to develop a representation scheme that is tailored and more compact than a general scheme. For example, for the recommendation system game, a highly compact representation would be one that simply states which agents know of which products, and let the algorithms that operate on the representation to compute the values of coalitions appropriately. For some problems, however, there may not be efficient algorithms for customized representations. By having a general representation and efficient algorithms that go with it, the representation will be useful as a prototyping tool for studying new economic situations.

1.3 Previous Work

The question of coalitional game representation has only been sparsely explored in the past [2, 3, 4]. In [4], Deng and Papadimitriou focused on the complexity of different solution concepts on coalitional games defined on graphs. While the representation is compact, it is not fully expressive. In [2], Conitzer and Sandholm looked into the problem of determining the emptiness of the core in superadditive games. They developed a compact representation scheme for such games, but again the representation is not fully expressive either. In [3], Conitzer and Sandholm developed a fully expressive representation scheme based on decomposition. Our work extends and generalizes the representation schemes in [3, 4] through decomposing the game into a set of rules that assign marginal contributions to groups of agents. We will give a more detailed review of these papers in section 2.2 after covering the technical background.

1.4 Summary of Our Contributions

- We develop the *marginal contribution networks* representation, a fully expressive representation scheme whose size scales according to the complexity of the interactions among the agents. We believe that the representation is also simple and intuitive.
- We develop an algorithm for computing the Shapley value of coalitional games under this representation that runs in time linear in the size of the input.
- Under the graphical interpretation of the representation, we develop an algorithm for determining the whether a payoff vector is in the core and the emptiness of the core in time exponential only in the treewidth of the graph.

2. PRELIMINARIES

In this section, we will briefly review the basics of coalitional game theory and its two primary solution concepts, the Shapley value and the core.[1] We will also review previous work on coalitional game representation in more detail. Throughout this paper, we will assume that the payoff to a group of agents can be freely distributed among its members. This assumption is often known as the *transferable utility* assumption.

2.1 Technical Background

We can represent a coalition game with transferable utility by the pair $\langle N, v\rangle$, where

- N is the set of agents; and
- $v : 2^N \mapsto \mathbb{R}$ is a function that maps each group of agents $S \subseteq N$ to a real-valued payoff.

This representation is known as the characteristic form. As there are exponentially many subsets, it will take space exponential in the number of agents to describe a coalitional game.

An *outcome* in a coalitional game specifies the utilities the agents receive. A *solution concept* assigns to each coalitional game a set of "reasonable" outcomes. Different solution concepts attempt to capture in some way outcomes that are stable and/or fair. Two of the best known solution concepts are the Shapley value and the core.

The Shapley value is a normative solution concept. It prescribes a "fair" way to divide the gains from cooperation when the grand coalition (i.e., N) is formed. The division of payoff to agent i is the average marginal contribution of agent i over all possible permutations of the agents. Formally, let $\phi_i(v)$ denote the Shapley value of i under characteristic function v, then[2]

$$\phi_i(v) = \sum_{S \subset N} \frac{s!(n-s-1)!}{n!} \left(v(S \cup \{i\}) - v(S)\right) \qquad (1)$$

The Shapley value is a solution concept that satisfies many nice properties, and has been studied extensively in the economic and game theoretic literature. It has a very useful axiomatic characterization.

Efficiency (EFF) A total of $v(N)$ is distributed to the agents, i.e., $\sum_{i \in N} \phi_i(v) = v(N)$.

Symmetry (SYM) If agents i and j are interchangeable, then $\phi_i(v) = \phi_j(v)$.

[1] The materials and terminology are based on the textbooks by Mas-Colell *et al.* [9] and Osborne and Rubinstein [11].

[2] As a notational convenience, we will use the lower-case letter to represent the cardinality of a set denoted by the corresponding upper-case letter.

Dummy (DUM) If agent i is a dummy player, i.e., his marginal contribution to all groups S are the same, $\phi_i(v) = v(\{i\})$.

Additivity (ADD) For any two coalitional games v and w defined over the same set of agents N, $\phi_i(v + w) = \phi_i(v) + \phi_i(w)$ for all $i \in N$, where the game $v + w$ is defined as $(v + w)(S) = v(S) + w(S)$ for all $S \subseteq N$.

We will refer to these axioms later in our proof of correctness of the algorithm for computing the Shapley value under our representation in section 4.

The core is another major solution concept for coalitional games. It is a descriptive solution concept that focuses on outcomes that are "stable." Stability under core means that no set of players can jointly deviate to improve their payoffs. Formally, let $x(S)$ denote $\sum_{i \in S} x_i$. An outcome $x \in \mathbb{R}^n$ is in the core if

$$\forall S \subseteq N \quad x(S) \geq v(S) \tag{2}$$

The core was one of the first proposed solution concepts for coalitional games, and had been studied in detail. An important question for a given coalitional game is whether the core is empty. In other words, whether there is any outcome that is stable relative to group deviation. For a game to have a non-empty core, it must satisfy the property of *balancedness*, defined as follows. Let $1_S \in \mathbb{R}^n$ denote the characteristic vector of S given by

$$(1_S)_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases}$$

Let $(\lambda_S)_{S \subseteq N}$ be a set of weights such that each λ_S is in the range between 0 and 1. This set of weights, $(\lambda_S)_{S \subseteq N}$, is a *balanced collection* if for all $i \in N$,

$$\sum_{S \subseteq N} \lambda_S (1_S)_i = 1$$

A game is *balanced* if for all balanced collections of weights,

$$\sum_{S \subseteq N} \lambda_S v(S) \leq v(N) \tag{3}$$

By the Bondereva-Shapley theorem, the core of a coalitional game is non-empty if and only if the game is balanced. Therefore, we can use linear programming to determine whether the core of a game is empty.

$$\begin{array}{lll} \underset{\lambda \in \mathbb{R}^{2^n}}{\text{maximize}} & \sum_{S \subseteq N} \lambda_S v(S) & \\ \text{subject to} & \sum_{S \subseteq N} \lambda_S 1_S = 1 & \forall i \in N \\ & \lambda_S \geq 0 & \forall S \subseteq N \end{array} \tag{4}$$

If the optimal value of (4) is greater than the value of the grand coalition, then the core is empty. Unfortunately, this program has an exponential number of variables in the number of players in the game, and hence an algorithm that operates directly on this program would be infeasible in practice. In section 5.4, we will describe an algorithm that answers the question of emptiness of core that works on the dual of this program instead.

2.2 Previous Work Revisited

Deng and Papadimitriou looked into the complexity of various solution concepts on coalitional games played on weighted graphs in [4]. In their representation, the set of agents are the nodes of the graph, and the value of a set of agents S is the sum of the weights of the edges spanned by them. Notice that this representation is concise since the space required to specify such a game is $O(n^2)$. However, this representation is not general; it will not be able to represent interactions among three or more agents. For example, it will not be able to represent the *majority game*, where a group of agents S will have value of 1 if and only if $s > n/2$. On the other hand, there is an efficient algorithm for computing the Shapley value of the game, and for determining whether the core is empty under the restriction of positive edge weights. However, in the unrestricted case, determining whether the core is non-empty is coNP-complete.

Conitzer and Sandholm in [2] considered coalitional games that are superadditive. They described a concise representation scheme that only states the value of a coalition if the value is *strictly* superadditive. More precisely, the semantics of the representation is that for a group of agents S,

$$v(S) = \max_{\{T_1, T_2, \ldots, T_n\} \in \Pi} \sum_i v(T_i)$$

where Π is the set of all possible partitions of S. The value $v(S)$ is only explicitly specified for S if $v(S)$ is greater than all partitioning of S other than the trivial partition ($\{S\}$). While this representation can represent all games that are superadditive, there are coalitional games that it cannot represent. For example, it will not be able to represent any games with substitutability among the agents. An example of a game that cannot be represented is the *unit game*, where $v(S) = 1$ as long as $S \neq \emptyset$. Under this representation, the authors showed that determining whether the core is non-empty is coNP-complete. In fact, even determining the value of a group of agents is NP-complete.

In a more recent paper, Conitzer and Sandholm described a representation that decomposes a coalitional game into a number of subgames whose sum add up to the original game [3]. The payoffs in these subgames are then represented by their respective characteristic functions. This scheme is fully general as the characteristic form is a special case of this representation. For any given game, there may be multiple ways to decompose the game, and the decomposition may influence the computational complexity. For computing the Shapley value, the authors showed that the complexity is linear in the input description; in particular, if the largest subgame (as measured by number of agents) is of size n and the number of subgames is m, then their algorithm runs in $O(m2^n)$ time, where the input size will also be $O(m2^n)$. On the other hand, the problem of determining whether a certain outcome is in the core is coNP-complete.

3. MARGINAL CONTRIBUTION NETS

In this section, we will describe the *Marginal Contribution Networks* representation scheme. We will show that the idea is flexible, and we can easily extend it to increase its conciseness. We will also show how we can use this scheme to represent the recommendation game from the introduction. Finally, we will show that this scheme is fully expressive, and generalizes the representation schemes in [3, 4].

3.1 Rules and Marginal Contribution Networks

The basic idea behind marginal contribution networks (MC-nets) is to represent coalitional games using sets of *rules*. The rules in MC-nets have the following syntactic

form:

$$Pattern \rightarrow value$$

A rule is said to *apply* to a group of agents S if S *meets* the requirement of the *Pattern*. In the basic scheme, these patterns are conjunctions of agents, and S meets the requirement of the given pattern if S is a superset of it. The value of a group of agents is defined to be the sum over the values of all rules that apply to the group. For example, if the set of rules are

$$\{a \wedge b\} \rightarrow 5$$
$$\{b\} \rightarrow 2$$

then $v(\{a\}) = 0$, $v(\{b\}) = 2$, and $v(\{a, b\}) = 5 + 2 = 7$.

MC-nets is a very flexible representation scheme, and can be extended in different ways. One simple way to extend it and increase its conciseness is to allow a wider class of patterns in the rules. A pattern that we will use throughout the remainder of the paper is one that applies only in the *absence* of certain agents. This is useful for expressing concepts such as substitutability or default values. Formally, we express such patterns by

$$\{p_1 \wedge p_2 \wedge \ldots \wedge p_m \wedge \neg n_1 \wedge \neg n_2 \wedge \ldots \wedge \neg n_n\}$$

which has the semantics that such rule will apply to a group S only if $\{p_i\}_{i=1}^m \in S$ and $\{n_j\}_{j=1}^n \notin S$. We will call the $\{p_i\}_{i=1}^m$ in the above pattern the *positive literals*, and $\{n_j\}_{j=1}^n$ the *negative literals*. Note that if the pattern of a rule consists solely of negative literals, we will consider that the empty set of agents will also satisfy such pattern, and hence $v(\emptyset)$ may be non-zero in the presence of negative literals.

To demonstrate the increase in conciseness of representation, consider the unit game described in section 2.2. To represent such a game without using negative literals, we will need 2^n rules for n players: we need a rule of value 1 for each individual agent, a rule of value -1 for each pair of agents to counter the double-counting, a rule of value 1 for each triplet of agents, etc., similar to the inclusion-exclusion principle. On the other hand, using negative literals, we only need n rules: value 1 for the first agent, value 1 for the second agent *in the absence of the first agent*, value 1 for the third agent *in the absence of the first two agents*, etc. The representational savings can be exponential in the number of agents.

Given a game represented as a MC-net, we can interpret the set of rules that make up the game as a graph. We call this graph the *agent graph*. The nodes in the graph will represent the agents in the game, and for each rule in the MC-net, we connect all the agents in the rule together and assign a value to the *clique* formed by the set of agents. Notice that to accommodate negative literals, we will need to annotate the clique appropriately. This alternative view of MC-nets will be useful in our algorithm for CORE-MEMBERSHIP in section 5.

We would like to end our discussion of the representation scheme by mentioning a trade-off between the expressiveness of patterns and the space required to represent them. To represent a coalitional game in characteristic form, one would need to specify all $2^n - 1$ values. There is no overhead on top of that since there is a natural ordering of the groups. For MC-nets, however, specification of the rules requires specifying both the patterns and the values. The patterns, if not represented compactly, may end up overwhelming the savings from having fewer values to specify. The space required for the patterns also leads to a trade-off between the expressiveness of the allowed patterns and the simplicity of representing them. However, we believe that for most naturally arising games, there should be sufficient structure in the problem such that our representation achieves a net saving over the characteristic form.

3.2 Example: Recommendation Game

As an example, we will use MC-net to represent the recommendation game discussed in the introduction. For each product, as the benefit of knowing about the product will count only once for each group, we need to capture substitutability among the agents. This can be captured by a scaled unit game. Suppose the value of the knowledge about product i is v_i, and there are n_i agents, denoted by $\{x_i^j\}$, who know about the product, the game for product i can then be represented as the following rules:

$$\{x_i^1\} \rightarrow v_i$$
$$\{x_i^2 \wedge \neg x_i^1\} \rightarrow v_i$$
$$\vdots$$
$$\{x_i^{n_i} \wedge \neg x_i^{n_i-1} \wedge \cdots \wedge \neg x_i^1\} \rightarrow v_i$$

The entire game can then be built up from the sets of rules of each product. The space requirement will be $O(mn^*)$, where m is the number of products in the system, and n^* is the maximum number of agents who knows of the same product.

3.3 Representation Power

We will discuss the expressiveness and conciseness of our representation scheme and compare it with the previous works in this subsection.

PROPOSITION 1. *Marginal contribution networks constitute a fully expressive representation scheme.*

PROOF. Consider an arbitrary coalitional game $\langle N, v\rangle$ in characteristic form representation. We can construct a set of rules to describe this game by starting from the singleton sets and building up the set of rules. For any singleton set $\{i\}$, we create a rule $\{i\} \rightarrow v(i)$. For any pair of agents $\{i, j\}$, we create a rule $\{i \wedge j\} \rightarrow v(\{i, j\}) - v(\{i\}) - v(\{j\}$. We can continue to build up rules in a manner similar to the inclusion-exclusion principle. Since the game is arbitrary, MC-nets are fully expressive. □

Using the construction outlined in the proof, we can show that our representation scheme can simulate the multi-issue representation scheme of [3] in almost the same amount of space.

PROPOSITION 2. *Marginal contribution networks use at most a linear factor (in the number of agents) more space than multi-issue representation for any game.*

PROOF. Given a game in multi-issue representation, we start by describing each of the subgames, which are represented in characteristic form in [3], with a set of rules.

We then build up the grand game by including all the rules from the subgames. Note that our representation may require a space larger by a linear factor due to the need to describe the patterns for each rule. On the other hand, our approach may have fewer than exponential number of rules for each subgame, depending on the structure of these subgames, and therefore may be more concise than multi-issue representation. □

On the other hand, there are games that require exponentially more space to represent under the multi-issue scheme compared to our scheme.

PROPOSITION 3. *Marginal contribution networks are exponentially more concise than multi-issue representation for certain games.*

PROOF. Consider a unit game over all the agents N. As explained in 3.1, this game can be represented in linear space using MC-nets with negative literals. However, as there is no decomposition of this game into smaller subgames, it will require space $O(2^n)$ to represent this game under the multi-issue representation. □

Under the agent graph interpretation of MC-nets, we can see that MC-nets is a generalization of the graphical representation in [4], namely from weighted graphs to weighted hypergraphs.

PROPOSITION 4. *Marginal contribution networks can represent any games in graphical form (under [4]) in the same amount of space.*

PROOF. Given a game in graphical form, G, for each edge (i, j) with weight w_{ij} in the graph, we create a rule $\{i, j\} \rightarrow w_{ij}$. Clearly this takes exactly the same space as the size of G, and by the additive semantics of the rules, it represents the same game as G. □

4. COMPUTING THE SHAPLEY VALUE

Given a MC-net, we have a simple algorithm to compute the Shapley value of the game. Considering each rule as a separate game, we start by computing the Shapley value of the agents for each rule. For each agent, we then sum up the Shapley values of that agent over all the rules. We first show that this final summing process correctly computes the Shapley value of the agents.

PROPOSITION 5. *The Shapley value of an agent in a marginal contribution network is equal to the sum of the Shapley values of that agent over each rule.*

PROOF. For any group S, under the MC-nets representation, $v(S)$ is defined to be the sum over the values of all the rules that apply to S. Therefore, considering each rule as a game, by the (ADD) axiom discussed in section 2, the Shapley value of the game created from aggregating all the rules is equal to the sum of the Shapley values over the rules. □

The remaining question is how to compute the Shapley values of the rules. We can separate the analysis into two cases, one for rules with only positive literals and one for rules with mixed literals.

For rules that have only positive literals, the Shapley value of the agents is v/m, where v is the value of the rule and m is the number of agents in the rule. This is a direct consequence of the (SYM) axiom of the Shapley value, as the agents in a rule are indistinguishable from each other.

For rules that have both positive and negative literals, we can consider the positive and the negative literals separately. For a given positive literal i, the rule will apply only if i occurs in a given permutation after the rest of the positive literals but before any of the negative literals. Formally, let ϕ_i denote the Shapley value of i, p denote the cardinality of the positive set, and n denote the cardinality of the negative set, then

$$\phi_i = \frac{(p-1)!n!}{(p+n)!}\, v = \frac{v}{p\binom{p+n}{n}}$$

For a given negative literal j, j will be responsible for cancelling the application of the rule if all positive literals come before the negative literals in the ordering, and j is the first among the negative literals. Therefore,

$$\phi_j = \frac{p!(n-1)!}{(p+n)!}\,(-v) = \frac{-v}{n\binom{p+n}{p}}$$

By the (SYM) axiom, all positive literals will have the value of ϕ_i and all negative literals will have the value of ϕ_j.

Note that the sum over all agents in rules with mixed literals is 0. This is to be expected as these rules contribute 0 to the grand coalition. The fact that these rules have no effect on the grand coalition may appear odd at first. But this is because the presence of such rules is to define the values of coalitions smaller than the grand coalition.

In terms of computational complexity, given that the Shapley value of any agent in a given rule can be computed in time linear in the pattern of the rule, the total running time of the algorithm for computing the Shapley value of the game is linear in the size of the input.

5. ANSWERING CORE-RELATED QUESTIONS

There are a few different but related computational problems associated with the solution concept of the core. We will focus on the following two problems:

Definition 1. (CORE-MEMBERSHIP) Given a coalitional game and a payoff vector x, determine if x is in the core.

Definition 2. (CORE-NON-EMPTINESS) Given a coalitional game, determine if the core is non-empty.

In the rest of the section, we will first show that these two problems are coNP-complete and coNP-hard respectively, and discuss some complexity considerations about these problems. We will then review the main ideas of tree decomposition as it will be used extensively in our algorithm for CORE-MEMBERSHIP. Next, we will present the algorithm for CORE-MEMBERSHIP, and show that the algorithm runs in polynomial time for graphs of bounded treewidth. We end by extending this algorithm to answer the question of CORE-NON-EMPTINESS in polynomial time for graphs of bounded treewidth.

5.1 Computational Complexity

The hardness of CORE-MEMBERSHIP and CORE-NON-EMPTINESS follows directly from the hardness results of games over weighted graphs in [4].

PROPOSITION 6. CORE-MEMBERSHIP *for games represented as marginal contribution networks is coNP-complete.*

PROOF. CORE-MEMBERSHIP in MC-nets is in the class of coNP since any set of agents S of which $v(S) > x(S)$ will serve as a certificate to show that x does not belong to the core. As for its hardness, given any instance of CORE-MEMBERSHIP for a game in graphical form of [4], we can encode the game in exactly the same space using MC-net due to Proposition 4. Since CORE-MEMBERSHIP for games in graphical form is coNP-complete, CORE-MEMBERSHIP in MC-nets is coNP-hard. □

PROPOSITION 7. CORE-NON-EMPTINESS *for games represented as marginal contribution networks is coNP-hard.*

PROOF. The same argument for hardness between games in graphical frm and MC-nets holds for the problem of CORE-NON-EMPTINESS. □

We do not know of a certificate to show that CORE-NON-EMPTINESS is in the class of coNP as of now. Note that the "obvious" certificate of a balanced set of weights based on the Bondereva-Shapley theorem is exponential in size. In [4], Deng and Papadimitriou showed the coNP-completeness of CORE-NON-EMPTINESS via a combinatorial characterization, namely that the core is non-empty if and only if there is no negative cut in the graph. In MC-nets, however, there need not be a negative hypercut in the graph for the core to be empty, as demonstrated by the following game ($N = \{1, 2, 3, 4\}$):

$$v(S) = \begin{cases} 1 & \text{if } S = \{1,2,3,4\} \\ 3/4 & \text{if } S = \{1,2\}, \{1,3\}, \{1,4\}, \text{ or } \{2,3,4\} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

Applying the Bondereva-Shapley theorem, if we let $\lambda_{12} = \lambda_{13} = \lambda_{14} = 1/3$, and $\lambda_{234} = 2/3$, this set of weights demonstrates that the game is not balanced, and hence the core is empty. On the other hand, this game can be represented with MC-nets as follows (weights on hyperedges):

$$w(\{1,2\}) = w(\{1,3\}) = w(\{1,4\}) = 3/4$$
$$w(\{1,2,3\}) = w(\{1,2,4\}) = w(\{1,3,4\}) = -6/4$$
$$w(\{2,3,4\}) = 3/4$$
$$w(\{1,2,3,4\}) = 10/4$$

No matter how the set is partitioned, the sum over the weights of the hyperedges in the cut is always non-negative.

To overcome the computational hardness of these problems, we have developed algorithms that are based on tree decomposition techniques. For CORE-MEMBERSHIP, our algorithm runs in time exponential only in the treewidth of the agent graph. Thus, for graphs of small treewidth, such as trees, we have a tractable solution to determine if a payoff vector is in the core. By using this procedure as a separation oracle, i.e., a procedure for returning the inequality violated by a candidate solution, to solving a linear program that is related to CORE-NON-EMPTINESS using the ellipsoid method, we can obtain a polynomial time algorithm for CORE-NON-EMPTINESS for graphs of bounded treewidth.

5.2 Review of Tree Decomposition

As our algorithm for CORE-MEMBERSHIP relies heavily on tree decomposition, we will first briefly review the main ideas in tree decomposition and treewidth.[3]

Definition 3. A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\mathcal{X}, T)$, where $T = (I, F)$ is a tree and $\mathcal{X} = \{X_i \mid i \in I\}$ is a family of subsets of V, one for each node of T, such that

- $\bigcup_{i \in I} X_i = V$;
- For all edges $(v, w) \in E$, there exists an $i \in I$ with $v \in X_i$ and $w \in X_i$; and
- (*Running Intersection Property*) For all $i, j, k \in I$: if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The treewidth of a tree decomposition is defined as the maximum cardinality over all sets in $\mathcal{X}$, less one. The treewidth of a graph is defined as the minimum treewidth over all tree decompositions of the graph.

Given a tree decomposition, we can convert it into a *nice* tree decomposition of the same treewidth, and of size linear in that of T.

Definition 4. A tree decomposition T is *nice* if T is rooted and has four types of nodes:

Leaf nodes i are leaves of T with $|X_i| = 1$.

Introduce nodes i have one child j such that $X_i = X_j \cup \{v\}$ of some $v \in V$.

Forget nodes i have one child j such that $X_i = X_j \setminus \{v\}$ for some $v \in X_j$.

Join nodes i have two children j and k with $X_i = X_j = X_k$.

An example of a (partial) nice tree decomposition together with a classification of the different types of nodes is in Figure 1. In the following section, we will refer to nodes in the tree decomposition as nodes, and nodes in the agent graph as agents.

5.3 Algorithm for Core Membership

Our algorithm for CORE-MEMBERSHIP takes as an input a nice tree decomposition T of the agent graph and a payoff vector x. By definition, if x belongs to the core, then for all groups $S \subseteq N$, $x(S) \geq v(S)$. Therefore, the difference $x(S) - v(S)$ measures how "close" the group S is to violating the core condition. We call this difference the *excess* of group S.

Definition 5. The *excess* of a coalition S, $e(S)$, is defined as $x(S) - v(S)$.

A brute-force approach to determine if a payoff vector belongs to the core will have to check that the excesses of all groups are non-negative. However, this approach ignores the structure in the agent graph that will allow an algorithm to infer that certain groups have non-negative excesses due to

[3]This is based largely on the materials from a survey paper by Bodlaender [1].

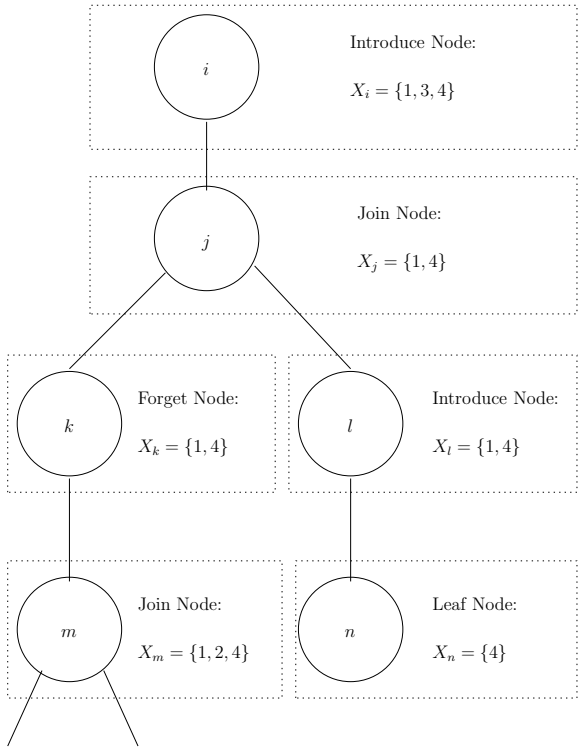

Figure 1: Example of a (partial) nice tree decomposition

the excesses computed elsewhere in the graph. Tree decomposition is the key to take advantage of such inferences in a structured way.

For now, let us focus on rules with positive literals. Suppose we have already checked that the excesses of all sets $R \subseteq U$ are non-negative, and we would like to check if the addition of an agent i to the set U will create a group with negative excess. A naïve solution will be to compute the excesses of all sets that include i. The excess of the group $(R \cup \{i\})$ for any group R can be computed as follows

$$e(R \cup \{i\}) = e(R) + x_i - v(c) \tag{6}$$

where c is the cut between R and i, and $v(c)$ is the sum of the weights of the edges in the cut.

However, suppose that from the tree decomposition, we know that i is only connected to a subset of U, say S, which we will call the *entry set* to U. Ideally, because i does not share any edges with members of $\bar{U} = (U \setminus S)$, we would hope that an algorithm can take advantage of this structure by checking only sets that are subsets of $(S \cup \{i\})$. This computational saving may be possible since $(x_i - v(c))$ in the update equation of (6) does not depend on $\bar{U}$. However, we cannot simply ignore $\bar{U}$ as members of $\bar{U}$ may still influence the excesses of groups that include agent i through group S. Specifically, if there exists a group $T \supset S$ such that $e(T) < e(S)$, then even when $e(S \cup \{i\})$ has non-negative excess, $e(T \cup \{i\})$ may have negative excess. In other words, the excess available at S may have been "drained" away due to T. This motivates the definition of the *reserve* of a group.

Definition 6. The *reserve* of a coalition S relative to a coalition U is the minimum excess over all coalitions between S and U, i.e., all $T : S \subseteq T \subseteq U$. We denote this value by $r(S, U)$. We will refer to the group T that has the minimum excess as $\arg r(S, U)$. We will also call U the *limiting set* of the reserve and S the *base set* of the reserve.

Our algorithm works by keeping track of the reserves of all non-empty subsets that can be formed by the agents of a node at each of the nodes of the tree decomposition. Starting from the leaves of the tree and working towards the root, at each node i, our algorithm computes the reserves of all groups $S \subseteq X_i$, limited by the set of agents in the subtree rooted at i, T_i, *except* those in $(X_i \backslash S)$. The agents in $(X_i \backslash S)$ are excluded to ensure that S is an entry set. Specifically, S is the entry set to $((T_i \setminus X_i) \cup S)$.

To accomodate for negative literals, we will need to make two adjustments. Firstly, the cut between an agent m and a set S at node i now refers to the cut among agent m, set S, and set $\neg(X_i \setminus S)$, and its value must be computed accordingly. Also, when an agent m is introduced to a group at an introduce node, we will also need to consider the change in the reserves of groups that do not include m due to possible cut involving $\neg m$ and the group.

As an example of the reserve values we keep track of at a tree node, consider node i of the tree in Figure 1. At node i, we will keep track of the following:

$$r(\{1\}, \{1, 2, \ldots\})$$
$$r(\{3\}, \{2, 3, \ldots\})$$
$$r(\{4\}, \{2, 4, \ldots\})$$
$$r(\{1, 3\}, \{1, 2, 3, \ldots\})$$
$$r(\{1, 4\}, \{1, 2, 4, \ldots\})$$
$$r(\{3, 4\}, \{2, 3, 4, \ldots\})$$
$$r(\{1, 3, 4\}, \{1, 2, 3, 4, \ldots\}$$

where the dots ... refer to the agents rooted under node m.

For notational use, we will use $r_i(S)$ to denote $r(S, U)$ at node i where U is the set of agents in the subtree rooted at node i excluding agents in $(X_i \setminus S)$. We sometimes refer to these values as the *r-values* of a node. The details of the r-value computations are in Algorithm 1.

To determine if the payoff vector x is in the core, during the r-value computation at each node, we can check if all of the r-values are non-negative. If this is so for all nodes in the tree, the payoff vector x is in the core. The correctness of the algorithm is due to the following proposition.

PROPOSITION 8. *The payoff vector x is not in the core if and only if the r-values at some node i for some group S is negative.*

PROOF. ($\Leftarrow$) If the reserve at some node i for some group S is negative, then there exists a coalition T for which $e(T) = x(T) - v(T) < 0$, hence x is not in the core.

($\Rightarrow$) Suppose x is not in the core, then there exists some group R^* such that $e(R^*) < 0$. Let X_{root} be the set of nodes at the root. Consider any set $S \in X_{root}$, $r_{root}(S)$ will have the base set of S and the limiting set of $((N \setminus X_{root}) \cup S)$. The union over all of these ranges includes all sets U for which $U \cap X_{root} \neq \emptyset$. Therefore, if R^* is not disjoint from X_{root}, the r-value for some group in the root is negative.

If R^* is disjoint from U, consider the forest $\{T_i\}$ resulting from removal of all tree nodes that include agents in X_{root}.

Algorithm 1 Subprocedures for Core Membership

LEAF-NODE(i)
1: $r_i(X_i) \leftarrow e(X_i)$

INTRODUCE-NODE(i)
2: $j \leftarrow$ child of i
3: $m \leftarrow X_i \setminus X_j$ {the introduced node}
4: **for all** $S \subseteq X_j, S \neq \emptyset$ **do**
5: $C \leftarrow$ all hyperedges in the cut of m, S, and $\neg(X_i \setminus S)$
6: $r_i(S \cup \{x\}) \leftarrow r_j(S) + x_m - v(C)$
7: $C \leftarrow$ all hyperedges in the cut of $\neg m$, S, and $\neg(X_i \setminus S)$
8: $r_i(S) \leftarrow r_j(S) - v(C)$
9: **end for**
10: $r(\{m\}) \leftarrow e(\{m\})$

FORGET-NODE(i)
11: $j \leftarrow$ child of i
12: $m \leftarrow X_j \setminus X_i$ {the forgotten node}
13: **for all** $S \subseteq X_i, S \neq \emptyset$ **do**
14: $r_i(S) = \min(r_j(S), r_j(S \cup \{m\}))$
15: **end for**

JOIN-NODE(i)
16: $\{j, k\} \leftarrow$ {left, right} child of i
17: **for all** $S \subseteq X_i, S \neq \emptyset$ **do**
18: $r_i(S) \leftarrow r_j(S) + r_k(S) - e(S)$
19: **end for**

By the running intersection property, the sets of nodes in the trees T_i's are disjoint. Thus, if the set $R^* = \bigcup_i S_i$ for some $S_i \in T_i$, $e(R^*) = \sum_i e(S_i) < 0$ implies some group S_i^* has negative excess as well. Therefore, we only need to check the r-values of the nodes on the individual trees in the forest.

But for each tree in the forest, we can apply the same argument restricted to the agents in the tree. In the base case, we have the leaf nodes of the original tree decomposition, say, for agent i. If $R^* = \{i\}$, then $r(\{i\}) = e(\{i\}) < 0$. Therefore, by induction, if $e(R^*) < 0$, some reserve at some node would be negative. □

We will next explain the intuition behind the correctness of the computations for the r-values in the tree nodes. A detailed proof of correctness of these computations can be found in the appendix under Lemmas 1 and 2.

PROPOSITION 9. *The procedure in Algorithm 1 correctly compute the r-values at each of the tree nodes.*

PROOF. (SKETCH) We can perform a case analysis over the four types of tree nodes in a nice tree decomposition.

Leaf nodes (i) The only reserve value to be computed is $r_i(X_i)$, which equals $r(X_i, X_i)$, and therefore it is just the excess of group X_i.

Forget nodes (i with child j) Let m be the forgotten node. For any subset $S \subseteq X_i$, $\arg r_i(S)$ must be chosen between the groups of S and $S \cup \{m\}$, and hence we choose between the lower of the two from the r-values at node j.

Introduce nodes (i with child j) Let m be the introduced node. For any subset $T \subseteq X_i$ that includes m, let S denote $(T \setminus \{m\})$. By the running intersection property, there are no rules that involve m and agents of the subtree rooted at node i except those involving m and agents in X_i. As both the base set and the limiting set of the r-values of node j and node i differ by $\{m\}$, for any group V that lies between the base set and the limiting set of node i, the excess of group V will differ by a constant amount from the corresponding group $(V \setminus \{m\})$ at node j. Therefore, the set $\arg r_i(T)$ equals the set $\arg r_j(S) \cup \{m\}$, and $r_i(T) = r_j(S) + x_m - v(\text{cut})$, where $v(\text{cut})$ is the value of the rules in the cut between m and S. For any subset $S \subset X_i$ that does not include m, we need to consider the values of rules that include $\neg m$ as a literal in the pattern. Also, when computing the reserve, the payoff x_m will not contribute to group S. Therefore, together with the running intersection property as argued above, we can show that $r_i(S) = r_j(S) - v(\text{cut})$.

Join nodes (i with left child j and right child k) For any given set $S \subseteq X_i$, consider the r-values of that set at j and k. If $\arg r_j(S)$ or $\arg r_k(S)$ includes agents not in S, then $argr_j(S)$ and $argr_k(S)$ will be disjoint from each other due to the running intersection property. Therefore, we can decompose $\arg r_i(S)$ into three sets, $(\arg r_j(S) \setminus S)$ on the left, S in the middle, and $(\arg r_k(S) \setminus S)$ on the right. The reserve $r_j(S)$ will cover the excesses on the left and in the middle, whereas the reserve $r_k(S)$ will cover those on the right and in the middle, and so the excesses in the middle is double-counted. We adjust for the double-counting by subtracting the excesses in the middle from the sum of the two reserves $r_j(S)$ and $r_k(S)$.

□

Finally, note that each step in the computation of the r-values of each node i takes time at most exponential in the size of X_i, hence the algorithm runs in time exponential only in the treewidth of the graph.

5.4 Algorithm for Core Non-emptiness

We can extend the algorithm for CORE-MEMBERSHIP into an algorithm for CORE-NON-EMPTINESS. As described in section 2, whether the core is empty can be checked using the optimization program based on the balancedness condition (3). Unfortunately, that program has an exponential number of variables. On the other hand, the dual of the program has only n variables, and can be written as follows:

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & \sum_{i=1}^n x_i \\ \text{subject to} & x(S) \geq v(S), \quad \forall S \subseteq N \end{array} \tag{7}$$

By strong duality, optimal value of (7) is equal to optimal value of (4), the primal program described in section 2. Therefore, by the Bondereva-Shapley theorem, if the optimal value of (7) is greater than $v(N)$, the core is empty.

We can solve the dual program using the ellipsoid method with CORE-MEMBERSHIP as a separation oracle, i.e., a procedure for returning a constraint that is violated. Note that a simple extension to the CORE-MEMBERSHIP algorithm will allow us to keep track of the set T for which $e(T) < 0$ during the r-values computation, and hence we can return the inequality about T as the constraint violated. Therefore, CORE-NON-EMPTINESS can run in time polynomial in the running time of CORE-MEMBERSHIP, which in turn runs in

time exponential only in the treewidth of the graph. Note that when the core is not empty, this program will return an outcome in the core.

6. CONCLUDING REMARKS

We have developed a fully expressive representation scheme for coalitional games of which the size depends on the complexity of the interactions among the agents. Our focus on general representation is in contrast to the approach taken in [3, 4]. We have also developed an efficient algorithm for the computation of the Shapley values for this representation. While CORE-MEMBERSHIP for MC-nets is coNP-complete, we have developed an algorithm for CORE-MEMBERSHIP that runs in time exponential only in the treewidth of the agent graph. We have also extended the algorithm to solve CORE-NON-EMPTINESS. Other than the algorithm for CORE-NON-EMPTINESS in [4] under the restriction of non-negative edge weights, and that in [2] for superadditive games when the value of the grand coalition is given, we are not aware of any explicit description of algorithms for core-related problems in the literature.

The work in this paper is related to a number of areas in computer science, especially in artificial intelligence. For example, the graphical interpretation of MC-nets is closely related to Markov random fields (MRFs) of the Bayes nets community. They both address the issue of of conciseness of representation by using the combinatorial structure of weighted hypergraphs. In fact, Kearns *et al.* first apply these idea to games theory by introducing a representation scheme derived from Bayes net to represent non-cooperative games [6]. The representational issues faced in coalitional games are closely related to the problem of expressing valuations in combinatorial auctions [5, 10]. The OR-bid language, for example, is strongly related to superadditivity. The question of the representation power of different patterns is also related to Boolean expression complexity [12]. We believe that with a better understanding of the relationships among these related areas, we may be able to develop more efficient representations and algorithms for coalitional games.

Finally, we would like to end with some ideas for extending the work in this paper. One direction to increase the conciseness of MC-nets is to allow the definition of equivalent classes of agents, similar to the idea of extending Bayes nets to probabilistic relational models. The concept of symmetry is prevalent in games, and the use of classes of agents will allow us to capture symmetry naturally and concisely. This will also address the problem of unpleasing assymetric representations of symmetric games in our representation.

Along the line of exploiting symmetry, as the agents within the same class are symmetric with respect to each other, we can extend the idea above by allowing functional description of marginal contributions. More concretely, we can specify the value of a rule as dependent on the number of agents of each relevant class. The use of functions will allow concise description of marginal diminishing returns (MDRs). Without the use of functions, the space needed to describe MDRs among n agents in MC-nets is $O(n)$. With the use of functions, the space required can be reduced to $O(1)$.

Another idea to extend MC-nets is to augment the semantics to allow constructs that specify certain rules cannot be applied simultaneously. This is useful in situations where a certain agent represents a type of exhaustible resource, and therefore rules that depend on the presence of the agent should not apply simultaneously. For example, if agent i in the system stands for coal, we can either use it as fuel for a power plant or as input to a steel mill for making steel, but not for both at the same time. Currently, to represent such situations, we have to specify rules to cancel out the effects of applications of different rules. The augmented semantics can simplify the representation by specifying when rules cannot be applied together.

7. ACKNOWLEDGMENT

The authors would like to thank Chris Luhrs, Bob McGrew, Eugene Nudelman, and Qixiang Sun for fruitful discussions, and the anonymous reviewers for their helpful comments on the paper.

8. REFERENCES

[1] H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proc. 22nd Symp. on Mathematical Foundation of Copmuter Science*, pages 19–36. Springer-Verlag LNCS 1295, 1997.

[2] V. Conitzer and T. Sandholm. Complexity of determining nonemptiness of the core. In *Proc. 18th Int. Joint Conf. on Artificial Intelligence*, pages 613–618, 2003.

[3] V. Conitzer and T. Sandholm. Computing Shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In *Proc. 19th Nat. Conf. on Artificial Intelligence*, pages 219–225, 2004.

[4] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 19:257–266, May 1994.

[5] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proc. 16th Int. Joint Conf. on Artificial Intelligence*, pages 548–553, 1999.

[6] M. Kearns, M. L. Littman, and S. Singh. Graphical models for game theory. In *Proc. 17th Conf. on Uncertainty in Artificial Intelligence*, pages 253–260, 2001.

[7] J. Kleinberg, C. H. Papadimitriou, and P. Raghavan. On the value of private information. In *Proc. 8th Conf. on Theoretical Aspects of Rationality and Knowledge*, pages 249–257, 2001.

[8] C. Li and K. Sycara. Algoirthms for combinatorial coalition formation and payoff division in an electronic marketplace. Technical report, Robotics Insititute, Carnegie Mellon University, November 2001.

[9] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, New York, 1995.

[10] N. Nisan. Bidding and allocation in combinatorial auctions. In *Proc. 2nd ACM Conf. on Electronic Commerce*, pages 1–12, 2000.

[11] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, Cambridge, Massachusetts, 1994.

[12] I. Wegener. *The Complexity of Boolean Functions*. John Wiley & Sons, New York, October 1987.

APPENDIX

We will formally show the correctness of the r-value computation in Algorithm 1 of introduce nodes and join nodes.

LEMMA 1. *The procedure for computing the r-values of introduce nodes in Algorithm 1 is correct.*

PROOF. Let node m be the newly introduced agent at i. Let U denote the set of agents in the subtree rooted at i. By the running intersection property, all interactions (the hyperedges) between m and U must be in node i. For all $S \subseteq X_i : m \in S$, let R denote $(U \setminus X_i) \cup S)$, and Q denote $(R \setminus \{m\})$.

$$\begin{aligned} r_i(S) &= r(S,R) \\ &= \min_{T:S\subseteq T\subseteq R} e(T) \\ &= \min_{T:S\subseteq T\subseteq R} x(T) - v(T) \\ &= \min_{T:S\subseteq T\subseteq R} x(T\setminus\{m\}) + x_m - v(T\setminus\{m\}) - v(\text{cut}) \\ &= \left(\min_{T':S\setminus\{m\}\subseteq T'\subseteq Q} e(T')\right) + x_m - v(\text{cut}) \\ &= r_j(S) + x_m - v(\text{cut}) \end{aligned}$$

The argument for sets $S \subseteq X_i : m \notin S$ is symmetric except x_m will not contribute to the reserve due to the absence of m. □

LEMMA 2. *The procedure for computing the r-values of join nodes in Algorithm 1 is correct.*

PROOF. Consider any set $S \subseteq X_i$. Let U_j denote the subtree rooted at the left child, R_j denote $((U_j \setminus X_j) \cup S)$, and Q_j denote $(U_j \setminus X_j)$. Let U_k, R_k, and Q_k be defined analogously for the right child. Let R denote $(U \setminus X_i) \cup S)$.

$$\begin{aligned} r_i(S) &= r(S,R) \\ &= \min_{T:S\subseteq T\subseteq R} x(T) - v(T) \\ &= \min_{T:S\subseteq T\subseteq R} \Big(x(S) + x(T\cap Q_j) + x(T\cap Q_k) \\ &\quad - v(S) - v(\text{cut}(S, T\cap Q_j) - v(\text{cut}(S, T\cap Q_k)\Big) \\ &= \min_{T:S\subseteq T\subseteq R} \Big(x(T\cap Q_j) - v(\text{cut}(S, T\cap Q_j))\Big) \\ &\quad + \min_{T:S\subseteq T\subseteq R} \Big(x(T\cap Q_k) - v(\text{cut}(S, T\cap Q_k))\Big) \\ &\quad + (x(S) - v(S)) \qquad (*) \\ &= \min_{T:S\subseteq T\subseteq R} \Big(x(T\cap Q_j) + x(S) - v(\text{cut}(S, T\cap Q_j)) - v(S)\Big) \\ &\quad + \min_{T:S\subseteq T\subseteq R} \Big(x(T\cap Q_k) + x(S) - v(\text{cut}(S, T\cap Q_k)) - v(S)\Big) \\ &\quad - (x(S) - v(S)) \\ &= \min_{T:S\subseteq T\subseteq R} e(T\cap R_j) + \min_{T:S\subseteq T\subseteq R} e(T\cap R_k) - e(S) \\ &= \min_{T':S\subseteq T'\subseteq R_j} e(T') + \min_{T'':S\subseteq T\subseteq R_k} e(T'') - e(S) \\ &= r_j(S) + r_k(S) - e(S) \end{aligned}$$

where (*) is true as $T \cap Q_j$ and $T \cap Q_k$ are disjoint due to the running intersection property of tree decomposition, and hence the minimum of the sum can be decomposed into the sum of the minima. □