Theoretical Computer Science 677 (2017) 69–82

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Efficient stabilization of cooperative matching games ☆

Takehiro Ito [a,1], Naonori Kakimura [b,*,2], Naoyuki Kamiyama [c,3], Yusuke Kobayashi [d,4], Yoshio Okamoto [e,5]

[a] *Tohoku University, Japan*
[b] *University of Tokyo, Japan*
[c] *Kyushu University/JST, PRESTO, Japan*
[d] *University of Tsukuba, Japan*
[e] *University of Electro-Communications, Japan*

ARTICLE INFO

Article history:
Received 1 August 2016
Received in revised form 3 February 2017
Accepted 23 March 2017
Available online 27 March 2017
Communicated by P. Krysta

Keywords:
Cooperative game
Core
Gallai–Edmonds decomposition
Matching
Network bargaining
Solution concept

ABSTRACT

Cooperative matching games have drawn much interest partly because of the connection with bargaining solutions in the networking environment. However, it is not always guaranteed that a network under investigation gives rise to a stable bargaining outcome. To address this issue, we consider a modification process, called *stabilization*, that yields a network with stable outcomes, where the modification should be as small as possible. Therefore, the problem is cast to a combinatorial-optimization problem in a graph. Recently, the stabilization by edge removal was shown to be **NP**-hard. On the contrary, in this paper, we show that other possible ways of stabilization, namely, edge addition, vertex removal and vertex addition, are all polynomial-time solvable. Thus, we obtain a complete complexity-theoretic classification of the natural four variants of the network stabilization problem. We further study weighted variants and prove that the variants for edge addition and vertex removal are **NP**-hard.

1. Introduction

1.1. Background

Matching markets play a central role in economics and game theory, and much work has been done. Among them, we concentrate on cooperative games with transferable utility derived from matchings in a network, called matching games, which are also known as stable roommate problems with side payments (when it is modeled as a cooperative game). The following example was given by Eriksson and Karlander [19] to motivate such games.

☆ A preliminary version appears in International Conference on Autonomous Agents and Multiagent Systems (AAMAS2016).

* Corresponding author.
E-mail address: kakimura@global.c.u-tokyo.ac.jp (N. Kakimura).

[1] Supported by JST CREST Grant Number JPMJCR1402 and JSPS KAKENHI Grant Numbers JP25330003, JP15H00849, JP16K00004.
[2] Supported by JST, ERATO, Kawarabayashi Large Graph Project, and by JSPS KAKENHI Grant Numbers JP25730001, JP24106002.
[3] Supported by JST, PRESTO.
[4] Supported by JST, ERATO, Kawarabayashi Large Graph Project, and by JSPS KAKENHI Grant Numbers JP24106002, JP24700004.
[5] Supported by Kayamori Foundation of Informational Science Advancement, JST CREST Grant Number JPMJCR1402, and JSPS KAKENHI Grant Numbers JP24106005, JP24700008, JP24220003, JP15K00009.

http://dx.doi.org/10.1016/j.tcs.2017.03.020
0304-3975/

In the professional tennis circuit, there are parallel singles and doubles tournaments. Although the prize money in the doubles is not as generous as in the singles, the sums are still impressive. The players are free to form pairs for the doubles as they choose. It is not necessarily so that the two best single players make the best team; the strength of a pair is a more complex function of the players abilities.

The players know each others' strengths and weaknesses, and can make good estimates of the expected prize money for each possible constellation. In the process of forming pairs, the players negotiate how to distribute the expected income within the pair. (At least, this is what rational tennis players should do.) The objective of each player in this process is to maximize his own expected prize money.

As a cooperative game with transferable utility, in a *matching game*, we are given a network (or an undirected graph), and each vertex of the network corresponds to a player of the game. A *matching* is a set of pairs of players such that every player is involved in at most one pair, and a pair can be formed only if there is an edge between two players. The network is often associated with edge weights so that the utility of forming a pair can be incorporated. The characteristic function value of a coalition is defined as the maximum possible total utility of a matching over the coalition.

Matching games have attracted researchers, and solution concepts for matching games have been studied through the algorithmic lens [20,11,10,27,25,15,5]. In particular, there is a polynomial-time algorithm to test whether a given matching game has a non-empty core [16], where the *core* is defined as the set of payoff vectors such that no coalition has a characteristic function value larger than the total payoff allocated to the players of the coalition. Thus, a "stable" payoff exists in a game with a non-empty core. We here note that if the underlying network is bipartite, then the game is called an *assignment game*, which is also well studied in the literature [31,41].

On the other hand, if the core of a game is empty, then no payoff is stable. Therefore, we often modify the game itself to achieve the core non-emptiness. The literature offers options of taxation [33,45], cost of stability [6], and least cores [27]. In this paper, we consider a structural modification of the underlying network, which better fits another motivation from network bargaining as explained later. The modification should be as small as possible so that the games themselves do not differ much.

There are several possible ways of modification. We consider the following four processes.

Edge removal: We remove a set of edges from the network. In this case, we want to remove as few edges as possible.
Edge addition: We add a set of edges to the network. In this case, we want to add as few edges as possible.
Vertex removal: We remove a set of vertices from the network. When we remove a vertex, all the edges incident to the vertex are removed too. In this case, we want to remove as few vertices as possible.
Vertex addition: We add a set of vertices to the network. When we add a vertex, we may also add edges from the new vertex to any of the existing vertices. In this case, we want to add as few vertices as possible.

The literature only studied edge removal. Namely, the problem was to find a smallest subset of edges whose removal results in a network with non-empty core. Biró et al. [10] proved that the problem is **NP**-hard for the weighted case. Later, Bock et al. [13] proved that the problem is still **NP**-hard for the unweighted case (as introduced above), and hard to approximate within factor less than two assuming the unique games conjecture [28]. Some approximation algorithms have been proposed [13,30], but the existence of a constant-factor approximation algorithm is left open.

1.2. Our results

We study the remaining three options, namely, edge addition, vertex removal, and vertex addition. For all of them, we prove that the problems can be solved in polynomial time (Sections 3, 4, and 5, respectively). In this way, we obtain the complete complexity classification of the problem variants, and reveal that edge removal studied in the literature is the only hard case.

Those polynomial-time algorithms are obtained by a thorough treatment of the so-called Gallai–Edmonds decompositions that possess useful information on the structure of maximum matchings in a graph. We also utilize the theory of linear programming, as explained later, to connect our problems with fractional matchings of a graph.

One may think that in vertex addition, we may add a lot of edges at the same time even though we add only one vertex. However, as it turns out, our algorithm finds a solution which adds as few edges as possible among all the possible solutions for the vertex addition variant.

We also consider a weighted variant. In the edge addition variant, each possible edge is associated with a non-negative real number. In the vertex removal variant, each vertex is associated with a non-negative real number. Those numbers represent costs of addition or removal. That models a more realistic situation where each possible modification operation is not equivalently manageable. With this setup, we prove that for the edge addition and the vertex removal, the problem is **NP**-hard (Section 6).

The results are summarized in Table 1.

Simultaneously with us, Ahmadian, Sanita, and Hosseinzadeh [1] consider the vertex removal variant, and prove its polynomial-time solvability and the **NP**-hardness of the weighted variant. Furthermore, they propose approximation algorithms for the weighted variant.