Core stable algorithms for coalition games with complementarities and peer effects

M. Touati[1,2,3], J.M. Kelif[1]
[1]Orange Labs,
Issy-les-Moulineaux, France

R. El-Azouzi[2]
[2]University of Avignon
(LIA/CERI), Avignon, France

M. Coupechoux[3]
[3]Telecom ParisTech and
CNRS LTCI, Paris, France

E. Altman[4]
[4]INRIA, Sophia-Antipolis,
France

1. ABSTRACT

In this short paper, we show two new algorithms for finding stable structures in ordinal coalition potential games. The first one is anytime and enumerative. It performs on a graph. The second one is a modified Deferred Acceptance Algorithm (DAA) using counter-proposals. It finds a many-to-one matching. We illustrate with the example of video caching from a content creator's servers to a service provider's servers.

2. INTRODUCTION

Game theorists have been interested in looking at the stability of the structures resulting from the coalition formation process and have shown the importance of the concept in real-life applications[1]. The stability exhibits the respect of the individual preferences and incentives. An unstable structure would result in deviations [3]. Coalition and matching games open the way through an important number of interesting applications, notably in wireless networks where much work remain to be done in the decentralized decision taking paradigm. These game-theoretic tools allow for more tractability in the formulations and a reduced complexity. Nevertheless, some of the commonly used assumptions (substitutability, responsive preferences, see [3]) limit the modeling to the cases without complementarities or peer effects. Recent theoretical works [9] (and references therein) overcame this difficulties and have paved the way through a suitable modeling of systems with complementarities and peer effects. As examples of such systems in wireless networks we have WiFi and its related anomaly, Device-2-Device with multi-hop relaying, virtual MIMO exploiting multi-users diversity, etc.

3. RELATED WORKS

Much work has been devoted to super-additive coalition games (where the grand-coalition forms [2]) and to the stabilization of structures by payoff distribution. Nevertheless, real-word problems may not verify the super-additivity assumption. More recently, researchers have been interested in the set of structures maximizing the social welfare in case of non-super-additive coalition games. This problem is called optimal coalition structure generation and is known to be an NP-hard problem with exponential complexity even for sub-optimal solution. Three classes are to be distinguished: dynamic programming, anytime property and heuristics. We have an interest in the first two classes. Dynamic programming has been used in [4, 7]. The worst case complexity is $O(3^n)$. On the opposite, any-time algorithms can be stopped at any-time to provide a sub-optimal result. The coalition structure graph representation[2] is used in [4] for coalition formation. The results are guaranteed to be within a bound from the optimum. Despite of improvements (see [8] and references therein), the complexity remains $O(n^n)$. Our compatibility graph is an alternative graph representation with coalitions as vertices and structures as maximal cliques. In [1], Gale and Shapley show the non-emptiness of the core of the stable marriage and the college admission problems with preferences over individuals. The Deferred Acceptance Algorithm (DAA) is introduced. In [3], Roth et a.l. survey the existing results related to matching games and develop the theoretical results to provide insights in the understanding of the matching markets. The core stable structures (no subset of players have an incentive to deviate and form a coalition with each others) is shown to be non-empty under some simplifying assumptions (e.g. responsive preferences). In [5], Cechlarova et a.l. propose two extensions of the preferences over individuals to sets and propose an algorithm close to Gale's top-trading cycles to find a strict core partition. In [6], Echenique et a.l. show a fixed-point-based algorithm to compute the set of core stable (if non-empty) many-to-one matchings in the case of preferences over colleagues. In [9], Pycia analytically tackles the problem of coalition formation and matchings with complementarities and peer effects. Among many other results, he shows that the pairwise aligned preferences and the Nash bargaining based allocation rules guarantee non-emptiness of the set of core stable structures in all states of nature.

4. CONTRIBUTIONS

[1]J.-M. Kelif and M. Coupechoux are partly supported by the french ANR project NETLEARN ANR-13-INFR-004

[2]The coalition structure graph is defined as the graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$, where the set of vertices is the set of coalition structures and the set of edges is defined such that it exists an edge $e = (S, S')$ between vertices S and S' if S' can be obtained from S by merging two coalitions of S or by splitting a coalition of S into two disjoint ones. By definition, the graph is of size $B(N)$ (Bell number).

The first algorithm is a centralized algorithm enumerating the set of core stable structures. It performs on a graph called the compatibility graph and does not require a tie-breaking rule (indifference between payoffs). The second algorithm is a decentralized algorithm finding a many-to-one core stable matching. It is a modified DAA that uses the pairwise alignment of the preferences over groups to reduce the preferences of the proposers over individuals.

5. POTENTIAL COALITION GAMES AND VIDEO CACHING

5.1 Potentials coalition games

Let $\Gamma = (\mathcal{N}, v, \{u_i\}_{i\in\mathcal{N}}\}$ define a coalition game in characteristic form. The set $\mathcal{N}$ denotes the set of players of cardinality N, $\{u_i\}_{i\in\mathcal{N}}$ denotes the set of their individual utilities and $v : \mathcal{N} \to \mathbb{R}$ is the characteristic function of the game. We define the set of coalitions $\mathcal{C}$. The potential function Φ is an ordinal coalition potential for the game Γ if for every player $i \in \mathcal{N}$,

$$u_i(C) > u_i(C') \text{ iff } \Phi(C) > \Phi(C'), \text{ for every } C, C' \in \mathcal{C} \quad (1)$$

where $u_i(C)$ is player i's utility of the payoff it receives when taking part in the coalition C.
A coalition game in characteristic form with set of players $\mathcal{N}$ admitting an ordinal potential Φ is called an ordinal coalition potential game in characteristic form and denoted,

$$\Gamma = (\mathcal{N}, v, \{u_i\}_{i\in\mathcal{N}}, \Phi) \quad (2)$$

5.2 An application: the video caching

As an example consider a video caching matching market between a content creator $\mathcal{P}$ and a service provider $\mathcal{S}$. The content creator $\mathcal{P}$ has a set $\mathcal{L}$ of L videos. The service provider $\mathcal{S}$ has a set $\mathcal{R}$ of R caching servers. The caching servers may differ in the Quality of Service (QoS) for the cached content. This differentiation is taken to be due to the hierarchical location of the servers in the caching tree. We define $\mathcal{T}$ the set of T QoS of the servers. The content creator's videos are stored in its own servers. They can also be duplicated in a subset of the service provider's servers.
We define the set of coalitions as,

$$\mathcal{C} = \{\{r\} \cup J,\ r \in \mathcal{R},\ J \subseteq \mathcal{L}, |J| \leq q_r\} \cup \{l \in \mathcal{L}\}$$

i.e. the set of subsets of videos and a single server. We furthermore require the quotas q_r of any server $r \in \mathcal{R}$ to be valued in $\{2, \dots, L-1\}$[3]. These quotas give for each server the maximum number of videos that can be cached.
We assume that the service provider can compute the impact of caching over the number of views of the videos in the coalition as the following fixed point equations,

$$n_l(C) = n_l(1)\gamma_r + \sum_{l'\in C\cap\mathcal{L}} a_{ll'} n_l(C)$$

where $a_{ll'}$ is an impact factor of l over l' and $\gamma_r \in \mathcal{T}$ is QoS gain obtained by caching in the server r. We define the characteristic function $v : \mathcal{C} \to \mathbb{R}$ of a coalition as,

$$v(C) = \sum_{l\in C\cap\mathcal{L}} n_l(C)\alpha$$

where α is the constant monetary income generated by a view of a video. We now assume $v(C)$ is shared among the players in C via a Nash bargaining (null threats) with concave individual utilities $u_l(x_l) = x_l^{\alpha_{\mathcal{P}}}$ for the content creator's videos and $u_r(x_r) = x_r^{\alpha_r}$ for the service provider's server r.The servers' bargaining powers increase in their factor of quality. We obtain that the individual payoffs increase in $\chi_C = \frac{v(C)}{|C\cap\mathcal{L}|\alpha_{\mathcal{P}}+\alpha_r}$ which is called the fear-of-ruin. In this case the fear-of-ruin is the potential Φ of the matching game between $\mathcal{P}$'s videos and $\mathcal{S}$'s servers. The game $\Gamma = (\mathcal{N} = \mathcal{L} \cup \mathcal{R}, v, \{u_i\}, \chi)$ is an ordinal coalition potential game.

6. AN ANYTIME ENUMERATIVE CLIQUES-BASED ALGORITHM

In this section we introduce an anytime cliques-based algorithm for enumerating the core stable coalition structures in the case of an ordinal coalition potential game Γ with a set of coalitions $\mathcal{C}$. The algorithm has been constructed over the game-theoritic sequential description of convergence to a stable structure provided in [9].

Define the coalitions weighted-vertices undirected graph $(\mathcal{G}, \Phi)$ such that the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the complement of the intersection graph of the coalitions in $\mathcal{C}$. The set of vertices $\mathcal{V}$ is in bijection α with the set of coalitions $\mathcal{C}$ but for the sake of simplicity and clarity we will further identify a coalition $C_i \in \mathcal{C}$ with the vertex $v_i \in \mathcal{V}$ of the graph. Let $\mathcal{V}' \subset \mathcal{V}$ be a subset of the nodes, we denote $\mathcal{G}' = (\mathcal{V}', \mathcal{E}(\mathcal{V}'))$ the subgraph induced by $\mathcal{V}'$. Each vertex $v_i \in \mathcal{V}$ is weighted by the value $\Phi(C_i)$ and the set of edges $\mathcal{E}$ is defined by the adjacency matrix $\mathbf{A} = (a_{ij})_{(i,j)\in\mathcal{C}^2}$ such that:

$$a_{ij} = \begin{cases} 1 & \text{if } C_i \neq C_j \text{ and } C_i \cap C_j = \{\emptyset\} \\ 0 & \text{if } C_i = C_j \text{ or } C_i \cap C_j \neq \{\emptyset\} \end{cases}$$

A clique σ of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a complete subgraph of $\mathcal{G}$. In case the clique σ may not be included in a superior sized clique σ' without loosing the completeness property, σ is said to be maximal. If σ is the clique with the highest cardinality (in terms of number of vertices $|\sigma|$) the clique is maximum. For the sake of clarity and simplicity, we will further identify a clique σ with the set of its coalitions given by $\mathcal{V}(\sigma)$. Any two coalitions C_1 and C_2 will be called *compatible* if they have no common players, i.e. $C_1 \cap C_2 = \emptyset$. In other words, if the nodes v_1 and v_2 are adjacent. More generally, the coalitions of $\mathcal{C}' \subset \mathcal{C}$ are compatible if the set of their nodes form a clique in the graph $\mathcal{G}$. Denote $\mathcal{G}_k$ the subgraph induced by the set of nodes of k^{th} highest weight. As an example, $\mathcal{G}_1$ is the subgraph induced by the nodes with maximum weights, i.e. the subgraph such that any node in $\mathcal{G}_1$ is a coalition giving each player its most preferred achievable share in the game.

The algorithm iteratively builds a forest (i.e. a set of disconnected trees) such that each tree is rooted by one of the maximal clique obtained at initialization and each vertex in each tree is a stable structure. A vertex that is not a leaf is a subgame stable structure (this is the anytime property of the algorithm) and a leaf is a stable structure for the game.

[3]See the regularity conditions over the set of coalitions for the non-emptiness of the set of core stable structures in all states of nature in [9].

Algorithm 1: Cliques-based algorithm for finding the set of stable structures

```
Data: Coalitions compatibility weighted graph $(\mathcal{G}, \Phi)$
Result: The set of stable structures $\mathcal{S}$
begin
    Step 1 (Initialize);
    $\mathcal{S} := \Sigma_1$ ($\Sigma_1$ set of maximal cliques in $\mathcal{G}_1$);
    $p(\mathcal{S}) = 0$ (vector of size $|\mathcal{S}|$, the structures in $\mathcal{S}$ have not been visited yet);
    Step 2 (Form the forest);
    while $\exists S \in \mathcal{S}$ s.t. $p(S) = 0$ do
        take $S$ s.t. $p(S) = 0$;
        $p(S) = 1$;
        if $\mathcal{W} \subset S$ or $\mathcal{F} \subset S$ (if the structure includes one or both of the set of players, no coalition can be added) then
            break;
        if $\mathcal{C}_S^{\max} = \emptyset$ ($\mathcal{C}_S^{\max}$ set of maximum valued coalitions compatible with $S$) then
            break;
        $\mathcal{S} := \mathcal{S} \backslash S$;
        for $S' \in \Sigma'_S$ ($\Sigma'_S$ set of maximal cliques in $\mathcal{C}_S^{\max}$) do
            $\mathcal{S} := \mathcal{S} \cup \{S' \cup S\}$ (complete the structures by the child $S' \cup S$);
            $p(S' \cup S) = 0$;
```

The set of leaves is the output of the algorithm. To go into further details, the algorithm starts by defining an initial set of structures, obtained by looking for the set Σ_1 of maximal cliques in the subgraph $\mathcal{G}_1$ (line 3). Each clique $S \in \Sigma_1$ is the root of a tree of stable structures (line 3). Starting from S, we look in $\mathcal{G} \backslash \mathcal{G}_1$ for the set $\mathcal{C}_S^{\max}$ of highest valued coalitions that are compatible with S (line 11). If $\mathcal{C}_S^{\max}$ is empty, the remaining players are left unmatched. Otherwise let Σ'_S be the set of maximal cliques in $\mathcal{C}_S^{\max}$ (line 14). A child of S in the structure tree is a structure $S \cup S'$ where $S' \in \Sigma'_S$ (line 15). Every time a structure is visited, it is tagged by a binary variable such that it will not be visited again (line 8). Furthermore, in case this structure becomes a parent, then the algorithm gets rid of it (line 13) to only hold the children (tagged as non-visited, line 16). The algorithm goes on iterating as long as it exists non-visited structures (line 6). We have the following propositions.

Proposition 1 *The algorithm converges in a finite number of iterations.*

Proposition 2 *The algorithm outputs stable structures.*

Proposition 3 *The algorithm outputs the set of stable structures.*

7. A DECENTRALIZED ALGORITHM: BACKWARD DEFERRED ACCEPTANCE

We now show that a modified version of the Gale and Shapley's Deferred Acceptance Algorithm (DAA) in its college-admission form with servers' preferences over groups of videos and videos' preferences over individual servers is a core stable matching mechanism for the many-to-one matching games with complementarities, peer effects and pairwise alignment of the preferences (see Algorithm 2).

The Backward Deferred Acceptance Algorithm (BDAA) is similar to the DAA in many aspects. It involves two sets of players that have to be matched. Every player from one side has a set of unacceptable players from the other side. In our case, a server and a video are always acceptable. As in DAA, the algorithm proceeds by proposals and corresponding acceptances or rejections. The main difference resides in the notion of counter-proposals, introduced to tackle the problem of complementarities.

To go into more details: Knowing the mutual impact factors of the videos and their own QoS gains, the servers compute all the possible coalitions they can form and the corresponding payoff vectors. They can thus build their preference lists over groups (Steps 1b). Then, every server r transmits to each of its acceptable videos the maximum achievable payoff it can achieve in the coalitions it can form with r (Step 1.c). Every video l can thus build its reduced list of preferences over individual servers: l prefers r_i to r_j if the maximum achievable throughput with r_i is strictly greater than its maximum achievable throughput with r_j (Step 1.d). BDDA then proceeds by rounds during which videos make proposals, servers make counter-proposals and videos accept or reject (from Step 2.a to Step 2.h). $L(r)$ is the list of all videos that have proposed at least once to server r. $L^*(r)$ is a dynamic list that is reinitialized to $L(r)$ at the beginning of every new round (Step 2.b). In each round of the algorithm, every unengaged video proposes to its most preferred server for which it has not yet proposed (Step 2.a). Every server receiving proposals adds the proposing players to its cumulated list of proposers and reinitializes its dynamic list (Step 2.b). Using $P^{\#}(r)$ it then searches for its most preferred coalition involving only videos from the dynamic list and emits a counter-proposal to these videos. This counter-proposal contains the payoff the videos can achieve in this coalition (Step 2.c). Each video compares the counter-proposals it just received with the best achievable payoffs obtained with the servers it has not proposed to yet (Step 2.d). If one of these best achievable payoffs is strictly greater than the best counter-proposal, the video rejects the counter-proposals and continues proposing (Step 2.d, Step 2.h). Otherwise, the video accepts its most preferred counter-proposal (Step 2.d). Given a counter-proposal, if all the videos accept it, then they are engaged to the server (Step 2.e). If at least one video does not agree, then the

Algorithm 2: Backward Deferred Acceptance

```
Data: For each server: The set of acceptable videos. For each video: The set of acceptable servers.
Result: A core stable structure $\mathcal{S}$
1  begin
2      Step 1: Initialization;
3          Step 1.a: All servers and videos are marked unengaged. $L(f) = L^*(f) = \emptyset$, $\forall f$;
4          Step 1.b: Every server $r$ computes possible coalitions with its acceptable videos, the respective videos' payoffs
           and emits its preference list $P^{\#}(r)$;
5          Step 1.c: Every server $r$ transmits to its acceptable users the highest payoff they can achieve in coalitions
           involving $r$;
6          Step 1.d: Every video $l$ emits its reduced list of preference $P'(l)$;
7      Step 2 (BDAA);
8          Step 2.a, Videos proposals: According to $P'(v)$, every unengaged video $v$ proposes to its most preferred
           acceptable server for which it has not yet proposed;
9          Step 2.b, Lists update: Every server $r$ updates its list with the set of its proposers:
           $L(r) \longleftarrow L(r) \cup \{\text{proposers}\}$ and $L^*(r) \longleftarrow L(r)$;
10         Step 2.c, Counter-proposals: Every server $r$ computes the set of coalitions with users in the dynamic list
           $L^*(r)$ and counter-proposes to the videos of their most preferred coalition according to $P^{\#}(r)$;
11         Step 2.d, Acceptance/Rejections: Based on these counter-proposals and the best achievable payoffs offered
           by the servers in Step 1.c to which they have not yet proposed, videoss accept or reject the counter-proposals;
12             Step 2.e: If all videos of the most preferred coalition accept the counter-proposal of an server $r$, all these
               videos and $r$ are marked engaged;
13             Step 2.f: Every unengaged server $r$ updates its dynamic list by removing videos both having rejected the
               counter-proposal and being engaged to another server: $L^*(r) \longleftarrow L^*(r) \backslash \{\text{engaged rejecters}\}$;
14         Step 2.g: Go to Step 2.c while there are unengaged servers with $L^*(r) \neq \emptyset$;
15         Step 2.h: Go to Step 2.a while there are unengaged videos that can propose.
```

server is unengaged (Step 2.e), it updates its dynamic list by removing the videos both having rejected its counter-proposal (Step 2.f) and being engaged with an other server. The counter-proposals continues up to the point when no unengaged server can emit any counter-proposal (Step 2.g). The current round ends and the algorithm enters a new round (Step 2.h). The algorithm stops when no more videos are rejected (Step 2.h). A stable matching is obtained.

Proposition 4 *In a many-to-one matching game with complementarities, peer effects and pairwise alignment of the preferences, the Backward Deferred Acceptance Algorithm (BDAA) is a core stable matching mechanism.*

Assume R servers and L videos . At each round, the number of counter-propositions by servers is at most R. From the pairwise alignment of the preferences, at each step there is at least one server whose counter-proposal has been accepted. Since the algorithm stops if no more videos are rejected or all rejected videos have already proposed to all their available servers, then in at most $L^2 - 2L + 1$, every non-engaged mobiles have proposed to all servers. Hence the complexity of our algorithm is $O(n^3)$ where $n = \max(R, L)$.

8. CONCLUSION

We have shown two new algorithms for finding core stable structures. The first algorithm is a centralized anytime enumerative one and the second is a decentralized deferred acceptance based one. We need to go further in the analysis of the complexity of the enumerative algorithm and search for the requirements over the coalitions compatibility graph to make it fall in classes of graphs with low complexity in the maximal cliques enumeration.

9. REFERENCES

[1] D. Gale and L.S. Shapley, College Admissions and the Stability of Marriage, The American Mathematical Monthly, Vol. 6, No. 1, pp 9-15, January 1962.

[2] A.Rapoportand and J.P.Kahan, Theories of Coalition Formation. Lawrence Erlbaum Associates, 1984.

[3] A.E. Roth and M.A.O. Sotomayor, Two-Sided Matching A Study In Game-Theoritic Modeling and Analysis, Econometric Society Monographs, No. 18, Cambridge University Press, 1990.

[4] T. Sandholm and K. Larson and M. Andesson and O. Shehory and F. Tohme, Coalition structure generation with worst case guarantees, Artificial Intelligence, Vol. 111, No. 1-2, pp 209-238, 1999.

[5] K. Cechlarova and A. Romero-Medina, Stability in coalition formation games, International Journal of Game Theory IJGT, Volume 20, pp 487-494, June 2001.

[6] F. Echenique and M.B. Yenmez, A solution to matching with preferences over colleagues, Volume 59, Issue 1, pp. 46 - 71, April 2007.

[7] T. Rahwan and N.R. Jennings, An improved dynamic programming algorithm for coalition structure generation, AAMAS, pp. 1417-1420, 2008.

[8] H. Keinanen, Algorithms for coalitional games, PhD Thesis, Turku School of Economics, 2011.

[9] M. Pycia, Stability and Preference Alignment in Matching and Coalition Formation, Econometrica, Vol. 80, No. 1, pp 323-362, January 2012.