The Least-Core of Threshold Network Flow Games

Yoram Bachrach

Abstract

Network flow games model domains where a commodity can flow through a network controlled by selfish agents. Threshold Network Flow Games (TNFGs) are a form of such games where an agent coalition wins if it manages to send a flow exceeding a certain threshold between a source and a target vertex. Cooperative game theory predicts the agents' actions in such settings with solutions such as the core, the set of stable distributions of a coalition's gains among its members. However, some games have empty cores, so every distribution is inherently unstable. When the core is empty, one must use a more relaxed notion of stability, such as the least-core. We examine several problems regarding the least-core in general and restricted TNFGs.

1 Introduction

Game theory analyzes interactions between selfish agents, with implications ranging from auctions to electronic commerce. A key aspect of agent interaction is *cooperation*. Cooperation between selfish agents requires careful planning as a stable coalition can only be formed if the resulting gains are distributed in appropriately. Game theory suggests possible distributions of the gains, formalized in *solution concepts* such as the core [11], and least-core [14].

We consider Threshold Network Flow Games (TNFGs), which model situations where a commodity (goods, information or traffic) can flow through a network where selfish agents control parts of the network. In TNFGs, agents form coalitions to guarantee a certain bandwidth (or a required flow) from a source to a target node. In our model, a principal offers a single unit of reward if a coalition can send a certain minimal flow between the source and the target, but is not willing to offer a higher reward for achieving a higher flow.

Our Contribution: We examine stable gain distributions in TNFGs. When the TNFG's core is empty , a less demanding solution, the ϵ-core, must be applied to find a "less unstable" distribution. We show that computing the ϵ-core of TNFGs is a hard, but provide polynomial algorithms for unit capacity networks and bounded layer-graphs with bounded integer capacities.

1.1 Preliminaries

A transferable utility coalitional game is composed of a set of n agents, $I = \{1, 2, \ldots, n\}$, and a characteristic function mapping any subset (coalition) of the agents to a rational value $v : 2^I \to \mathbb{Q}$, indicating the total utility these agents achieve together. We denote the set of all the agents except agent i as $I_{-i} = I \setminus \{i\}$. A game is *monotone* if for all coalitions $C' \subset C$ we have $v(C') \leq v(C)$. In a *simple* game, v only gets values of 0 or 1 ($v : 2^I \to \{0, 1\}$). We say a coalition $C \subset I$ *wins* if $v(C) = 1$, and say it *loses* if $v(C) = 0$. In simple monotone games we say i is a veto agent if she is present in all winning coalitions, so when $v(C) = 1$ we have $i \in C$. It is easy to see that in such games, i is a veto agent iff $v(I) = 1$ but $v(I_{-i}) = 0$. The characteristic function only defines the gains a *coalition* achieves, but does not say how they are to be distributed among the coalition's agents. An *imputation* $(p_1, \ldots, p_n)$ is a division of the gains of the grand coalition I among the agents, where $p_i \in \mathbb{Q}$, such that $\sum_{i=1}^{n} p_i = v(I)$. We call p_i the payoff of agent i, and denote the payoff of a coalition C as $p(C) = \sum_{i \in C} p_i$. Cooperative Game theory allows choosing the appropriate imputation for the game.

A basic requirement for a good imputation is *individual rationality*, stating that for all agents $i \in C$, we have $p_i \geq v(\{i\})$—otherwise, some agent is incentivized to work alone. Similarly, we say a coalition B *blocks* the payoff vector $(p_1, \ldots, p_n)$ if $p(B) < v(B)$, since B's members can split from the original coalition, derive the gains of $v(B)$ in the game, give each member $i \in B$ its previous gains p_i—and still some utility remains, so each member can get more utility. If a blocked payoff vector is chosen, the coalition is unstable. A prominent solution focusing on stability is the core [11]. The core of a game is the set of all imputations $(p_1, \ldots, p_n)$ that are not blocked by any coalition, so that for any coalition C, we have: $p(C) \geq v(C)$. The core can be empty, so every imputation is blocked by some coalition. In this case we must relax the core's requirements to get a solution. One model for this assumes that coalitions that only have a small incentive to drop-off from the grand coalition would not do so. This solution, which slightly core's inequalities is the ϵ-core [14]. The ϵ-core is the set of all imputations $(p_1, \ldots, p_n)$ such that for any coalition $C \subseteq I$, $p(C) \geq v(C) - \epsilon$.

Given an imputation p, the *excess* of C is the difference between C's value and payoff: $e(C) = v(C) - p(C)$. Under an ϵ-core imputation, the excess of *any* coalition is at most ϵ. If ϵ is large enough, the ϵ-core is non-empty. A natural question is finding the smallest ϵ such that the ϵ-core is non-empty, known as the *least-core*. Given a game G we consider $\{\epsilon | \text{the } \epsilon\text{-core of G is not empty}\}$. It is easy to see that this set is compact, and has a minimal element ϵ_{min}. The least-core of the game G is the ϵ_{min}-core of G.

2 The Least-Core in TNFGs

Network domains give rise to natural game theoretic problems. A flow network consists of a directed graph $G = \langle V, E \rangle$ with capacities on the edges $c : E \to \mathbb{Q}+$,

a distinguished source vertex $s \in V$ and a sink vertex $t \in V$. A flow through the network is a function $f : E \rightarrow \mathbb{Q}+$ which obeys the capacity constraints and conserves the flow at each vertex (except for the source and sink), meaning that the total flow entering a vertex must equal the total flow leaving that vertex. The value of a flow f (denoted $|f|$) is the net amount flowing out of the source (and into the sink). A maximum flow is a valid flow f^* whose magnitude $|f^*|$ is maximal of all possible flows, so for any valid flow f' we have $|f^*| \geq |f'|$.

A *threshold network flow domain* consists of a network flow graph $G = \langle V, E \rangle$, with capacities on the edges $c : E \rightarrow \mathbb{Q}$, a source vertex s, a target vertex t, and a set I of agents, where agent i controls the edge $e_i \in E$, and flow threshold k. A coalition $C \subseteq I$, controls the edges $E_C = \{e_i | i \in C\}$. Given a coalition C we denote by $G_C = \langle V, E_C \rangle$ the induced graph where the only edges are the edges that belong to the agents in C. We denote by f_C the maximum flow between s and t in G_C. In a *threshold network flow game (TNFG)*, a coalition C wins if it can achieve a k-flow between s and t and loses otherwise. The characteristic function of the game, v, is:

$$v(C) = \begin{cases} 1 & \text{if } f_C \geq k, \text{ so } E_C \text{ allows a flow of } k \text{ from } s \text{ to } t; \\ 0 & \text{otherwise;} \end{cases}$$

TNFGs are simple games, since v can only get a value of 0 or 1. A *Cardinal* Network Flow Game (CNFG) is defined in a similar manner, except the value of a coalition is the maximum flow it can send from s to t. In CNFGs the characteristic function is $v_{CNFG}(C) = f_C$.

We consider stable payoff distributions in TNFGs. Given a reward for allowing a certain flow between the source and the target, the coalition must decide how to distribute this reward to the agents who own the links. We wish to distribute the gains in a way that minimizes the incentives of any subset of agents to split off from the main coalition and form their own network. The appropriate game theoretic solution for this is the core. TNFGs are *simple* and *monotone* games, where the core is closely related to veto players. A folklore theorem (see [15]) states that in simple monotone games, if there are no veto agents then the core is empty, and if there are veto agents then the core is the set of imputations that distribute all the gains solely to the veto agents. Thus, computing the core in such games simply requires returning a list of veto players in that game, and checking if the core is non-empty simply requires testing if the game has any veto players. Unfortunately, TNFGs rarely have veto agents. If there are two disjoint edge subsets that allow a k-flow, the core is empty [1].

Computational problems in TNFGS: we now define several important problems regarding TNFGs: IMP-C, IMP-EC, IMP-LC, CNE, ECNE, LCV, ME. The core, ϵ-core and the least-core are all solution concepts that may contain more than one possible imputation (or even, an infinite number of imputations). One key question given a TNFG is testing whether a certain imputation $p = (p_1, \ldots, p_n)$ is stable (i.e. whether it is in the core, ϵ-core or the least-core).

[1] Consider a unit capacity graph, and a target flow of one unit. In this case, if there are two or more edge disjoint paths from the source to the target, then the core is empty.

We denote these problems for the core, $\epsilon - core$ and least-core as IMP-C, IMP-EC and IMP-LC. The core and ϵ-core may be empty, so another problem is testing for their emptiness, which we denote as CNE (for core non-emptiness) and ECNE (ϵ-core non-emptiness). Another problem is LCV (least-core value), computing the ϵ_{min} value of the least-core (i.e the minimal ϵ such that the ϵ-core is non-empty). A solution for IMP-C and CNE in TNFGs was given in [5], using a simple polynomial method for finding veto agents. Since the core in TNFGs may be empty, we propose using the relaxed stability notions of the ϵ-core and the least-core. We show the above problems are closely related to the problem ME (maximal excees), where we are given a TNFG with characteristic function v and an imputation $p = (p_1, \ldots, p_n)$ and are asked to compute the maximal excess of any coalition: $\max_{C \subseteq I} v(C) - p(C)$. Both LCV and ME require outputting a rational value, but we consider the decision version of these problems, where in the input we are given a rational number q and are asked whether the solution to the LCV or ME problem is greater than the specified value.

We consider the relation between TNFGs and another class of games, Weighted Voting Games (WVGs). We show that the problem of LCV is NP-hard in TNFGs, using a similar hardness result [8] for WVGs. WVGs are simple games with n agents $I = \{1, 2, \ldots, n\}$, where each agent i has a weight w_i. Given a coalition $C \subseteq I$, we denote $w(C) = \sum_{i \in C} w_i$. The game has a threshold k, and a coalition C wins if $w(C) \geq k$, and loses otherwise. We use the fact that TNFGs can easily express any WVG to obtain several negative results.

Theorem 1. *LCV in TNFGs is NP-hard*

Proof. Elkind et al. show [8] that LCV is NP-hard in WVGs. We reduce an instance of LCV in a WVG to an instance of LCV in a TNFG. Given the WVG instance with weights $w_1, \ldots, w_n$, we construct a TNFG. The source s is connected to the target t by n edges, so that edge $e_i = (s, t)$ has capacity c_i equal to the weight w_i in the WVG. Any coalition C achieves a flow of $\sum_{\{i|e_i \in E\}} c_i = \sum_{i \in C} w_i = w(C)$. The TNFG has the same threshold as the WVG threshold. Under these thresholds, coalition C wins in the TNFG iff it wins in the original WVG. The original and reduced instances define the same game (have an identical characteristic function) and have the same least-core value, so LCV in a TNFG is NP-hard as well. □

We now provide a negative result for the ϵ-core, and then provide positive results for restricted TNFGs. We begin with the IMP-EC problem where we are given a TNFG, an imputation $p = (p_1, \ldots, p_n)$ and a value ϵ, and are asked whether p is in the $\epsilon - core$ of the game.

Theorem 2. *IMP-EC and ME are coNP-complete*

Proof. IMP-EC is *in* coNP, as one can compute the excess of a given coalition using any polynomial max-flow algorithm and test if it is higher than ϵ or not. The results in [8] show that testing if an imputation is in the ϵ-core in WVGs is NP-Hard. Using construction of Theorem 1, we express any WVG as a TNFG. An imputation in the constructed TNFG is in the ϵ-core iff it is in the ϵ-core of

the original WVG. Finally, reducing IMP-EC to ME is trivial: by definition an imputation is in the ϵ-core if the maximal excess is at most ϵ. □

ME is coNP-complete in general graphs, so we examine restricted cases. One important restricted case is *unit-capacity* graphs, where all the edges have a capacity of 1. Several results for *cardinal* network flow games (CNFGs) in unit-capacity graphs are given in [13]. We provide positive results for *threshold* network flow games (TNFGs). One result regarding CNFGs on unit-capacity graphs is that they always have non-empty cores. However, is not the case in TNFGs, as shown in the following example.

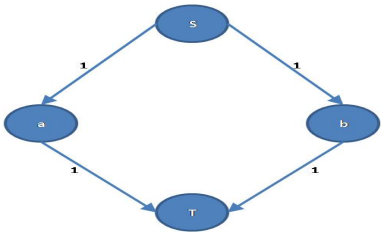

Figure 1: Example of a unit capacity TNFG with empty core.

Consider distributing a total reward of 1 among the edges in Figure 1. Under any such imputation, either the path $A = (s, a), (a, t)$ or the path $B = (s, b), (b, t)$ has a payoff of at most 0.5. Thus, the excess of at least one coalition C_A or C_B must be at least 0.5, and for any $\epsilon > 0.5$, the ϵ-core is empty, so the core is also empty. Note that the 0.5-core is not empty, as the imputation giving each edge 0.25 has a excess of exactly 0.5 for the coalitions C_A and C_B, and any other winning coalition has a lower excess.

We show that ME *is solvable in polynomial time* for *unit-capacity* graphs, and use this to compute the least-core for unit-capacity TNFGs. We first examine the relation between ME and the min-cost-flow problem [17], defined as follows. We are given a network flow graph, a target flow value d and a cost function $a : E \to \mathbb{Q}$, mapping each edge to the cost of sending a *unit* of flow through that edge. Given a flow $f(e)$ through edge e, the cost of the flow through the edge is $f(e) \cdot a(e)$. We define the cost of a flow f as $a(f) = \sum_{e \in E} f(e) \cdot a(e)$. We are asked to find a d-flow through the network, of minimal cost.

The Min-Cost Flow uses a pricing scheme where the cost of an edge is proportional to the flow sent through the edge. We call this pricing scheme *linear* pricing, as the cost of an edge increases linearly in the flow. A different pricing scheme is requiring a payment for *using* an edge, so the cost of an edge is $a(e)$ if $f(e) > 0$, and the cost is 0 if $f(e) = 0$. We call this pricing scheme the *All-Or-Nothing scheme* (AON pricing), as it either takes the full cost of an edge (for edges e with flow $f(e) > 0$), or takes nothing at all (for edges e for which the flow is $f(e) = 0$). The problem of Min-Cost Flow under AON-pricing is defined as follows. We are given a network flow graph, a target flow value d and a cost function $a : E \to \mathbb{Q}$, mapping each edge to the cost of sending *any non-zero*

flow through that edge. Given a flow $f(e)$ through edge e, the AON-cost of the flow through e is $n(e) = a(e)$ if $f(e) > 0$ and $n(e) = 0$ if $f(e) = 0$. We define the AON-cost of a flow f as $a^{AON}(f) = \sum_{e \in E} n(e)$. We are asked to find a d-flow through the network, f, of minimal AON-cost (i.e. to minimize $a^{AON}(f)$).

Lemma 1. *ME and AON-Min-Cost-Flow are equivalent problems when we set the AON-Min-Cost-Flow edge costs to be the edge payoff in the ME imputation: $a(e) = p_e$ for any edge e.*

Proof. Consider a k-flow f sending flow only through edges C_f. The payoff $p(C_f)$ under the imputation is exactly the AON-cost of f, $a^{AON}(f)$. ME finds the minimal $p(C_f)$ where f is a k-flow, and AON-Min-Cost finds the minimal $a^{AON}(f)$ where f is a k-flow, so their solution is the same. □

Theorem 2 shows that ME is coNP-complete, so due to Lemma 1, AON-Min-Cost-Flow is also coNP-complete. However, the similar Min-Cost Flow problem (under *linear* pricing) is solvable in polynomial time [17]. We show that for *unit-capacity* graphs, AON-Min-Cost is solvable in polynomial time. We use a lemma from [17] used in the analysis of the Min-Cost Flow problem, regarding augmenting min-cost flows [2]. A min-cost flow is a flow f, such that for any other flow f' such that $|f| = |f'|$ we have $a(f') \geq a(f)$, i.e. it is impossible to achieve a flow of equal magnitude with a lower cost (under linear pricing).

Lemma 2. *If f is a min-cost flow, then any flow obtained from f by augmenting along a minimum-cost path is also a min-cost flow.*

We show that in unit-capacity graphs, one of the optimal k-flows is an all-or-nothing flow. An *All-Or-Nothing Flow (AON Flow)* is a flow f such that any edge is used up to its maximal capacity, or completely unused, so for any edge e we either have $f(e) = c(e)$ or we have $f(e) = 0$.

Theorem 3. *For any unit-capacity graph G and an achievable integer flow threshold k, there is a min-cost k-flow f (under* linear *pricing) that is an AON-flow, i.e. either fully uses an edge, sending the maximal possible flow of 1 unit, or does not use the edge at all, sending a flow of 0 through it.*

Proof. One possible way of finding a min-cost flow is starting with an empty flow, and repeating the process of augmenting the flow through a min-cost augmenting path until the target flow threshold of k is reached [17]. Due to Lemma 2, after each augmentation we end up with a min-cost flow. The initial residual graph has capacities of 1 on all the edges. Thus, the first augmenting path has a capacity of one unit on all the edges along it. The first augmentation fully uses the edges along a certain path. Thus, in the residual graph after the augmentation, all the edges have either a capacity of 1, or -1 for the reverse direction. This, in turn, means that the next augmenting path would also have a capacity

[2] For a complete introduction to the min-cost network flow problem we refer the reader to [17]. A min-cost augmenting path is simply a minimal cost s-t path in the residual graph. We use the lemma to prove certain properties of min-cost flow solutions.

of 1 all along it. Thus the next augmentation either fully uses some edges, or completely stops using some previously unused edges. Thus, each augmentation is done through a path of capacity one, and after each augmentation some edges are changed from being unused to completely used, or from being fully used to completely unused. Since the flow's magnitude increases by one unit after each step, the process terminates after k such steps, with a flow such that each edge is either fully used or completely unused. □

Lemma 3. *In unit-capacity graphs, under the same cost function $a(e)$, the AON-cost of a flow f, $a^{AON}(F)$, is bounded from below by the linear cost of a flow, $a(f)$, i.e $a(f) \leq a^{AON}(f)$*

Proof. The graph is a unit-capacity graph, so for any edge $f(e) \leq 1$. We note that the cost of each edge only increases when we use AON-pricing. For used edges the full cost is incurred even if the edge is only partially used, and for unused edges both pricing schemes charge nothing: $a(f) = \sum_{e \in E} a(e) \cdot f(e) = \sum_{e \in E | f(e) > 0} a(e) \cdot f(e) + \sum_{e \in E | f(e) = 0} a(e) \cdot f(e) \leq \sum_{e \in E | f(e) > 0} a(e) \cdot 1 + \sum_{e \in E | f(e) = 0} a(e) \cdot f(e) = a^{AON}(f)$ □

Lemma 4. *In unit-capacity graphs, under the same cost function $a(e)$, the cost of an AON-flow is the same under linear-pricing and AON-pricing. In other words, if f is a AON-flow, then $a(f) = a^{AON}(f)$.*

Proof. Similarly to the analysis in Lemma 3 we decompose the flow to used and unused edges, and note that the cost for each edge is the same under the two pricing models. Under both models, an unused edge incurs no cost, and a fully used edge incurs a cost of $a(e)$, since for a AON-flow if $f(e) > 0$ then $f(e) = c(e) = 1$. Thus the following holds: $a(f) = \sum_{e \in E} a(e) \cdot f(e)$ $= \sum_{e \in E | f(e) > 0} a(e) \cdot f(e) + \sum_{e \in E | f(e) = 0} a(e) \cdot f(e) = \sum_{e \in E | f(e) > 0} a(e) \cdot 1 + \sum_{e \in E | f(e) = 0} a(e) \cdot 0 = a^{AON}(f)$ □

Theorem 4. *In unit-capacity graphs, under a certain cost function $a(e)$, a min-cost k-flow (under linear pricing) is also a AON-min-cost k-flow.*

Proof. Due to Theorem 3, a certain AON-flow f^* is a min-cost k-flow, under linear pricing. Thus, for any other k-flow f' we have $a(f') \geq a(f^*)$. Due to Lemma 4, since f^* is a AON-flow, then $a(f^*) = a^{AON}(f^*)$. Due to Lemma 3, for any k-flow f we have $a^{AON}(f) \geq a(f)$. Thus for any k-flow f, $a^{AON}(f) \geq a(f) \geq a(f^*) = a^{AON}(f^*)$, so f^* is a AON-min-cost k-flow. □

Corollary 1. *ME is in P for unit-capacity TNFGs.*

Proof. Lemma 1 shows that ME and AON-Min-Cost-Flow are equivalent. Due to Theorem 4 solving AON-Min-Cost simply requires solving Min-Cost Flow (under the linear pricing model). Polynomial algorithms for solving Min-Cost-Flow that rely on the all-or-nothing augmenting paths used in Theorem 3 are given in [17]. Note that the proof in Theorem 3 is constructive, and allows finding such a Min-Cost Flow which is also an AON-flow for unit capacity graphs, thus solving the ME problem for such graphs. □

Theorem 5. *ECNE is in P for unit-capacity TNFGs.*

Proof. We consider an exponential-size program linear program (LP) for computing an imputation in the ϵ-core. Consider all 2^n possible coalitions over n players, $C_1, \ldots, C_{2^n}$. The ϵ-core's definition is a LP over the variables $p_1, \ldots, p_n$, with a constraint for each coalition:

Table 1 Exponential LP for the ϵ-core

Feasible $(p_1, \ldots, p_n)$ **s.t.:**	
$\forall C : v(C_1) - \sum_{i \in C} p_i \leq \epsilon$	(Coalition constraints)
$\sum_{i=1}^{n} p_i = v(I)$	(Imputation constraint)

A violated constraint is a coalition C_j, such that $v(C_j) - \sum_{i \in C_j} p_i > \epsilon$. TNFGs are simple games, so any losing coalition C_j cannot yield a violated constraint, as $v(C_j) = 0$. Thus the violated constrain is due to a winning coalition such that $p(C_j) < v(C_j) - \epsilon = 1 - \epsilon$. The algorithm of Corollary 1 returns a maximal excess coalition C, where $e(C) = v(C) - \sum_{i \in C} p_i$ is maximal. If $e(C) > \epsilon$, we have a violating constraint: $v(C) - \sum_{i \in C_2} p_i > \epsilon$. If $e(C) \leq \epsilon$, then all the constrains hold: $e(C)$ is the maximal excess, so for any $C_j \neq C$ we have $d(C_j) = v(C_j) - \sum_{i \in C_j} p_i \leq e(C) \leq \epsilon$. Thus this algorithm can serve as a separation oracle for the above LP, and we can solve it in polynomial time. Thus, given ϵ we can, in polynomial time, compute an imputation in the ϵ-core if one exists, or reply that no such imputation exists. □

Corollary 2. *LCV is in P for unit-capacity TNFGs.*

Proof. Due to Theorem 5, for a given ϵ we can test if the ϵ-core is non-empty. We can perform a binary search on the minimal value of ϵ such that the ϵ-core is not empty. Since the maximal value of a coalition in TNFGs is 1, we search for ϵ between 0 and 1. This finds the minimal ϵ such that the ϵ-core is not empty, up to any given degree of accuracy. We can then use the algorithm from Theorem 5 to return a least-core imputation. Alternatively, we can consider program in Theorem 5's proof as a program over the variables $p_1, \ldots, p_n, \epsilon$ and set the target function to be $\min \epsilon$ (rather than being a feasibility program). □

We propose a polynomial algorithm for computing the least-core of a TNFG with *non-unit capacities* under restrictions on the graph's structure. A graph $G = \langle V, E \rangle$ is a *layer graph* with w layers if we can partition the vertices into layers $L_0, L_1, L_2, \ldots, L_w$ (where each L_i is a subset of vertices $L_i \subseteq V$ and the L_i's are disjoint so if $i \neq j$ then $L_i \cap L_j = \emptyset$) such that edges only occur between vertices of consecutive layers. In other words, we require that if $(u, v) \in E$ then we have $u \in L_i$ and $v \in L_{i+1}$ for some $0 \leq i < w$. A layer graph is *h-bounded* if each layer L_i has at most h vertices (so $|L_i| \leq h$ for any $0 \leq i \leq w$). For flow networks we denote the source s as the first layer so $L_0 = \{s\}$ and the target t as the last layer $L_w = \{t\}$ (in layer graphs vertices other than s or t in the source/target layers do not influence the maximum flow). The bound h

only applies to the number of vertices in each layer so the number of layers is unbounded. A flow network has *c-bounded integer capacities* if all edge capacities are integers bounded from above by c, so for any $e \in E$, $c(e) \in \{0, 1, 2, \ldots, c\}$.

In Lemma 1 we noted that solving ME is equivalent to AON-Min-Cost-Flow, where for any edge e we have $a(e) = p_e$ (edge costs are the payoffs in the ME imputation). We show that AON-Min-Cost-Flow can be solved in polynomial time in bounded layer-graphs with bounded integer capacities.

Theorem 6. *AON-Min-Cost-Flow is in P for bounded layer-graphs with bounded integer capacities.*

Proof. Consider a layer-graph with $w + 1$ layers $L_0, L_1, \ldots, L_w$. Denote the source s as $L_0 = \{s\}$, the target as $L_w = \{t\}$, the bound on the number of vertices in each layer as h and the maximal capacity as c. Denote by $p(e)$ the AON-price of an edge e in the AON-Min-Cost-Flow input. The maximal achievable flow is at most $b = h \cdot c$ as there are at most h edges going out of s and each has a capacity at most c. The border between any two layers forms an $s - t$-cut in the graph, so the incoming flow into any vertex is at most $b = h \cdot c$, the bound on the flow leaving s. Given a layer $L_i \subset V$ of l_i vertices $v_1^i, v_2^i, \ldots, v_{l_i}^i$ we define an *incoming flow configuration* for that layer, representing the amount of flow entering each vertex in that layer. An *incoming flow configuration* for layer L_i is a sequence $(f_1, f_2, \ldots, f_{l_i})$ of integers where $0 \leq l_j \leq b$. A layer has at most h vertices and each has a possible incoming integer flow of $0 \leq f_j \leq b$, so there are at most $(b + 1)^h = (h \cdot c + 1)^h$ possible flow configurations for any layer L_i. Denote the bound on the possible flow configurations for any layer as $q = (b + 1)^h = (h \cdot c + 1)^h$. Since h and c are constants, q is constant.

We provide an algorithm to compute the minimal cost to achieve a d-flow to the target. It operates by iterating through the layers and computing the minimal cost to achieve a specific flow configuration. We say an edge $e = (u, v)$ occurs before layer L_i if $u \in L_a$ and $v \in L_b$ where $a \leq i$ and $b \leq i$ (since no edges are allowed *within* a layer, this also means that $a \leq i - 1$). We say a flow configuration $f = (f_1, f_2, \ldots, f_{l_i})$ for L_i is achievable with price p if there is a subset of edges E' that all occur before layer L_i with total cost $\sum_{e \in E'} p(e) \leq p$ which allows *simultaneously* sending a flow of f_1 to v_1^i, f_2 to v_2^i and so on through a flow of f_{l_i} to $v_{l_i}^i$ using only the edges of E'.

Consider a flow configuration $f^i = (f_1^i, f_2^i, \ldots, f_{l_i}^i)$ for L_i, a flow configuration $f^{i+1} = (f_1^{i+1}, f_2^{i+1}, \ldots, f_{l_{i+1}}^{i+1})$ for layer L_{i+1} and an edge subset E' that occur between L_i and L_{i+1}. Suppose that p is the minimal price such that f_i is achievable for layer L_i with price p. Given f_i and E' we may achieve various flow configurations in L_{i+1}: if each vertex in L_i has an incoming flow as specified in f_i, we may route this flow through E' to the vertices in L_{i+1} in several ways. It is possible to check if f^{i+1} is achievable from f^i through the edges E' in polynomial time. To do this, we create a small flow network with a source s connected to each of the vertices in L_1 through edge with capacities as specified in f_i (i.e. s is connected to $v_j \in L_i$ through an edge of capacity f_j^i), maintain the edges E' connecting L_i and L_{i+1} with their current capacities and connect

each vertex in L_{i+1} to t with capacities as specified in f^{i+1} (i.e. $v_j \in L_{i+1}$ is connected to t through an edge of capacity f^{i+1}_j). If this created network allows a flow of $\sum_{j=1}^{l_{i+1}} f^{i+1}_j$ then given flow configuration f^i for L_i we can achieve the flow configuration f^{i+1} for layer L_{i+1} by using only the edges E' between L_i and L_{i+1}. This check can be done in polynomial time using a max-flow algorithm. The total cost of the edge subset E' is $p(E') = \sum_{e \in E'} p_e$, so if f^i is achievable with cost p then f^{i+1} is achievable with cost of at most $p + p(E')$ by simply adding the edges E' and routing the flow through them appropriately. We denote by $achievable-flow(f^i_j, E', f^{i+1}_k)$ the algorithm that checks whether given a flow configuration f^i_j for layer i and an edge subset E' (occurring between L_i and L_{i+1}) we can achieve a flow configuration f^{i+1}_k for layer L_{i+1}.

For each layer L_i, our algorithm maintains a bound p^i_j for the minimal cost to achieve any of the possible q flow configurations (for $j \in \{1, \ldots, q\}$). We denote the set of all flow configurations for layer i as Q^i. Layer L_w contains the single target vertex so $L_w = \{t\}$ and a valid flow configurations for L_w is simply the possible flow values to the target vertex. We denote the flow configuration where a flow of d enters the target vertex as f_d so we denote the cost of achieving a flow of k is $p^w_{f_d}$. Layer L_0 contains the single source vertex so $L_0 = \{s\}$ so similarly a valid flow configuration for L_0 is simply the possible flow values to the target vertex. We denote the cost of a flow configuration where s has an "incoming" flow of x as $p^0_{f_x}$. Since s is the source it may achieve an any "incoming" flow at a zero cost. When iterating through layer L_i our algorithm relaxes the bounds by examining any possible edge subset E' occurring between L_i and L_{i+1}. We denote by q^i_j the j'th flow configuration for the $i'th$ layer in the graph. There are at most q flow configurations for each layer, so $1 \leq j \leq q$.

1. For $i = 1$ to w:
   (a) For each $q^i_j \in Q^i$: $p^i_j = \infty$
   (b) For $x = 0$ to $b = h \cdot c$: $p^0_{f_x} = 0$
2. For $i = 1$ to w:
   (a) For each $q^i_j \in Q^i$, each $q^{i+1}_k \in Q^{i+1}$ and each $E' \subseteq E_{L_i}$:
      i. If $achievable - flow(q^i_j, E', q^{i+1}_k)$: $p^{i+1}_k = \min(p^{i+1}_k, p^i_j + p(E'))$
3. return $p^w_{f_d}$

An induction through the layers shows that the algorithm correctly computes p^i_j for any layer i and configuration j for layer i. It returns $p^w_{f_d}$, the minimal cost for a d-flow to the t, i.e. the AON-Min-Cost-Flow solution. The update loop's runtime is $w \cdot |Q^i| \cdot |Q^{i+1}| \cdot 2^{|E_{L_i}|}$. For any i the configuration number $|Q^i|$ is bounded by the constant q. Also, $|E_{L_i}|$ is the number of edges between L_i and L_{i+1}, bounded by the constant h^2. Thus the runtime is bounded by $w \cdot q^2 \cdot 2^{h^2}$ or by $g \cdot w$ where g is a constant, so this is a polynomial algorithm. The runtime is bounded by $w \cdot q^2 \cdot 2^{h^2}$ where $q = (b+1)^h = (h \cdot c + 1)^h$, so it can be expressed as $O(f(h, c) \cdot w)$ so this is a *fixed parameter tractable* (FPT) algorithm[3]. □

[3]Fixed parameter tractability with parameter α (α-FPT) means the running time is $f(\alpha) \cdot$

The equivalence of ME and AON-Min-Cost-Flow obtains the following.

Corollary 3. *ME is in P for TNFGs over bounded layer-graphs with bounded integer capacities.*

Theorem 7. *ECNE and LCV are in P for TNFGs over bounded layer-graphs with bounded integer capacities.*

Proof. Similarly to Theorem 5, we use the ME algorithm of Corollary 3 as a separation oracle for the ϵ-core LP, so ECNE can be solved in polynomial time. Similarly to Theorem 2, we perform a binary search on the minimal value of ϵ that makes the ϵ-core non-empty, so LCV is solvable in polynomial time. □

3 Related Work and Conclusions

We examined computing the core,ϵ-core and the least-core in TNFGs. The core was introduced in [11], and the ϵ-core and least-core relaxations for the case where it is empty were presented in [14]. An early treatment of computational aspects of such solutions was given in [7]. One restricted domain that has received much attention is the weighted voting games (WVG) model of decision making. Computing the least-core of WVGs was analyzed in [9].

We focused on solving TNFGs. The similar CNFG model was introduced in [13], showing that unit capacity CNFGs have non-empty cores. Further core related questions in CNFGs were studied in [10]. Other network task games, such as network formation, connectivity and security were studied in [18, 12, 6, 3]. Despite the superficial similarity between CNFGs and TNFGs, they have very different computational and game theoretic properties. TNFGs were introduced in [4], which considered the Banzhaf index. An extended version of that paper [5] also considered computing the core of TNFGs. Further work [16] focused on solving TNFGs using the core-related Cost of Stability introduced in [2, 1] for weighted voting games and general cooperative games.

We showed that, as opposed to CNFGs, in TNFGs the core can be non-empty even if the graph has unit-capacities, and that for general graphs, even finding the maximal excess of a coalition under an imputation (the ME problem) is coNP-complete. We provided polynomial-time algorithms for computing the ϵ-core and least-core for unit-capacity graphs and bounded layer-graphs with bounded integer capacities. Some questions remain open. First, one might examine other game theoretic solutions in TNFGs. Also, it would be interesting to examine other restrictions that allow tractably computing the least-core.

References

[1] Y. Bachrach, E. Elkind, R. Meir, D. Pasechnik, M. Zuckerman, J. Rothe, and J. Rosenschein. The cost of stability in coalitional games. *SAGT*, 2009.

$p(n)$ where $f(\alpha)$ may be any function (e.g. an exponential or even superexponential function), and $p(n)$ is a polynomial function in the input length n.

[2] Y. Bachrach, R. Meir, M. Zuckerman, J. Rothe, and J. Rosenschein. The cost of stability in weighted voting games. In *AAMAS*, 2009.

[3] Y. Bachrach and E. Porat. Path disruption games. In *AAMAS*, 2010.

[4] Y. Bachrach and J. S. Rosenschein. Computing the Banzhaf power index in network flow games. In *AAMAS*, 2007.

[5] Y. Bachrach and J. S. Rosenschein. Power in threshold network flow games. *Autonomous Agents and Multi-Agent Systems*, 2009.

[6] Y. Bachrach, J. S. Rosenschein, and E. Porat. Power and stability in connectivity games. In *AAMAS*, 2008.

[7] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 19(2):257–266, 1994.

[8] E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. Wooldridge. Computational complexity of weighted threshold games. In *AAAI*, 2007.

[9] E. Elkind and D. Pasechnik. Computing the nucleolus of weighted voting games. In *SODA*, 2009.

[10] Q. Fang, R. Fleischer, J. Li, and X. Sun. Algorithms for core stability, core largeness, exactness, and extendability of flow games. *Computing and Combinatorics*, pages 439–447, 2007.

[11] D. B. Gillies. *Some theorems on n-person games*. PhD thesis, 1953.

[12] S. Goyal and F. Vega-Redondo. Network formation and social coordination. *Games and Economic Behavior*, 50(2):178–207, 2005.

[13] E. Kalai and E. Zemel. Generalized network problems yielding totally balanced games. *Operations Research*, 30:998–1008, September 1982.

[14] M. Maschler, B. Peleg, and L. S. Shapley. Geometric properties of the kernel, nucleolus, and related solution concepts. *Math. Oper. Res.*, 1979.

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

[16] E. Resnick, Y. Bachrach, R. Meir, and J. Rosenschein. The cost of stability in network flow games. *MFCS*, 2009.

[17] R. E. Tarjan. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, 1983.

[18] A. van den Nouweland. *Models of network formation in cooperative games*. Cambridge University Press, 2005.