

Algorithms for Core Stability, Core Largeness, Exactness, and Extendability of Flow Games^{*}

Qizhi Fang¹, Rudolf Fleischer², Jian Li³, and Xiaoxun Sun⁴

¹ Department of Mathematics, Ocean University of China, Qingdao, China
Email: qfang@ouc.edu.cn

² Department of Computer Science and Engineering, Fudan University, Shanghai, China
Email: rudolf@fudan.edu.cn

³ Department of Computer Science, University of Maryland, College Park, USA
Email: lijian@cs.umd.edu

⁴ Department of Mathematics and Computing University of Southern Queensland
Toowoomba, Australia
Email: sunx@usq.edu.au

Abstract. We study core stability and some related properties of flow games defined on simple networks (all edge capacities are equal) from an algorithmic point of view. We first present a sufficient and necessary condition that can be tested efficiently for a simple flow game to have a stable core. We also prove the equivalence of the properties of core largeness, extendability, and exactness of simple flow games and provide an equivalent graph theoretic characterization which allows us to decide these properties in polynomial time.

1 Introduction

Originating with the pioneering work of Ford and Fulkerson [11], network flow models have been thoroughly investigated and found applications in various fields. In cooperative game theory, flow games were first discussed by Kalai and Zemel [15, 16]. They arise from the profit distribution problem related to the maximum flow in a network. Afterwards, many results have been obtained on flow games, most of them focusing on the core, the most important solution concept in cooperative game theory. Basically, the *core* of a game is a set of profit allocations such that no subset of the players could obtain a bigger profit by leaving the grand coalition of all players. The characterization of the core of flow games is due to Kalai and Zemel [15, 16] and Deng *et al.* [4]. They showed that flow games are totally balanced (which means that every subgame has a non-empty core) and the profit allocations corresponding to minimum cuts in the network always belong to the core. On the other hand, for flow games with general edge capacities it is $co\mathcal{NP}$ -complete to check whether an allocation belongs to the core [9].

^{*} Different parts of this paper have appeared in a preliminary form in [24] and [10].

Von Neumann and Morgenstern proposed stable sets as an important solution concept of cooperative games [19], which turned out to be very useful in the analysis of bargaining situations. A *stable set* is a set of profit allocations not dominating each other, and any other profit allocation is dominated by some element in this set. Unfortunately, stable set may not always exist [18], and it seems difficult to characterize its fundamental properties.

Although, in general, the core and the stable set are different, Shapley [21] proved that for convex games the core is the unique stable set. Hence the question arises: when do the core and the stable set coincide, and how can we decide core stability? Several sufficient conditions for core stability have been discussed in the literature. *Subconvexity* of the game and *largeness* of the core were introduced by Sharkey [22] who showed that subconvexity implies largeness of the core, which in turn implies core stability. In an unpublished paper, Kikuta and Shapley [17] investigated another concept, later called *extendability* of the game by Gellekom *et al.* [12], and proved that it is necessary for core largeness and sufficient for core stability.

Only few results are known about the core stability of concrete cooperative game models. Solymosi and Raghavan [23] studied assignment games, and Bietenhader and Okamoto [2] studied minimum coloring games defined on perfect graphs. Jain and Vohra [13] recently showed that there exist algorithms for testing the existence of a stable set and core stability for general cooperative games, solving an open problem by Deng and Papadimitriou [5]. However, it is unlikely that these algorithms will be efficient.

The main purpose of this paper is to identify those flow games whose cores are stable and to investigate the algorithmic issues on core stability and related properties. We restrict our attention to flow games defined on simple networks, called *simple flow games*, where all edge capacities are equal. Our key contributions can be summarized as follows:

1. We show that a simple flow game has a stable core if and only if the network is a pseudo balanced directed acyclic graph (see Section 3.2 for definitions), which can easily be tested in $O(\min(n^{3/2}, m^{1/2})m + nm)$ time, where n is the number of nodes and m the number of edges in the network.
2. We show that core largeness, extendability, and exactness are equivalent for simple flow games. These properties are further equivalent to the graph theoretic property that the flow network is *cut-normal*, i.e., every minimal (s, t) -cut is already a minimum (s, t) -cut. We show how to test this property in $O(m^2\alpha(m, n))$ time, where $\alpha(m, n)$ is the inverse of the Ackermann function.

This paper is organized as follows. In Section 2, we give the definitions and review some known results on the cores of flow games. In Section 3, we characterize those flow games with stable cores. In Section 4, we discuss the equivalence of core largeness, extendability, and exactness of simple flow games, and propose an efficient algorithm for testing these properties. We conclude with some open problems in Section 5.

2 Definitions and Preliminaries

2.1 Graphs

A *flow network* is a directed graph $G = (V, E; \omega; s, t)$, where V is the node set, E is the edge set, $\omega : E \rightarrow \mathbb{R}^+$ is the edge capacity function, and $s, t \in V$ are the source and the sink of the network, respectively. Without loss of generality, we may assume that every node lies on some (s, t) -path. G is a *simple flow network* if all edge capacities are equal (we may assume they are all equal to one). Throughout this paper, we denote a simple flow network by $G = (V, E; s, t)$.

For $F \subseteq E$, $G[F]$ denotes the subgraph of G with node set V and edge set F , and $G \setminus F$ the subgraph of G with node set V and edge set $E \setminus F$. For $v \in V$, $out(v)$ ($in(v)$) is the set of outgoing (incoming) edges at v .

If node sets S and T partition the node set V into two parts such that $s \in S$ and $t \in T$, then the set F of edges going from a node in S to a node in T is an (s, t) -cut. We denote the indicator vector of F by $\mathbf{1}_F \in \{0, 1\}^{|E|}$, where $\mathbf{1}_F(e) = 1$ if $e \in F$, and 0 otherwise. The *capacity* of F is the sum of its edge capacities. F is a *minimal* cut if no proper subset of F is also an (s, t) -cut; it is a *minimum* cut if it has the smallest capacity among all (s, t) -cuts. A simple flow network G is *cut-normal* if every minimal (s, t) -cut is already a minimum (s, t) -cut, or equivalently, if every (s, t) -cut contains a minimum (s, t) -cut.

A directed graph is a *DAG* (*directed acyclic graph*) if it does not contain a directed cycle. A *2-terminal DAG* is a DAG with two distinguished nodes (called *terminals*), say s and t , such that $in(s) = out(t) = \emptyset$ and every other node of G appears in at least one simple (s, t) -path (a simple path is a path where no node appears more than once). A 2-terminal DAG is *balanced* if $|in(v)| = |out(v)|$ for each $v \in V \setminus \{s, t\}$.

A *2-terminal directed series-parallel graph* (*2-DSPG*) is a directed graph with two nodes (*terminals*) s and t that is obtained inductively as follows:

1. A basic 2-DSPG consists of two terminals s and t , connected by an edge (s, t) ;
2. If G_1 and G_2 are 2-DSPGs with terminals s_i and t_i , for $i = 1, 2$, then we can combine them in *series* by identifying t_1 with s_2 to obtain a 2-DSPG network with terminals s_1 and t_2 , or in *parallel* by identifying s_1 with s_2 , called s , and t_1 with t_2 , called t , to obtain a 2-DSPG network with terminals s and t .

2.2 Cooperative Games and Core Stability

A *cooperative (profit) game* $\Gamma = (N, v)$ consists of a player set $N = \{1, 2, \dots, n\}$ and a profit function $p : 2^N \rightarrow \mathbb{R}$ with $p(\emptyset) = 0$. A *coalition* S is a nonempty subset of N , and $p(S)$ represents the profit that can be achieved by the players in S without help of other players. The *induced subgame* (S, p_S) on S has profit function $p_S(T) = p(T)$, for all $T \subseteq S$.

The central problem in cooperative game theory is how to allocate the total profit $p(N)$ among the individual players in a ‘fair’ way. An *allocation* is a vector

$x \in \mathbb{R}^n$ with $x(N) = p(N)$, where $x(S) = \sum_{i \in S} x_i$ for any $S \subseteq N$. Different requirements for fairness and rationality lead to different set of allocations which are generally referred to as *solution concepts*.

An allocation $x \in \mathbb{R}^n$ is called an *imputation* if it satisfies *individual rationality*, i.e., $x_i \geq p(\{i\})$ for each player $i \in N$. That is, each player gains more from cooperating than acting alone. We denote by $\mathcal{I}(\Gamma)$ the set of imputations of Γ . The *core* $\mathcal{C}(\Gamma)$ of Γ is the set of imputations satisfying *coalition rationality*, i.e.,

$$\mathcal{C}(\Gamma) = \{x \in \mathbb{R}^n \mid x(N) = p(N) \text{ and } x(S) \geq p(S), \text{ for all } S \subseteq N\}.$$

That is, no coalition S can increase their profit by splitting off from the grand coalition N and going their own way. The game is called *balanced*, if $\mathcal{C}(\Gamma) \neq \emptyset$. It is called *totally balanced* if all its subgames are balanced, i.e., all its subgames have non-empty cores.

In their classical work on game theory, von Neumann and Morgenstern [19] introduced the concept of a stable set. Let $\Gamma = (N, p)$ be a cooperative game with n players, and let x and y be imputations. We say that x *dominates* y if there is a nonempty coalition S such that $x(S) \leq p(S)$, and $x_i > y_i$ for all $i \in S$. A set \mathcal{I} of imputations is *stable* if no two imputations in \mathcal{I} dominate each other (*internal stability*), and any imputation not in \mathcal{I} is dominated by some imputation in \mathcal{I} (*external stability*). In particular, the core $\mathcal{C}(\Gamma)$ is stable if for any $y \in \mathcal{I}(\Gamma) \setminus \mathcal{C}(\Gamma)$ there is an $x \in \mathcal{C}(\Gamma)$ and a nonempty coalition $S \subset N$ such that $x(S) = p(S)$ and $x_i > y_i$, for all $i \in S$. That is, Γ has a stable core if any imputation not in $\mathcal{C}(\Gamma)$ is dominated by some core imputation.

There are three other concepts closely related to core stability. The core $\mathcal{C}(\Gamma)$ is *large* if for any $y \in \mathbb{R}^n$ with $y(S) \geq p(S)$, for all $S \subseteq N$, there exists an $x \in \mathcal{C}(\Gamma)$ such that $x \leq y$. Γ is *extendable* if, for every nonempty $S \subset N$ and every core element y of the induced subgame (S, p_S) , there exists an $x \in \mathcal{C}(\Gamma)$ such that $x_i = y_i$, for all $i \in S$. Γ is *exact* if, for every $S \subset N$, there exists an $x \in \mathcal{C}(\Gamma)$ such that $x(S) = p(S)$.

Kikuta and Shapley [17] showed that a balanced game with a large core is extendable, and an extendable balanced game must have a stable core. Sharkey [22] proved that a totally balanced game with a large core is exact. Biswas *et al.* [3] pointed out that extendability also implies exactness. Note that the flow games discussed in this paper are totally balanced. We summarize these results in the following theorem.

Theorem 1 ([3, 17, 22]). *Let $\Gamma = (N, p)$ be a totally balanced game. Then core largeness implies extendability, and extendability implies exactness and core stability.*

2.3 Flow Games

Flow games were introduced by Kalai and Zemel [15, 16]. They arise from the profit distribution problem related to the maximum flow in a network. Consider

a flow network $G = (V, E; \omega; s, t)$. We assume that each player controls one edge, i.e., we can identify the set of edges with the set of players. Then the associated flow game $\Gamma = (E, p)$ is defined as

1. The player set is E ;
2. For all $S \subseteq E$, $p(S)$ is the value of a maximum (s, t) -flow that only uses edges in S .

A flow game is *simple* if the underlying network is a simple flow network. In this paper, we focus on simple flow games. These games belong to the class of packing/covering games introduced by Deng *et al.* [4].

Theorem 2 ([4, 16]). *Let $\Gamma = (E, p)$ be the flow game defined on $G = (V, E; \omega; s, t)$. Then Γ is totally balanced. If G is a simple flow network, the core $\mathcal{C}(\Gamma)$ is exactly the convex hull of the indicator vectors of the minimum (s, t) -cuts of G .*

Corollary 3 ([4]). *Let $\Gamma = (E, p)$ be a simple flow game. Then $x \in \mathcal{C}(\Gamma)$ if and only if $x(e) \geq 0$, for all $e \in E$, and $x(P) \geq 1$, for all (s, t) -paths P .*

3 Core Stability of Flow Games

In this section, we discuss the core stability of simple flow games. We first introduce the notion of dummy edges which is crucial for characterizing the stable core of a flow game. We note that dummy edges were introduced in [6] where they play an important role in the efficient computation of the nucleolus of flow games.

3.1 Dummy Edges

Let $G = (V, E; s, t)$ be a simple network and $\Gamma = (E, p)$ be the corresponding flow game. We assume that each edge is contained in some (s, t) -path. An edge $e \in E$ is called a *dummy edge (player)* of G if $p(E \setminus \{e\}) = p(E)$, i.e., the profit does not change if we exclude e . The following lemma follows immediately from the the max-flow min-cut theorem [11] and Theorem 2.

Lemma 4. *Let $e \in E$ be an edge. The following three conditions are equivalent:*

1. e is a dummy edge of G .
2. e is not contained in any minimum (s, t) -cut of G .
3. $x(e) = 0$, for all core imputations x of Γ .

Theorem 5. *$G = (V, E; s, t)$ is a balanced 2-terminal DAG if and only if it contains no dummy edges.*

Proof. Suppose G is a balanced 2-terminal DAG. Let f be a maximum 0-1 (s, t) -flow in G (which is also a maximum (s, t) -flow). Consider the subgraph H of G induced by the set $F = \{e \in E \mid f(e) = 0\}$ of edges not used by flow f . Since G is a balanced DAG and f satisfies the flow conservation property in every node except s and t , H is also acyclic and balanced. Thus, if F is not empty, there must be a simple (s, t) -path in H . But then we can push one more unit of flow along this path, contradicting the maximality of f . Hence, $F = \emptyset$, i.e., G contains no dummy edges.

Now assume G has no dummy edges. The flow conservation property and the fact that all edges have capacity one imply that for each maximum 0-1 (s, t) -flow f , at least one edge incident to an unbalanced node is not used by f . Thus, G is balanced, otherwise it would contain dummy edges. If G contains a directed cycle C , we can assume that f does not use edges on C (we could cancel one unit of flow around the cycle without violating the flow conservation), i.e., G contains dummy edges. Thus, G is a balanced DAG. \square

3.2 Core Stability

Let $G = (V, E; s, t)$ be a simple flow network. Let f be a maximum integral flow from s to t . We may w.l.o.g. assume f does not contain any directed cycles. In this section, all flows are integral and cycle-free. Since all edge capacities are 1, an edge is either saturated or not used by f . Let F denote the set of saturated edges and \bar{F} the set of unused edges. Clearly, F defines a DAG with source s and sink t and a partial order $<_f$ on the nodes (edge $(u, v) \in F$ implies $u <_f v$). Note that the edges in \bar{F} are not involved in defining the partial order. For any node without f passing through, its order is not defined.

We call a flow network G a *pseudo balanced directed acyclic graph (PBDAG)* if $v <_f u$ for all (u, v) -paths $P \subset \bar{F}$ where u and v are nodes used by f . Intuitively, the edges in \bar{F} cannot be used to push flow closer to t , they can only push flow back along a path on which the flow is flowing from s to t . We show next that being a PBDAG does not depend on the particular choice of f .

Lemma 6. *If G is a PBDAG, then G has a unique maximum (s, t) -flow.*

Proof. Remember that we only consider integral flows, so an edge is either used at full capacity 1, or it is not used at all by the flow. Assume G has two different maximum (s, t) -flows f and f' . Let u be a minimal node with respect to $<_f$ such that there is a (u, v) -path $P \subset \bar{F}$ to a node v used by f which is used by f' but not by f . Since G is a PBDAG, $v <_f u$, i.e., P closes a directed cycle in f' , contradicting the assumption that the flows are cycle-free. Note that in the last argument we implicitly used the fact that f is a maximum flow, because $v <_f u$ is only guaranteed for maximum flows in a PBDAG. \square

Note that the unique maximum (s, t) -flow in the previous lemma defines a set of $p(E)$ edge-disjoint paths. We can now state the main theorem in this section.

Theorem 7. *Let $G = (V, E; s, t)$ be a simple flow network. Then the associated flow game $\Gamma = (E, p)$ has a stable core if and only if G is a PBDAG.*

Corollary 8. *In time $O(\min(n^{3/2}, m^{1/2})m + nm)$ we can decide core stability of a simple flow game, where n and m are the number of nodes and edges in the underlying network, respectively.*

Proof. By Theorem 7, we only need to test whether the given network is a PBDAG. We first compute in time $O(\min(n^{3/2}, m^{1/2})m)$ a maximum integral cycle-free (s, t) -flow f using Even and Tarjan's maximum flow algorithm [8]. Then we delete all edges used by f and run an all-pairs shortest path algorithm for unit length graphs on the remaining graph. If there exists a (u, v) -path such that $v \not\prec_f u$, for two nodes u and v used by f , then we conclude that the network is not a PBDAG. Note that all-pairs shortest paths in unit length graphs only takes $O(nm)$ time, for example by doing n BFS explorations from different starting nodes. \square

To prove Theorem 7, we need several lemmas.

Lemma 9. *If G is a PBDAG, then the set of dummy edges is exactly \overline{F} , where F is the set of edges used by the unique maximum (s, t) -flow f in G .*

Proof. All edges in \overline{F} are dummy edges because they are not used by f . It remains to show that F contains no dummy edges. We can think of F as a union of $p(E)$ edge-disjoint (s, t) -paths. For an edge $e \in F$, let P_e be the (s, t) -path containing e . Let the flow f' be the flow obtained from f by subtracting one unit of flow along P_e . Consider the residual graph of G with respect to f' . It is easy to see that P_e is the only (s, t) -path in the residual graph, otherwise f would not have been a maximum flow in G . Here we use the fact that G is a PBDAG, because the edges of P_e are the only edges that can bring us from smaller (w.r.t. order \prec_f) nodes to larger nodes. Hence, f' is a maximum flow in $G \setminus \{e\}$, which means that e is not a dummy edge. \square

Note that the proof of this lemma also implies Lemma 6, but we stated Lemma 6 earlier for clarity. The following lemma strengthens Corollary 3 for PBDAGs.

Lemma 10. *If G is a PBDAG, then an imputation $x \in \mathcal{I}(\Gamma)$ is in the core $\mathcal{C}(\Gamma)$ if and only if $x(e) \geq 0$ for all $e \in E$, and $x(P) \geq 1$ for all (s, t) -paths $P \subseteq F$.*

Proof. If x is in the core, then $x(P) \geq 1$ for all (s, t) -paths $P \subseteq F$ by Corollary 3. To show the opposite direction, let x be an arbitrary imputation such that $x(P) \geq 1$ for all (s, t) -paths $P \subseteq F$. Since $x(E) = p(E)$ and F consists of $p(E)$ edge-disjoint (s, t) -paths, we must have $x(P) = 1$ for these paths P , and $x(e) = 0$ for all $e \in \overline{F}$. Further, $x(P_1) = x(P_2)$ for any two paths $P_1, P_2 \subseteq F$ from s to some node u . Now assume there exists an (s, t) -path P in G such

that $x(P) < 1$. Let P be such a path minimizing $x(P)$, and among all such paths minimizing the number of edges in \overline{F} . P must contain an edge in \overline{F} ; let $e = (u, v')$ be the first such edge in P . Note that u is used by f . Let v be the first node after u on P which is also used by f . Since G is a PBDAG, we know that $v <_f u$. Let P_1 be a path from s to v in F , and let P_2 be the first part of P from s to v . Since $v <_f u$, there is a path in F from v to u , and there is at least one node w on that path which also belongs to P_2 . Let P_3 be the path from s to w . Then, $x(P_1) \leq x(P_3) \leq x(P_2)$. If we replace P_2 in P by P_1 , the x -weight does not increase but the number of edges in \overline{F} in P decreases, contradicting the minimal choice of P . Thus, every (s, t) path P in G has $x(P) \geq 1$. \square

The next lemma follows directly from the complementary slackness condition of linear programming for the maximum flow problem. It states that any (s, t) -path without dummy edges must cross any minimum (s, t) -cut exactly once.

Lemma 11. *Let P be an (s, t) -path without dummy edges. Then $|P \cap C| = 1$ for any minimum (s, t) -cut C in G .*

The next lemma states that non-core elements in a simple flow game can be dominated by some core element via some (s, t) -path.

Lemma 12. *Let $\Gamma = (E, p)$ be a simple flow game. Then, $\mathcal{C}(\Gamma)$ is stable if and only if for any $x \in \mathcal{I}(\Gamma) \setminus \mathcal{C}(\Gamma)$ there exists a core element $z \in \mathcal{C}(\Gamma)$ and an (s, t) -path P such that $z(P) = 1$ and $z(e) > x(e)$ for all $e \in P$.*

Proof. Sufficiency is obvious. To see necessity, suppose that $\mathcal{C}(\Gamma)$ is stable. If $x \in \mathcal{I}(\Gamma) \setminus \mathcal{C}(\Gamma)$ is an imputation outside the core, then stability of the core implies the existence of a core element $z \in \mathcal{C}(\Gamma)$ and a nonempty coalition S such that $z(S) = p(S)$ and $z(e) > x(e)$ for all $e \in S$. Since $p(S) = z(S) > x(S) \geq 0$, there are $p(S)$ edge-disjoint (s, t) -paths in $G[S]$, denoted by $P_1, P_2, \dots, P_{p(S)}$. Let $S' = P_1 \cup P_2 \cup \dots \cup P_{p(S)}$ be the edge set of all these paths. Then, $z(S') = p(S) = p(S')$, thus there must be a path P_k such that $z(P_k) \leq 1$, for some $1 \leq k \leq p(S)$. Since $z \in \mathcal{C}(\Gamma)$, we actually have $z(P_k) = 1$. Since $z(e) > x(e)$ for all $e \in S$ and $P_k \subseteq S' \subseteq S$, we have $z(e) > x(e)$ for all $e \in P_k$. \square

Note that the path P in the previous lemma does not contain dummy edges (a dummy edge e must have $z(e) = 0$ by Lemma 4). We are now ready to prove the main theorem.

Proof (Theorem 7).

We first prove sufficiency. Suppose G is a PBDAG. For any $x \in \mathcal{I}(\Gamma) \setminus \mathcal{C}(\Gamma)$, Lemma 10 implies that there is an (s, t) -path $P = \{e_1, \dots, e_k\} \subseteq F$ such that $x(P) < 1$ and $x(e_j) \geq 0$, for $j = 1, \dots, k$. By Lemma 9, P does not contain dummy edges. By Lemma 4, each edge e_j of P is contained in some minimum (s, t) -cut C_j . By Lemma 11, each C_j does not contain other edges in P except for e_j . Thus, $C_i \neq C_j$ if $i \neq j$, for $i, j = 1, \dots, k$. Let

$$\lambda_j = x(e_j) + \frac{1 - x(P)}{k}$$

for $j = 1, \dots, k$, and

$$z = \sum_{j=1}^k \lambda_j \mathbf{1}_{C_j}.$$

Obviously, $\lambda_j > 0$ and $\sum_{j=1}^k \lambda_j = 1$. By Theorem 1, we know that $z \in \mathcal{C}(\Gamma)$. Moreover, we have

$$z(P) = \sum_{j=1}^k \left(x(e_j) + \frac{1 - x(P)}{k} \right) \mathbf{1}_{C_j}(P) = x(P) + k \cdot \frac{1 - x(P)}{k} = 1$$

and $z(e_j) > x(e_j)$, for $j = 1, \dots, k$. Hence, by Lemma 12, $\mathcal{C}(\Gamma)$ is stable.

It remains to show necessity. Suppose G is not a PBDAG, and let f be some maximum (s, t) -flow using edge set F . Then there is a (u, v) -path $\tilde{P} \subseteq \bar{F}$ with $v \not\prec_f u$ and u and v used by f . By definition, all edges in \tilde{P} are dummy edges. We will now construct an imputation $x \in \mathcal{I}(\Gamma) \setminus \mathcal{C}(\Gamma)$ such that $x(P) \geq 1$ for any (s, t) -path P without dummy edges. By Lemma 12, this implies that $\mathcal{C}(\Gamma)$ is not stable.

Since $v \not\prec_f u$, adding \tilde{P} to F does not create a directed cycle. Let $\tilde{\prec}_f$ be the partial order induced by $\tilde{P} \cup F$. We now construct a total order $s = v_1, \dots, v_n = t$ consistent with $\tilde{\prec}_f$, i.e., $v_i \tilde{\prec}_f v_j$ if and only if $i < j$. Suppose $e = (v_i, v_j)$ is an edge in \tilde{P} with $i < j$. Consider the (s, t) -cut C consisting of all edges from $\{v_1, \dots, v_i\}$ to $\{v_{i+1}, \dots, v_n\}$. C contains exactly $p(E)$ edges of F . Denote the set of these edges by C' . $C - C'$ contains only dummy edges, and it is not empty because $e \in C$. Let the imputation $x = \mathbf{1}_{C'}$ be the indicator vector of C' . If $x \in \mathcal{C}(\Gamma)$, then Lemma 4 implies that the set of edges e with $x(e) > 0$ must contain an (s, t) -cut. But C' is not an (s, t) -cut of G (we can reach t from v_i via (v_i, v_j)), therefore $x \in \mathcal{I}(\Gamma) \setminus \mathcal{C}(\Gamma)$. Since C is an (s, t) -cut, each (s, t) -path P must cross C ; if P has no edges in \tilde{P} , it must contain at least one edge of C' , which means $x(P) \geq 1$. \square

4 Exactness, Extendability and Core Largeness

4.1 Equivalence of Exactness, Extendability, and Core Largeness

In this subsection, we prove that the properties of exactness, extendability, and core largeness are equivalent for simple flow games. Since flow games are totally balanced, this equivalence strictly implies the stability of the core of a flow game.

Theorem 13. *Let $G = (V, E; s, t)$ be a simple flow network, and $\Gamma = (E, p)$ be the associated flow game. Then the following statements are equivalent:*

- (a) The game Γ is exact;
- (b) The game Γ is extendable;
- (c) The core $\mathcal{C}(\Gamma)$ is large;
- (d) The network G is cut-normal.

To prove this theorem, we need a few more lemmas. The first one is due to van Gellekom *et al.* [12]. For a cooperative game $\Gamma = (N, p)$, the set of *upper vectors* is defined as

$$L(\Gamma) = \{y \in \mathbb{R}^{|N|} \mid y(S) \geq p(S), \forall S \subseteq N\}.$$

Lemma 14 ([12]). *Let $\Gamma = (N, p)$ be a balanced game. Then, Γ has a large core if and only if $y(N) \leq p(N)$ for all extreme points y of $L(\Gamma)$.*

In order to characterize the extreme points of $L(\Gamma)$ of the flow game $\Gamma = (E, p)$, we define another polyhedron (which turns out to be identical to $L(\Gamma)$). Let \mathcal{P} be the set of (s, t) -paths in G . Define the polyhedron $L'(\Gamma)$ as

$$L'(\Gamma) = \{y \in \mathbb{R}^{|E|} \mid y(P) \geq 1, \forall P \in \mathcal{P}, \text{ and } y(e) \geq 0, \forall e \in E\}.$$

Lemma 15. *For any simple flow game Γ , $L(\Gamma) = L'(\Gamma)$.*

Proof. Clearly, $L(\Gamma) \subseteq L'(\Gamma)$. The other inclusion follows from the fact that S contains $p(S)$ edge-disjoint (s, t) -paths, for any $S \subseteq E$. \square

In particular, $L(\Gamma)$ has the same extreme points as $L'(\Gamma)$. In fact, the polyhedron $L'(\Gamma)$ is exactly the dominant of the (s, t) -cut polytope [20, Chapter 13] which immediately implies the following lemma.

Lemma 16. *Let Γ be a simple flow game. Then each extreme point of $L'(\Gamma)$ is an indicator vector of some minimal (s, t) -cut of G .*

Proof (Theorem 13). Since flow games are totally balanced by Theorem 2, the implications “(c) \Rightarrow (b) \Rightarrow (a)” follow from Theorem 1.

“(a) \Rightarrow (d)”. A 2-terminal DAG with terminals s and t is cut-normal if and only if every (s, t) -cut contains a minimum (s, t) -cut. Suppose Γ is exact. Let S be an (s, t) -cut of G and $\bar{S} = E \setminus S$. Then, by the definition of exactness, there exists an $x \in \mathcal{C}(\Gamma)$ such that $x(\bar{S}) = p(\bar{S}) = 0$. This implies $x(S) = x(E) - x(\bar{S}) = p(E) - x(\bar{S}) = p(E)$. Let \mathcal{M} denote the set of minimum (s, t) -cuts of G . We know from Theorem 2 that x can be written as

$$x = \sum_{C \in \mathcal{M}} \lambda_C \mathbf{1}_C,$$

where $\lambda_C \geq 0$ for each $C \in \mathcal{M}$ and $\sum_{C \in \mathcal{M}} \lambda_C = 1$. It follows that

$$p(E) = x(S) = \sum_{C \in \mathcal{M}} \lambda_C \mathbf{1}_C(S) = \sum_{C \in \mathcal{M}} \lambda_C |S \cap C| \leq \sum_{C \in \mathcal{M}} \lambda_C |C| = |C|.$$

Since C is a minimum (s, t) -cut, we have $p(E) = |C|$. Thus, in the inequality above, we actually have equality and therefore $S \cap C = C$, for all $C \in \mathcal{M}$ with $\lambda_C > 0$. That is, S contains at least one minimum (s, t) -cut of G .

“(d) \Rightarrow (c)”. Suppose G is cut-normal. Let y be some extreme point of $L(\Gamma)$. By Lemmas 15 and 16, y is the indicator vector of some minimal (s, t) -cut of G , say C , which is also a minimum (s, t) -cut. Thus, $\mathbf{1}_C(E) = |C| = p(E)$. Lemma 14 now implies that the core of Γ is large. \square

4.2 An Efficient Algorithm

In this subsection, we give a polynomial time algorithm to decide exactness, extendability, and core largeness for simple flow games. Since these properties imply core stability, in view of Theorem 7 we can therefore w.l.o.g. assume that the flow networks in this subsection are PBDAGs. We distinguish two types of dummy edges in a PBDAG.

1. Type I: Dummy edges that lie on some simple (s, t) -path;
2. Type II: Other dummy edges.

We can now state our main theorem. We will prove it in Subsection 4.3.

Theorem 17. *Let $G = (V, E; s, t)$ be a PBDAG. Let G' be obtained from G by deleting all dummy edges. Then, G is cut-normal if and only if G only contains type II dummy edges and G' is a balanced 2-DSPG.*

Corollary 18. *We can decide the properties of exactness, extendability, and core largeness of a simple flow game in $O(m^2\alpha(m, n))$ time, where n and m are the numbers of nodes and edges in the underlying network, respectively, and $\alpha(m, n)$ is the inverse of the Ackermann function.*

Proof. For each edge $(u, v) \in E$, we test whether it is a type II dummy edge. We can do this in time $O(n + m\alpha(m, n))$ by computing two node-disjoint paths, one from s to u and the other one from v to t , using Tholey’s Algorithm [25]. If we find two such paths, (u, v) is a type I edge. After deleting all type II edges, we only need to test whether the remaining graph is a balanced DAG which can be done in linear time. If this is the case, then we test whether the DAG is a 2-DSPG, which also takes linear time [26]. The correctness follows from Theorems 13 and 17. \square

4.3 Proof of Theorem 17

Let $G = (V, E; s, t)$ be a PBDAG. First, we need some lemmas. The following lemma is trivial.

Lemma 19. *Let W be the set of type II dummy edges of G . No minimal (s, t) -cut contains any edge in W . Moreover, any (s, t) -cut of $G \setminus W$ is also an (s, t) -cut of G .*

Lemma 20. *If G is cut-normal, then it does not contain any type I dummy edges.*

Proof. Suppose $e = (u, v)$ is a type I dummy edge and P is a simple (s, t) -path containing e . Let P_{su} and P_{vt} denote the subpaths of P from s to u and from v to t , respectively.

Let U be the set of nodes in P_{su} , and C be the (s, t) -cut induced by the partition U and $V \setminus U$. Since C contains the dummy edge $e = (u, v)$, by Lemma 4, it is not a minimum (s, t) -cut. On the other hand, $C \setminus \{(u, v)\}$ is not an (s, t) -cut because $(C \setminus \{(u, v)\}) \cap P = \emptyset$. Hence, (u, v) occurs in any minimal (s, t) -cut C' contained in C , and thus C' is not a minimum (s, t) -cut either, contradicting the assumption that G is cut-normal. \square

Two graphs are *homeomorphic* if they can be made isomorphic by inserting new nodes of degree two into edges, i.e., substituting edges by directed paths (which does not change the topology of the graph). Let H denote the graph shown in Fig. 1, which is the *forbidden subgraph* for 2-DSPGs.

Theorem 21 ([7, 14, 26]). *A 2-terminal DAG is a 2-DSPG if and only if it does not contain a subgraph homeomorphic to H .*

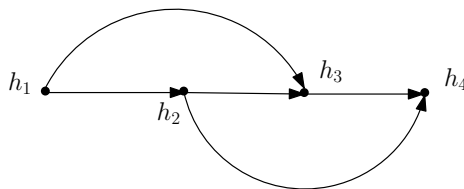


Fig. 1. The forbidden network H .

We can now characterize balanced 2-DSPGs.

Theorem 22. *Let $G = (V, E; s, t)$ be a balanced 2-terminal DAG with terminals s and t . Then G is cut-normal if and only if it is a 2-DSPG.*

Proof. Sufficiency. It is easy to see that any balanced 2-DSPG can be generated inductively with the additional constraint that the combination in series step (where we identify the two terminals t_1 and s_2) requires that the indegree of t_1 is equal to the outdegree of s_2 .

We now prove sufficiency by induction on $|E|$. When $|E| = 1$, the graph is obviously cut-normal. Suppose all balanced 2-DSPGs with fewer than $|E|$ edges are cut-normal. If G is generated from $G_1 = (V_1, E_1; s_1, t_1)$ and $G_2 = (V_2, E_2; s_2, t_2)$ by combination in parallel (with $s = s_1 = s_2$ and $t = t_1 = t_2$), then any minimal (s, t) -cut C in G consists of a minimal (s_1, t_1) -cut $C_1 = C \cap E_1$

in G_1 and a minimal (s_2, t_2) -cut $C_2 = C \cap E_2$ in G_2 . By induction hypothesis, C_1 and C_2 are minimum (s, t) -cuts in G_1 and G_2 , respectively. Thus, C is a minimum (s, t) -cut in G .

If G is generated from G_1 and G_2 by combination in series (with $s = s_1$ and $t = t_2$), then a minimal (s, t) -cut C in G is either a minimal (and thus minimum) (s_1, t_1) -cut in G_1 or a minimal (and thus minimum) (s_2, t_2) -cut in G_2 . Note that $in(t_1)$ and $out(s_2)$ are a minimal (and thus minimum) (s_1, t_1) -cut in G_1 and (s_2, t_2) -cut in G_2 , respectively. Since $|in(t_1)| = |out(s_2)|$ in G , a minimum (s_1, t_1) -cut in G_1 has the same capacity as a minimum (s_2, t_2) -cut in G_2 . Thus, C is a minimum (s, t) -cut in G .

Necessity. Since G is a balanced 2-terminal DAG, there exists a topological ordering of G , say $V = \{s = v_1, \dots, v_n = t\}$. Let $k = |out(s)|$.

Suppose that G is not a 2-DSPG. Then we can construct a minimal (s, t) -cut of size larger than k , contradicting the assumption that G is cut-normal. By Theorem 21, G contains a subgraph homeomorphic to H (see Fig. 1) which we denote by $H[a, b, c, d, P_{ab}, P_{bc}, P_{cd}, P_{ac}, P_{bd}]$, where v_a, v_b, v_c and v_d are the nodes corresponding to h_1, h_2, h_3 and h_4 in H , respectively, and the node-disjoint (except at their endpoints) paths $P_{ab}, P_{bc}, P_{cd}, P_{ac}$ and P_{bd} correspond to the edges $(h_1, h_2), (h_2, h_3), (h_3, h_4), (h_1, h_3)$, and (h_2, h_4) in H , respectively. Among all subgraphs homeomorphic to H we choose one, denoted by G_H , with largest index b .

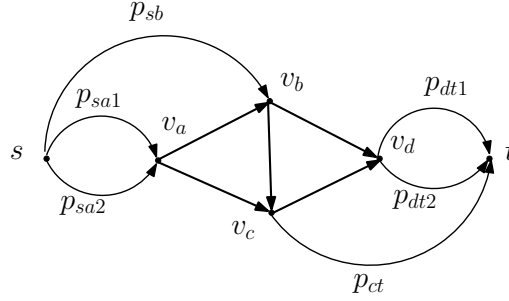


Fig. 2. The network \tilde{G} .

Obviously, G_H is not balanced. But G is balanced, so we can find six pairwise edge-disjoint paths in G (they are also edge-disjoint with G_H), namely $P_{sa1}, P_{sa2}, P_{sb}, P_{ct}, P_{dt1}$, and P_{dt2} that can be added to G_H to obtain a balanced subgraph \tilde{G} (see Fig. 2). Note that \tilde{G} is the union of three edge-disjoint (s, t) -paths $P_{sa1} + P_{ab} + P_{bd} + P_{dt1}$, $P_{sa2} + P_{ac} + P_{cd} + P_{dt2}$, and $P_{sb} + P_{bc} + P_{ct}$ (we use $P_1 + P_2$ to denote the concatenation of paths P_1 and P_2).

Consider the (s, t) -cut C_b in G induced by the partition of V into $\{v_1, \dots, v_{b-1}\}$ and $\{v_b, \dots, v_n\}$. Since G is acyclic and all nodes lie on some (s, t) -path, C_b is a minimal (s, t) -cut. If $|C_b| > k$, we have a minimal (s, t) -cut that is not a minimum cut. So we may assume that $|C_b| = k$. We partition the edges of C_b into

two classes, namely C_b^1 and C_b^2 . An edge e belongs to C_b^1 if every (s, t) -path containing e passes through v_c , otherwise it belongs to C_b^2 . We first show two properties of C_b^1 .

Claim 1. $C_b^1 \neq \emptyset$. In particular, C_b^1 contains the edge $e = P_{ac} \cap C_b$.

Proof of Claim 1. Assume there is an (s, t) -path P_e containing e but not v_c . Let v_r be the node in $P_{ac} \cap P_e$ with the largest subscript, v_q be the first node in \tilde{G} that we encounter on P_e when starting walking at v_r . Note that such a node exists because P_e ends at $t \in \tilde{G}$. Let P_{rq} be the path from v_r to v_q .

Since $e \in C_b$, we have $r \geq b$. Actually, $r > b$ because v_b is not on P_{ac} . On the other hand, $r < c$ because $v_r \in P_{ac}$ and $v_c \notin P_e$. But then we can find another subgraph $H[a, r, v_c, \dots]$ in G , a contradiction because $r > b$. To see this, we distinguish five cases, depending on the location of v_q (see Figs. 3 and 4).

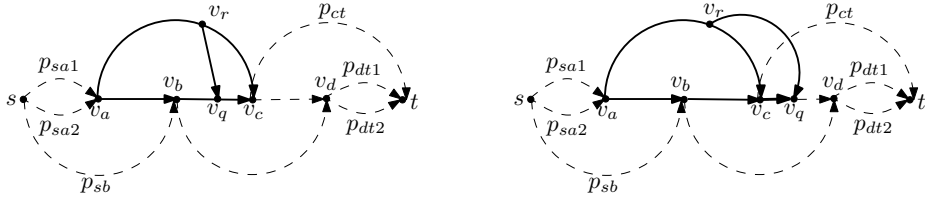


Fig. 3. Cases (1) and (2) in the proof of Theorem 22.

Let $P_{xy}(i, j)$ denote the subpath of P_{xy} from v_i to v_j , where $x, y \in \{a, b, c, d\}$.

- (1) $v_q \in P_{bc}$: $H[a, r, q, c, P_{ac}(a, r), P_{rq}, P_{bc}(q, c), P_{ab} + P_{bc}(b, q), P_{ac}(r, c)]$;
- (2) $v_q \in P_{cd}$: $H[a, r, c, q, P_{ac}(a, r), P_{ac}(r, c), P_{cd}(c, q), P_{ab} + P_{bc}, P_{rq}]$;
- (3) $v_q \in P_{bd}$: $H[a, r, q, d, P_{ac}(a, r), P_{rq}, P_{bd}(q, d), P_{ab} + P_{bd}(b, q), P_{ac}(r, c) + P_{cd}]$;
- (4) $v_q \in P_{ct}$: $H[a, r, c, q, P_{ac}(a, r), P_{ac}(r, c), P_{ct}(c, q), P_{ab} + P_{bc}, P_{rq}]$;
- (5) $v_q \in P_{dt1}$ or $v_q \in P_{dt2}$: $H[a, r, d, q, P_{ac}(a, r), P_{ac}(r, c) + P_{cd}, P_{dq}, P_{ab} + P_{bd}, P_{rq}]$.

This finishes the proof of Claim 1. \square

Claim 2. $|C_b^1| < c_{\text{out}}$, where c_{out} is the outdegree of v_c in G .

Proof of Claim 2. Let U be a maximal set of edge-disjoint (s, t) -paths containing v_c that includes the path $P_{sa1} + P_{ab} + P_{bc} + P_{cd} + P_{dt1}$. Note that $|U| \leq c_{\text{out}}$. Clearly, C_b^1 is a subset of $U \cap C_b$ by the definition of C_b^1 . Since the last edge of P_{ab} belongs to $(U \cap C_b) \setminus C_b^1$, C_b^1 is a proper subset of $U \cap C_b$, and thus $|C_b^1| < c_{\text{out}}$. This completes the proof of Claim 2. \square

Let C^* be the set of edges in C_b^2 plus all outgoing edges at v_c . Then C^* is an (s, t) -cut because every (s, t) -path contains either an edge in C_b^2 or the node v_c . Since $C_b^1 \neq \emptyset$, each edge in $\text{out}(v_c)$ is necessarily in C^* . Since each edge in C_b^2 is necessary, C^* is a minimal (s, t) -cut. The size of C^* is

$$|C^*| = |C_b^2| + c_{\text{out}} > |C_b^2| + |C_b^1| = |C_b| = k.$$

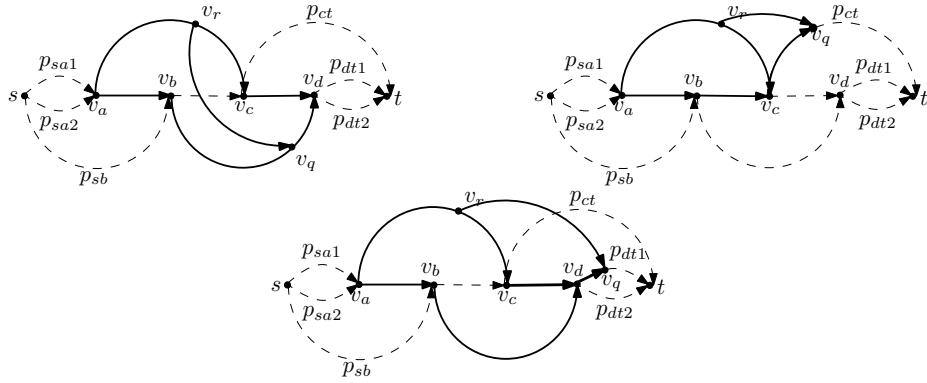


Fig. 4. Cases (3), (4) and (5) in the proof of Theorem 22.

Thus, C^* is not a minimum (s, t) -cut, a contradiction. Therefore, the assumption that G is not a 2-DSPG must be wrong. This concludes the proof of Theorem 22. \square

Theorem 17 follows directly from Lemma 19, Lemma 20, and Theorem 22.

5 Further Discussion

In this paper, we propose structural characterizations of exact game, extendable game, large core and stable core for simple flow games and obtain polynomial time algorithms to test these properties. Currently, little is known about core stability of flow games on networks with arbitrary capacities. Although it is $co\mathcal{NP}$ -complete to decide whether an imputation belongs to the core, this does not rule out the possibility that core stability can be decided efficiently. We leave it as an open problem.

Acknowledgements

This research was supported by NCET (No. 05-0598), the National Natural Science Foundation of China (No. 10771200 and 60573025), the Shanghai Leading Academic Discipline Project (project number B114), and partially supported by NSFC (No. 60573028).

References

1. R.K. Ahuja, T.L. Magnanti and J.B. Orlin, Network Flows, Prentice Hall, Inc., Englewood Cliffs, NJ, 1993.

2. Bietenhader, T., Okamoto, Y.: Core stability of minimum coloring games. *Proceedings of 30th International Workshop on Graph-Theoretic Concept in Computer Science*, Lecture notes in computer science **3353** (2004) 389–401
3. Biswas, A.K., Parthasarathy, T., Potters, J.A.M., Voorneveld, M.: Large cores and exactness. *Game and Economic Behavior*, **28** (1999) 1–12
4. Deng, X., Ibaraki, T., Nagamochi, H.: Algorithmic aspects of the core of combinatorial optimization games. *Mathematics of Operations Research*, **24** (1999) 751–766
5. Deng, X., Papadimitriou, C. H.: On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, **19** (1994) 257–266
6. Deng, X, Fang, Q. and Sun, X: Finding nucleolus of flow game. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'06)*, pages 124–131, 2006.
7. Duffin, R. Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications*, **10** (1965) 303–318
8. Even, S, and Tarjan, R. E.: Network flow and testing graph connectivity. *SIAM J. Comput.*, **4** (1975) 507-518
9. Fang, Q., Zhu, S., Cai, M., Deng, X.: Membership for core of LP games and other games. *Lecture notes in computer science 2108*, (2001) 247–256
10. Fang, Q., Fleischer, R., Li, J. and Sun, X.: Algorithms for core stability, core largeness, exactness and extendability of flow games. In *Proceedings of the 13th Annual International Computing and Combinatorics Conference*, (2007) 439-447
11. Ford, L.R., Fulkerson, D.R.: *Flows in Networks*, Princeton University Press, Princeton, New Jersey (1962)
12. van Gellekom, J. R. G., Potters, J. A. M., Reijnierse, J. H.: Prosperity properties of TU-games. *International Journal of Game Theory*, **28** (1999) 211–227
13. Jain, K. and Vohra, R. V. On stability of the core. *Manuscript*, (2006) <http://www.kellogg.northwestern.edu/faculty/vohra/ftp/newcore.pdf>
14. Jakoby, A. and Li, M.:skiewicz, and R. Reischuk. Space efficient algorithms for directed series-parallel graphs. *Journal of Algorithms*, **60** (2006) 85–114
15. Kalai, E., Zemel, E.: Totally balanced games and games of flow. *Mathematics of Operations Research*, **7** (1982) 476–478
16. Kalai, E., Zemel, E.: Generalized network problems yielding totally balanced games. *Operations Research*, **30** (1982) 998–1008
17. Kikuta, K., Shapley, L.S.: Core stability in n -person games. *Manuscript*, (1986)
18. Lucas, W.F.: A game with no solution. *Bulletin of the American Mathematical Society*, **74** (1968) 237–239
19. von Neumann, J., Morgenstern, O.: *Theory of Games and Economic Behaviour*. Princeton Univeristy Press, Princeton (1944)
20. Schrijver, A.: *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag, Berlin Heidelberg (2003)
21. Shapley, L.S.: Cores and convex games. *International Journal of Game Theory*, **1** (1971) 11-26
22. Sharkey, W.W.: Cooperative games with large cores. *International Journal of Game Theory*, **11** (1982) 175–182
23. Solymosi, T., Raghavan, T.E.S.: Assignment games with stable cores. *International Journal of Game Theory*, **30** (2001) 177-185
24. Sun, X. and Fang, Q.: Core stability of flow games. In *Proceedings of the 2005 China-Japan Conference on Discrete Geometry, Combinatorics and Graph Theory*, (2007) 189–199
25. Tholey, T.:Solving the 2-disjiont paths problem in nearly linear time. *Theory of Computing Systems*, **39** (2006) 51–78

26. Valdes, J., Tarjan, R. E. and Lawler, E. L.: The recognition of series-parallel digraphs. *In Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC'79)*, (1979) 1–12