N

Nash Equilibria and Dominant Strategies in Routing
2005; Wang, Li, Chu

Weizhao Wang[1], Xiang-Yang Li[2], Xiaowen Chu[3]
[1] Google Inc., Irvine, CA, USA
[2] Department of Computer Science, Illinois Institute of Tech., Chicago, IL, USA
[3] Department of Computer Science, Hong Kong Baptist University, Hong Kong, China

Keywords and Synonyms

Strategyproof; Truthful; Nash; BB

Problem Definition

This problem is concerned with the multicast routing and cost sharing in a selfish network composed of relay terminals and receivers. This problem is motivated by the recent observation that the selfish behavior of the network could largely degraded existing system performance, even dysfunction. The work of Wang, Li and Chu [7] first presented some negative results of the strategyproof mechanism in multicast routing and sharing, and then proposed a new solution based on Nash Equilibrium that could greatly improve the performance.

Wang, Li and Chu modeled a network by a link weighted graph $G = (V, E, \mathbf{c})$, where V is the set of all nodes and $\mathbf{c}$ is the cost vector of the set E of links. For a multicast session, let Q denote the set of all receivers. In game theoretical networking literatures, usually there are two models for the multicast cost/payment sharing.

Axiom Model (AM) All receivers must receive the service, or equivalently, each receiver has an infinity valuation [3]. In this model, a sharing method ξ computes how much each receiver should pay when the receiver set is R and cost vector is $\mathbf{c}$.

Valuation Model (VM) There is a set $Q = \{q_1, q_2, \cdots, q_r\}$ of r possible receivers. Each receiver $q_i \in Q$ has a valuation $\boldsymbol{\eta}_i$ for receiving the service. Let $\boldsymbol{\eta} = (\eta_1, \eta_2, \ldots, \eta_r)$ be the valuation vector and $\boldsymbol{\eta}_R$ be the valuation vector of a set $R \subseteq Q$ of receivers. In this model, they are interested in a sharing mechanism S consisting of a *selection scheme* $\sigma(\boldsymbol{\eta}, \mathbf{c})$ and a *sharing method* $\xi(\boldsymbol{\eta}, \mathbf{c})$. $\sigma_i(\boldsymbol{\eta}, \mathbf{c})$ denotes whether receiver i receives the service or not, and $\xi_i(\boldsymbol{\eta}, \mathbf{c})$ computes how much the receiver q_i should pay for the multicast service. Let $\mathbb{P}(\boldsymbol{\eta}, \mathbf{c})$ be the total payment for providing the service to the receiver set.

In the valuation model, a receiver who is willing to receive the service is not guaranteed to receive the service. For notational simplicity, $\sigma(\boldsymbol{\eta}, \mathbf{c})$ is used to denote the set of actual receivers. Under the Valuation Model, a *fair* sharing according to the following criteria is studied.

- **Budget Balance**: For the receiver set $R = \sigma(\boldsymbol{\eta}, \mathbf{c})$, $\mathbb{P}(\boldsymbol{\eta}, \mathbf{c}) = \sum_{q_i \in Q} \xi_i(\boldsymbol{\eta}, \mathbf{c})$. If $\alpha \cdot \mathbb{P}(\boldsymbol{\eta}, \mathbf{c}) \le \sum_{i \in R} \xi_i(\boldsymbol{\eta}, \mathbf{c}) \le \mathbb{P}(\boldsymbol{\eta}, \mathbf{c})$, for some given parameter

1: Compute path $\mathsf{LCP}(s, q_j, \mathbf{d})$ and set $\phi_j = \frac{\omega(\mathrm{B}_{mm}(s, q_j, \mathbf{d}), \mathbf{d})}{r}$ for every $q_j \in Q$.
2: Set $\mathcal{O}_i^{\mathsf{DM}}(\eta, \mathbf{d}) = 0$ and $\mathcal{P}_i^{\mathsf{DM}}(\eta, \mathbf{d}) = 0$ for each link $e_i \notin \mathsf{LCP}(s, q_j, \mathbf{d})$.
3: **for** each receiver q_j **do**
4: **if** $\eta_j \ge \phi_j$ **then**
5: Receiver q_j is granted the service and charged $\xi_j^{\mathsf{DM}}(\eta, \mathbf{d})$, set $R = R \cup q_j$.
6: **else**
7: Receiver q_j is not granted the service and is charged 0.
8: **end if**
9: **end for**
10: Set $\mathcal{O}_i^{\mathsf{DM}}(\eta, \mathbf{d}) = 1$ and $\mathcal{P}_i^{\mathsf{DM}}(\eta, \mathbf{d}) = \mathcal{P}_i^{\mathsf{LCPT}}(\eta_R^{=\infty}, \mathbf{d})$ for each link $e_i \in \mathsf{LCPT}(R, \mathbf{d})$.

Nash Equilibria and Dominant Strategies in Routing, Algorithm 1
The multicast system $\Psi^{\mathsf{DM}} = (\mathcal{M}^{\mathsf{DM}}, S^{\mathsf{DM}})$ based on multicast tree LCPT

$0 < \alpha \leq 1$, then $S = (\sigma, \xi)$ is called α-budget-balance. If budget balance is not achievable, then a sharing scheme S may need to be α-budget-balance instead of budget balance.

- **No Positive Transfer** (NPT): Any receiver q_i's sharing should not be negative.
- **Free Leaving**: (FR) The potential receivers who do not receive the service should not pay anything.
- **Consumer Sovereignty** (CS): For any receiver q_i, if $\boldsymbol{\eta}_i$ is sufficiently large, then q_i is guaranteed to be an actual receiver.
- **Group-Strategyproof** (GS): Assume that $\boldsymbol{\eta}$ is the valuation vector and $\boldsymbol{\eta}' \neq \boldsymbol{\eta}$. If $\xi_i(\boldsymbol{\eta}', \mathbf{c}) \geq \xi_i(\boldsymbol{\eta}, \mathbf{c})$ for each $q_i \in \boldsymbol{\eta}$, then $\xi_i(\boldsymbol{\eta}', \mathbf{c}) = \xi_i(\boldsymbol{\eta}, \mathbf{c})$.

Notations

The path with the lowest cost between two odes s and t is denoted as $\mathsf{LCP}(s, t, \mathbf{c})$, and its cost is dented as $|\mathsf{LCP}(s, t, \mathbf{c})|$. Given a simple path P in the graph G with cost vector $\mathbf{c}$, the sum of the cost of links on path P is denoted as $|\mathsf{P}(\mathbf{c})|$. For a simple path $P = v_i \rightsquigarrow v_j$, if $\mathsf{LCP}(s, t, \mathbf{c}) \bigcap P = \{v_i, v_j\}$, then P is called a *bridge* over $\mathsf{LCP}(s, t, \mathbf{c})$. This bridge P *covers* link e_k if $e_k \in \mathsf{LCP}(v_i, v_j, \mathbf{c})$. Given a link $e_i \in \mathsf{LCP}(s, t, \mathbf{c})$, the path with the minimum cost that covers e_i is denoted as $\mathrm{B}_{\min}(e_i, \mathbf{c})$. The bridge $\mathrm{B}_{mm}(s, t, \mathbf{c}) = \max_{e_i \in \mathsf{LCP}(s,t,\mathbf{c})} \mathrm{B}_{\min}(e_i, \mathbf{c})$ is the *max-min cover* of the path $\mathsf{LCP}(s, t, \mathbf{c})$.

A bridge set $\mathcal{B}$ is a *bridge cover* for $\mathsf{LCP}(s, t, \mathbf{c})$, if for every link $e_i \in \mathsf{LCP}(s, t, \mathbf{c})$, there exists a bridge $\mathrm{B} \in \mathcal{B}$ such that $e_i \in \mathsf{LCP}(v_{s(\mathrm{B})}, v_{t(\mathrm{B})}, \mathbf{c})$. The *weight* of a bridge cover $\mathcal{B}(s, t, \mathbf{c})$ is defined as $|\mathcal{B}(s, t, \mathbf{c})| = \sum_{B \in \mathcal{B}(s,t,\mathbf{c})} \sum_{e_i \in B} c_i$. A bridge cover is a *least bridge cover* (LB), denoted by $\mathbb{LB}(s, t, \mathbf{c})$, if it has the smallest weight among all bridge covers that cover $\mathsf{LCP}(s, t, \mathbf{c})$.

Key Results

Theorem 1 *If $\Psi = (\mathcal{M}, S)$ is an α-stable multicast system, then $\alpha \leq 1/n$.*

Theorem 2 *Multicast system Ψ^{DM} is $1/(r \cdot n)$-stable, where r is the number of receivers.*

Theorem 1 gives an upper bound for α for any α-stable unicast system Ψ. It is not difficult to observe that even the receivers are cooperative, Theorem 1 still holds. Theorem 2 showed that there exists a multicast system is $1/(r \cdot n)$-stable. When $r = 1$, the problem become traditional unicast system and the bound is tight. When relaxing the dominant strategy to the Nash Equilibria requirement, a First Price Auction (FPA) mechanism is proposed by Wang et al. under the Axiom Model that has many nice properties.

1: Each terminal bids a price b_i.
2: Every link sends a unit size dummy packet with property $\rho = \tau \cdot (n \cdot b_u - \sum_{e_i \in G} b_i)$ and receives payment $f_i(s, q_1, \mathbf{b}) = \tau \cdot \left[b_u \cdot (n \cdot b_u - \sum_{e_j \in G - e_i} b_j) - \frac{h_i^2}{2} \right]$. Here, b_u is the maximum cost any link can declare.
3: Compute the unique path $\mathsf{LCP}(s, q_1, \mathbf{b}')$ by applying certain fixed tie-breaking rule consistently.
4: Each terminal bids again for a price b_i'.
5: **for** each link e_i **do**
6: It is select to relay the packet and receives payment b_i' if and only if e_i is on path $\mathsf{LCP}(s, q_1, \mathbf{b}')$.
7: **end for**

Nash Equilibria and Dominant Strategies in Routing, Algorithm 2
FPA Mechanism $\mathcal{M}^{\mathsf{AUC}}$

1: Execute Line 1 − 3 in Algorithm 2.
2: Compute $\mathbb{LB}(s, q_1, \mathbf{b})$, and set $\phi = \frac{|\mathbb{LB}(s,q_1,\mathbf{b})|}{2}$.
3: If $\phi \leq \eta_1$ then set $\sigma_1^{\mathsf{AU}}(\eta_1, \widetilde{\mathbf{b}}) = 1$ and $\xi_1^{\mathsf{AU}}(\eta_1, \widetilde{\mathbf{b}}) = \phi$. Every relay link on LCP is selected and receives an extra payment b_i'.
4: For each link $e_i \notin \mathsf{LCP}(s, q_1, \mathbf{b}')$, it receives a payment $\mathcal{P}_i^{\mathsf{AU}}(\eta_1, \widetilde{\mathbf{b}}) - \gamma \cdot (b_i' - b_i)^2$.

Nash Equilibria and Dominant Strategies in Routing, Algorithm 3
FPA based unicast system

Theorem 3 *There exists NE for FPA mechanism $\mathcal{M}^{AUC}$ and for any NE, (a) each link bids his true cost as the first bid b_i, (b) the actual shortest path is always selected, (c) the total cost for different NE differs at most 2 times.*

Based on the FPA Mechanism Ψ^{AUC}, Wang, Li and Chu design a unicast system as follows.

Theorem 4 *The FPA based unicast system not only has Nash Equilibria, but also is $\frac{1}{2}$-NE-stable with ϵ additive, for any given ϵ.*

By treating each receiver as a separate receiver and applying the similar process as in the unicast system, Wang, Li and Chu extended the unicast system to a multicast system.

Theorem 5 *The FPA based multicast system not only has Nash Equilibria, but also is $1/(2 \cdot r)$-NE-stable with ϵ additive, for any given ϵ.*

Applications

More and more research effort has been done to study the non-cooperative games recently. Among these various forms of games, the unicast/multicast routing game [2,5,6] and multicast cost sharing game [1,3,4] have received a considerable amount of attentions over the past few year due to its application in the Internet. However, both unicast/multicast routing game and multicast cost sharing game are one folded: the unicast/multicast routing game does not take the receivers into account while the multicast cost sharing game does not treat the links as non-cooperative. In this paper, they study the scenario, which was called *multicast system*, in which both the links and the receivers could be non-cooperative. Solving this problem paving a way for the real world commercial multicast and unicast application. A few examples are, but not limited to, the multicast of the video content in wireless mesh network and commercial WiFi system; the multicast routing in the core Internet.

Open Problems

A number of problems related to the work of Wang, Li and Chu [7] remain open. The first and foremost, the upper bound and lower bound on α still have a gap of r if the multicast system is α-stable; and a gap of $2r$ if the multicast system is α-Nash stable.

The second, Wang, Li and Chu only showed the existence of the Nash Equilibrium under their systems. They have not characterized the convergence of the Nash Equilibrium and the strategies of the user, which are not only interesting but also important problems.

Cross References

► Non-approximability of Bimatrix Nash Equilibria

Recommended Reading

1. Feigenbaum, J., Papadimitriou, C.H., Shenker, S.: Sharing the cost of multicast transmissions. J. Comput. Syst. Sci. **63**, 21–41 (2001)
2. Kao, M.-Y., Li, X.-Y., Wang, W.: Towards truthful mechanisms for binary demand games: A general framework. In: ACM EC, pp. 213–222, Vancouver, Canada (2005)
3. Herzog, S., Shenker, S., Estrin, D.: Sharing the "cost" of multicast trees: an axiomatic analysis. IEEE/ACM Trans. Netw. **5**, 847–860 (1997)
4. Moulin, H., Shenker, S.: Strategyproof sharing of submodular costs: budget balance versus efficiency. Econ. Theory **18**, 511–533 (2001)
5. Wang, W., Li, X.-Y., Sun, Z., Wang, Y.: Design multicast protocols for non-cooperative networks. In: Proceedings of the 24th IEEE INFOCOM. vol. 3, pp. 1596–1607, Miami, USA (2005)
6. Wang, W., Li, X.-Y., Wang, Y.: Truthful multicast in selfish wireless networks. In: Proceedings of the 10th ACM MOBICOM, pp. 245–259, Philadelphia, USA (2004)
7. Wang, W., Li, X.-Y., Chu, X.: Nash equilibria, dominant strategies in routing. In: Workshop for Internet and Network Economics (WINE). Lecture Notes in Computer Science, vol. 3828, pp 979–988. Springer, Hong Kong, China (2005)

Navigation

► Mobile Agents and Exploration
► Robotics

Nearest Neighbor Interchange and Related Distances

1999; DasGupta, He, Jiang, Li, Tromp, Zhang

Bhaskar DasGupta[1], Xin He[2], Tao Jiang[3], Ming Li[4], John Tromp[5], Louxin Zhang[6]

[1] Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA
[2] Department of Computer Science and Engineering, University at Buffalo The State University of New York, Buffalo, NY, USA
[3] Department of Computer Science and Engineering, University of California at Riverside, Riverside, CA, USA
[4] Department of Computer Science, University of Waterloo, Waterloo, ON, Canada
[5] CWI, Amsterdam, Netherlands
[6] Department of Mathematics, National University of Singapore, Singapore, Singapore

Keywords and Synonyms

Comparison of phylogenies; Network models of evolution

Problem Definition

In this chapter, the authors state results on some transformation based distances for *evolutionary trees*. Several distance models for evolutionary trees have been proposed in the literature. Among them, the best known is perhaps the *nearest neighbor interchange* (nni) distance introduced independently in [10] and [9]. The authors will focus on the nni distance and a closely related distance called the *subtree-transfer* distance originally introduced in [5,6]. Several papers that involved DasGupta, He, Jiang,

Li, Tromp and Zhang essentially showed the following results:

- A correspondence between the nni distance and the linear-cost subtree-transfer distance on unweighted trees;
- Computing the nni distance is NP-hard, but admits a fixed-parameter tractability and a logarithmic ratio approximation algorithms;
- A 2-approximation algorithm for the linear-cost subtree-transfer distance on weighted evolutionary trees.

The authors first define the nni and linear-cost subtree-transfer distances for unweighted trees. Then the authors extend the nni and linear-cost subtree-transfer distances to weighted trees. For the purpose of this chapter, an evolutionary tree (also called *phylogeny*) is an *unordered* tree, has uniquely labeled leaves and unlabeled interior nodes, can be *unrooted* or *rooted*, can be *unweighted* or *weighted*, and has all internal nodes of degree 3.

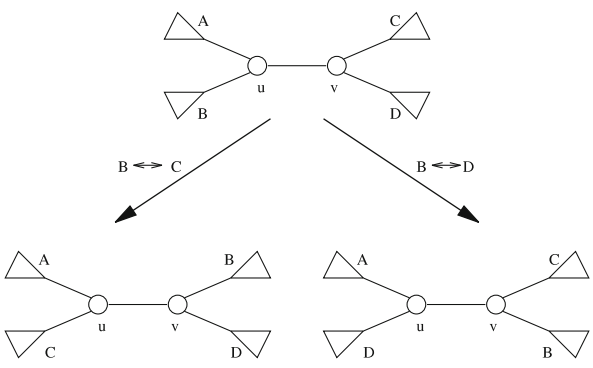
Unweighted Trees

An *nni* operation swaps two subtrees that are separated by an internal edge (u, v), as shown in Fig. 1.

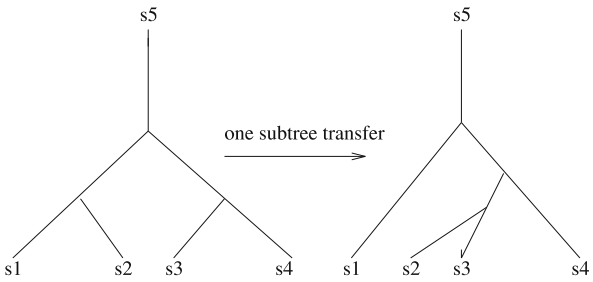
The nni operation is said to *operate* on this internal edge. The nni distance, $D_{\text{nni}}(T_1, T_2)$, between two trees T_1 and T_2 is defined as the *minimum* number of nni operations required to transform one tree into the other.

An nni operation can also be viewed as moving a subtree past a neighboring internal node. A more general operation is to transfer a subtree from one place to another arbitrary place. Figure 2 shows such a *subtree-transfer* operation.

The subtree-transfer distance between two trees T_1 and T_2 is the minimum number of subtrees one need to move to transform T_1 into T_2 [5,6,7]. It is sometimes



Nearest Neighbor Interchange and Related Distances, Figure 1
The two possible nni operations on an internal edge (u, v): exchange $B \leftrightarrow C$ or $B \leftrightarrow D$



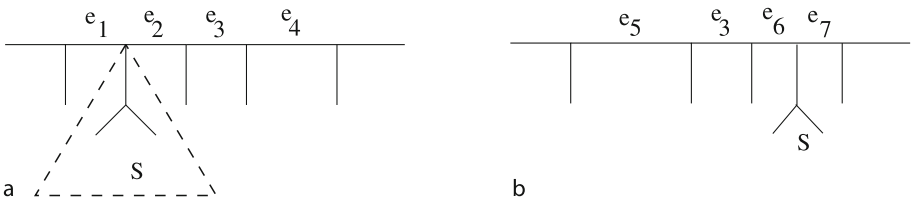
Nearest Neighbor Interchange and Related Distances, Figure 2
An example of subtree-transfer

appropriate in practice to discriminate among subtree-transfer operations as they occur with different frequencies. In this case, one can charge each subtree-transfer operation a cost equal to the distance (the number of nodes passed) that the subtree has moved in the current tree. The *linear-cost* subtree-transfer distance, $D_{\text{lcst}}(T_1, T_2)$, between two trees T_1 and T_2 is then the minimum total cost required to transform T_1 into T_2 by subtree-transfer operations [1,2].

Weighted Trees

Both the linear-cost subtree-transfer and nni models can be naturally extended to weighted trees. The extension for nni is straightforward: an nni operation is simply charged a cost equal to the weight of the edge it operates on. For feasibility of weighted nni transformation between two given weighted trees T_1 and T_2, one also requires that the following conditions are satisfied: (**1**) for each leaf label a, the weight of the edge in T_1 incident on a is the same as the weight of the edge in T_2 incident on a and (**2**) the multisets of weights of internal edges of T_1 and T_2 are the same.

In the case of linear-cost subtree-transfer, although the idea is immediate, i. e., a moving subtree should be charged for the weighted distance it travels, the formal definition needs some care and is given below. Consider (unrooted) trees in which each edge e has a weight $w(e) \geq 0$. To ensure feasibility of transforming a tree into another, One requires the total weight of all edges to equal one. A subtree-transfer is now defined as follows. Select a subtree S of T at a given node u and select an edge $e \notin S$. Split the edge e into two edges e_1 and e_2 with weights $w(e_1)$ and $w(e_2)$ $(w(e_1), w(e_2) \geq 0, w(e_1) + w(e_2) = w(e))$, and move S to the common end-point of e_1 and e_2. Finally, merge the two remaining edges e' and e'' adjacent to u into one edge with weight $w(e') + w(e'')$. The cost of this subtree-transfer is the total weight of all the edges over which S



Nearest Neighbor Interchange and Related Distances, Figure 3
Subtree-transfer on weighted phylogenies. Tree b is obtained from tree a with one subtree-transfer

is moved. Figure 1 gives an example. The edge-weights of the given tree are normalized so that their total sum is 1. The subtree S is transferred to split the edge e_4 to e_6 and e_7 such that $w(e_6), w(e_7) \geq 0$ and $w(e_6) + w(e_7) = w(e_4)$; finally, the two edges e_1 and e_2 are merged to e_5 such that $w(e_5) = w(e_1) + w(e_2)$. The cost of transferring S is $w(e_2) + w(e_3) + w(e_6)$.

Note that for weighted trees, the linear-cost subtree-transfer model is more general than the nni model in the sense that one can slide a subtree along an edge with subtree-transfers. Such an operation is not realizable with nni moves.

Key Results

Let T_1 and T_2 be the two trees, each with n nodes, that are being used in the distance computation.

Theorem 1 ([2,3,4]) *Assume that T_1 and T_2 are unweighted. Then, the following results hold:*

- $D_{nni}(T_1, T_2) = D_{lcst}(T_1, T_2)$.
- *Computing $D_{nni}(T_1, T_2)$ is NP-complete.*
- *Suppose that $D_{nni}(T_1, T_2) \leq d$. Then, an optimal sequence of nni operations transforming T_1 into T_2 can be computed in $O(n^2 \log n + n \cdot 2^{23d/2})$ time.*
- *$D_{nni}(T_1, T_2)$ can be approximated to within a factor of $\log n + O(1)$ in polynomial time.*

Theorem 2 ([1,2,3,4]) *Assume that T_1 and T_2 are weighted. Then, the following results hold:*

- *$D_{nni}(T_1, T_2)$ can be approximated to within a factor of $6 + 6 \log n$ in $O(n^2 \log n)$ time.*
- *Assume that T_1 and T_2 are allowed to have leaves that are not necessarily uniquely labeled. Then, computing $D_{lcst}(T_1, T_2)$ is NP-hard.*
- *$D_{lcst}(T_1, T_2)$ can be approximated to within a factor of 2 in $O(n^2 \log n)$ time.*

Applications

The results reported here are on transformation based distances for evolutionary trees. Such a tree is can be *rooted* if the evolutionary origin is known and can be *weighted* if the evolutionary length on each edge is known. Reconstructing the correct evolutionary tree for a set of species is one of the fundamental yet difficult problems in evolutionary genetics. Over the past few decades, many approaches for reconstructing evolutionary trees have been developed, including (not exhaustively) parsimony, compatibility, distance and maximum likelihood approaches. The outcomes of these methods usually depend on the data and the amount of computational resources applied. As a result, in practice they often lead to different trees on the same set of species [8]. It is thus of interest to compare evolutionary trees produced by different methods, or by the same method on different data.

Another motivation for investigating the linear-cost subtree transfer distance comes from the following motivation. When *recombination* of DNA sequences occurs in an evolution, two sequences meet and generate a new sequence, consisting of genetic material taken left of the recombination point from the first sequence and right of the point from the second sequence [5,6]. From a phylogenetic viewpoint, before the recombination, the ancestral material on the present sequence was located on two sequences, one having all the material to the left of the recombination point and another having all the material to the right of the breaking point. As a result, the evolutionary history can no longer be described by a single tree. The recombination event partitions the sequences into two neighboring regions. The history for the left and the right regions could be described by separate evolutionary trees. The recombination makes the two evolutionary trees describing neighboring regions differ. However, two neighbor trees cannot be arbitrarily different, one must be obtainable from the other by a *subtree-transfer operation*. When more than one recombination occurs, one can describe an evolutionary history using a list of evolutionary trees, each corresponds to some region of the sequences and each can be obtained by several subtree-transfer operations from its predecessor [6]. The computation of a linear-cost subtree-transfer distance is useful in reconstructing such a list of trees based on parsimony [5,6].

Open Problems

1. Is there a constant ratio approximation algorithm for the nni distance on unweighted evolutionary trees or is the $O(\log n)$-approximation the best possible?
2. Is the linear-cost subtree-transfer distance NP-hard to compute on weighted evolutionary trees if leaf labels are not allowed to be non-unique?
3. Can one improve the approximation ratio for linear-cost subtree-transfer distance on weighted evolutionary trees?

Cross References

- ▶ Constructing a Galled Phylogenetic Network
- ▶ Maximum Agreement Subtree (of 2 Binary Trees)
- ▶ Maximum Agreement Subtree (of 3 or More Trees)
- ▶ Phylogenetic Tree Construction from a Distance Matrix

Recommended Reading

1. DasGupta, B., He, X., Jiang, T., Li, M., Tromp, J.: On the linear-cost subtree-transfer distance. Algorithmica **25**(2), 176–195 (1999)
2. DasGupta, B., He, X., Jiang, T., Li, M., Tromp, J., Zhang, L.: On distances between phylogenetic trees, 8th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 427–436 (1997)
3. DasGupta, B., He, X., Jiang, T., Li, M., Tromp, J., Wang, L., Zhang, L.: Computing Distances between Evolutionary Trees. In: Du, D.Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization. Kluwer Academic Publishers, Norwell, **2**, 35–76 (1998)
4. DasGupta, B., He, X., Jiang, T., Li, M., Tromp, J., Zhang, L.: On Computing the Nearest Neighbor Interchange Distance. In: Du, D.Z., Pardalos, P.M., Wang, J. (eds.) Proceedings of the DIMACS Workshop on Discrete Problems with Medical Applications, DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Am. Math. Soc. **55**, 125–143 (2000)
5. Hein, J.: Reconstructing evolution of sequences subject to recombination using parsimony. Math. Biosci. **98**, 185–200 (1990)
6. Hein, J.: A heuristic method to reconstruct the history of sequences subject to recombination. J. Mol. Evol. **36**, 396–405 (1993)
7. Hein, J., Jiang, T., Wang, L., Zhang, K.: On the complexity of comparing evolutionary trees. Discret. Appl. Math. **71**, 153–169 (1996)
8. Kuhner, M., Felsenstein, J.: A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. Mol. Biol. Evol. **11**(3), 459–468 (1994)
9. Moore, G.W., Goodman, M., Barnabas, J.: An iterative approach from the standpoint of the additive hypothesis to the dendrogram problem posed by molecular data sets. J. Theor. Biol. **38**, 423–457 (1973)
10. Robinson, D.F.: Comparison of labeled trees with valency three. J Combinator. Theory Series B **11**, 105–119 (1971)

Negative Cycles in Weighted Digraphs

1994; Kavvadias, Pantziou, Spirakis, Zaroliagis

Christos Zaroliagis
Computer Engineering & Informatics,
University of Patras, Patras, Greece

Problem Definition

Let $G = (V, E)$ be an n-vertex, m-edge directed graph (digraph), whose edges are associated with a real-valued cost function $wt : E \to \mathbb{R}$. The cost, $wt(P)$, of a path P in G is the sum of the costs of the edges of P. A simple path C whose starting and ending vertices coincide is called a cycle. If $wt(C) < 0$, then C is called a *negative cycle*. The goal of the negative cycle problem is to detect whether there is such a cycle in a given digraph G with real-valued edge costs, and if indeed exists to output the cycle.

The negative cycle problem is closely related to the shortest path problem. In the latter, a minimum cost path between two vertices s and t is sought. It is easy to see that an s-t shortest path exists if and only if no s-t path in G contains a negative cycle [1,13]. It is also well-known that shortest paths from a given vertex s to all other vertices form a tree called *shortest path tree* [1,13].

Key Results

For the case of general digraphs, the best algorithm to solve the negative cycle problem (or to compute the shortest path tree, if such a cycle does not exist) is the classical Bellman–Ford algorithm that takes $O(nm)$ time (see e. g., [1]). Alternative methods with the same time complexity are given in [4,7,12,13]. Moreover, in [11, Chap. 7] an extension of the Bellman–Ford algorithm is described which, in addition to detecting and reporting the existing negative cycles (if any), builds a shortest path tree rooted a some vertex s reaching those vertices u whose shortest s-u path does not contain a negative cycle. If edge costs are integers larger than $-L$ ($L \geq 2$), then a better algorithm was given in [6] that runs in $O(m\sqrt{n}\log L)$ time, and it is based on bit scaling.

A simple deterministic algorithm that runs in $O(n^2 \log n)$ expected time with high probability is given in [10] for a large class of input distributions, where the edge costs are chosen randomly according to the endpoint-independent model (this model includes the common case where all edge costs are chosen independently from the same distribution).

Better results are known for several important classes of sparse digraphs (i. e., digraphs with $m = O(n)$ edges) such as planar digraphs, outerplanar digraphs, digraphs of small genus, and digraphs of small treewidth.

For general sparse digraphs, an algorithm is given in [8] that solves the negative cycle problem in $O(n + \tilde{\gamma}^{1.5} \log \tilde{\gamma})$ time, where $\tilde{\gamma}$ is a topological measure of the input sparse digraph G, and whose value varies from 1 up to $\Theta(n)$. Informally, $\tilde{\gamma}$ represents the minimum number of outerplanar subgraphs, satisfying certain separation properties, into which G can be decomposed. In particular, $\tilde{\gamma}$ is proportional to $\gamma(G) + q$, where G is supposed to be embedded into an orientable surface of genus $\gamma(G)$ so as to minimize the number q of faces that collectively cover all vertices. For instance, if G is outerplanar, then $\tilde{\gamma} = 1$, which implies an optimal $O(n)$ time algorithm for this case. The algorithm in [8] does not require such an embedding to be provided by the input. In the same paper, it is shown that random $G_{n,p}$ graphs with threshold function $1/n$ are planar with probability one and have an expected value for $\tilde{\gamma}$ equal to $O(1)$. Furthermore, an efficient parallelization of the algorithm on the CREW PRAM model of computation is provided in [8].

Better bounds for planar digraphs are as follows. If edge costs are integers, then an algorithm running in $O(n^{4/3} \log(nL))$ time is given in [9]. For real edge costs, an $O(n \log^3 n)$-time algorithm was given in [5].

An optimal $O(n)$-time algorithm is given in [3] for the case of digraphs with small treewidth (and real edge costs). Informally, the treewidth t of a graph G is a parameter which measures how close is the structure of G to a tree. For instance, the class of graphs of small treewidth includes series-parallel graphs ($t = 2$) and outerplanar graphs ($t = 2$). An optimal parallel algorithm for the same problem, on the EREW PRAM model of computation, is provided in [2].

Applications

Finding negative cycles in a digraph is a fundamental combinatorial and network optimization problem that spans a wide range of applications including: shortest path computation, two dimensional package element, minimum cost flows, minimal cost-to-time ratio, model verification, compiler construction, software engineering, VLSI design, scheduling, circuit production, constraint programming and image processing. For instance, the isolation of negative feedback loops is imperative in the design of VLSI circuits. It turns out that such loops correspond to negative cost cycles in the so-called amplifier-gain graph of the circuit. In constraint programming, it is required to check the feasibility of sets of constraints. Systems of difference constraints can be represented by constraint graphs, and one can show that such a system is feasible if and only if there are no negative cost cycles in its corresponding constraint graph. In zero-clairvoyant scheduling, the problem of checking whether there is a valid schedule in such a scheduling system can be reduced to detecting negative cycles in an appropriately defined graph. For further discussion on these and other applications see [1,12,14].

Open Problems

The negative cycle problem is closely related to the shortest path problem. The existence of negative edge costs makes the solution of the negative cycle problem or the computation of a shortest path tree more difficult and thus more time consuming compared to the time required to solve the shortest path tree problem in digraphs with non-negative edge costs. For instance, for digraphs with real edge costs, compare the $O(nm)$-time algorithm in the former case with the $O(m + n \log n)$-time algorithm for the latter case (Dijkstra's algorithm implemented with an efficient priority queue; see e. g., [1]).

It would therefore be interesting to try to reduce the gap between the above two time complexities, even for special classes of graphs or the case of integer costs.

The only case where these two complexities coincide concerns the digraphs of small treewidth [3], making it the currently most general such class of graphs. For planar digraphs, the result in [5] is only a polylogarithmic factor away from the $O(n)$-time algorithm in [9] that computes a shortest path tree when the edge costs are non-negative.

Experimental Results

An experimental study for the negative cycle problem is conducted in [4]. In that paper, several methods that combine a shortest path algorithm (based on the Bellman–Ford approach) with a cycle detection strategy are investigated, along with some new variations of them. It turned out that the performance of algorithms for the negative cycle problem depends on the number and the size of the negative cycles. This gives rise to a collection of problem families for testing negative cycle algorithms.

A follow-up of the above study is presented in [14], where two new heuristics are introduced and are incorporated on three of the algorithms considered in [4] (the original Bellman–Ford and the variations in [13] and [7]), achieving dramatic improvements. The data sets considered in [14] are those in [4].

Data Sets

Data set generators and problem families are described in [4], and are available from http://www.avglab.com/andrew/soft.html.

URL to Code

The code used in [4] is available from http://www.avglab.com/andrew/soft.html.

Cross References

- ▶ All Pairs Shortest Paths in Sparse Graphs
- ▶ All Pairs Shortest Paths via Matrix Multiplication
- ▶ Single-Source Shortest Paths

Recommended Reading

1. Ahuja, R., Magnanti, T., Orlin, J.: Network Flows. Prentice-Hall, Englewood Cliffs (1993)
2. Chaudhuri, S., Zaroliagis, C.: Shortest Paths in Digraphs of Small Treewidth. Part II: Optimal Parallel Algorithms. Theor. Comput. Sci. **203**(2), pp. 205–223 (1998)
3. Chaudhuri, S., Zaroliagis, C.: Shortest Paths in Digraphs of Small Treewidth. Part I: Sequential Algorithms. Algorithmca **27**(3), pp. 212–226 (2000)
4. Cherkassky, B.V., Goldberg, A.V.: Negative-Cycle Detection Algorithms. Math. Program. **85**, pp. 277–311 (1999)
5. Fakcharoenphol, J., Rao, S.: Planar Graphs, Negative Weight Edges, Shortest Paths, and near Linear Time. In: Proc. 42nd IEEE Symp. on Foundations of Computer Science – FOCS (2001), pp. 232–241. IEEE Computer Society Press, Los Alamitos (2001)
6. Goldberg, A.V.: Scaling Algorithms for the Shortest Paths Problem. SIAM J. Comput. **24**, pp. 494–504 (1995)
7. Goldberg, A.V., Radzik, T.: A Heuristic Improvement of the Bellman—Ford Algorithm. Appl. Math. Lett. **6**(3), pp. 3–6 (1993)
8. Kavvadias, D., Pantziou, G., Spirakis, P., Zaroliagis, C.: Efficient Sequential and Parallel Algorithms for the Negative Cycle Problem. In: Algorithms and Computation – ISAAC'94. Lect. Notes Comput. Sci., vol. 834, pp.270–278. Springer, Heidelberg (1994)
9. Klein, P., Rao, S., Rauch, M., Subramanian, S.: Faster shortest-path algorithms for planar graphs. J. Comput. Syst. Sci. **5**(1), pp. 3–23 (1997)
10. Kolliopoulos, S.G., Stein, C.: Finding Real-Valued Single-Source Shortest Paths in $o(n^3)$ Expected Time. J. Algorithms **28**, pp. 125–141 (1998)
11. Mehlhorn, K., Näher, S.: LEDA: A Platform for Combinatorial and Geometric Computing. Cambridge University Press, Cambridge (1999)
12. Spirakis, P., Tsakalidis, A.: A Very Fast, Practical Algorithm for Finding a Negative Cycle in a Digraph. In Proc. of 13th ICALP, pp. 397–406 (1986)
13. Tarjan, R.E.: Data Structures and Network Algorithms. SIAM, Philadelphia (1983)
14. Wong, C.H., Tam, Y.C.: Negative Cycle Detection Problem. In: Algorithms – ESA 2005. Lecture Notes in Computer Science, vol. 3669, pp. 652–663. Springer, Heidelberg (2005)

Non-approximability of Bimatrix Nash Equilibria

2006; Chen, Deng, Teng

Xi Chen[1], Xiaotie Deng[2]

[1] Computer Science and Technology, Tsinghua University, Beijing, Beijing, China

[2] Department of Computer Science, City University of Hong Kong, Hong Kong, China

Keywords and Synonyms

Approximate Nash equilibrium

Problem Definition

In this entry, the following two problems are considered: 1) the problem of finding an approximate Nash equilibrium in a positively normalized bimatrix (or two-player) game; and 2) the smoothed complexity of finding an exact Nash equilibrium in a bimatrix game. It turns out that these two problems are strongly correlated [3].

Let $\mathcal{G} = (\mathbf{A}, \mathbf{B})$ be a bimatrix game, where $\mathbf{A} = (a_{i,j})$ and $\mathbf{B} = (b_{i,j})$ are both $n \times n$ matrices. Game $\mathcal{G}$ is said to be positively normalized, if $0 \le a_{i,j}, b_{i,j} \le 1$ for all $1 \le i, j \le n$.

Let $\mathbb{P}^n$ denote the set of all probability vectors in $\mathbb{R}^n$, i. e., non-negative vectors whose entries sum to 1. A Nash equilibrium [8] of $\mathcal{G} = (\mathbf{A}, \mathbf{B})$ is a pair of mixed strategies $(\mathbf{x}^* \in \mathbb{P}^n, \mathbf{y}^* \in \mathbb{P}^n)$ such that for all $\mathbf{x}, \mathbf{y} \in \mathbb{P}^n$,

$$(\mathbf{x}^*)^{\mathrm{T}}\mathbf{A}\mathbf{y}^* \ge \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{y}^* \quad \text{and} \quad (\mathbf{x}^*)^{\mathrm{T}}\mathbf{B}\mathbf{y}^* \ge (\mathbf{x}^*)^{\mathrm{T}}\mathbf{B}\mathbf{y}\,,$$

while an ϵ-approximate Nash equilibrium is a pair $(\mathbf{x}^* \in \mathbb{P}^n, \mathbf{y}^* \in \mathbb{P}^n)$ that satisfies

$$(\mathbf{x}^*)^{\mathrm{T}}\mathbf{A}\mathbf{y}^* \ge \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{y}^* - \epsilon \quad \text{and}$$
$$(\mathbf{x}^*)^{\mathrm{T}}\mathbf{B}\mathbf{y}^* \ge (\mathbf{x}^*)^{\mathrm{T}}\mathbf{B}\mathbf{y} - \epsilon\,, \quad \text{for all} \quad \mathbf{x}, \mathbf{y} \in \mathbb{P}^n\,.$$

In the smoothed analysis [11] of bimatrix games, a perturbation of magnitude $\sigma > 0$ is first applied to the input game: For a positively normalized $n \times n$ game $\mathcal{G} = (\overline{\mathbf{A}}, \overline{\mathbf{B}})$, let $\mathbf{A}$ and $\mathbf{B}$ be two matrices with

$$a_{i,j} = \overline{a}_{i,j} + r^A_{i,j} \quad \text{and} \quad b_{i,j} = \overline{b}_{i,j} + r^B_{i,j}, \quad \forall 1 \le i, j \le n,$$

while $r^A_{i,j}$ and $r^B_{i,j}$ are chosen independently and uniformly from interval $[-\sigma, \sigma]$ or from Gaussian distribution with variance σ^2. These two kinds of perturbations are referred to as σ-*uniform* and σ-*Gaussian perturbations*, respectively. An algorithm for bimatrix games has *polynomial smoothed complexity* (under σ-uniform or σ-Gaussian perturbations) [11], if it finds a Nash equilibrium of game $(\mathbf{A}, \mathbf{B})$ in expected time poly$(n, 1/\sigma)$, for all $(\overline{\mathbf{A}}, \overline{\mathbf{B}})$.

Key Results

The complexity class **PPAD** [9] is defined in entry ► Bimatrix Nash. The following theorems are proved in [3].

Theorem 1 *For any constant $c > 0$, the problem of computing a $1/n^c$-approximate Nash equilibrium of a positively normalized $n \times n$ bimatrix game is* **PPAD***-complete.*

Theorem 2 *The problem of computing a Nash equilibrium in a bimatrix game is not in smoothed polynomial time, under uniform or Gaussian perturbations, unless* **PPAD** $\subseteq$ **RP**.

Corollary 1 *The smoothed complexity of the Lemke-Howson algorithm is not polynomial, under uniform or Gaussian perturbations, unless* **PPAD** $\subseteq$ **RP**.

Applications

See entry ► Bimatrix Nash.

Open Problems

There remains a complexity gap on the approximation of Nash equilibria in bimatrix games: The result of [7] shows that, an ϵ-approximate Nash equilibrium can be computed in $n^{O(\log n/\epsilon^2)}$-time, while [3] show that no algorithm can find an ϵ-approximate Nash equilibrium in poly$(n, 1/\epsilon)$-time for ϵ of order 1/poly(n), unless **PPAD** is in **P**. However, the hardness result of [3] does not cover the case when ϵ is a constant between 0 and 1. Naturally, it is unlikely that the problem of finding an ϵ-approximate Nash equilibrium is **PPAD**-complete when ϵ is an absolute constant, for otherwise, all the search problems in **PPAD** would be solvable in $n^{O(\log n)}$-time, due to the result of [7]. An interesting open problem is that, for every constant $\epsilon > 0$, is there a polynomial-time algorithm for finding an ϵ-approximate Nash equilibrium? The following conjectures are proposed in [3]:

Conjecture 1 *There is an $O(n^{k+\epsilon^{-c}})$-time algorithm for finding an ϵ-approximate Nash equilibrium in a bimatrix game, for some constants c and k.*

Conjecture 2 *There is an algorithm to find a Nash equilibrium in a bimatrix game with smoothed complexity $O(n^{k+\sigma^{-c}})$ under perturbations with magnitude σ, for some constants c and k.*

It is also conjectured in [3] that Corollary 1 remains true without any complexity assumption on class **PPAD**. A positive answer would extend the result of [10] to the smoothed analysis framework.

Cross References

► Complexity of Bimatrix Nash Equilibria
► General Equilibrium
► Leontief Economy Equilibrium

Recommended Reading

1. Chen, X., Deng, X.: 3-Nash is PPAD-complete. ECCC, TR05-134 (2005)
2. Chen, X., Deng, X.: Settling the complexity of two-player Nash equilibrium. In: FOCS'06: Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, pp. 261–272
3. Chen, X., Deng, X., Teng, S.H.: Computing Nash equilibria: approximation and smoothed complexity. In: FOCS'06: Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, pp. 603–612
4. Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a Nash equilibrium. In: STOC'06: Proceedings of the 38th ACM Symposium on Theory of Computing, 2006, pp. 71–78
5. Daskalakis, C., Papadimitriou, C.H.: Three-player games are hard. ECCC, TR05-139 (2005)
6. Goldberg, P.W., Papadimitriou, C.H.: Reducibility among equilibrium problems. In: STOC'06: Proc. of the 38th ACM Symposium on Theory of Computing, 2006, pp. 61–70
7. Lipton, R., Markakis, E., Mehta, A.: Playing large games using simple strategies. In: Proc. of the 4th ACM conference on Electronic commerce, 2003, pp. 36–41
8. Nash, J.F.: Equilibrium point in n-person games. Proc. Natl. Acad. Sci. USA **36**(1), 48–49 (1950)
9. Papadimitriou, C.H.: On the complexity of the parity argument and other inefficient proofs of existence. J. Comput. Syst. Sci. **48**, 498–532 (1994)
10. Savani, R., von Stengel, B.: Exponentially many steps for finding a Nash equilibrium in a bimatrix game. In: FOCS'04: Proc. of the 45th Annual IEEE Symposium on Foundations of Computer Science, 2004, pp. 258–267
11. Spielman, D.A., Teng, S.H.: Smoothed analysis of algorithms and heuristics: progress and open questions. In: Pardo, L.M., Pinkus, A., Süli, E., Todd, M.J. (eds.) Foundations of Computational Mathematics, pp. 274–342. Cambridge University Press, Cambridge, UK (2006)

Non-shared Edges

1985; Day

Wing-Kai Hon
Department of Computer Science, National Tsing Hua University, Hsin Chu, Taiwan

Keywords and Synonyms

Robinson–Foulds distance; Robinson–Foulds metric

Problem Definition

Phylogenies are binary trees whose leaves are labeled with distinct leaf labels. This problem in this article is concerned with a well-known measurement, called *non-shared edge distance*, for comparing the dissimilarity between two phylogenies. Roughly speaking, the non-shared edge distance counts the number of edges that differentiate one phylogeny from the other.

Let e be an edge in a phylogeny T. Removing e from T splits T into two subtrees. The leaf labels are partitioned into two subsets according to the subtrees. The edge e is said to *induce* a partition of the set of leaf labels. Given two phylogenies T and T' having the same number of leaves with the same set of leaf labels, an edge e in T is *shared* if there exists some edge e' in T' such that the edges e and e' induce the same partition of the set of leaf labels in their corresponding tree. Otherwise, e is *non-shared*. Notice that T and T' has the same number of edges, so that the number of non-shared edges in T (with respect to T') is the same as the number of non-shared edges in T' (with respect to T). Such a number is called the *non-shared edge distance* between T and T'. Two problems are defined as follows:

Non-shared Edge Distance Problem
INPUT: Two phylogenies on the same set of leaf labels
OUTPUT: The non-shared edge distance between the two input phylogenies

All-Pairs Non-shared Edge Distance Problem
INPUT: A collection of phylogenies on the same set of leaf labels
OUTPUT: The non-shared edge distance between each pair of the input phylogenies

Extension

Phylogenies that are commonly used in practice have weights associated to the edges. The notion of non-shared edge can be easily extended for edge-weighted phylogenies. In this case, an edge e will induce a partition of the set of leaf labels as well as the multi-set of edge weights (here, edge weights are allowed to be non-distinct). Given two edge-weighted phylogenies R and R' having the same set of leaf labels and the same multi-set of edge weights, an edge e in R is *shared* if there exists some edge e' in R' such that the edges e and e' induce the same partition of the set of leaf labels and the multi-set of edge weights. Otherwise, e is *non-shared*. The *non-shared edge distance* between R and R' are similarly defined, giving the following problem:

General Non-shared Edge Distance Problem
INPUT: Two edge-weighted phylogenies on the same set of leaf labels and same multi-set of edge weights
OUTPUT: The non-shared edge distance between the two input phylogenies

Key Results

Day [3] proposed the first linear-time algorithm for the Non-shared Edge Distance Problem.

Theorem 1 *Let T and T' be two input phylogenies with the same set of leaf labels, and n be the number of leaves in each phylogeny. The non-shared edge distance between T and T' can be computed in $O(n)$ time.*

Let Δ be a collection of k phylogenies on the same set of leaf labels, and n be the number of leaves in each phylogeny. The All-Pairs Non-shared Edge Distance Problem can be solved by applying Theorem 1 on each pair of phylogenies, thus solving the problem in a total of $O(k^2 n)$ time. Pattengale and Moret [9] proposed a randomized result based on [7] to solve the problem approximately, whose running time is faster when $n \leq k \leq 2^n$.

Theorem 2 *Let ε be a parameter with $\varepsilon > 0$. Then, there exists a randomized algorithm such that with probability at least $1 - k^{-2}$, the non-shared edge distance between each pair of phylogenies in Δ can be approximated within a factor of $(1 + \varepsilon)$ from the actual distance; the running time of the algorithm is $O(k(n^2 + k \log k)/\varepsilon^2)$.*

For general phylogenies, let R and R' be two input phylogenies with the same set of leaf labels and the same multi-set of edge weights, and n be the number of leaves in each phylogeny. The General Non-shared Distance Problem can be solved easily in $O(n^2)$ time by applying Theorem 1 in a straightforward manner. The running time is improved by Hon et al. in [5].

Theorem 3 *The non-shared edge distance between R and R' can be computed in $O(n \log n)$ time.*

Applications

Phylogenies are commonly used by biologists to model the evolutionary relationship among species. Many reconstruction methods (such as maximum parsimony, maximum likelihood, compatibility, distance matrix) produce different phylogenies based on the same set of species, and it is interesting to compute the dissimilarities between them. Also, through the comparison, information about rare genetic events such as recombinations or gene conversions may be uncovered. The most common dissimi-

larity measure is the Robinson–Foulds metric [11], which is exactly the same as the non-shared edge distance.

Other dissimilarity measures, such as the nearest-neighbor interchange (NNI) distance and the subtree-transfer (STT) distance (see [2] for details), are also proposed in the literature. These measures are sometimes preferred by the biologists since they can be used to deduce the biological events that create the dissimilarity. Nevertheless, these measures are usually difficult to compute. In particular, computing the NNI distance and the STT distance are shown to be NP-hard by DasGupta et al. [1,2]. Approximation algorithms are devised for these problems (NNI: [4,8], STT: [1,6]). Interestingly, all these algorithms make use of the non-shared edge distance to bound their approximation ratios.

Cross References

A related problem is to measure the *similarity* between two input phylogenies. ► Maximum Agreement Subtree for references.

Recommended Reading

1. DasGupta, B., He, X., Jiang, T., Li, M., Tromp, J.: On the Linear-Cost Subtree-Transfer Distance between Phylogenetic Trees. Algorithmica **25**(2–3), 176–195 (1999)
2. DasGupta, B., He, X., Jiang, T., Li, M., Tromp, J., Zhang, L.: On Distances between Phylogenetic Trees. In: Proceedings of the Eighth ACM-SIAM Annual Symposium on Discrete Algorithms (SODA), New Orleans, pp. 427–436. SIAM, Philadelphia (1997)
3. Day, W.H.E.: Optimal Algorithms for Comparing Trees with Labeled Leaves. J. Classif. **2**, 7–28 (1985)
4. Hon, W.K., Lam, T.W.: Approximating the Nearest Neighbor Intercharge Distance for Non-Uniform-Degree Evolutionary Trees. Int. J. Found. Comp. Sci. **12**(4), 533–550 (2001)
5. Hon, W.K., Kao, M.Y., Lam, T.W., Sung, W.K., Yiu, S.M.: Non-shared Edges and Nearest Neighbor Interchanges revisited. Inf. Process. Lett. **91**(3), 129–134 (2004)
6. Hon, W.K., Lam, T.W., Yiu, S.M., Kao, M.Y., Sung, W.K.: Subtree Transfer Distance For Degree-D Phylogenies. Int. J. Found. Comp. Sci. **15**(6), 893–909 (2004)
7. Johnson, W., Lindenstrauss, J.: Extensions of Lipschitz Mappings into a Hilbert Space. Contemp. Math. **26**, 189–206 (1984)
8. Li, M., Tromp, J., Zhang, L.: Some Notes on the Nearest Neighbour Interchange Distance. J. Theor. Biol. **26**(182), 463–467 (1996)
9. Pattengale, N.D., Moret, B.M.E.: A Sublinear-Time Randomized Approximation Scheme for the Robinson–Foulds Metric. In: Proceedings of the Tenth ACM Annual International Conference on Research in Computational Molecular Biology (RECOMB), pp. 221–230. Venice, Italy, April 2–5 2006
10. Robinson, D.F.: Comparison of Labeled Trees with Valency Three. J. Comb. Theor. **11**, 105–119 (1971)
11. Robinson, D.F., Foulds, L.R.: Comparison of Phylogenetic Trees. Math. Biosci. **53**, 131–147 (1981)

Nucleolus

2006; Deng, Fang, Sun

Qizhi Fang
Department of Mathematics, Ocean University of China, Qingdao, China

Keywords and Synonyms

Nucleon; Kernel

Problem Definition

Cooperative game theory considers how to distribute the total income generated by a set of participants in a joint project to individuals. The Nucleolus, trying to capture the intuition of minimizing dissatisfaction of players, is one of the most well-known solution concepts among various attempts to obtain a unique solution. In Deng, Fang and Sun's work [2], they study the Nucleolus of flow games from the algorithmic point of view. It is shown that, for a flow game defined on a simple network (arc capacity being all equal), computing the Nucleolus can be done in polynomial time, and for flow games in general cases, both the computation and the recognition of the Nucleolus are $\mathcal{NP}$-hard.

A cooperative (profit) game (N, v) consists of a player set $N = \{1, 2, \cdots, n\}$ and a characteristic function $v : 2^N \to R$ with $v(\emptyset) = 0$, where the value $v(S)$ $(S \subseteq N)$ is interpreted as the profit achieved by the collective action of players in S. Any vector $x \in R^n$ with $\sum_{i \in N} x_i = v(N)$ is an allocation. An allocation x is called an *imputation* if $x_i \geq v(\{i\})$ for all $i \in N$. Denote by $\mathcal{I}(v)$ the set of imputations of the game.

Given an allocation x, the *excess* of a coalition $S(S \subseteq N)$ at x is defined as

$$e(S, x) = x(S) - v(S) ,$$

where $x(S) = \sum_{i \in S} x_i$ for $S \subseteq N$. The value $e(S, x)$ can be interpreted as a measure of satisfaction of coalition S with the allocation x. The core of the game (N, v), denoted by $C(v)$, is the set of allocations whose excesses are all non-negative. For an allocation x of the game (N, v), let $\theta(x)$ denote the $(2^n - 2)$-dimensional vector whose components are the non-trivial excesses $e(S, x)$, $\emptyset \neq S \neq N$, arranged in a non-decreasing order. That is, $\theta_i(x) \leq \theta_j(x)$, for $1 \leq i < j \leq 2^n - 2$. Denote by $\succeq_l$ the "lexicographically greater than" relationship between vectors of the same dimension.

Definition 1 The Nucleolus $\eta(v)$ of game (N, v) is the set of imputations that lexicographically maximize $\theta(x)$ over

the set of all imputations $x \in \mathcal{I}(v)$. That is,

$$\eta(v) = \{x \in \mathcal{I}(v) : \theta(x) \succeq_l \theta(y) \text{ for all } y \in \mathcal{I}(v)\}\ .$$

Even though, the Nucleolus may contain multiple points by the definition, it was proved by Schmeidler [12] that the Nucleolus of a game with non-empty imputation set contains exactly one element. Kopelowitz [10] proposed that the Nucleolus can be obtained by recursively solving a sequential linear programs (SLP):

$$LP_k: \quad \begin{array}{ll} max\ \varepsilon & \\ x(S) = v(S) + \varepsilon_r & \forall S \in \mathcal{J}_r\ \ r = 0, 1, \cdots, k-1 \\ x(S) \geq v(S) + \varepsilon & \forall S \in 2^N \setminus \bigcup_{r=0}^{k-1} \mathcal{J}_r \\ x \in \mathcal{I}(v). & \end{array}$$

Here, $\mathcal{J}_0 = \{\emptyset, N\}$ and $\varepsilon_0 = 0$ initially; the number ε_r is the optimum value of the r-th program (LP_r), and $\mathcal{J}_r = \{S \in 2^N : x(S) = v(S) + \varepsilon_r \text{ for every } x \in X_r\}$, where $X_r = \{x \in \mathcal{I}(v) : (x, \varepsilon_r) \text{ is an optimal solution to } LP_r\}$. It can be shown that after at most $n-1$ iterations, one arrives at a unique optimal solution (x^*, ε_k), where x^* is just the Nucleolus of the game. In addition, the set of optimal solutions X_1 to the first program LP_1 is called the least core of the game.

The definition of the Nucleolus entails comparisons between vectors of exponential length, and with linear programming approach, each linear programs in (SLP) may possess exponential size in the number of players. Clearly, both do not provide an efficient solution in general.

Flow games, first studied in Kailai and Zemel [8,9], arise from the profit distribution problem related to the maximum flow in a network. Let $D = (V, E; \omega; s, t)$ be a directed flow network, where V is the vertex set, E is the arc set, $\omega : E \to R^+$ is the arc capacity function, s and t are the source and the sink of the network, respectively. The network D is *simple* if $\omega(e) = 1$ for each $e \in E$, which is denoted briefly by $D = (V, E; s, t)$.

Definition 2 The flow game $\Gamma_f = (E, v)$ associated with network $D = (V, E; \omega; s, t)$ is defined by

(i) The player set is E;
(ii) $\forall S \subseteq E$, $v(S)$ is the value of a maximum flow from s to t in the subnetwork of D consisting only of arcs belonging to S.

Problem 1 (Computing the Nucleolus)
INSTANCE: A flow network $D = (V, E; \omega; s, t)$.
QUESTION: Is there a polynomial time algorithm to compute the Nucleolus of the flow game associated with D?

Problem 2 (Recognizing the Nucleolus)
INSTANCE: A flow network $D = (V, E; \omega; s, t)$ and $y : E \to R^+$.
QUESTION: Is it true that y is the Nucleolus of the flow game associated with D?

Key Results

Theorem 1 *Let $D = (V, E; s, t)$ be a simple network and $\Gamma_f = (E, v)$ be the associated flow game. Then the Nucleolus $\eta(v)$ can be computed in polynomial time.*

By making use of duality technique in linear programming, Kalai and Zemel [9] gave an characterization on the core of a flow game. They further conjectured that their approach may serve as a practical basis for computing the Nucleolus. In fact, the proof of Theorem 1 in the work of Deng Fang and Sun [2] is just an elegant application of Kalai and Zemel's approach (especially the duality technique), and hence settling their conjecture.

Theorem 2 *Given a flow game $\Gamma_f = (E, v)$ defined on network $D = (V, E; \omega; s, t)$, computing the Nucleolus $\eta(v)$ is $\mathcal{NP}$-hard.*

Theorem 3 *Given a flow game $\Gamma_f = (E, v)$ defined on network $D = (V, E; \omega; s, t)$, and an imputation $y \in \mathcal{I}(v)$, checking whether y is the Nucleolus of Γ_f is $\mathcal{NP}$-hard.*

Although a flow game can be formulated as a linear production game [1], the size of reduction may in general be exponential in space, and consequently, their complexity results on the Nucleolus are independent. However, in the $\mathcal{NP}$-hardness proof of Theorem 2 and 3, the flow game constructed possesses a polynomial size formulation of linear production game [2]. Therefore, as a direct corollary, the same $\mathcal{NP}$-hardness conclusions for linear production games are obtained. That is, both computing and recognizing the Nucleolus of a linear production game are $\mathcal{NP}$-hard.

Applications

As an important solution concept in economics and game theory, the Nucleolus and related solution concepts have been applied to study insurance policies, real estate and bankruptcy, etc. However, it is a challenging problem in mathematical programming to decide what classes of cooperative games permit polynomial time computation of the Nucleolus.

The first polynomial time algorithm for Nucleolus in a special tree game was proposed by Megiddo [11], in advocation of efficient algorithms for cooperative game solutions. Subsequently, some efficient algorithms have been developed for computing the Nucleolus, such as, for assignment games [13] and matching games [7]. On the negative side, $\mathcal{NP}$-hardness result was obtained for minimum cost spanning tree games [3].

Granot, Granot and Zhu [6] observed that most of the efficient algorithms for computing the Nucleolus are based on the fact that the information needed to completely characterize the Nucleolus is much less than that dictated by its definition. Therefore, they introduced the concept of a characterization set for the Nucleolus to embody the notion of "minimum" relevant information needed for determining the Nucleolus. Furthermore, based on the sequential linear programs (SLP), they established a general relationship between the size of a characterization set and the complexity of computing the Nucleolus. Following this line of development, some known efficient algorithms for computing the Nucleolus are derived directly.

Another approach to computing the Nucleolus is taken by Faigle, Kern and Kuipers [4], which is motivated by Schmeidler's observation that the Nucleolus of a game lies in the kernel [12]. In the case where the kernel of the game contains exactly one core vector and the minimum excess for any given allocation can be compute efficiently, their approach derives a polynomial time algorithm for the Nucleolus. This also generalizes some known results on the computation of the Nucleolus. However, their algorithm uses the ellipsoid method as a subroutine, it implies that the efficiency of the algorithm is of a more theoretical kind.

Open Problems

The field of combinatorial optimization has much to offer for the study of cooperative games. It is usually the case that the value of subgroup of players can be obtained via a combinatorial optimization problem, where the game is called a combinatorial optimization game. This class of games leads to the applications of a variety of combinatorial optimization techniques in design and analysis of algorithms, as well as establishing complexity results. One of the most interesting result is the LP duality characterization of the core [1]. However, little work dealt with the Nucleolus by using the duality technique so far. Hence, the work of Deng, Fang and Sun [2] on computing the Nucleolus may be of independent interest.

There are still many unsolved complexity questions concerning the Nucleolus. For the computation of the Nucleolus of matching games, Kern and Paulusma [7] proposed an efficient algorithm in unweighted case, and conjectured that it is in general $\mathcal{NP}$-hard. Since both simple flow game and matching game fall into the class of packing/covering games, it is interesting to know the complexity of computing the Nucleolus for other game models in this class, such as, vertex covering games and minimum coloring games.

For cooperative games arising from $\mathcal{NP}$-hard combinatorial optimization problems, the computation of the Nucleolus may in general be a hard task. For example, in a traveling salesman game, nodes of the graph are the players and an extra node 0, and the value of a subgroup S of players is the length of a minimum Hamiltonian tour in the subgraph induced by $S \cup \{0\}$ [1]. It would not be surprising if one shows that both the computation and the recognition of the Nucleolus for this game model are $\mathcal{NP}$-hard. However, this is not known yet. The same questions are proposed for facility location games [5], though there have been efficient algorithms for some special cases.

Moreover, when the computation of the Nucleolus is difficult, it is also interesting to seek for meaningful approximation concepts of the Nucleolus, especially from the political and economic background.

Cross References

- ► Complexity of Core
- ► Routing

Recommended Reading

1. Deng, X.: Combinatorial Optimization and Coalition Games. In: Du, D., Pardalos, P.M. (eds.) Handbook of combinatorial optimization, vol 2, pp 77–103, Kluwer, Boston (1998)
2. Deng, X., Fang, Q., Sun, X.: Finding Nucleolus of Flow Games. Proceedings of the 17th annual ACM-SIAM symposium on Discrete algorithm (SODA 2006). Lect. Notes in Comput. Sci. **3111**, 124–131 (2006)
3. Faigle, U., Kern, W., Kuipers, J.: Computing the Nucleolus of Min-cost Spanning Tree Games is $\mathcal{NP}$-hard. Int. J. Game Theory **27**, 443–450 (1998)
4. Faigle, U., Kern, W., Kuipers, J.: On the Computation of the Nucleolus of a Cooperative Game. Int. J. Game Theory **30**, 79–98 (2001)
5. Goemans, M.X., Skutella, M.: Cooperative Facility Location Games. J. Algorithms **50**, 194–214 (2004)
6. Granot, D., Granot, F., Zhu, W.R.: Characterization Sets for the Nucleolus. Int. J. Game Theory **27**, 359–374 (1998)
7. Kern, W., Paulusma, D.: Matching Games: The Least Core and the Nucleolus. Math. Oper. Res. **28**, 294–308 (2003)
8. Kalai, E., Zemel, E.: Totally Balanced Games and Games of Flow. Math. Oper. Res. **7**, 476–478 (1982)
9. Kalai, E., Zemel, E.: Generalized Network Problems Yielding Totally Balanced Games. Oper. Res. **30**, 998–1008 (1982)

10. Kopelowitz, A.: Computation of the Kernels of Simple Games and the Nucleolus of *n*-person Games. RM-31, Math. Dept., The Hebre University of Jerusalem (1967)
11. Megiddo, N.: Computational Complexity and the Game Theory Approach to Cost Allocation for a Tree. Math. Oper. Res. **3**, 189–196 (1978)
12. Schmeidler, D.: The Nucleolus of a Characteristic Function Game. SIAM J. Appl. Math. **17**, 1163–1170 (1969)
13. Solymosi, T., Raghavan, T.E.S.: An Algorithm for Finding the Nucleolus of Assignment Games. Int. J. Game Theory **23**, 119–143 (1994)