M

Majority Equilibrium

2003; Chen, Deng, Fang, Tian

QIZHI FANG
Department of Mathematics, Ocean University of China, Qingdao, China

Keywords and Synonyms

Condorcet winner

Problem Definition

Majority rule is arguably the best decision mechanism for public decision making, which is employed not only in public management but also in business management. The concept of majority equilibrium captures such a democratic spirit in requiring that no other solutions would please more than half of the voters in comparison to it. The work of Chen, Deng, Fang, and Tian [1] considers a public facility location problem decided via a voting process under the majority rule on a discrete network. This work distinguishes itself from previous work by applying the computational complexity approach to the study of majority equilibrium. For the model with a single public facility located in trees, cycles, and cactus graphs, it is shown that the majority equilibrium can be found in linear time. On the other hand, when the number of public facilities is taken as the input size (not a constant), finding a majority equilibrium is shown to be $\mathcal{NP}$-hard.

Consider a network $G = ((V, \omega), (E, l))$ with vertex and edge weight functions $\omega : V \to R^+$ and $l : E \to R^+$, respectively. Each vertex $i \in V$ represents a community, and $\omega(i)$ represents the number of voters that reside there. For each $e \in E$, $l(e) > 0$ represents the length of the road $e = (i, j)$ connecting two communities i and j. For two vertices $i, j \in V$, the distance between i and j, denoted by $d_G(i, j)$, is the length of a shortest path joining them. The location of a public facility such as a library, community center, etc., is to be determined by the public via a voting process under the majority rule. Here, each member of the community desires to have the public facility close to himself, and the decision has to be agreed upon by a majority of the voters. Denote the vertex set of G by $V = \{v_1, v_2, \cdots, v_n\}$. Then each $v_i \in V$ has a preference order $\geq_i$ on V induced by the distance on G. That is, $x \geq_i y$ if and only if $d_G(v_i, x) \leq d_G(v_i, y)$ for two vertices $x, y \in V$; similarly, $x >_i y$ if and only if $d_G(v_i, x) < d_G(v_i, y)$. Under such a preference profile, four types of majority equilibrium, called Condorcet winners, are defined as follows.

Definition 1 Let $v_0 \in V$, then v_0 is called:

(1) *a weak quasi-Condorcet winner*, if for every $u \in V$ distinct of v_0,

$$\omega(\{v_i \in V : v_0 \geq_i u\}) \geq \sum_{v_i \in V} \omega(v_i)/2;$$

(2) *a strong quasi-Condorcet winner*, if for every $u \in V$ distinct of v_0,

$$\omega(\{v_i \in V : v_0 \geq_i u\}) > \sum_{v_i \in V} \omega(v_i)/2;$$

(3) *a weak Condorcet winner*, if for every $u \in V$ distinct of v_0,

$$\omega(\{v_i \in V : v_0 >_i u\}) \geq \omega(\{v_i \in V : u >_i v_0\});$$

(4) *a strong Condorcet winner*, if for every $u \in V$ distinct of v_0,

$$\omega(\{v_i \in V : v_0 >_i u\}) > \omega(\{v_i \in V : u >_i v_0\}).$$

Under the majority voting mechanism described above, the problem is to develop efficient ways for determining the existence of Condorcet winners and finding such a winner when one exists.

Problem 1 ***(Finding Condorcet Winners)***
INPUT: A network $G = ((V, w), (E, l))$.
OUTPUT: A Condorcet winner $v \in V$, or nonexistence of Condorcet winners.

Key Results

The mathematical results given in this section depend deeply on the understanding of combinatorial structures of underlying networks. Theorem 1, 2, and 3 below are given for weak quasi-Condorcet winners in the model with a single facility to be located. Other kinds of Condorcet winners can be discussed similarly.

Theorem 1 *Every tree has one weak quasi-Condorcet winner, or two adjacent weak quasi-Condorcet winners, which can be found in linear time.*

Theorem 2 *Let C_n be a cycle of order n with vertex-weight function $\omega : V(C_n) \to R^+$. Then $v \in V(C_n)$ is a weak quasi-Condorcet winner of C_n if and only if the weight of each $\lfloor \frac{n+1}{2} \rfloor$-interval containing v is at least $\frac{1}{2} \sum_{v \in C_n} \omega(v)$. Furthermore, the problem of finding a weak quasi-Condorcet winner of C_n is solvable in linear time.*

Given a graph $G = (V, E)$, a vertex v of G is a *cut vertex* if $E(G)$ can be partitioned into two nonempty subsets E_1 and E_2 such that the induced graphs $G[E_1]$ and $G[E_2]$ have just the vertex v in common. A *block* of G is a connected subgraph of G that has no cut vertices and is maximal with respect to this property. Every graph is the union of its blocks. A graph G is called a *cactus graph*, if G is a connected graph in which each block is an edge or a cycle.

Theorem 3 *The problem of finding a weak quasi-Condorcet winner of a cactus graph with vertex-weight function is solvable in linear time.*

In general, the problem can be extended to the cases where a number of public facilities are required to be located during one voting process, and the definitions of Condorcet winners can also be extended accordingly. In such cases, the public facilities may be of the same type, or different types; and the utility functions of the voters may be of different forms.

Theorem 4 *If there are a bounded constant number of public facilities to be located at one voting process under the majority rule, then the problem of finding a Condorcet winner (any of the four types) can be solved in polynomial time.*

Theorem 5 *If the number of public facilities to be located is not a constant but considered as the input size, the problem of finding a Condorcet winner is $\mathcal{NP}$-hard; and the corresponding decision problem: deciding whether a candidate set of public facilities is a Condorcet winner, is co-$\mathcal{NP}$-complete.*

Applications

Damange [2] first reviewed continuous and discrete spatial models of collective choice, aiming at characterizing the public facility location problem as a result of the pubic voting process. Although the network models in Chen et al. [1] have been studied for some problems in economics [3,4], the principal point of departure in Chen et al.'s work is the computational complexity and algorithmic approach. This approach can be applied to more general public decision-making processes.

For example, consider a public road repair problem, pioneered by Tullock [5] to study redistribution of tax revenue under a majority rule system. An edge-weighted graph $G = (V, E, w)$ represents a network of local roads, where the weight of each edge represents the cost of repairing the road. There is also a distinguished vertex $s \in V$ representing the entry point to the highway system. The majority decision problem involves a set of agents $A \subseteq V$ situated at vertices of the network who would choose a subset F of edges. The cost of repairing F, which is the sum of the weights of edges in F, will be shared by all n agents, each an n-th of the total. In this model, a majority stable solution under the majority rule is a subset $F \subseteq E$ that connects s to a subset $A_1 \subset A$ of agents with $|A_1| > |A|/2$ such that no other solution H connecting s to a subset of agents $A_2 \subset A$ with $|A_2| > |A|/2$ satisfies the conditions that $\sum_{e \in H} w(e) \leq \sum_{e \in F} w(e)$, and for each agent in A_2, its shortest path to s in solution H is not longer than that in solution F, and at least one of the inequalities is strict. It is shown in Chen et al. [1] that for this model, finding a majority equilibrium is $\mathcal{NP}$-hard for general networks, and is polynomially solvable for tree networks.

Cross References

- ▶ General Equilibrium
- ▶ Leontief Economy Equilibrium
- ▶ Local Search for K-medians and Facility Location

Recommended Reading

1. Chen, L., Deng, X., Fang, Q., Tian, F.: Majority equilibrium for public facility allocation. Lect. Notes Comput. Sci. **2697**, 435–444 (2002)
2. Demange, G.: Spatial Models of Collective Choice. In: Thisse, J.F., Zoller, H.G. (eds.) Locational Analysis of Public Facilities, North-Holland Publishing Company, North Holland, Amsterdam (1983)
3. Hansen, P., Thisse, J.F.: Outcomes of voting and planning: condorcet, weber and rawls locations. J. Publ. Econ. **16**, 1–15 (1981)
4. Schummer, J., Vohra, R.V.: Strategy-proof location on a network. J. Econ. Theor. **104**, 405–428 (2002)
5. Tullock, G.: Some problems of majority voting. J. Polit. Econ. **67**, 571–579 (1959)

Market Games and Content Distribution

2005; Mirrokni

Vahab S. Mirrokni
Theory Group, Microsoft Research, Redmond, WA, USA

Keywords and Synonyms

Market sharing games; Valid-Utility games; Congestion games; Stable matching

Problem Definition

This chapter studies market games for their performance and convergence of the equilibrium points. The main application is the content distribution in cellular networks in which a service provider needs to provide data to users. The service provider can use several cache locations to store and provide the data. The assumption is that cache locations are selfish agents (resident subscribers) who want to maximize their own profit. Most of the results apply to a general framework of monotone two-sided markets.

Uncoordinated Two-Sided Markets

Various economic interactions can be modeled as two-sided markets. A two-sided market consists of two disjoint groups of agents: active agents and passive agents. Each agent has a preference list over the agents of the other side, and can be matched to one (or many) of the agents in the other side. A central solution concept to these markets are *stable matchings*, introduced by Gale and Shapley [5]. It is well known that stable matchings can be achieved using a centralized polynomial-time algorithm. Many markets, however, do not have any centralized matching mechanism to match agents. In those markets, matchings are formed by actions of self-interested agents. Knuth [9] introduced uncoordinated two-sided markets. In these markets, cycles of better or best responses exist, but random better response and best response dynamics converge to a stable matching with probability one [2,10,14]. Our model for content distribution corresponds to a special class of uncoordinated two-sided markets that is called *the distributed caching games*.

Before introducing the distributed caching game as an uncoordinated two-sided market, the distributed caching problem and some game theoretic notations are defined.

Distributed Caching Problem

Let U be a set of n cache locations with given available capacities A_i and given available bandwidths B_i for each cache location i. There are k request types;[1] each request type t has a size a_t ($1 \le t \le k$). Let H be a set of m requests with a reward R_j, a required bandwidth b_j, a request type t_j for each request j, and a cost c_{ij} for connecting each cache location i to each request j. The profit of providing request j by cache location i is $f_{ij} = R_j - c_{ij}$. A cache location i can service a set of requests S_i, if it satisfies the *bandwidth constraint*: $\sum_{j \in S_i} b_j \le B_i$, and the *capacity constraint*: $\sum_{t \in \{t_j | j \in S_i\}} a_t \le A_i$ (this means that the sum of the sizes of the request types of the requests in cache location i should be less than or equal to the available capacity of cache location i). A set S_i of requests is feasible for cache location i if it satisfies both of these constraints. The goal of the DCP problem is to find a feasible assignment of requests to cache locations to maximize the total profit; i. e., the total reward of requests that are provided minus the connection costs of these requests.

Strategic Games

A *strategic game* $\mathcal{G}$ is defined as a tuple $\mathcal{G}(U, \{F_i | i \in U\}, \{\alpha_i() | i \in U\})$ where (i) U is the set of n players or agents, (ii) F_i is a family of feasible *(pure) strategies* or *actions* for player i and (iii) $\alpha_i : \Pi_{i \in U} F_i \to \mathbb{R}^+ \cup \{0\}$ is the (private) *payoff* or *utility* function for agent i, given the set of strategies of all players. Player i's strategy is denoted by $s_i \in F_i$. A *strategy profile* or a (strategy) *state*, denoted by $S = (s_1, s_2, \ldots, s_n)$, is a vector of strategies of players. Also let $S \oplus s'_i := (s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_k)$.

Best-Response Moves

In a *non-cooperative* game, each agent wishes to maximize its own payoff. For a strategy profile $S = (s_1, s_2, \ldots, s_n)$, a *better response move* of player i is a strategy s'_i such that $\alpha_i(S \oplus s'_i) \ge \alpha_i(S)$. In a *strict better response move*, the above inequality is strict. Also, for a strategy profile $S = (s_1, s_2, \ldots, s_n)$ a *best response* of player i in S is a better response move $s^*_i \in F_i$ such that for any strategy $s_i \in F_i$, $\alpha_i(S \oplus s^*_i) \ge \alpha_i(S \oplus s_i)$.

Nash Equilibria

A pure strategy Nash equilibrium (PSNE) of a strategic game is a strategy profile in which each player plays his best response.

[1] Request type can be thought of as different files that should be delivered to clients.

State Graph

The state graph, $\mathcal{D} = (\mathcal{F}, \mathcal{E})$, of a strategic game $\mathcal{G}$, is an arc-labeled directed graph, where the vertex set $\mathcal{F}$ corresponds to the set of strategy profiles or states in $\mathcal{G}$, and there is an arc from state S to state S' with label i if the only difference between S and S' is in the strategy of player i; and player i plays one of his best responses in strategy profile S'. A *best-response walk* is a directed walk in the state graph.

Price of Anarchy

Given a strategic game, $\mathcal{G}(U, \{F_i | i \in U\}, \{\alpha() | i \in U\})$, and a maximization social function $\gamma : \Pi_{i \in U} F_i \to \mathbb{R}$, the price of anarchy, denoted by $\mathsf{poa}(\mathcal{G}, \gamma)$, is the worst ratio between the social value of a pure Nash equilibrium and the optimum.

Distributed Caching Games

The distributed caching game can be formalized as a two-sided market game: active agents correspond to n resident subscribers or cache locations, and passive agents correspond to m requests from transit subscribers. Formally, given an instance of the DCP problem, a strategic game $\mathcal{G}(U, \{F_i | i \in U\}, \{\alpha_i | i \in U\})$ is defined as follows. The set of players (or active agents) U is the set of cache locations. The family of feasible strategies F_i of a cache location i is the family of subsets s_i of requests such that $\sum_{j \in s_i} b_j \leq B_i$ and $\sum_{t \in \{t_j | j \in s_i\}} a_t \leq A_i$. Given a vector $S = (s_1, s_2, \ldots, s_n)$ of strategies of cache locations, the *favorite* cache locations for request j, denoted by $\mathsf{FAV}(j)$, is the set of cache locations i such that $j \in s_i$ and f_{ij} has the maximum profit among the cache locations that have request j in their strategy set, i. e., $f_{ij} \geq f_{i'j}$ for any i' such that $j \in s_{i'}$. For a strategy profile $S = (s_1, \ldots, s_n)$ $\alpha_i(S) = \sum_{j : i \in \mathsf{FAV}(j)} f_{ij} / |\mathsf{FAV}(j)|$. Intuitively, the above definition implies that the profit of each request goes to the cache locations with the minimum connection cost (or equivalently with the maximum profit) among the set of cache locations that provide this request. If more than one cache location have the maximum profit (or minimum connection cost) for a request j, the profit of this request is divided equally between these cache locations. The payoff of a cache location is the sum of profits from the requests it actually serves. A player i *serves* a request j if $i \in \mathsf{FAV}(j)$. The social value of strategy profile S, denoted by $\gamma(S)$, is the sum of profits of all players. This value $\gamma(S)$ is a measure of the efficiency of the assignment of requests and request types to cache locations.

Special Cases

In this paper, the following variants and special cases of the DCP problem are also studied: The CapDCP problem is a special case of DCP problem without bandwidth constraints. The BanDCP problem is a special case of DCP problem without capacity constraints. In the uniform BanDCP problem, the bandwidth consumption of all requests is the same. In the uniform CapDC problem, the size of all request types is the same.

Many-to-One Two-Sided Markets with Ties

In the distributed caching game, active and passive agents correspond to cache locations and requests respectively. The set of feasible strategies for each active agent correspond to a set of solutions to a packing problem. Moreover, the preferences of both active and passive agents is determined from the profit of requests to cache locations. In many-to-one two-sided markets, the preference of passive and active agents as well as the feasible family of strategies are arbitrary. The preference list of agents may have ties as well.

Monotone and Matroid Markets

In *monotone many-to-one two-sided markets*, the preferences of both active and passive agents are determined based on payoffs $p_{ij} = p_{ji}$ for each active agent i and passive agent j (similar to the DCP game). An agent i prefers j to j' if $p_{ij} > p_{ij'}$. In *matroid two-sided markets*, the feasible set of strategies of each active agent is the set of independent sets of a matroid. Therefore, uniform BanDCP game is a matroid two-sided market game.

Key Results

In this section, the known results for these problems are summarized.

Centralized Approximation Algorithm

The distributed caching problem generalizes the multiple knapsack problem and the generalized assignment problem [3] and as a result is an APX-hard problem.

Theorem 1 ([4]) *There exists a linear programming based* $1 - \frac{1}{e}$*-approximation algorithm and a local search* $\frac{1}{2}$*-approximation algorithm for the* DCP *problem.*

The $1 - \frac{1}{e}$-approximation for this problem is based on rounding an exponentially large configuration linear program [4]. On the basis of some reasonable complexity theoretic assumptions, this approximation factor of $1 - \frac{1}{e}$ is tight for this problem. More formally,

Theorem 2 ([4]) *For any $\epsilon > 0$, there exists no $1 - \frac{1}{e} - \epsilon$-approximation algorithm for the* DCP *problem unless NP $\subseteq DTIME(n^{O(\log \log n)})$.*

Price of Anarchy

Since the DCP game is a strategic game, it possesses mixed Nash equilibria [12]. The DCP game is a valid-utility game with a submodular social function as defined by Vetta [16]. This implies that the performance of any mixed Nash equilibrium of this game is at least $\frac{1}{2}$ of the optimal solution.

Theorem 3 ([4,11]) *The* DCP *game is a valid-utility game and the price of anarchy for mixed Nash equilibria is $\frac{1}{2}$. Moreover, this result holds for all monotone many-to-one two-sided markets with ties.*

A direct proof of the above price of anarchy bound for the DCP game can be found in [11].

Pure Nash Equilibria: Existence and Convergence

This part surveys known results for existence and convergence of pure Nash equilibria.

Theorem 4 ([11]) *There are instances of the* IBDC *game that have no pure Nash equilibrium.*

Since, IBDC is a special case of CapDCP, the above theorem implies that there are instances of the CapDCP game that have no pure Nash equilibrium. In the above theorem, the bandwidth consumption of requests are not uniform, and this was essential in finding the example. The following gives theorems for the uniform variant of these games.

Theorem 5 ([1,11]) *Any instance of the uniform* BanDCP *game does not contain any cycle of strict best-response moves, and thus possess a pure Nash equilibrium. On the other hand, there are instances of the uniform* CapDCP *game with no pure Nash equilibria.*

The above result for the uniform BanDCP game can be generalized to matroid two-sided markets with ties as follows.

Theorem 6 ([1]) *Any instance of the monotone matroid two-sided market game with ties is a potential game, and possess pure Nash equilibria. Moreover, any instance of the matroid two-sided market game with ties possess pure Nash equilibria.*

Convergence Time to Equilibria

This section proves that there are instances of the uniform CapDCP game in which finding a pure Nash equilibrium is PLS-hard [8]. The definition of PLS-hard problems can be found in papers by Yannakakis et al. [8,15].

Theorem 7 ([11]) *There are instances of the uniform* CapDCP *game with pure Nash equilibria[2] for which finding a pure Nash equilibrium is PLS-hard.*

Using the above proof and a result of Schaffer and Yannakakis [13,15], it is possible to show that in some instances of the uniform CapDCP game, there are states from which all paths of best responses have exponential length.

Corollary 1 ([11]) *There are instances of the uniform* CapDCP *game that have pure Nash equilibria with states from which any sequence of best-response moves to any pure Nash equilibrium (or sink equilibrium) has an exponential length.*

The above theorems show exponential convergence to pure Nash equilibria in general DCP games. For the special case of the uniform BanDCP game, the following is a positive result for the convergence time to equilibria.

Theorem 8 ([2]) *The expected convergence time of a random best-response walk to pure Nash equilibria in matroid monotone two-sided markets (without ties) is polynomial.*

Since the uniform BanDCP game is a special case of matroid monotone two-sided markets with ties, the above theorem indicates that for the BanDCP game with no tie in the profit of requests, the convergence time of a random best-response walk is polynomial. Finally, we state a theorem about the convergence time of the general (non-monotone) matroid two-sided market games.

Theorem 9 ([2]) *In the matroid two-sided markets (without ties), a random best response dynamic of players may cycle, but it converges to a Nash equilibrium with probability one. However, it may take exponential time to converge to a pure Nash equilibrium.*

Pure Nash equilibria of two-sided market games correspond to stable matchings in two-sided markets and vice-versa [2]. The fact that better response dynamics of players in two-sided market games may cycle, but will converge to a stable matching has been proved in [9,14]. Ackermann et al. [2] extend these results for best-response dynamics, and show an exponential lower bound for expected convergence time to pure Nash equilibria.

[2]It is also possible to say that finding a *sink equilibrium* is PLS-hard. A sink equilibrium is a set of strategy profiles that is closed under best-response moves. A pure equilibrium is a sink equilibrium with exactly one profile. This equilibrium concept is formally defined in [7].

Applications

The growth of the Internet, the World Wide Web, and wide-area wireless networks allow an increasing number of users to access vast amounts of information in different geographic areas. As one of the most important functions of the service provider, content delivery can be performed by caching popular items in cache locations close to the users. Performing such a task in a decentralized manner in the presence of self-interested entities in the system can be modeled as an uncoordinated two-sided market game.

The 3G subscriber market can be categorized into groups with shared interest in location-based services, e. g., the preview of movies in a theater or scenes of the mountain nearby. Since the 3G radio resources are limited, it is expensive to repeatedly transmit large quantities of data over the air interface from the base station (BS). It is more economical for the service provider to offload such repeated requests on to the ad-hoc network comprised of its subscribers where some of them recently acquired a copy of the data. In this scenario, the goal for the service provider is to give incentives for peer subscribers in the system to cache and forward the data to the requesting subscribers. Since each data item is large in size and transit subscribers are mobile, we assume that the data transfer occurs in a close range of a few hops.

In this setting, envision a system consisting of two groups of subscribers: resident and transit subscribers. Resident subscribers are less mobile and mostly confined to a certain geographical area. Resident subscribers have incentives to cache data items that are specific to this geographical region since the service provider gives monetary rewards for satisfying the queries of transit subscribers. Transit subscribers request their favorite data items when they visit a particular region. Since the service provider does not have knowledge of the spatial and temporal distribution of requests, it is difficult if not impossible for the provider to stipulate which subscriber should cache which set of data items. Therefore, the decision of what to cache is left to each individual subscriber. The realization of this content distribution system depends on two main issues. First, since subscribers are selfish agents, they may act to increase their individual payoff and decrease the performance of the system. Here, we provide a framework for which we can prove that in an equilibrium situation of this framework, we use the performance of the system efficiently. The second issue is that the payoff of each request for each agent must be a function of the set of agents that have this request in their strategy, since these agents compete on this request and the profit of this request should be divided among these agents in an appropriate way. Therefore, each selfish agent may change the set of items it cached in response to the set of items cached by others. This model leads to a non-cooperative caching scenario that can be modeled on a two-sided market game, studied and motivated in the context of market sharing games and distributed caching games [4,6,11].

Open Problems

It is known that there exist instances of the distributed caching game with no pure Nash equilibria. It is also known that best response dynamics of players may take exponential time to converge to pure Nash equilibria. An interesting question is to study the performance of sink equilibria [7,11] or the price of sinking [7,11] for these games. The distributed caching game is a valid-utility game. Goemans, Mirrokni, and Vetta [7] show that despite the price of anarchy of $\frac{1}{2}$ for valid-utility games, the performance of sink equilibria (or price of sinking) for these games is $\frac{1}{n}$. We conjecture that the price of sinking for DCP games is a constant. Moreover, it is interesting to show that after polynomial rounds of best responses of players the approximation factor of the solution is a constant. We know that one round of best responses of players is not sufficient to get constant-factor solutions. It might be easier to show that after a polynomial number of random best responses of players, the expected total profit of players is at least a constant factor of the optimal solution. Similar positive results for sink equilibria and random best responses of players are known for congestion games [7,11].

The complexity of verifying if a given state of the distributed caching game is in a sink equilibrium or not is an interesting question to explore. Also, given a distributed caching game (or a many-to-one two-sided market game), an interesting problem is to check if the set of all sink equilibria is pure Nash equilibria or not. Finally, an interesting direction of research is to classify classes of two-sided market games for which pure Nash equilibria exists or best-response dynamics of players converge to a pure Nash equilibrium.

Cross References

- ▶ Stable Marriage
- ▶ Stable Marriage with Ties and Incomplete Lists
- ▶ Best Response Algorithms for Selfish Routing

Recommended Reading

1. Ackermann, H., Goldberg, P., Mirrokni, V., Röglin, H., Vöcking, B.: A unified Approach to Congestion Games and Two-sided markets. In: 3rd Workshop of Internet Economics (WINE), pp. 30–41. San Diego, CA, USA (2007)

2. Ackermann, H., Goldberg, P., Mirrokni, V., Röglin, H., Vöcking, B.: Uncoordinated two-sided markets. ACM Electronic Commerce (ACM EC) (2008)
3. Chekuri, C., Khanna, S.: A PTAS for the multiple knapsack problem. In 11th ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 213–222 (2000)
4. Fleischer, L., Goemans, M., Mirrokni, V.S., Sviridenko, M.: Tight approximation algorithms for maximum general assignment problems. In: Proceedings of the 16th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA), pp. 611–620 (2006)
5. Gale, D., Shapley, L.: College admissions and the stability of marriage. Am. Math. Mon. **69**, 9–15 (1962)
6. Goemans, M., Li, L., Mirrokni, V.S., Thottan, M.: Market sharing games applied to content distribution in ad-hoc networks. In: Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 1020–1033 (2004)
7. Goemans, M., Mirrokni, V.S., Vetta, A.: Sink equilibria and convergence. In: 46th Conference on Foundations of Computer Science (FOCS), pp. 123–131 (2005)
8. Johnson, D., Papadimitriou, C.H., Yannakakis, M.: How easy is local search? J. Comp. Syst. Sci. **37**, 79–100 (1988)
9. Knuth, D.: Marriage Stables et leurs relations avec d'autres problèmes Combinatories. Les Presses de l'Université de Montréal (1976)
10. Kojima, F., Unver, Ü.: Random paths to pairwise stability in many-to-many matching problems: A study on market equilibration. Intern. J. Game Theor. (2006)
11. Mirrokni, V.S.: Approximation Algorithms for Distributed and Selfish Agents. Ph.D. thesis, Massachusetts Institute of Technology (2005)
12. Nash, J.F.: Non-cooperative games. Ann. Math. **54**, 268–295 (1951)
13. Papadimitriou, C.H., Schaffer, A., Yannakakis, M.: On the complexity of local search. In: 22nd Symp. on Theory of Computing (STOC), pp. 438 – 445 (1990)
14. Roth, A.E., Vande Vate, J.H.: Random paths to stability in two-sided matching. Econometrica **58**(6), 1475–1480 (1990)
15. Schaffer, A. Yannakakis, M.: Simple local search problems that are hard to solve. SIAM J. Comput. **20**(1), 56–87 (1991)
16. Vetta, A.: Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In: 43rd Symp. on Foundations of Computer Science (FOCS), pp. 416–425 (2002)

Max Cut

1994; Goemans, Williamson
1995; Goemans, Williamson

Alantha Newman
Department of Algorithms and Complexity,
Max-Planck Institute for Computer Science,
Saarbrücken, Germany

Keywords and Synonyms

Maximum bipartite subgraph

Problem Definition

Given an undirected edge-weighted graph, $G = (V, E)$, the maximum cut problem (Max-Cut) is to find a bipartition of the vertices that maximizes the weight of the edges crossing the partition. If the edge weights are non-negative, then this problem is equivalent to finding a maximum weight subset of the edges that forms a bipartite subgraph, i. e. the maximum bipartite subgraph problem. All results discussed in this article assume non-negative edge weights. Max-Cut is one of Karp's original NP-complete problems [19]. In fact, it is NP-hard to approximate to within a factor better than 16/17[16,33].

For nearly twenty years, the best-known approximation factor for Max-Cut was half, which can be achieved by a very simple algorithm: Form a set S by placing each vertex in S with probability half. Since each edge crosses the cut $(S, V \setminus S)$ with probability half, the expected value of this cut is half the total edge weight. This implies that for any graph, there exists a cut with value at least half of the total edge weight. In 1976, Sahni and Gonzalez presented a deterministic half-approximation algorithm for Max-Cut, which is essentially a de-randomization of the aforementioned randomized algorithm [31]: Iterate through the vertices and form sets S and $\bar{S}$ by placing each vertex in the set that maximizes the weight of cut $(S, \bar{S})$ thus far. After each iteration of this process, the weight of this cut will be at least half of the weight of the edges with both endpoints in $S \cup \bar{S}$.

This simple half-approximation algorithm uses the fact that for any graph with non-negative edge weights, the total edge weight of a given graph is an upper bound on the value of its maximum cut. There exist classes of graphs for which a maximum cut is arbitrarily close to half the total edge weight, i. e. graphs for which this "trivial" upper bound can be close to twice the true value of an optimal solution. An example of such a class of graphs are complete graphs on n vertices, K_n. In order to obtain an approximation factor better than half, one must be able to compute an upper bound on the value of a maximum cut that is better, i. e. smaller, than the trivial upper bound for such classes of graphs.

Linear Programming Relaxations

For many optimization (maximization) problems, linear programming has been shown to yield better (upper) bounds on the value of an optimal solution than can be obtained via combinatorial methods. There are several well-studied linear programming relaxations for Max-Cut. For example, a classical integer program has a variable x_e for each edge and a constraint for each odd cycle, requir-

ing that an odd cycle C contribute at most $|C| - 1$ edges to an optimal solution.

$$\max \sum_{e \in E} w_e x_e$$
$$\sum_{e \in C} x_e \leq |C| - 1 \quad \forall \text{ odd cycles } C$$
$$x_e \in \{0, 1\} .$$

The last constraint can be relaxed so that each x_e is required to lie between 0 and 1, but need not be integral, i. e. $0 \leq x_e \leq 1$. Although this relaxation may have exponentially many constraints, there is a polynomial-time separation oracle (equivalent to finding a minimum weight odd-cycle), and thus, the relaxation can be solved in polynomial time [13]. Another classical integer program contains a variable x_{ij} for each pair of vertices. In any partition of the vertices, either zero or two edges from a 3-cycle cross the cut. This requirement is enforced in the following integer program. If edge $(i, j) \notin E$, then w_{ij} is set to 0.

$$\max \sum_{i,j \in V} w_{ij} x_{ij}$$
$$x_{ij} + x_{jk} + x_{ki} \leq 2 \quad \forall i, j, k \in V$$
$$x_{ij} + x_{jk} - x_{ki} \geq 0 \quad \forall i, j, k \in V$$
$$x_{ij} \in \{0, 1\} .$$

Again, the last constraint can be relaxed so that each x_{ij} is required to lie between 0 and 1. In contrast to the aforementioned cycle-constraint based linear program, this linear programming relaxation has a polynomial number of constraints.

Both of these relaxations actually have the same optimal value for any graph with non-negative edge weights [3,26,30]. (For a simplified proof of this, see [25].) Poljak showed that the integrality gap for each of these relaxations is arbitrarily close to 2 [26]. In other words, there are classes of graphs that have a maximum cut containing close to half of the edges, but for which each of the above relaxations yields an upper bound close to all the edges, i. e. no better than the trivial "all-edges" bound. In particular, graphs with a maximum cut close to half the edges and with high girth can be used to demonstrate this gap. A comprehensive look at these linear programming relaxations is contained in the survey of Poljak and Tuza [30].

Eigenvalue Upper Bounds

Delorme and Poljak [7] presented an eigenvalue upper bound on the value of a maximum cut, which was a strengthened version of a previous eigenvalue bound considered by Mohar and Poljak [24]. Computing Delorme and Poljak's upper bound is equivalent to solving an eigenvalue minimization problem. They showed that their bound is computable in polynomial time with arbitrary precision. In a series of work, Delorme, Poljak and Rendl showed that this upper bound behaves "differently" from the linear-programming-based upper bounds. For example, they studied classes of sparse random graphs (e. g. $G(n, p)$ with $p = 50/n$) and showed that their upper bound is close to optimal on these graphs [8]. Since graphs of this type can also be used to demonstrate an integrality gap arbitrarily close to 2 for the aforementioned linear programming relaxations, their work highlighted contrasting behavior between these two upper bounds. Further computational experiments on other classes of graphs gave more evidence that the bound was indeed stronger than previously studied bounds [27,29]. Delorme and Poljak conjectured that the 5-cycle demonstrated the worst-case behavior for their bound: a ratio of $32/(25 + 5\sqrt{5}) \approx .88445$ between their bound and the optimal integral solution. However, they could not prove that their bound was strictly less than twice the value of a maximum cut in the worst case.

Key Results

In 1994, Goemans and Williamson presented a randomized .87856-approximation algorithm for Max-Cut [11]. Their breakthrough work was based on rounding a semidefinite programming relaxation and was the first use of semidefinite programming in approximation algorithms. Poljak and Rendl showed that the upper bound provided by this semidefinite relaxation is equivalent to the eigenvalue bound of Delorme and Poljak [28]. Thus, Goemans and Williamson's proved that the eigenvalue bound of Delorme and Poljak is no more than 1.138 times the value of a maximum cut.

A Semidefinite Relaxation

Max-Cut can be formulated as the following quadratic integer program, which is NP-hard to solve. Each vertex $i \in V$ is represented by a variable y_i, which is assigned either 1 or -1 depending on which side of the cut it occupies.

$$\max \frac{1}{2} \sum_{(i,j) \in E} w_{ij}(1 - y_i y_j)$$
$$y_i \in \{-1, 1\} \quad \forall i \in V .$$

Goemans and Williamson considered the following relaxation of this integer program, in which each vertex is rep-

resented by a unit vector.

$$\max \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - v_i \cdot v_j)$$
$$v_i \cdot v_i = 1 \quad \forall i \in V$$
$$v_i \in \mathcal{R}^n \quad \forall i \in V\,.$$

They showed that this relaxation is equivalent to a semidefinite program. Specifically, consider the following semidefinite relaxation:

$$\max \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - y_{ij})$$
$$y_{ii} = 1 \quad \forall i \in V$$
$$Y \quad \text{positive semidefinite}\,.$$

The equivalence of these two relaxations is due to the fact that a matrix Y is positive semidefinite if and only if there is a matrix B such that $B^{\mathrm{T}}B = Y$. The latter relaxation can be solved to within arbitrary precision in polynomial time via the Ellipsoid Algorithm, since it has a polynomial-time separation oracle [14]. Thus, a solution to the first relaxation can be obtained by finding a solution to the second relaxation and finding a matrix B such that $B^{\mathrm{T}}B = Y$. If the columns of B correspond to the vectors $\{v_i\}$, then $y_{ij} = v_i \cdot v_j$, yielding a solution to the first relaxation.

Random-Hyperplane Rounding

Goemans and Williamson showed how to round the semidefinite programming relaxation of MAX-CUT using a new technique that has since become known as "random-hyperplane rounding" [11]. First obtain a solution to the first relaxation, which consists of a set of unit vectors $\{v_i\}$, one vector for each vertex. Then choose a random vector $r \in \mathcal{R}^n$ in which each coordinate of r is chosen from the standard normal distribution. Finally, set $S = \{i | v_i \cdot r \geq 0\}$ and output the cut $(S, V \setminus S)$.

The probability that a particular edge $(i, j) \in E$ crosses the cut is equal to the probability that the dot products $v_i \cdot r$ and $v_j \cdot r$ differ in sign. This probability is exactly equal to θ_{ij}/π, where θ_{ij} is the angle between vectors v_i and v_j. Thus, the expected weight of edges crossing the cut is equal to $\sum_{(i,j)\in E} \theta_{ij}/\pi$. How large is this compared to the objective value given by the semidefinite programming relaxation, i. e. what is the approximation ratio?

Define α_{gw} as the worst-case ratio of the expected contribution of an edge to the cut, to its contribution to the objective function of the semidefinite programming relaxation. In other words: $\alpha_{gw} = \min_{0\leq\theta\leq\pi} \frac{2}{\pi} \frac{\theta}{1-\cos\theta}$. It can be shown that $\alpha_{gw} > .87856$. Thus, the expected value of a cut is at least $\alpha_{gw} \cdot SDP_{OPT}$, resulting in an approximation ratio of at least .87856 for MAX-CUT. The same analysis applies to weighted graphs with non-negative edge weights.

This algorithm was de-randomized by Mahajan and Hariharan [23]. Goemans and Williamson also applied their random-hyperplane rounding techniques to give improved approximation guarantees for other problems such as MAX-DICUT and MAX-2SAT.

Integrality Gap and Hardness

Karloff showed that there exist graphs for which the best hyperplane is only a factor α_{gw} of the maximum cut [18], showing that there are graphs for which the analysis in [11] is tight. Since the optimal SDP value for such graphs equals the optimal value of a maximum cut, these graphs can not be used to demonstrate an integrality gap. However, Feige and Schechtman showed that there exist graphs for which the maximum cut is a α_{gw} fraction of the SDP bound [9], thereby establishing that the approximation guarantee of Goemans and Williamson's algorithm matches the integrality gap of their semidefinite programming relaxation. Recently, Khot, Kindler, Mossel and O'Donnell [21] showed that if the Unique Games Conjecture of Khot [20] is assumed to be true, then it is NP-hard to approximate MAX-CUT to within any factor larger than α_{gw}.

Applications

The work of Goemans and Williamson paved the way for the further use of semidefinite programming in approximation algorithms, particularly for graph partitioning problems. Methods based on the random-hyperplane technique have been successfully applied to many optimization problems that can be categorized as partition problems. A few examples are 3-COLORING [17], MAX-3-CUT [10,12,22], MAX-BISECTION [15], CORRELATION-CLUSTERING [5,32], and SPARSEST-CUT [2]. Additionally, some progress has been made in extending semidefinite programming techniques outside the domain of graph partitioning to problems such as BETWEENNESS [6], BANDWIDTH [4], and LINEAR EQUATIONS mod p [1].

Cross References

- ▶ Graph Coloring
- ▶ Maximum Two-Satisfiability
- ▶ Sparsest Cut

Recommended Reading

1. Andersson, G., Engebretsen, L., Håstad, J.: A new way to use semidefinite programming with applications to linear equations mod *p*. J. Algorithms **39**, 162–204 (2001)
2. Arora, S., Rao, S., Vazirani, U.: Expander flows, geometric embeddings and graph partitioning. In: Proceedings of the 36th Annual Symposium on the Theory of Computing (STOC), Chicago 2004, pp. 222–231
3. Barahona, F.: On cuts and matchings in planar graphs. Math. Program. **60**, 53–68 (1993)
4. Blum, A., Konjevod, G., Ravi, R., Vempala, S.: Semi-definite relaxations for minimum bandwidth and other vertex-ordering problems. Theor. Comput. Sci. **235**, 25–42 (2000)
5. Charikar, M., Guruswami, V., Wirth, A.: Clustering with qualitative information. In: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Boston 2003, pp. 524–533
6. Chor, B., Sudan, M.: A geometric approach to betweeness. SIAM J. Discret. Math. **11**, 511–523 (1998)
7. Delorme, C., Poljak, S.: Laplacian eigenvalues and the maximum cut problem. Math. Program. **62**, 557–574 (1993)
8. Delorme, C., Poljak, S.: The performance of an eigenvalue bound in some classes of graphs. Discret. Math. **111**, 145–156 (1993). Also appeared in: Proceedings of the Conference on Combinatorics, Marseille, 1990
9. Feige, U., Schechtman, G.: On the optimality of the random hyperplane rounding technique for MAX-CUT. Random Struct. Algorithms **20**(3), 403–440 (2002)
10. Frieze, A., Jerrum, M.R.: Improved approximation algorithms for MAX-*k*-CUT and MAX BISECTION. Algorithmica **18**, 61–77 (1997)
11. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**, 1115–1145 (1995)
12. Goemans, M.X., Williamson, D.P.: Approximation algorithms for MAX-3-CUT and other problems via complex semidefinite programming. STOC 2001 Special Issue of J. Comput. Syst. Sci. **68**, 442–470 (2004)
13. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica **1**, 169–197 (1981)
14. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Berlin (1988)
15. Halperin, E., Zwick, U.: A unified framework for obtaining improved approximation algorithms for maximum graph bisection problems. Random Struct. Algorithms **20**(3), 382–402 (2002)
16. Håstad, J.: Some optimal inapproximability results. J. ACM **48**, 798–869 (2001)
17. Karger, D.R., Motwani, R., Sudan, M.: Improved graph coloring via semidefinite programming. J. ACM **45**(2), 246–265 (1998)
18. Karloff, H.J.: How good is the Goemans-Williamson MAX CUT algorithm? SIAM J. Comput. **29**(1), 336–350 (1999)
19. Karp, R.M.: Reducibility Among Combinatorial Problems. In: Complexity of Computer Computations, pp. 85–104. Plenum Press, New York (1972)
20. Khot, S.: On the power of unique 2-prover 1-round games. In: Proceedings of the 34th Annual Symposium on the Theory of Computing (STOC), Montreal 2002, pp. 767–775
21. Khot, S., Kindler, G., Mossel, E., O'Donnell, R.: Optimal inapproximability results for MAX CUT and other 2-variable CSPs? In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Rome 2004, pp. 146–154
22. de Klerk, E., Pasechnik, D., Warners, J.: On approximate graph colouring and MAX-*k*-CUT algorithms based on the θ function. J. Combin. Optim. **8**(3), 267–294 (2004)
23. Mahajan, R., Hariharan, R.: Derandomizing semidefinite programming based approximation algorithms. In: Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Milwaukee 1995, pp. 162–169
24. Mohar, B., Poljak, S.: Eigenvalues and the max-cut problem. Czechoslov Math. J. **40**(115), 343–352 (1990)
25. Newman, A.: A note on polyhedral relaxations for the maximum cut problem (2004). Unpublished manuscript
26. Poljak, S.: Polyhedral and eigenvalue approximations of the max-cut problem. Sets, Graphs and Numbers. Colloqiua Mathematica Societatis Janos Bolyai **60**, 569–581 (1992)
27. Poljak, S., Rendl, F.: Node and edge relaxations of the max-cut problem. Comput. **52**, 123–137 (1994)
28. Poljak, S., Rendl, F.: Nonpolyhedral relaxations of graph-bisection problems. SIAM J. Opt. **5**, 467–487 (1995)
29. Poljak, S., Rendl, F.: Solving the max-cut using eigenvalues. Discret. Appl. Math. **62**(1–3), 249–278 (1995)
30. Poljak, S., Tuza, Z.: Maximum cuts and large bipartite subgraphs. DIMACS Ser. Discret. Math. Theor. Comput. Sci. **20**, 181–244 (1995)
31. Sahni, S., Gonzalez, T.: P-complete approximation problems. J. ACM **23**(3), 555–565 (1976)
32. Swamy, C.: Correlation clustering: maximizing agreements via semidefinite programming. In: Proceedings of 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), New Orleans 2004, pp. 526–527
33. Trevisan, L., Sorkin, G., Sudan, M., Williamson, D.: Gadgets, approximation, and linear programming. SIAM J. Comput. **29**(6), 2074–2097 (2000)

Maximum Agreement Subtree (of 2 Binary Trees)

1996; Cole, Hariharan

Ramesh Hariharan
Strand Life Sciences, Bangalore, India

Keywords and Synonyms

Isomorphism; Tree agreement

Problem Definition

Consider two rooted trees T_1 and T_2 with n leaves each. The internal nodes of each tree have at least two children each. The leaves in each tree are labeled with the same set of labels and further, no label occurs more than once

in a particular tree. An *agreement subtree* of T_1 and T_2 is defined as follows. Let L_1 be a subset of the leaves of T_1 and let L_2 be the subset of those leaves of T_2 which have the same labels as leaves in L_1. The subtree of T_1 *induced* by L_1 is an agreement subtree of T_1 and T_2 if and only if it is *isomorphic* to the subtree of T_2 induced by L_2. The Maximum Agreement Subtree problem (henceforth called *MAST*) asks for the largest agreement subtree of T_1 and T_2.

The terms *induced subtree* and *isomorphism* used above need to be defined. Intuitively, the subtree of T induced by a subset L of the leaves of T is the topological subtree of T restricted to the leaves in L, with branching information relevant to L preserved. More formally, for any two leaves $a\,b$ of a tree T, let $\mathrm{lca}_T(a, b)$ denote their lowest common ancestor in T. If $a = b$, $\mathrm{lca}_T(a, b) = a$. The *subtree* U of T *induced* by a subset L of the leaves is the tree with leaf set L and interior node set $\{\mathrm{lca}_T(a, b) | a, b \in L\}$ inheriting the ancestor relation from T, that is, for all $a, b \in L$, $\mathrm{lca}_U(a, b) = \mathrm{lca}_T(a, b)$.

Intuitively, two trees are isomorphic if the children of each node in one of the trees can be reordered so that the leaf labels in each tree occur in the same order and the shapes of the two trees become identical. Formally, two trees U_1 and U_2 with the same leaf labels are said to be *isomorphic* if there is a 1–1 mapping μ between their nodes mapping leaves to leaves with the same labels and such that for any two different leaves $a\,b$ of U_1, $\mu(\mathrm{lca}_{U_1}(a, b)) = \mathrm{lca}_{U_2}(\mu(a), \mu(b))$.

Key Results

Previous Work

Finden and Gordon [8] gave a heuristic algorithm for the *MAST* problem on binary trees which had an $O(n^5)$ running time and did not guarantee an optimal solution. Kubicka, Kubicki and McMorris [13] gave an $O(n^{(.5+\epsilon)\log n})$ algorithm for the same problem. The first polynomial time algorithm for this problem was given by Steel and Warnow [15]; it had a running time of $O(n^2)$. Steel and Warnow also considered the case of non-binary and unrooted trees. Their algorithm takes $O(n^2)$ time for fixed degree rooted and unrooted trees and $O(n^{4.5}\log n)$ for arbitrary degree rooted and unrooted trees. They also give a linear reduction from the rooted to the unrooted case. Farach and Thorup gave an $O(nc^{\sqrt{\log n}})$ time algorithm for the *MAST* problem on binary trees; here c is a constant greater than 1. For arbitrary degree trees, their algorithm takes $O(n^2 c^{\sqrt{\log n}})$ time for the unrooted case [6] and $O(n^{1.5}\log n)$ time for the rooted case [7]. Farach, Przytycka, and Thorup [4] obtained an $O(n\log^3 n)$ algorithm for the *MAST* problem on binary trees. Kao [12] obtained an algorithm for the same problem which takes $O(n\log^2 n)$ time. This algorithm takes $O(\min\{nd^2\log d\log^2 n, nd^{\frac{3}{2}}\log^3 n\})$ for degree d trees.

The *MAST* problem for more than two trees has also been studied. Amir and Keselman [1] showed that the problem is *NP*-hard for even 3 unbounded degree trees. However, polynomial bounds are known [1,5] for three or more bounded degree trees.

Our Contribution

An $O(n\log n)$ algorithm for the *MAST* problem for two binary trees is presented here. This algorithms is obtained by improving upon the $O(n\log^3 n)$ algorithm from [4] (in fact, the final journal version [3] combines both papers). The $O(n\log^3 n)$ algorithm of [4] can be viewed as taking the following approach (although the authors do not describe it this way). It identifies two special cases and then solves the general case by interpolating between these cases.

Special Case 1: The internal nodes in both trees form a path. The *MAST* problem reduces to essentially a size n Longest Increasing Subsequence Problem in this case. As is well known, this can be solved in $O(n\log n)$ time.

Special Case 2: Both trees T_1 and T_2 are complete binary trees. For each node v in T_2, only certain nodes u in T_1 can be usefully mapped to v, in the sense that the subtree of T_1 rooted at u and the subtree of T_2 rooted at v have a non-empty Agreement Subtree. There are $O(n\log^2 n)$ such pairs (u, v). This can be seen as follows. Note that for (u, v) to be such a pair, the subtree of T_1 rooted at u and the subtree of T_2 rooted at v must have a leaf-label in common. For each label, there are only $O(\log^2 n)$ such pairs obtained by pairing each ancestor of the leaf with this label in T_1 with each ancestor of the leaf with this label in T_2. The total number of interesting pairs is thus $O(n\log^2 n)$. For each pair, computing the *MAST* takes $O(1)$ time, as it is simply a question of deciding the best way of pairing their children.

The interpolation process takes a centroid decomposition of the two trees and compares pairs of centroid paths, rather than individual nodes as in the complete tree case. The comparison of a pair of centroid paths requires finding matchings with special properties in appropriately defined bipartite graphs, a non-trivial generalization of the Longest Increasing Subsequence problem. This process

creates $O(n \log^2 n)$ interesting (u, v) pairs, each of which takes $O(\log n)$ time to process.

This work provides two improvements, each of which gains a $\log n$ factor.

Improvement 1: The complete tree special case is improved to $O(n \log n)$ time as follows. A pair of nodes (u, v), $u \in T_1$, $v \in T_2$, is said to be *interesting* if there is an agreement subtree mapping u to v. As is shown below, for complete trees, the total number of interesting pairs (u, v) is just $O(n \log n)$. Consider a node v in T_2. Let L_2 be the set of leaves which are descendants of v. Let L_1 be the set of leaves in T_1 which have the same labels as the leaves in L_2. The only nodes that may be mapped to v are the nodes u in the subtree of T_1 induced by L_1. The number of such nodes u is O(size of the subtree of T_2 rooted at v). The total number of interesting pairs is thus the sum of the sizes of all subtrees of T_2, which is $O(n \log n)$.

This reduces the number of interesting pairs (u, v) to $O(n \log n)$. Again, processing a pair takes $O(1)$ time (this is less obvious, for identifying the descendants of u which root the subtrees with which the two subtrees of v can be matched is non-trivial). Constructing the above induced subtree itself can be done in $O(|L_1|)$ time, as will be detailed later. The basic tool here is to preprocess trees T_1 and T_2 in $O(n)$ time so that least common ancestor queries can be answered in $O(1)$ time.

Improvement 2: As in [4], when the trees are not complete binary trees, the algorithm takes centroid paths and matches pairs of centroid paths. The $O(\log n)$ cost that the algorithm in [4] incurs in processing each such interesting pair of paths arises when there are large (polynomial in n size) instances of the generalized Longest Increasing Subsequence Problem. At first sight, it is not clear that large instances of these problems can be created for sufficiently many of the interesting pairs; unfortunately, this turns out to be the case. However, these problem instances still have some useful structure. By using (static) weighted trees, pairs of interesting vertices are processed in $O(1)$ time per pair, on the average, as is shown by an appropriately parametrized analysis.

The Multiple Degree Case

The techniques can be generalized to higher degree bounds $d > 2$, by combining it with techniques from ([6, Sect. 2]) for unbounded degrees. This appears to yield an algorithm with running time $O(\min\{n\sqrt{d}\log^2 n, nd \log n \log d\})$. The conjecture, however, is that there is an algorithm with running time $O(n\sqrt{d}\log n)$.

Applications

Motivation

The *MAST* problem arises naturally in biology and linguistics as a measure of consistency between two evolutionary trees over species and languages, respectively. An evolutionary tree for a set of *taxa*, either species or languages, is a rooted tree whose leaves represent the taxa and whose internal nodes represent ancestor information. It is often difficult to determine the true phylogeny for a set of taxa, and one way to gain confidence in a particular tree is to have different lines of evidence supporting that tree. In the biological taxa case, one may construct trees from different parts of the DNA of the species. These are known as *gene trees*. For many reasons, these trees need not entirely agree, and so one is left with the task of finding a consensus of the various gene trees. The maximum agreement subtree is one method of arriving at such a consensus. Notice that a gene is usually a binary tree, since DNA replicates by a binary branching process. Therefore, the case of binary trees is of great interest.

Another application arises in automated translation between two languages [10]. The two trees are the parse trees for the same meaning sentences in the two languages. A complication that arises in this application (due in part to imperfect dictionaries) is that words need not be uniquely matched, i. e., a word at the leaf of one tree could match a number (usually small) of words at the leaves of the other tree. The aim is to find a maximum agreement subtree; this is done with the goal of improving context-using dictionaries for automated translation. So long as each word in one tree has only a constant number of matches in the other tree (possibly with differing weights), the algorithm given here can be used and its performance remains $O(n \log n)$. More generally, if there are m word matches in all, the performance becomes $O((m + n) \log n)$. Note however, that if there are two collections of equal meaning words in the two trees of sizes k_1 and k_2 respectively, they induce $k_1\, k_2$ matches.

Cross References

- ► Maximum Agreement Subtree (of 3 or More Trees)
- ► Maximum Agreement Supertree

Recommended Reading

1. Amir, A., Keselman, D.: Maximum agreement subtree in a set of evolutionary trees. SIAM J. Comput. **26**(6), 1656–1669 (1997)
2. Cole, R., Hariharan, R.: An $O(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees. Proc. of the 7th ACM-SIAM SODA, pp. 323–332 (1996)

3. Cole, R., Farach-Colton, M., Hariharan, R., Przytycka, T., Thorup, M.: An $O(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees. SIAM J. Comput. **30**(5), 1385–1404 (2000)
4. Farach, M., Przytycka, T., Thorup, M.: The maximum agreement subtree problem for binary trees. Proc. of 2nd ESA (1995)
5. Farach, M., Przytycka, T., Thorup, M.: Agreement of many bounded degree evolutionary trees. Inf. Process. Lett. **55**(6), 297–301 (1995)
6. Farach, M., Thorup, M.: Fast comparison of evolutionary trees. Inf. Comput. **123**(1), 29–37 (1995)
7. Farach, M., Thorup, M.: Sparse dynamic programming for evolutionary-tree comparison. SIAM J. Comput. **26**(1), 210–230 (1997)
8. Finden, C.R., Gordon, A.D.: Obtaining common pruned trees. J. Classific. **2**, 255–276 (1985)
9. Fredman, M.L.: Two applications of a probabilistic search technique: sorting $X + Y$ and building balanced search trees. Proc. of the 7th ACM STOC, pp. 240–244 (1975)
10. Grishman, R., Yangarber, R.: Private Communication. NYU (1995)
11. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. SIAM J. Comput. **13**(2), 338–355 (1984)
12. Kao, M.-Y.: Tree contractions and evolutionary trees. SIAM J. Comput. **27**(6), 1592–1616 (1998)
13. Kubicka, E., Kubicki, G., McMorris, F.R.: An algorithm to find agreement subtrees. J. Classific. **12**, 91–100 (1995)
14. Mehlhorn, K.: A best possible bound for the weighted path length of binary search trees. SIAM J. Comput. **6**(2), 235–239 (1977)
15. Steel, M., Warnow, T.: Kaikoura tree theorems: computing the maximum agreement subtree. Inf. Process. Lett. **48**, 77–82 (1993)

Maximum Agreement Subtree (of 3 or More Trees)

1995; Farach, Przytycka, Thorup

TERESA M. PRZYTYCKA
Computational Biology Branch, NCBI, NIH, Bethesda, MD, USA

Keywords and Synonyms

Tree alignment

Problem Definition

The Maximum Agreement Subtree problem for k trees (k-MAST) is a generalization of a similar problem for two trees (MAST). Consider a tuple of k rooted leaf-labeled trees $(T_1, T_2 \ldots T_k)$. Let $A = \{a_1, a_2, \ldots a_n\}$ be the set of leaf labels. Any subset $B \subseteq A$ uniquely determines the so called *topological restriction* $T|B$ of the three T to B. Namely, $T|B$ is the topological subtree of T spanned by all leaves labeled with elements from B and lowest common ancestors of all pairs of these leaves. In particular, the ancestor relation in $T|B$ is defined so that it agrees with the ancestor relation in T. A subset B of A such $T^1|B, \ldots, T^k|B$ are isomorphic is called an *agreement set*.

Problem 1 (k-MAST)
INPUT: *A tuple $\vec{T} = (T^1, \ldots, T^k)$ of leaf-labeled trees, with a common set of labels $A = \{a_1, \ldots, a_n\}$, such that for each tree T^i there exists one-to-one mapping between the set of leaves of that tree and the set of labels A.*
OUTPUT: *k-MAST($\vec{T}$) equal to the maximum cardinality agreement set of $\vec{T}$.*

Key Results

In the general setting, k-MAST problem is NP-complete for $k \geq 3$ [1]. Under the assumption that the degree of at least one of the trees is bounded, Farach et al. proposed an algorithm leading to the following theorem:

Theorem 1 *If the degree of the trees in the tuple $\vec{T} = (T^1, \ldots, T^k)$ is bounded by d then the k-MAST($\vec{T}$) can be computed in $O(kn^3 + n^d)$ time.*

In what follows, the problem is restricted to finding the cardinality of the maximum agreement set rather than the set itself. The extension of this algorithm to an algorithm that finds the agreement set (and subsequently the agreement subtree) within the same time bounds is relatively straightforward.

Recall that the classical $O(n^2)$ dynamic programming algorithm for MAST of two binary trees [11] processes all pairs of internal nodes of the two trees in a bottom up fashion. For each pair of such nodes it computes the MAST value for the subtrees rooted at this pair. There are $O(n^2)$ pairs of nodes and the MAST value for the subtrees rooted at a given pair of nodes can be computed in constant time from MAST values of previously processed pairs of nodes.

To set the stage for the more general case, let k-MAST($\vec{v}$) be the solution to the k-MAST problem for the subtrees of $T^1(v_1), \ldots, T^k(v_k)$ where $T^i(v_i)$ is the subtree if T^i rooted at v_i. If, for all i, u_i is a strict ancestor of v_i in T^i then, $\vec{v}$ is *dominated* by $\vec{u}$ (denoted $\vec{v} \prec \vec{u}$).

A naive extension of the algorithm for two trees to an algorithm for k trees would require computing k-MAST($\vec{v}$) for all possible tuples $\vec{v}$ by processing these tuples in the order consistent with the domination relation. The basic idea that allows to avoid $\Omega(n^k)$ complexity is to replace the computation of k-MAST($\vec{v}$) with the computation of a related value, mast($\vec{v}$), defined to be the size of the maximum agreement set for the subtrees

of $(T^1, \ldots, T^k)$ rooted at $(v_1, \ldots v_k)$ subject to the additional restriction that the agreement subtrees themselves are rooted at $v_1, \ldots v_k$ respectively. Clearly

$$\text{k-MAST}(T^1, \ldots, T^k) = \max_{\vec{v}} \text{mast}(\vec{v}) \ .$$

The benefit of computing mast rather than k-MAST follows from the fact that most of mast values are zero and it is possible to identify (very efficiently) $\vec{v}$ with non-zero mast values.

Remark 1 If $\text{mast}(\vec{v}) > 0$ then $\vec{v} = (\text{lca}^{T^1}(a, b), \ldots \text{lca}^{T^k}(a, b))$ for some leaf labels a, b where $\text{lca}^{T^i}(a, b)$ is the lowest common ancestor of leaves labeled by a and b in the tree T^i.

A tuple $\vec{v}$ such that $\vec{v} = (\text{lca}^{T^1}(a, b), \ldots \text{lca}^{T^k}(a, b))$ for some $a, b \in A$ is called an *lca-tuple*. By Remark 1 it suffices to compute mast values for the lca-tuples only. Just like in the naive approach, $\text{mast}(\vec{v})$ is computed from mast values of other lca-tuples dominated by $\vec{v}$. Another important observation is that only some lca-tuples dominated by $\vec{v}$ are needed to compute $\text{mast}(\vec{v})$. To capture this, Farach et al. define the so called *proper domination* relation (introduced formally below) and show that the mast value for any lca-tuple $\vec{v}$ can be computed from mast values of lca-tuples properly dominated by $\vec{v}$ and that the proper domination relation has size $O(n^3)$.

Proper Domination Relation

Index the children of each internal node of any tree in an arbitrary way. Given a pair $\vec{v}, \vec{w}$ of lca-tuples such that $\vec{w} \prec \vec{v}$ the corresponding domination relation has associated with it *direction* $\vec{\delta}_{\vec{w} \prec \vec{v}} = (\delta_1, \ldots, \delta_k)$ where w_i descends from the child of v_i indexed with δ_i. Let $v_i(j)$ be the child of v_i with index j. The direction domination is termed *active* is if the subtrees rooted at the $v_1(\delta_1), \ldots, v_k(\delta_k)$ have at least one leaf label in common. Note that each leaf label can witness only one active direction, and consequently each lca-tuple can have at most n active domination directions. Two directions $\vec{\delta}_{\vec{w} \prec \vec{v}}$ and $\vec{\delta}_{\vec{u} \prec \vec{v}}$ are called *compatible* if and only if the direction vectors differ in all coordinates.

Definition 1 $\vec{v}$ properly denominates $\vec{u}$ (denoted $\vec{u} < \vec{v}$) if $\vec{v}$ dominates $\vec{u}$ along an active direction $\vec{\delta}$ and there exists another tuple $\vec{w}$ which is also dominated by $\vec{v}$ along an active direction $\vec{\delta}_{\perp}$ compatible with δ.

From the definition of proper domination and from the fact that each leaf label can witness only one active domination direction, the following observations can be made:

Remark 2 The strong domination relation $<$ on lca-tuples has size $O(n^3)$. Furthermore, the relation can be computed in $O(kn^3)$ time.

Remark 3 For any lca-tuple $\vec{v}$, if $\text{mast}(\vec{v}) > 0$ then either $\vec{v}$ is an lca-tuple composed of leaves with the same label or $\vec{v}$ properly dominates some lca-tuple.

It remains to show how the values $\text{mast}(\vec{v})$ are computed. For each lca-tuple $\vec{v}$, the so called compatibility graph $G(\vec{v})$ is constructed. The nodes of $G(\vec{v})$ are active directions from $\vec{v}$ and there is an edge between two such nodes if and only if corresponding directions are compatible. The vertices of $G(\vec{v})$ are weighted and the weight of a vertex corresponding to an active direction $\vec{\delta}$ equals the maximum mast value of a lca-tuple dominated by $\vec{v}$ along the this direction. Let $MWC(G(\vec{v}))$ be the maximum weight clique in $G(\vec{v})$.

The bottom-up algorithm for computing non-zero mast values based on the following recursive dependency whose correctness follows immediately from the corresponding definitions and Remark 3:

Lemma 2 *For any lca-tuple* $\vec{v}$

$$\text{mast}(\vec{v}) = \max \begin{cases} 1 & \text{if all elemets of } \vec{v} \text{ are leaves} \\ MWC(G(\vec{v})) & \text{otherwise} \end{cases} \ . \quad (1)$$

The final step is to demonstrate that once the lca-tuples and the strong domination relation is pre-computed, the computation all non-zero mast values can be preformed in $O(n^d)$ time. This is done by generating all possible cliques for all $G(\vec{v})$. Using the fact that the degree of at least one tree is bounded by d one can show that all the cliques can be generated in $O(\sum_{l \le d} \binom{n}{l}) = O(d^3(ne/d)^d)$ time and that there is $O(d(ne/d)^d)$ of them [6].

Applications

The k-MAST problem is motivated by the need to compare evolutionary trees. Recent advances in experimental techniques in molecular biology provide diverse data that can be used to construct evolutionary trees. This diversity of data combined with the diversity of methods used to construct evolutionary trees often leads to the situation when the evolution of the same set of species is explained by different evolutionary trees. Maximum Agreement Subtree problem has emerged as a measure of the agreement between such trees and as a method to identify subtree which is common for these trees. The problem was first defined by Finden and Gordon in the context of two

binary trees [7]. These authors also gave a heuristic algorithm to solve the problem. The $O(n^2)$ dynamic programming algorithm for computing MAST values for two binary trees has been given in [11]. Subsequently, a number of improvements leading to fast algorithms for computing MAST value of two trees under various assumption about rooting and tree degrees [5,10,8] and references therein.

The MAST problem for three or more unbounded degree trees is NP-complete [1]. Amir and Keselman report an $O(kn^{d+1} + n^{2d})$ time algorithm for the agreement of k bounded degree trees. The work described here provides a $O(kn^3 + n^d)$ for the case where the number of trees is k and the degree of at least one tree is bounded by d. For $d = 2$ the complexity of the algorithm is dominated by the first term. An $O(kn^3)$ algorithm for this case was also given by Bryant [4] and $O(n^2 \log^{k-1} n)$ implementation of this algorithm was proposed in [9].

k-MAST problem is fixed parameter tractable in p, the smallest number of leafs labels such that removal of the corresponding leaves produces agreement (see [3] and references therein). Approximability of the MAST and related problem has been studied in [2] and references therein.

Cross References

- ► Maximum Agreement Subtree (of 2 Binary Trees)
- ► Maximum Agreement Supertree
- ► Maximum Compatible Tree

Acknowledgments

This work was supported by the Intramural Research Program of the National Instituts of Health, National Library of Medicine.

Recommended Reading

1. Amir, A., Keselman, D.: Maximum agreement subtree in a set of evolutionary trees: Metrics and efficient algorithms. SIAM J. Comput. **26**(6), 1656–1669 (1997)
2. Berry, V., Guillemot, S., Nicolas, F., Paul, C.: On the approximation of computing evolutionary trees. In: COCOON, pp. 115–125. (2005)
3. Berry, V., Nicolas, F.: Improved parameterized complexity of the maximum agreement subtree and maximum compatible tree problems. IEEE/ACM Trans. Comput. Biology Bioinform. **3**(3), 289–302 (2006)
4. Bryand, D.: Building trees, hunting for trees, and comparing trees: theory and methods in phylogenetic analysis. In: Ph. D. thesis, Dept. Math., University of Canterbury (1997)
5. Cole, R., Farach-Colton, M., Hariharan, R., Przytycka, T., Thorup, M.: An $o(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees. SIAM J. Comput., pp. 1385–1404. (2001)
6. Farach, M., Przytycka, T.M., Thorup, M.: On the agreement of many trees. Inf. Process. Lett. **55**(6), 297–301 (1995)
7. Finden, C.R., Gordon, A.D.: Obtaining common pruned trees. J. Classific. **2**, 255–276 (1985)
8. Kao, M.-Y., Lam, T.-W., Sung, W.-K., Ting, H.-F.: An even faster and more unifying algorithm for comparing trees via unbalanced bipartite matchings. J. Algorithms **40**(2), 212–233 (2001)
9. Lee, C.-M., Hung, L.-J., Chang, M.-S., Tang, C.-Y.: An improved algorithm for the maximum agreement subtree problem. BIBE, p. 533 (2004)
10. Przytycka, T.M.: Transforming rooted agreement into unrooted agreement. J. Comput. Biol. **5**(2), 335–349 (1998)
11. Steel, M.A., Warnow, T.: Kaikoura tree theorems: Computing the maximum agreement subtree. Inf. Process. Lett. **48**(2), 77–82 (1993)

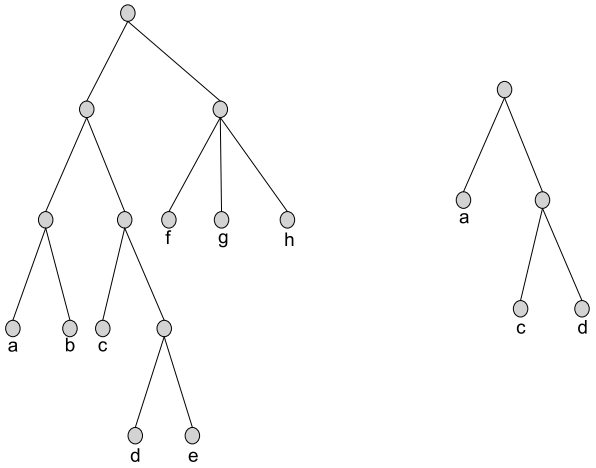
Maximum Agreement Supertree

2005; Jansson, Ng, Sadakane, Sung

WING-KIN SUNG
Department of Computer Science, National University of Singapore, Singapore, Singapore

Problem Definition

Let T be a tree whose leaves are distinctly labeled by a set of taxa S. By distinctly labeled, we mean that no two leaves in T have the same label. Given a subset S' of S, *the topological restriction of T to S'* (denoted by $T|S'$) is the tree obtained by deleting from T all nodes which are not on any path from the root to a leaf in S' along with their incident edges, and then contracting every edge between a node having just one child and its child (see Fig. 1). For any tree T, denote its set of leaves by $\Lambda(T)$.



Maximum Agreement Supertree, Figure 1
Let T be the tree on the *left*. Then $T|\{a, c, d\}$ is the tree shown on the *right*

The maximum agreement supertree problem (MASP) [8] is defined as follows.

Problem 1 *Let $D = \{T_1, T_2, \ldots, T_k\}$ be a set of rooted, unordered trees, where each T_i is distinctly leaf-labeled and where the sets $\Lambda(T_i)$ may overlap. The maximum agreement supertree problem (MASP) is to construct a distinctly leaf-labeled tree Q with leaf set $\Lambda(Q) \subseteq \bigcup_{T_i \in D} \Lambda(T_i)$ such that $|\Lambda(Q)|$ is maximized and for each $T_i \in D$, the topological restriction of T_i to $\Lambda(Q)$ is isomorphic to the topological restriction of Q to $\Lambda(T_i)$.*

Below discussion uses the following notations: $n = \left|\bigcup_{T_i \in D} \Lambda(T_i)\right|$, $k = |D|$, and $D = \max_{T_i \in D}\{\deg(T_i)\}$ where $deg(T_i)$ is the degree of T_i.

Key Results

The following lemma gives the relationship between the maximum agreement supertree problem and the maximum agreement subtree problem.

Lemma 1 ([8]) *For any set $D = \{T_1, T_2, \ldots, T_k\}$ of distinctly leaf-labeled, rooted, unordered trees such that $\Lambda(T_1) = \Lambda(T_2) = \ldots = \Lambda(T_k)$, an optimal solution to MASP for D is an optimal solution to MAST for D and vice versa.*

The above lemma implies the following theorem for computing the maximum agreement supertree for two trees.

Theorem 2 ([8]) *When $k = 2$ (there are two trees), the maximum agreement supertree can be found in $O(T_{MAST} + n)$ time where T_{MAST} is the time required for computing maximum agreement subtree of two $O(n)$-leaf trees. Note that $T_{MAST} = O\left(\sqrt{D}n \log(2n/D)\right)$ (see [9]).*

[1] generalized Theorem 2 and gave the following solution.

Theorem 3 ([1]) *For any fixed $k > 2$, if every leaf in D appears in either 1 or k trees, the maximum agreement supertree can be found in $O(T'_{MAST} + kn)$ time where T'_{MAST} is the time required for computing maximum agreement subtree of k trees leaf-labeled by $\bigcap_{T_i \in D} \Lambda(T_i)$. Note that $T'_{MAST} = O(km^3 + m^D)$ where $n = \left|\bigcap_{T_i \in D} \Lambda(T_i)\right|$ (see [4]).*

In general, the following two theorems showed that the maximum agreement supertree problem is NP-hard.

Theorem 4 ([8,1]) *For any fixed $k \geq 3$, MASP with unrestricted D is NP-hard. Even stronger, MASP is still NP-hard even if restricted to rooted triplets. (A rooted triplet is a distinctly leaf-labeled, binary, rooted, unordered tree with three leaves.)*

Theorem 5 ([1]) *MASP cannot be approximated in polynomial time within a constant factor, unless $P = NP$.*

Though the MASP problem is NP-hard, approximation algorithm for this problem exists.

Theorem 6 ([8]) *MASP can be approximated within a factor of $\frac{n}{\log n}$ in $O(n^2) \cdot \min\left\{O(k \cdot (\log\log n)^2),\ O(k + \log n \cdot \log\log n)\right\}$ time. MASP restricted to rooted triplets can be approximated within a factor of $\frac{n}{\log n}$ in $O(k + n^2 \log^2 n)$ time.*

Fixed parameter polynomial time algorithms for computing MASP also exist. For the case where a set of k binary trees T labeled by n distinct labels is given, a number of works have been done. Jansson et al.[8] first gave an $O(k(2n^2)^{3k^2})$-time algorithm to compute the MASP of T. Recently, Guillemot and Berry [5] improved the time complexity to $O((8n)^k)$. Hoang and Sung [7] further improved the time complexity to $O((6n)^k)$ as summarized by Theorem 7.

Theorem 7 ([7]) *Given a set of k binary trees T which are labeled by n distinct labels, their maximum agreement supertree can be computed in $O((6n)^k)$ time.*

For the case where a set of k trees T are of degree D and are labeled by n distinct labels, Hoang and Sung [7] gave the following fixed-parameter polynomial time solution to compute the MASP of T.

Theorem 8 ([7]) *Given a set of k trees T of degree D which are labeled by n distinct labels, their maximum agreement supertree can be computed in $O((kD)^{kD+3}(2n)^k)$ time.*

Applications

An important objective in phylogenetics is to develop good methods for merging a collection of phylogenetic trees on overlapping sets of taxa into a single supertree so that no (or as little as possible) branching information is lost. Ideally, the resulting supertree can then be used to deduce evolutionary relationships between taxa which do not occur together in any one of the input trees. Supertree methods are useful because most individual studies investigate relatively few taxa [11] and because sample bias leads to certain taxa being studied much more frequently than others [2]. Also, supertree methods can combine trees constructed for different types of data or under different models of evolution. Furthermore, although computationally expensive methods for constructing reliable phylogenetic trees are infeasible for large sets of taxa, they can be applied to obtain highly accurate trees for smaller, overlapping subsets of the taxa which may then be merged using

computationally less intense, supertree-based techniques (see, e. g., [3,6,10]).

Since the set of trees which is to be combined may in practice contain contradictory branching structure (for example, if the trees have been constructed from data originating from different genes or if the experimental data contains errors), a supertree method needs to specify how to resolve conflicts. One intuitive idea is to identify and remove a smallest possible subset of the taxa so that the remaining taxa can be combined without conflicts. In this way, one would get an indication of which ancestral relationships can be regarded as resolved and which taxa need to be subjected to further experiments. The above biological problem can be formalized as a computational problem called the maximum agreement supertree problem (MASP).

A related problem is the maximum compatible supertree problem (MCSP) [1], which is defined as follows.

Problem 2 *Let $D = \{T_1, T_2, \ldots, T_k\}$ be a set of rooted, unordered trees, where each T_i is distinctly leaf-labeled and where the sets $\Lambda(T_i)$ may overlap. The maximum compatible supertree problem (MCSP) is to construct a distinctly leaf-labeled tree Q with leaf set $\Lambda(Q) \subseteq \bigcup_{T_i \in D} \Lambda(T_i)$ such that $|\Lambda(Q)|$ is maximized and for each $T_i \in D$, The topological restriction Q_i' of Q to $\Lambda(T_i)$ refines the topological restriction T_i' of T_i, that is, T_i' can be obtained by collapsing certain edges of Q_i'.*

Open Problems

The current fixed parameter polynomial time algorithms for MASP are not practical. It is important to provide heuristics or to further improve the time complexity of current fixed-parameter polynomial time algorithms.

Cross References

- ► Maximum Agreement Subtree (of 2 Binary Trees)
- ► Maximum Agreement Subtree (of 3 or More Trees)
- ► Maximum Compatible Tree

Recommended Reading

1. Berry, V., Nicolas, F.: Maximum agreement and compatible supertrees. J. Discret. Algorithms (2006)
2. Bininda-Emonds, O., Gittleman, J., Steel, M.: The (super)tree of life: Procedures, problems, and prospects. Ann. Rev. Ecol. System. **33**, 265–289 (2002)
3. Chor, B., Hendy, M., Penny, D.: Analytic solutions for three-taxon ML_{MC} trees with variable rates across sites. In: Proceedings of the 1st Workshop on Algorithms in Bioinformatics (WABI 2001). Lecture Notes in Computer Science, vol. 2149, pp. 204–213. Springer (2001)
4. Farach, M., Przytycka, T., Thorup, M.: On the agreement of many trees. Information Process. Lett. **55**, 297–301 (1995)
5. Guillemot, S., Berry, V.: Fixed-parameter tractability of the maximum agreement supertrees. In: Proceedings of the 18th Annual Symposium on Combinatorial Pattern Matching (CPM 2007). Lecture Notes in Computer Science. Springer, (2007)
6. Henzinger, M.R., King, V., Warnow, T.: Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. Algorithmica **24**(1), 1–13 (1999)
7. Hoang, V.T., Sung, W.K.: Fixed Parameter Polynomial Time Algorithms for Maximum Agreement and Compatible Supertrees. In: Albers, S., Weil, P., 25th International Symposium on Theoretical Aspects of Computer Science (STACS 2008). Dagstuhl, Germany (2007)
8. Jansson, J., Joseph, H., Ng, K., Sadakane, K., Sung, W.-K.: Rooted maximum agreement supertrees. Algorithmica **43**(4), 293–307 (2005)
9. Kao, M.-Y., Lam, T.-W., Sung, W.-K., Ting, H.-F.: An even faster and more unifying algorithm for comparing trees via unbalanced bipartite matchings. J. Algorithms **40**(2), 212–233 (2001)
10. Kearney, P.: Phylogenetics and the quartet method. In: Jiang, T., Xu, Y., Zhang, M.Q. (eds.) Current Topics in Computational Molecular Biology. The MIT Press, Massachusetts, pp. 111–133 (2002)
11. Sanderson, M.J., Purvis, A., Henze, C.: Phylogenetic supertrees: assembling the trees of life. TRENDS in Ecology & Evolution, **13**(3), 105–109 (1998)

Maximum Compatible Tree

2001; Ganapathy, Warnow

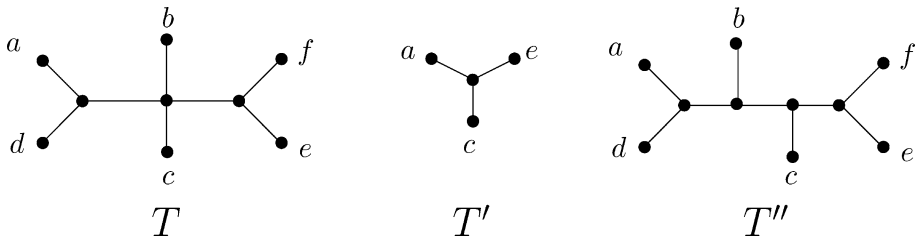
Vincent Berry
LIRMM, University of Montpellier, Montpellier, France

Keywords and Synonyms

Maximum refinement subtree (MRST)

Problem Definition

This problem is a pattern matching problem on leaf-labeled trees. Each input tree is considered as a branching pattern inducing specific groups of leaves. Given a tree collection with identical leaf sets, the goal is to find a largest subset of leaves on the branching pattern of which the input trees do not disagree. A *maximum compatible tree* is a tree with such a leaf-set and with the branching patterns of the input trees for these leaves. The Maximum Compatible Tree problem (MCT) is to find such a tree or, equivalently, its leaf set. The main motivation for this problem is in phylogenetics, to measure the similarity between evolutionary trees, or to represent a consensus of a set of trees. The problem was introduced in [9] and [10], under the MRST acronym. Previous related works concern the well-known Maximum Agreement Subtree prob-



Maximum Compatible Tree, Figure 1
Three unrooted trees. A tree T, a tree T' such that $T' = T|\{a, c, e\}$ and a tree T'' such that $T'' \trianglerighteq T$

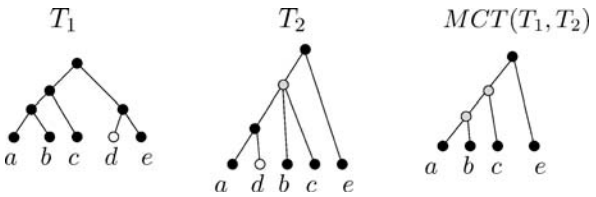
lem (MAST). Solving MAST is finding a largest subset of leaves on which all input trees *exactly* agree. More precisely, MAST seeks a tree whose branching information is isomorphic to that of a subtree in each of the input trees, while MCT seeks a tree that contains the branching information (i. e. groups) of a subtree of each input tree. This difference allows the tree obtained for MCT to be more informative, as it can include branching information present in one input tree but not in the others, as long as this information is compatible with them. Both problems are equivalent when all input trees are binary. Ganapathy and Warnow [5] were the first to give an algorithm to solve MCT in its general form. Their algorithm relies on a simple dynamic programming approach similar to a work on MAST [12] and has a running time exponential in the number of input trees and in the maximum degree of a node in the input trees. Later, [2] proposed a fixed-parameter algorithm using one parameter only. Approximation results have also been obtained [1,6], the result being low-cost polynomial-time algorithms that approximate the complement of MCT within a constant threshold.

Notations Trees considered here are evolutionary trees (*phylogenies*). Such a tree T has its leaf set $L(T)$ in bijection with a label set and is either rooted, in which case all internal nodes have at least two children each, or unrooted, in which case internal nodes have a degree of at least three. The size of $|T|$ of a tree T is the number of its leaves. Given a set L of labels and a tree T, the *restriction* of T to L, denoted $T|L$, is the tree obtained in the following way: take the smallest induced subgraph of T connecting leaves with labels in $L \cap L(T)$, then remove any degree two (non-root) node to make the tree homeomorphically irreducible. Two trees T, T' are *isomorphic*, denoted $T = T'$, if and only if there is a graph isomorphism $T \mapsto T'$ preserving leaf labels (and the root if both trees are rooted). A tree T *refines* a tree T', denoted $T \trianglerighteq T'$, whenever T can be transformed into T' by collapsing some of its internal edges (*collapsing* an edge means removing it and merging its extremities). See Fig. 1 for examples of these relations between trees.

Note that a tree T properly refining another tree T', agrees with the entire evolutionary history of T', while containing additional information absent from T': at least one high degree node of T' is replaced in T by several nodes of lesser degree, hence T contains more speciation events than T'. Given a collection $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of input trees with identical leaf sets L, a tree T with leaves in L is said to be *compatible with* $\mathcal{T}$ if and only if $\forall T_i \in \mathcal{T}$, $T \trianglerighteq T_i|L(T)$. If there is a tree T compatible with $\mathcal{T}$ such that $L(T) = L$, then the collection $\mathcal{T}$ is said to be *compatible*. Knowing whether a collection is compatible is a problem for which linear-time algorithms have been known for a long time e. g. [8]. The MAXIMUM COMPATIBLE TREE problem is a natural optimization version of this problem to deal with incompatible collections of trees.

Problem 1 (MAXIMUM COMPATIBLE TREE – MCT)
Input: *A collection $\mathcal{T}$ of trees with the same leaf sets.*
Output: *A tree compatible with $\mathcal{T}$ having the largest number of leaves. Such a tree is denoted* $\mathrm{MCT}(\mathcal{T})$.

See Fig. 2 for an example. Note that $\forall \mathcal{T}, |\mathrm{MCT}(\mathcal{T})| \geq |\mathrm{MAST}(\mathcal{T})|$ and that MCT is equivalent to MAST when input trees are binary. Note also that instances of MCT and MAST can have several optimum solutions.



Maximum Compatible Tree, Figure 2
An incompatible collection of two input trees $\{T_1, T_2\}$ and their maximum compatible tree, $T = MCT(T_1, T_2)$. Removing the leaf d renders the input trees compatible, hence $L(T) = \{a, b, c, e\}$. Here, T strictly refines T_2 restricted to $L(T)$, which is expressed by the fact that a node in T_2 (the grey one) has its child subtrees distributed between several connected nodes of T (grey nodes). Note also that here $|MCT(T_1, T_2)| > |MAST(T_1, T_2)|$

Key Results

Exact Algorithms

The MCT problem was shown to be NP-hard on 6 trees in [9], then on 2 trees in [10]. The NP-hardness holds as long as one of the input trees is not of bounded degree. For two bounded-degree trees, Hein et al. mention a polynomial-time algorithm based on *aligning* trees [10]. The work of Ganapathy and Warnow [5] proposed an exponential algorithm for solving MCT in the general case. Given two trees T_1, T_2, they show how to compute a binary MCT of any pair of subtrees ($S_1 \in T_1, S_2 \in T_2$) by dynamic programming. Subtrees whose root is of high degree are handled by considering every possible partition of the roots's children in two sets. This leads the complexity bound to have a term exponential in d, the maximum degree of a node in the input trees. When dealing with k input trees, k-tuples of subtrees are considered, and the simultaneous bipartitions of the roots's children for k subtrees are considered. Hence, the complexity bound is also exponential in k.

Theorem 1 ([5]) *Let L be a set of n leaves. The* MCT *problem for a collection of k rooted trees on L in which each tree has degree at most $d+1$, can be solved in $O(2^{2kd} n^k)$ time.*

The result easily extends to unrooted trees by considering each of the n leaves in turn as a possible root for all trees of the collection.

Theorem 2 ([5]) *Given a collection of k unrooted trees with degree at most $d+1$ on an n-leaf set, the* MCT *problem can be solved in $O(2^{2kd} n^{k+1})$.*

Let $\mathcal{T}$ be a collection on a leaf-set L, [2] considered the following decision problem, denoted MCT$_p$: given $\mathcal{T}$ and $p \in [0, n]$, does $|MCT(\mathcal{T})| \geq n - p$?

Theorem 3 ([2])

1. MCT$_p$ *on rooted trees can be solved in* $O(\min\{3^p kn, 2.27^p + kn^3\})$ *time.*
2. MCT$_p$ *on unrooted trees can be solved in* $O\big((p+1) \times \min\{3^p kn, 2.27^p + kn^3\}\big)$ *time.*

The 3^{pkn} term comes from an algorithm that first locates in $O(kn)$ time a 3-leaf set S on which the input trees conflict, then recursively obtains a maximum compatible tree T_1, resp. T_2, T_3 for each of the three collections $\mathcal{T}_1$, resp. $\mathcal{T}_2, \mathcal{T}_3$ obtained by removing from the input trees a leaf in S, and last returning the T_i such that $|T_i|$ is maximum (for $i \in [1, 3]$). The $2.27^p + kn^3$ term comes from an algorithm reducting MCT to 3-HITTING SET. Negative results have been obtained by Guillemot and Nicolas concerning the fixed-parameter tractability of MCT wrt the maximum degree D of the input trees.

Theorem 4 ([7])

1. MCT *is $W[1]$-hard with respect to D.*
2. MCT *can not be solved in $O(N^{o(2^{D/2})})$ time unless SNP $\subseteq$ SE, where N denotes the input length, i. e. $N = O(kn)$.*

The MCT problem also admits a variant that deals with *supertrees*, i. e. trees having different (but overlapping) sets of leaves. The resulting problem is $W[2]$-hard with respect to p [3].

Approximation Algorithms

The idea of locating and then eliminating successively all the conflicts between the input trees has also led to approximation algorithms for the *complement* version of the MCT problem, denoted CMCT. Let L be the leaf set of each tree in an input collection $\mathcal{T}$, CMCT aims at selecting the smallest number of leaves $S \subseteq L$ such that the collection $\{T_i | (L - S) : T_i \in \mathcal{T}\}$ is compatible.

Theorem 5 ([6]) *Given a collection $\mathcal{T}$ of k rooted trees on an n-leaf set L, there is a 3-approximation algorithm for* CMCT *that runs in $O(k^2 n^2)$ time.*

The running time of this algorithm was later improved:

Theorem 6 ([1]) *There is an $O(kn + n^2)$ time 3-approximation algorithm for* CMCT *on a collection of k rooted trees with n leaves.*

Note also that working on rooted or unrooted trees does not change the achievable approximation threshold for CMCT [1].

Applications

In bioinformatics, the MCT problem (and similarly MAST) is used to reach different practical goals. The first motivation is to measure the similarity of a set of trees. These trees can represent RNA secondary structures [10,11] or estimates of a phylogeny inferred from different datasets composed of molecular sequences (e. g. genes) [13]. The gap between the size of a maximum compatible tree and the number of input leaves indicates the degree of disimilarity of the input trees. Concerning the phylogenetic applications, quite often some edges of the trees inferred from the datasets have been collapsed due to insufficient statistical support, resulting in some higher-degree nodes in the trees considered. Each such node does not indicate a multi-speciation event but rather the uncertainty with respect to the branching pattern to be chosen for its child subtrees. In such a situation, the MCT problem is to be preferred to MAST, as it correctly handles high degree nodes, enabling them to be resolved according to branching information present in other input

trees. As a result, more leaves are conserved in the output tree, hence a larger degree of similarity is detected between the input trees. Note also that a low similarity value between the input trees can be due to horizontal gene transfers. When these events are not too numerous, identifying species subject to such effects is done by first suspecting leaves discarded from a maximum compatible tree.

The shape of a maximum compatible tree, i. e. not just its size, also has an application in systematic biology to obtain a consensus of a set of phylogenies that are optimal for some tree-building criterion. For instance, the maximum parsimony and maximum likelihood criteria can provide several dozens (sometimes hundreds) of optimal or near-optimal trees. In practice, these trees are first grouped into islands of neighboring trees, and a consensus tree is obtained for each island by resorting to a classical consensus tree method, e. g. the majority-rule or strict consensus. The trees representing the islands form a collection of which a consensus is then sought. However, consensus methods keeping all input leaves tend to create trees that lack of resolution. An alternative approach lies in proposing a representative tree that contains a largest possible subset of leaves on the position of which the trees of the collection agree. Again, MCT is more suited than MAST as the input trees can contain some high-degree nodes, with the same meaning as discussed above.

Open Problems

A direction for future work is to examine the variant of MCT where some leaves are imposed in the output tree. This question arises when a biologist wants to ensure that the species central to his study are contained in the output tree. For MAST on two trees, this constrained variant of the problem was shown in a natural way to be of the same complexity as the standart version [4]. For MCT however, such a constraint can lead to several optimization problems that need to be sorted out. Another important work to be done is a set of experiments to measure the range of parameters for which the algorithms proposed to solve or approximate MCT are useful.

URL to Code

A beta-version of a Perl program can be asked to the author of this entry.

Cross References

- ► Maximum Agreement Subtree (of 2 Binary Trees)
- ► Maximum Agreement Subtree (of 3 or More Trees)

Recommended Reading

1. Berry, V., Guillemot, S., Nicolas, F., Paul, C.: On the approximation of computing evolutionary trees. In: Wang, L. (ed.) Proc. of the 11th Annual International Conference on Computing and Combinatorics (COCOON'05). LNCS, vol. 3595, pp. 115–125. Springer, Berlin (2005)
2. Berry, V., Nicolas, F.: Improved parametrized complexity of the maximum agreement subtree and maximum compatible tree problems. IEEE/ACM Trans. Comput. Biol. Bioinform. **3**(3), 289–302 (2006)
3. Berry, V., Nicolas, F.: Maximum agreement and compatible supertrees. J. Discret. Algorithms. Algorithmica, Springer, New York (2008)
4. Berry, V., Peng, Z.S., Ting, H.-F.: From constrained to unconstrained maximum agreement subtree in linear time. Algorithmica, to appear (2006)
5. Ganapathy, G., Warnow, T.J.: Finding a maximum compatible tree for a bounded number of trees with bounded degree is solvable in polynomial time. In: Gascuel, O., Moret, B.M.E. (eds.) Proc. of the 1st International Workshop on Algorithms in Bioinformatics (WABI'01), pp. 156–163 (2001)
6. Ganapathy, G., Warnow, T.J.: Approximating the complement of the maximum compatible subset of leaves of *k* trees. In: Proc. of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX'02), LCNS, vol. 2462, pp. 122–134., Springer, Berlin (2002)
7. Guillemot, S., Nicolas, F.: Solving the maximum agreement subtree and the maximum compatible tree problems on many bounded degree trees. In: Lewenshtein, M., Valiente, G. (eds.) Proc. of the 17th Combinatorial Pattern Matching Symposium (CPM'06). LNCS, vol. 4009, pp. 165–176. Springer, Berlin (2006)
8. Gusfield, D.: Efficient algorithms for inferring evolutionary trees. Networks **21**, 19–28 (1991)
9. Hamel, A.M., Steel, M.A.: Finding a maximum compatible tree is NP-hard for sequences and trees. Appl. Math. Lett. **9**(2), 55–59 (1996)
10. Hein, J., Jiang, T., Wang, L., Zhang, K.: On the complexity of comparing evolutionary trees. Discrete Appl. Math. **71**(1–3), 153–169 (1996)
11. Jiang, T., Wang, L., Zhang, K.: Alignment of trees – an alternative to tree edit. Theor. Comput. Sci. **143**(1), 137–148 (1995)
12. Steel, M.A., Warnow, T.J.: Kaikoura tree theorems: Computing the maximum agreement subtree. Inform. Process. Lett. **48**(2), 77–82 (1993)
13. Swofford, D.L., Olsen, G.J., Wadell, P.J., Hillis, D.M.: Phylogenetic inference. In: Hillis, D.M., Moritz, D.M., Mable, B.K. (eds.) Molecular systematics, 2nd edn. pp. 407–514. Sunderland, USA (1996)

Maximum-Density Segment

1994; Huang

Kun-Mao Chao
Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

Keywords and Synonyms

Maximum-average segment

Problem Definition

Given a sequence of numbers, $A = \langle a_1, a_2, \ldots, a_n \rangle$, and two positive integers L, U, where $1 \leq L \leq U \leq n$, the maximum-density segment problem is to find a consecutive subsequence, i. e. a segment or substring, of A with length at least L and at most U such that the average value of the numbers in the subsequence is maximized.

Key Results

If there is no length constraint, then obviously the maximum-density segment is the maximum number in the sequence. Let's first consider the problem where only the length lower bound L is imposed. By observing that the length of the shortest maximum-density segment with length at least L is at most $2L - 1$, Huang [7] gave an $O(nL)$-time algorithm. Lin et al. [10] proposed a new technique, called the *right-skew decomposition*, to partition each suffix of A into *right-skew* segments of strictly decreasing averages. The right-skew decomposition can be done in $O(n)$ time, and it can answer, for each position i, a consecutive subsequence of A starting at that position such that the average value of the numbers in the subsequence is maximized. On the basis of the right-skew decomposition, Lin et al. [10] devised an $O(n \log L)$-time algorithm for the maximum-density segment problem with a lower bound L, which was improved to $O(n)$ time by Goldwasser et al. [6]. Kim [8] gave another $O(n)$-time algorithm by reducing the problem to the maximum-slope problem in computation geometry. As for the problem which takes both L and U into consideration, Chung and Lu [4] bypassed the construction of the right-skew decomposition and gave an $O(n)$-time algorithm.

It should be noted that a closely related problem in data mining, which basically deals with a binary sequence, was independently formulated and studied by Fukuda et al. [5].

An Extension to Multiple Segments

Given a sequence of numbers, $A = \langle a_1, a_2, \ldots, a_n \rangle$, and two positive integers L and k, where $k \leq \frac{n}{L}$, let $d(A[i, j])$ denote the *density* of segment $A[i, j]$, defined as $(a_i + a_{i+1} + \cdots + a_j)/(j - i + 1)$. The problem is to find k disjoint segments $\{s_1, s_2, \ldots, s_k\}$ of A, each has a length of at least L, such that $\sum_{1 \leq i \leq k} d(s_i)$ is maximized. Chen et al. [3] proposed an $O(nkL)$-time algorithm and an improved $O(nL + k^2L^2)$-time algorithm was given by Bergkvist and Damaschke [2]. Liu and Chao [11] gave an $O(n + k^2 L \log L)$-time algorithm.

Applications

In all organisms, the GC base composition of DNA varies between 25–75%, with the greatest variation in bacteria. Mammalian genomes typically have a GC content of 45–50%. Nekrutenko and Li [12] showed that the extent of the compositional heterogeneity in a genomic sequence strongly correlates with its GC content. Genes are found predominantly in the GC-richest isochore classes. Hence, finding GC-rich regions is an important problem in gene recognition and comparative genomics.

Given a DNA sequence, one would attempt to find segments of length at least L with the highest C+G ratio. Specifically, each of nucleotides C and G is assigned a score of 1, and each of nucleotides A and T is assigned a score of 0.

```
DNA sequence:    ATGACTCGAGCTCGTCA
Binary sequence: 00101011011011010
```

The maximum-average segments of the binary sequence correspond to those segments with the highest GC ratio in the DNA sequence. Readers can refer to [1,9,10,13,14,15] for more applications.

Open Problems

The best asymptotic time bound of the algorithms for the multiple maximum-density segments problem is $O(n + k^2 L \log L)$. Can this problem be solved in $O(n)$ time?

Cross References

► Maximum-scoring Segment with Length Restrictions

Recommended Reading

1. Arslan A., Eğecioğlu, Ö, Pevzner, P.: A new approach to sequence comparison: normalized sequence alignment. Bioinformatics **17**, 327–337 (2001)
2. Bergkvist, A., Damaschke, P.: Fast algorithms for finding disjoint subsequences with extremal densities. In: Proceedings of the 16th Annual International Symposium on Algorithms and Computation. LNCS, vol. 3827, pp. 714–723 (2005)
3. Chen, Y.H., Lu, H.I., Tang, C.Y.: Disjoint segments with maximum density. In: Proceedings of the 5th Annual International Conference on Computational Science, pp. 845–850 (2005)
4. Chung, K.-M., Lu, H.-I.: An optimal algorithm for the maximum-density segment problem. SIAM. J. Comput. **34**, 373–387 (2004)

5. Fukuda, T., Morimoto, Y., Morishita, S., Tokuyama, T.: Mining Optimized Association Rules for Numeric Attributes. J. Comput. Syst. Sci. **58**, 1–12 (1999)
6. Goldwasser, M.H., Kao, M.-Y., Lu, H.-I.: Linear-time algorithms for computing maximum-density sequence segments with bioinformatics applications. J. Comput. Syst. Sci. **70**, 128–144 (2005)
7. Huang, X.: An algorithm for identifying regions of a DNA sequence that satisfy a content requirement. Comput. Appl. Biosci. **10**, 219–225 (1994)
8. Kim, S.K.: Linear-time algorithm for finding a maximum-density segment of a sequence. Inf. Process. Lett. **86**, 339–342 (2003)
9. Lin, Y.-L., Huang, X., Jiang, T., Chao, K.-M.: MAVG: locating non-overlapping maximum average segments in a given sequence. Bioinformatics **19**, 151–152 (2003)
10. Lin, Y.-L., Jiang, T., Chao, K.-M.: Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis. J. Comput. Syst. Sci. **65**, 570–586 (2002)
11. Liu, H.-F., Chao, K.-M.: On locating disjoint segments with maximum sum of densities. In: Proceedings of the 17th Annual International Symposium on Algorithms and Computation. LNCS, vol. 4288, pp. 300–307 (2006)
12. Nekrutenko, A., Li, W.H.: Assessment of compositional heterogeneity within and between eukaryotic genomes. Genome Res. **10**, 1986–1995 (2000)
13. Stojanovic, N., Florea, L., Riemer, C., Gumucio, D., Slightom, J., Goodman, M., Miller, W., Hardison, R.: Comparison of five methods for finding conserved sequences in multiple alignments of gene regulatory regions. Nucl. Acid. Res. **19**, 3899–3910 (1999)
14. Stojanovic, N., Dewar, K.: Identifying multiple alignment regions satisfying simple formulas and patterns. Bioinformatics **20**, 2140–2142 (2005)
15. Zhang, Z., Berman, P., Wiehe, T., Miller, W.: Post-processing long pairwise alignments. Bioinformatics **15**, 1012–1019 (1999)

Maximum Matching

2004; Mucha, Sankowski

Marcin Mucha
Faculty of Mathematics, Informatics and Mechanics, Institute of Informatics, Warsaw, Poland

Problem Definition

Let $G = (V, E)$ be an undirected graph, and let $n = |V|$, $m = |E|$. A *matching* in G is a subset $M \subseteq E$, such that no two edges of M have a common endpoint. A *perfect matching* is a matching of cardinality $n/2$. The most basic matching related problems are: finding a *maximum matching* (i. e. a matching of maximum size) and, as a special case, finding a *perfect matching* if one exists. One can also consider the case where a weight function $w\colon E \to \mathbb{R}$ is given and the problem is to find a *maximum weight matching*.

The maximum matching and maximum weight matching are two of the most fundamental algorithmic graph problems. They have also played a major role in the development of combinatorial optimization and algorithmics. An excellent account of this can be found in a classic monograph [10] by Lovász and Plummer devoted entirely to matching problems. A more up-to-date, but also more technical discussion of the subject can be found in [18].

Classical Approach

Solving the maximum matching problem in time polynomial in n is a highly non-trivial task. The first such solution was given by Edmonds [3] in 1965 and has time complexity $O(n^3)$. Edmond's ingenious algorithm uses a combinatorial approach based on augmenting paths and blossoms. Several improvements followed, culminating in the algorithm with complexity $O(m\sqrt{n})$ given by Micali and Vazirani [11] in 1980 (a complete proof of the correctness of this algorithm was given much later by Vazirani [19], a nice exposition of the algorithm and its generalization to the weighted case can be found in a work of Gabow and Tarjan [4]). Beating this bound proved very difficult, several authors managed to achieve only a logarithmic speed-up for certain values of m and n. All these algorithms essentially follow the combinatorial approach introduced by Edmonds.

The maximum matching problem is much simpler for bipartite graphs. The complexity of $O(m\sqrt{n})$ was achieved for this case already in 1971 by Hopcroft and Karp [6], while the key ideas of the first polynomial algorithms date back to 1920's and the works of König and Egerváry (see [10] and [18]).

Algebraic Approach

Around the time Micali and Vazirani introduced their matching algorithm, Lovász gave a randomized (Monte Carlo) reduction of the problem of testing whether a given n-vertex graph has a perfect matching to the problem of computing a certain determinant of a $n \times n$ matrix. Using the Hopcroft-Bunch fast Gaussian elimination algorithm [1] this determinant can be computed in time $MM(n) = O(n^\omega)$ – time required to multiply two $n \times n$ matrices. Since $\omega < 2.38$ (see [2]), for dense graphs this algorithm is asymptotically faster than the matching algorithm of Micali and Vazirani.

However, Lovász's algorithm only tests for perfect matching, it does not find it. Using it to find perfect/maximum matchings in a straightforward fashion yields algorithm with complexity $O(mn^\omega) = O(n^{4.38})$. A major

open problem in the field was thus: can maximum matchings be actually found in $O(n^\omega)$ time?

The first step in this direction was taken in 1989 by Rabin and Vazirani [15]. They showed that maximum matchings can be found in time $O(n^{\omega+1}) = O(n^{3.38})$.

Key Results

The following theorems state the key results of [12].

Theorem 1 *Maximum matching in a n-vertex graph G can be found in $O(n^3)$ time (Las Vegas) by performing Gaussian elimination on a certain matrix related to G.*

Theorem 2 *Maximum matching in an n-vertex bipartite graph can be found in $\tilde{O}(n^\omega)$ time (Las Vegas) by performing a Hopcroft-Bunch fast Gaussian elimination on a certain matrix related to G.*

Theorem 3 *Maximum matching in an n-vertex graph can be found in $\tilde{O}(n^\omega)$ time (Las Vegas).*

Note: $\tilde{O}$ notation suppresses polylogarithmic factors, so $\tilde{O}(f(n))$ means $O(f(n)\log^k(n))$ for some k.

Let us briefly discuss these results. Theorem 1 shows that effective matching algorithms can be simple. This is in large contrast to augmenting paths/blossoms based algorithms which a generally regarded quite complicated.

The other two theorems show that, for dense graphs, the algebraic approach is asymptotically faster than the combinatorial one.

The algorithm for the bipartite case is very simple. It's only non-elementary part is the fast matrix multiplication algorithm used as black box by the Hopcroft-Bunch algorithm. The general algorithm, however, is complicated and uses strong structural results from matching theory. A natural question is whether or not it is possible to give a simpler and/or purely algebraic algorithm. This has been positively answered by Harvey [5].

Several other related results followed. Mucha and Sankowski [13] showed that maximum matchings in planar graphs can be found in time $\tilde{O}(n^{\omega/2}) = \tilde{O}(n^{1.19})$ which is currently fastest known. Yuster and Zwick [20] extended this to any excluded minor class of graphs. Sankowski [16] gave an RNC work-efficient matching algorithm (see also Mulmuley et al. [14] and Karp et al. [8] for earlier, less efficient RNC matching algorithms, and Karloff [7] for a description of a general technique for making such algorithm Las Vegas). He also generalized Theorem 2 to the case of weighted bipartite graphs with integer weights from $[0, \ldots, W]$, showing that in this case maximum weight matchings can be found in time $\tilde{O}(Wn^\omega)$ (see [17]).

Applications

The maximum matching problem has numerous applications, both in practice and as a subroutine in other algorithms. A nice discussion of practical applications can be found in the monograph [10] by Lovász and Plummer. It should be noted, however, that algorithms based on fast matrix multiplication are completely impractical, so the results discussed here are not really useful in these applications.

On the theoretical side, faster maximum (weight) matching algorithms yield faster algorithms for related problems: disjoint s-t paths problem, the minimum (weight) edge cover problem, the (maximum weight) b-matching problem, the (maximum weight) b-factor problem, the maximum (weight) T-join or the Chinese postman problem. For detailed discussion of all these applications see [10] and [18].

The algebraic algorithm of Theorem 1 also has a significant educational value. The combinatorial algorithms for the general maximum matching problem are generally regarded too complicated for an undergraduate course. That is definitely not the case with the algebraic $O(n^3)$ algorithm.

Open Problems

One of the most important open problems in the area is generalizing the results discussed above to weighted graphs. Sankowski [17] gives a $\tilde{O}(Wn^\omega)$ algorithm for bipartite graphs with integer weights from the interval $[0..W]$. The complexity of this algorithm is really bad in terms of W. No effective algebraic algorithm is known for general weighted graphs.

Another interesting, but most likely very hard problem is the derandomization of the algorithms discussed.

Cross References

- ► All Pairs Shortest Paths via Matrix Multiplication
- ► Assignment Problem

Recommended Reading

1. Bunch, J., Hopcroft, J.: Triangular Factorization and Inversion by Fast Matrix Multiplication. Math. Comput. **125**, 231–236 (1974)
2. Coppersmith, D., Winograd, S.: Matrix Multiplication via Arithmetic Progressions. In: Proceedings of the 19th Annual ACM Conference on Theory of Computing (STOC), 1987, pp. 1–6
3. Edmonds, J.: Paths, Trees, and Flowers. Canad. J. Math. **17**, 449–467 (1965)
4. Gabow, H.N., Tarjan, R.E.: Faster scaling algorithms for general graph matching problems. J. ACM **38**(4), 815–853 (1991)

5. Harvey, N.: Algebraic Structures and Algorithms for Matching and Matroid Problems. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2006
6. Hopcroft, J.E., Karp, R.M.: An $O(n^{5/2})$ Algorithm for Maximum Matchings in Bipartite Graphs. SIAM J. Comput. **2**, 225–231 (1973)
7. Karloff, H.: A Las Vegas RNC algorithm for maximum matching. Combinatorica **6**, 387–391 (1986)
8. Karp, R., Upfal, E., Widgerson, A.: Constructing a perfect matching is in Random NC. Combinatorica **6**, 35–48 (1986)
9. Lovász, L.: On Determinants, Matchings and Random Algorithms. In: Budach, L. (ed.) Fundamentals of Computation Theory, FCT'79, pp. 565–574. Akademie-Verlag, Berlin (1979)
10. Lovász, L., Plummer, M.D.: Matching Theory. Akadémiai Kiadó – North Holland, Budapest (1986)
11. Micali, S., Vazirani, V.V.: An $O(\sqrt{V}E)$ Algorithm for Finding Maximum Matching in General Graphs. In: Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science (FOCS), 1980, pp. 17–27
12. Mucha, M., Sankowski, P.: Maximum Matchings via Gaussian Elimination. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2004 pp. 248–255
13. Mucha, M., Sankowski, P.: Maximum Matchings in Planar Graphs via Gaussian Elimination. Algorithmica **45**, 3–20 (2006)
14. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching is as easy as matrix inversion. In: Proceedings of the 19th Annual ACM Conference on Theory of Computing, pp. 345–354. ACM Press, New York (1987)
15. Rabin, M.O., Vazirani, V.V.: Maximum Matchings in General Graphs Through Randomization. J. Algorithms **10**, 557–567 (1989)
16. Sankowski, P.: Processor Efficient Parallel Matching. In: Proceeding of the 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2005, pp. 165–170
17. Sankowski, P.: Weighted Bipartite Matching in Matrix Multiplication Time. In: Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, 2006, pp. 274–285
18. Schrijver, A.: Combinatorial optimization: polyhedra and efficiency. Springer, Berlin Heidelberg (2003)
19. Vazirani, V.V.: A Theory of Alternating Paths and Blossoms for Proving Correctness of the $O(\sqrt{V}E)$ Maximum Matching Algorithm. Combinatorica **14**(1), 71–109 (1994)
20. Yuster, R., Zwick, U.: Maximum Matching in Graphs with an Excluded Minor. In: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), 2007

Maximum-scoring Segment with Length Restrictions

2002; Lin, Jiang, Chao

Kun-Mao Chao
Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

Keywords and Synonyms

Shortest path; Longest path

Problem Definition

Given a sequence of numbers, $A = \langle a_1, a_2, \ldots, a_n \rangle$, and two positive integers L, U, where $1 \leq L \leq U \leq n$, the maximum-sum segment problem is to find a consecutive subsequence, i. e. a segment or substring, of A with length at least L and at most U such that the sum of the numbers in the subsequence is maximized.

Key Results

The maximum-sum segment problem without length constraints is linear-time solvable by using Kadane's algorithm [2]. Huang extended the recurrence relation used in [2] for solving the maximum-sum segment problem, and derived a linear-time algorithm for computing the maximum-sum segment with length at least L. Lin et al. [10] proposed an $O(n)$-time algorithm for the maximum-sum segment problem with both L and U constraints, and an online version was given by Fan et al. [8].

An Extension to Multiple Segments

Computing the k largest sums over all possible segments is a natural extension of the maximum-sum segment problem. This extension has been considered from two perspectives, one of which allows the segments to overlap, while the other disallows.

Linear-time algorithms for finding all the non-overlapping maximal segments were given in [3,12]. On the other hand, one may focus on finding the k maximum-sum segments whose overlapping is allowed. A naïve approach is to choose the k largest from the sums of all possible contiguous subsequences which requires $O(n^2)$ time. Bae and Takaoka [1] presented an $O(kn)$-time algorithm for the k maximum segment problem. Liu and Chao [11] noted that the k maximum-sum segments problem can be solved in $O(n + k)$ time [7], and gave an $O(n + k)$-time algorithm for the Length-Constrained k Maximum-Sum Segments Problem.

Applications

The algorithms for the maximum-sum segment problem have applications in finding GC-rich regions in a genomic DNA sequence, postprocessing sequence alignments, and annotating multiple sequence alignments. Readers can refer to [3,4,5,6,10,12,13,14,15] for more details.

Open Problems

It would be interesting to consider the higher dimensional cases.

Cross References

► Maximum-Density Segment

Recommended Reading

1. Bae, S.E., Takaoka, T.: Algorithms for the problem of k maximum sums and a VLSI algorithm for the k maximum subarrays problem. Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks, pp. 247–253 (2004)
2. Bentley, J.: Programming Pearls. Addison-Wesley, Reading (1986)
3. Chen, K.-Y., Chao, K.-M.: On the range maximum-sum segment query problem. Proceedings of the 15th International Symposium on Algorithms And Computation. LNCS **3341**, 294–305 (2004)
4. Chen, K.-Y., Chao, K.-M.: Optimal algorithms for locating the longest and shortest segments satisfying a sum or an average constraint. Inf. Process. Lett. **96**, 197–201 (2005)
5. Cheng, C.-H., Chen, K.-Y., Tien, W.-C., Chao, K.-M.: Improved algorithms for the *k* maximum-sum problems. Proceedings of the 16th International Symposium on Algorithms And Computation. Theoret. Comput. Sci. 362: 162–170 (2006)
6. Csűrös, M.: Maximum-scoring segment sets. IEEE/ACM Trans. Comput. Biol. Bioinform. **1**, 139–150 (2004)
7. Eppstein, D.: Finding the *k* Shortest Paths. SIAM J. Comput. **28**, 652–673 (1998)
8. Fan, T.-H., Lee, S., Lu, H.-I., Tsou, T.-S., Wang, T.-C., Yao, A.: An optimal algorithm for maximum-sum segment and its application in bioinformatics. Proceedings of the Eighth International Conference on Implementation and Application of Automata. LNCS **2759**, 251–257 (2003)
9. Huang, X.: An algorithm for identifying regions of a DNA sequence that satisfy a content requirement. Comput. Appl. Biosci. **10**, 219–225 (1994)
10. Lin, Y.-L., Jiang, T., Chao, K.-M.: Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis. J. Comput. Syst. Sci. **65**, 570–586 (2002)
11. Liu, H.-F., Chao, K.-M.: Algorithms for Finding the Weight-Constrained *k* Longest Paths in a Tree and the Length-Constrained *k* Maximum-Sum Segments of a Sequence. Theoret. Comput. Sci. in revision (2008)
12. Ruzzo, W.L., Tompa, M.: A linear time algorithm for finding all maximal scoring subsequences. Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology, pp. 234–241 (1999)
13. Stojanovic, N., Florea, L., Riemer, C., Gumucio, D., Slightom, J., Goodman, M., Miller, W., Hardison, R.: Comparison of five methods for finding conserved sequences in multiple alignments of gene regulatory regions. Nucleic Acids Res. **19**, 3899–3910 (1999)
14. Stojanovic, N., Dewar, K.: Identifying multiple alignment regions satisfying simple formulas and patterns. Bioinformatics **20**, 2140–2142 (2005)
15. Zhang, Z., Berman, P., Wiehe, T., Miller, W.: Post-processing long pairwise alignments. Bioinformatics **15**, 1012–1019 (1999)

Maximum Two-Satisfiability

2004; Williams

Ryan Williams
Department of Computer Science,
Carnegie Mellon University, Pittsburgh, PA, USA

Keywords and Synonyms

Max 2-SAT

Problem Definition

In the maximum 2-satisfiability problem (abbreviated as Max 2-Sat), one is given a Boolean formula in conjunctive normal form, such that each clause contains at most two literals. The task is to find an assignment to the variables of the formula such that a maximum number of clauses is satisfied.

Max 2-Sat is a classic optimization problem. Its decision version was proved NP-complete by Garey, Johnson, and Stockmeyer [7], in stark contrast with 2-Sat which is solvable in linear time [2]. To get a feeling for the difficulty of the problem, the NP-completeness reduction is sketched here. One can transform any 3-Sat instance F into a Max 2-Sat instance F', by replacing each clause of F such as

$$c_i = (\ell_1 \vee \ell_2 \vee \ell_3)\,,$$

where ℓ_1, ℓ_2, and ℓ_3 are arbitrary literals, with the collection of 2-CNF clauses

$$(\ell_1), (\ell_2), (\ell_3), (c_i), (\neg\ell_1 \vee \neg\ell_2), (\neg\ell_2 \vee \neg\ell_3),$$
$$(\neg\ell_1 \vee \neg\ell_3), (\ell_1 \vee c_i), (\ell_2 \vee c_i), (\ell_3 \vee c_i)\,,$$

where c_i is a new variable. The following are true:

- If an assignment satisfies c_i, then exactly seven of the ten clauses in the 2-CNF collection can be satisfied.
- If an assignment does not satisfy c_i, then exactly six of the ten clauses can be satisfied.

If F is satisfiable then there is an assignment satisfying 7/10 of the clauses in F', and if F is not satisfiable then no assignment satisfies more than 7/10 of the clauses in F'. Since 3-Sat reduces to Max 2-Sat, it follows that Max 2-Sat (as a decision problem) is NP-complete.

Notation

A CNF formula is represented as a set of clauses.

The symbols $\mathbb{R}$ and $\mathbb{Z}$ denote the sets of reals and integers, respectively. The letter ω denotes the smallest real number such that for all $\epsilon > 0$, n by n matrix multiplication over a ring can be performed in $O(n^{\omega+\epsilon})$ ring operations. Currently, it is known that $\omega < 2.376$ [4]. The ring matrix product of two matrices A and B is denoted by $A \times B$.

Let A and B be matrices with entries from $\mathbb{R} \cup \{\infty\}$. The *distance product* of A and B (written shorthand as $A \otimes_d B$) is the matrix C defined by the formula

$$C[i,j] = \min_{k=1,\ldots,n} \{A[i,k] + B[k,j]\} .$$

A word on m's and n's: in reference to graphs, m and n denote the number of edges and the number of nodes in the graph, respectively. In reference to CNF formulas, m and n denote the number of clauses and the number of variables, respectively.

Key Result

The primary result of this article is a procedure solving MAX 2-SAT in $O(m \cdot 2^{\omega n/3})$ time. The method can be generalized to *count* the number of solutions to *any* constraint optimization problem with at most two variables per constraint (cf. [17]), though the presentation in this article shall be somewhat different from the reference, and much simpler. There are several other known exact algorithms for MAX 2-SAT that are more effective in special cases, such as sparse instances [3,8,9,11,12,13,15,16]. The procedure described below is the only one known (to date) that runs in $O(poly(m) \cdot 2^{\delta n})$ time (for some fixed $\delta < 1$) in all possible cases.

Key Idea

The algorithm gives a reduction from MAX 2-SAT to the problem MAX TRIANGLE, in which one is given a graph with integer weights on its nodes and edges, and the goal is to output a 3-cycle of maximum weight. At first, the existence of such a reduction sounds strange, as MAX TRIANGLE can be trivially solved in $O(n^3)$ time by trying all possible 3-cycles. The key is that the reduction exponentially increases the problem size, from a MAX 2-SAT instance with m clauses and n variables, to a MAX TRIANGLE instance having $O(2^{2n/3})$ edges, $O(2^{n/3})$ nodes, and weights in the range $\{-m, \ldots, m\}$.

Note that if MAX TRIANGLE required $\Theta(n^3)$ time to solve, then the resulting MAX 2-SAT algorithm would take $\Theta(2^n)$ time, rendering the above reduction pointless. However, it turns out that the brute-force search of $O(n^3)$ for MAX TRIANGLE is not the best one can do– using fast matrix multiplication, there is an algorithm for MAX TRIANGLE that runs in $O(Wn^{\omega})$ time on graphs with weights in the range $\{-W, \ldots, W\}$.

Main Algorithm

First, a reduction from MAX 2-SAT to MAX TRIANGLE is described, arguing that each triangle of weight K in the resulting graph is in one-to-one correspondence with an assignment that satisfies K clauses of the MAX 2-SAT instance. Let a, b be reals, and let $\mathbb{Z}[a,b] := [a,b] \cap \mathbb{Z}$

Lemma 1 *If* MAX TRIANGLE *on graphs with n nodes and weights in* $\mathbb{Z}[-W, W]$ *is solvable in* $O(f(W) \cdot g(n))$ *time, for polynomials f and g, then* MAX 2-SAT *is solvable in* $O(f(m) \cdot g(2^{n/3}))$ *time, where m is the number of clauses and n is the number of variables.*

Proof Let C be a given 2-CNF formula. Assume without loss of generality that n is divisible by 3. Let F be an instance of MAX 2-SAT. Arbitrarily partition the n variables of F into three sets P_1, P_2, P_3, each having $n/3$ variables. For each P_i, make a list L_i of all $2^{n/3}$ assignments to the variables of P_i.

Define a graph $G = (V, E)$ with $V = L_1 \cup L_2 \cup L_3$ and $E = \{(u,v) | u \in P_i, v \in P_j, i \neq j\}$. That is, G is a complete tripartite graph with $2^{n/3}$ nodes in each part, and each node in G corresponds to an assignment to $n/3$ variables in C. Weights are placed on the nodes and edges of G as follows. For a node v, define $w(v)$ to be the number of clauses that are satisfied by the partial assignment denoted by v. For each edge $\{u,v\}$, define $w(\{u,v\}) = -W_{uv}$, where W_{uv} is the number of clauses that are satisfied by *both* u and v.

Define the weight of a triangle in G to be the total sum of all weights and nodes in the triangle.

Claim 1 There is a one-to-one correspondence between the triangles of weight K in G and the variable assignments satisfying exactly K clauses in F.

Proof Let a be a variable assignment. Then there exist unique nodes $v_1 \in L_1, v_2 \in L_2$, and $v_3 \in L_3$ such that a is precisely the concatenation of v_1, v_2, v_3 as assignments. Moreover, any triple of nodes $v_1 \in L_1, v_2 \in L_2$, and $v_3 \in L_3$ corresponds to an assignment. Thus there is a one-to-one correspondence between triangles in G and assignments to F.

The number of clauses satisfied by an assignment is exactly the weight of its corresponding triangle. To see this, let $T_a = \{v_1, v_2, v_3\}$ be the triangle in G corresponding to

assignment a. Then

$$\begin{aligned} w(T_a) &= w(v_1) + w(v_2) + w(v_3) + w(\{v_1, v_2\}) \\ &\quad + w(\{v_2, v_3\}) + w(\{v_1, v_3\}) \\ &= \sum_{i=1}^{3} |\{c \in F | v_i \text{ satisfies } F\}| \\ &\quad - \sum_{i,j:i\neq j} |\{c \in F | v_i \text{ and } v_j \text{ satisfy } F\}| \\ &= |\{c \in F | a \text{ satisfies } F\}| , \end{aligned}$$

where the last equality follows from the inclusion-exclusion principle. □

Notice that the number of nodes in G is $3 \cdot 2^{n/3}$, and the absolute value of any node and edge weight is m. Therefore, running a MAX TRIANGLE algorithm on G, a solution to MAX 2-SAT is obtained in $O(f(m) \cdot g(3 \cdot 2^{n/3}))$, which is $O(f(m) \cdot g(2^{n/3}))$ since g is a polynomial. This completes the proof of Lemma 1. □

Next, a procedure is described for finding a maximum triangle faster than brute-force search, using fast matrix multiplication. Alon, Galil, and Margalit [1] (following Yuval [20]) showed that the distance product for matrices with entries drawn from $\mathbb{Z}[-W, W]$ can be computed using fast matrix multiplication as a subroutine.

Theorem 2 (Alon, Galil, Margalit [1]) *Let A and B be $n \times n$ matrices with entries from $\mathbb{Z}[-W, W] \cup \{\infty\}$. Then $A \otimes_d B$ can be computed in $O(Wn^{\omega} \log n)$ time.*

Proof (Sketch) One can replace ∞ entries in A and B with $2W + 1$ in the following. Define matrices A' and B', where

$$A'[i, j] = x^{3W - A[i,j]} , \quad B'[i, j] = x^{3W - B[i,j]} ,$$

and x is a variable. Let $C = A' \times B'$. Then

$$C[i, j] = \sum_{k=1}^{n} x^{6W - A[i,k] - B[k,j]} .$$

The next step is to pick a number x that makes it easy to determine, from the sum of arbitrary powers of x, the largest power of x appearing in the sum; this largest power immediately gives the minimum $A[i, k] + B[k, j]$. Each $C[i, j]$ is a polynomial in x with coefficients from $\mathbb{Z}[0, n]$. Suppose each $C[i, j]$ is evaluated at $x = (n + 1)$. Then each entry of $C[i, j]$ can be seen as an $(n + 1)$-ary number, and the position of this number's most significant digit gives the minimum $A[i, k] + B[k, j]$.

In summary, $A \otimes_d B$ can be computed by constructing

$$A'[i, j] = (n + 1)^{3W - A[i,j]} , \quad B'[i, j] = (n + 1)^{3W - B[i,j]}$$

in $O(W \log n)$ time per entry, computing $C = A' \times B'$ in $O(n^{\omega} \cdot (W \log n))$ time (as the sizes of the entries are $O(W \log n)$), then extracting the minimum from each entry of C, in $O(n^2 \cdot W \log n)$ time. Note if the minimum for an entry $C[i, j]$ is at least $2W + 1$, then $C[i, j] = \infty$. □

Using the fast distance product algorithm, one can solve MAX TRIANGLE faster than brute-force. The following is based on an algorithm by Itai and Rodeh [10] for detecting if an unweighted graph has a triangle in less than n^3 steps. The result can be generalized to *counting* the number of *k-cliques*, for arbitrary $k \geq 3$. (To keep the presentation simple, the counting result is omitted. Concerning the k-clique result, there is unfortunately no asymptotic runtime benefit from using a k-clique algorithm instead of a triangle algorithm, given the current best algorithms for these problems.)

Theorem 3 MAX TRIANGLE *can be solved in $O(W n^{\omega} \log n)$, for graphs with weights drawn from $\mathbb{Z}[-W, W]$.*

Proof First, it is shown that a weight function on nodes and edges can be converted into an equivalent weight function with weights on only edges. Let w be the weight function of G, and redefine the weights to be:

$$w'(\{u, v\}) = \frac{w(u) + w(v)}{2} + w(\{u, v\}) , \quad w'(u) = 0 .$$

Note the weight of a triangle is unchanged by this reduction.

The next step is to use a fast distance product to find a maximum weight triangle in an edge-weighted graph of n nodes. Construe the vertex set of G as the set $\{1, \ldots, n\}$. Define A to be the $n \times n$ matrix such that $A[i, j] = -w(\{i, j\})$ if there is an edge $\{i, j\}$, and $A[i, j] = \infty$ otherwise. The claim is that there is a triangle through node i of weight at least K if and only if $(A \otimes_d A \otimes_d A)[i, i] \leq -K$. This is because $(A \otimes_d A \otimes_d A)[i, i] \leq -K$ if and only if there are distinct j and k such that $\{i, j\}, \{j, k\}, \{k, i\}$ are edges and $A[i, j] + A[j, k] + A[k, i] \leq -K$, i. e., $w(\{i, j\}) + w(\{j, k\}) + w(\{k, i\}) \geq K$.

Therefore, by finding an i such that $(A \otimes_d A \otimes_d A)[i, i]$ is minimized, one obtains a node i contained in a maximum triangle. To obtain the actual triangle, check all m edges $\{j, k\}$ to see if $\{i, j, k\}$ is a triangle. □

Theorem 4 MAX 2-SAT *can be solved in $O(m \cdot 1.732^n)$ time.*

Proof Given a set of clauses C, apply the reduction from Lemma 1 to get a graph G with $O(2^{n/3})$ nodes and weights from $\mathbb{Z}[-m, m]$. Apply the algorithm of Theorem 3 to output a max triangle in G in $O(m \cdot 2^{\omega n/3} \log(2^{n/3})) =$

$O(m \cdot 1.732^n)$ time, using the $O(n^{2.376})$ matrix multiplication of Coppersmith and Winograd [4]. □

Applications

By modifying the graph construction, one can solve other problems in $O(1.732^n)$ time, such as MAX CUT, MINIMUM BISECTION, and SPARSEST CUT. In general, any constraint optimization problem for which each constraint has at most two variables can be solved faster using the above approach. For more details, see [17] and the recent survey by Woeginger [19]. Techniques similar to the above algorithm have also been used by Dorn [6] to speed up dynamic programming for some problems on planar graphs (and in general, graphs of bounded branchwidth).

Open Problems

- Improve the space usage of the above algorithm. Currently, $\Theta(2^{2n/3})$ space is needed. A very interesting open question is if there is a $O(1.99^n)$ time algorithm for MAX 2-SAT that uses only *polynomial* space. This question would have a positive answer if one could find an algorithm for solving the k-CLIQUE problem that uses polylogarithmic space and $n^{k-\delta}$ time for some $\delta > 0$ and $k \geq 3$.
- Find a faster-than-2^n algorithm for MAX 2-SAT that does not require fast matrix multiplication. The fast matrix multiplication algorithms have the unfortunate reputation of being impractical.
- Generalize the above algorithm to work for MAX k-SAT, where k is any positive integer. The current formulation would require one to give an efficient algorithm for finding a small hyperclique in a hypergraph. However, no general results are known for this problem. It is conjectured that for all $k \geq 2$, MAX k-SAT is in $\tilde{O}(2^{n(1-\frac{1}{k+1})})$ time, based on the conjecture that matrix multiplication is in $n^{2+o(1)}$ time [17].

Cross References

- ▶ All Pairs Shortest Paths via Matrix Multiplication
- ▶ Max Cut
- ▶ Minimum Bisection
- ▶ Sparsest Cut

Recommended Reading

1. Alon, N., Galil, Z., Margalit, O.: On the exponent of the all-pairs shortest path problem. J. Comput. Syst. Sci. **54**, 255–262 (1997)
2. Aspvall, B., Plass, M.F., Tarjan R.E.: A linear-time algorithm for testing the truth of certain quantified boolean formulas. Inf. Proc. Lett. **8**(3), 121–123 (1979)
3. Bansal, N., Raman, V.: Upper bounds for Max Sat: Further Improved. In: Proceedings of ISAAC. LNCS, vol. 1741, pp. 247–258. Springer, Berlin (1999)
4. Coppersmith, D., Winograd S.: Matrix Multiplication via Arithmetic Progressions. JSC **9**(3), 251–280 (1990)
5. Dantsin, E., Wolpert, A.: Max SAT for formulas with constant clause density can be solved faster than in $O(2^n)$ time. In: Proc. of the 9th International Conference on Theory and Applications of Satisfiability Testing. LNCS, vol. 4121, pp. 266–276. Springer, Berlin (2006)
6. Dorn, F.: Dynamic Programming and Fast Matrix Multiplication. In: Proceedings of 14th Annual European Symposium on Algorithms. LNCS, vol. 4168, pp. 280–291. Springer, Berlin (2006)
7. Garey, M., Johnson, D., Stockmeyer, L.: Some simplified NP-complete graph problems. Theor. Comput. Sci. **1**, 237–267 (1976)
8. Gramm, J., Niedermeier, R.: Faster exact solutions for Max2Sat. In: Proceedings of CIAC. LNCS, vol. 1767, pp. 174–186. Springer, Berlin (2000)
9. Hirsch, E.A.: A $2^{m/4}$-time Algorithm for Max 2-SAT: Corrected Version. Electronic Colloquium on Computational Complexity Report TR99-036 (2000)
10. Itai, A., Rodeh, M.: Finding a Minimum Circuit in a Graph. SIAM J. Comput. **7**(4), 413–423 (1978)
11. Kneis, J., Mölle, D., Richter, S., Rossmanith, P.: Algorithms Based on the Treewidth of Sparse Graphs. In: Proc. Workshop on Graph Theoretic Concepts in Computer Science. LNCS, vol. 3787, pp. 385–396. Springer, Berlin (2005)
12. Kojevnikov, A., Kulikov, A.S.: A New Approach to Proving Upper Bounds for Max 2-SAT. In: Proc. of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 11–17 (2006)
13. Mahajan, M., Raman, V.: Parameterizing above Guaranteed Values: MAXSAT and MAXCUT. J. Algorithms **31**(2), 335–354 (1999)
14. Niedermeier, R., Rossmanith, P.: New upper bounds for maximum satisfiability. J. Algorithms **26**, 63–88 (2000)
15. Scott, A., Sorkin, G.: Faster Algorithms for MAX CUT and MAX CSP, with Polynomial Expected Time for Sparse Instances. In: Proceedings of RANDOM-APPROX 2003. LNCS, vol. 2764, pp. 382–395. Springer, Berlin (2003)
16. Williams, R.: On Computing k-CNF Formula Properties. In: Theory and Applications of Satisfiability Testing. LNCS, vol. 2919, pp. 330–340. Springer, Berlin (2004)
17. Williams, R.: A new algorithm for optimal 2-constraint satisfaction and its implications. Theor. Comput. Sci. **348**(2–3), 357–365 (2005)
18. Woeginger, G.J.: Exact algorithms for NP-hard problems: A survey. In: Combinatorial Optimization – Eureka! You shrink! LNCS, vol. 2570, pp. 185–207. Springer, Berlin (2003)
19. Woeginger, G.J.: Space and time complexity of exact algorithms: some open problems. In: Proc. 1st Int. Workshop on Parameterized and Exact Computation (IWPEC 2004). LNCS, vol. 3162, pp. 281–290. Springer, Berlin (2004)
20. Yuval, G.: An Algorithm for Finding All Shortest Paths Using $N^{2.81}$ Infinite-Precision Multiplications. Inf. Process. Lett. **4**(6), 155–156 (1976)

Max Leaf Spanning Tree

2005; Estivill-Castro, Fellows, Langston, Rosamond

FRANCES ROSAMOND
Parameterized Complexity Research Unit,
University of Newcastle, Callaghan, NSW, Australia

Keywords and Synonyms

Maximum leaf spanning tree; Connected dominating set; Extremal structure

Problem Definition

The MAX LEAF SPANNING TREE problem asks us to find a spanning tree with at least k leaves in an undirected graph. The decision version of parameterized MAX LEAF SPANNING TREE is the following:

MAX LEAF SPANNING TREE
INPUT: A connected graph G, and an integer k.
PARAMETER: An integer k.
QUESTION: Does G have a spanning tree with at least k leaves?

The parameterized complexity of the nondeterministic polynomial-time complete MAX LEAF SPANNING TREE problem has been extensively studied [2,3,9,11] using a variety of kernelization, branching and other fixed-parameter tractable (FPT) techniques. The authors are the first to propose an extremal structure method for hard computational problems. The method, following in the sense of Grothendieck and in the spirit of the graph minors project of Robertson and Seymour, is that a mathematical project should unfold as a series of small steps in an overall trajectory that is described by the appropriate "mathematical machine." The authors are interested in statements of the type: Every connected graph on n vertices that satisfies a certain set of properties has a spanning tree with at least k leaves, and this spanning tree can be found in time $O(f(k) + n^c)$, where c is a constant (independent of k) and f is an arbitrary function.

In parameterized complexity, the value k is called the *parameter* and is used to capture some structure of the input or other aspect of the computational objective. For example, k might be the number of edges to be deleted in order to obtain a graph with no cycles, or k might be the number of DNA sequences to be aligned in an alignment, or k may be the maximum type-declaration nesting depth of a compiler, or $k = 1/\epsilon$ may be the parameterization in the analysis of approximation, or k might be a composite of several variables.

There are two important ways of comparing FPT algorithms, giving rise to two FPT *races*. In the "$f(k)$" race, the competition is to find ever more slowing growing parameter functions $f(k)$ governing the complexity of FPT algorithms. The "kernelization race" refers to the following lemma stating that a problem is in FPT if and only if the input can be preprocessed (*kernelized*) in "ordinary" polynomial time into an instance whose size is bounded by a function of k only.

Lemma 1 *A parameterized problem Π is in FPT if and only if there is a polynomial-time transformation (in both n and k) that takes (x, k) to (x', k') such that:*
(1) (x, k) is a yes-instance of Π if and only if (x', k') is a yes-instance of Π,
(2) $k' \leq k$, and
(3) $|x'| \leq g(k)$ for some fixed function g.

In the situation described by the lemma, say that we can *kernelize* to instances of size at most $g(k)$. Although the two races are often closely related, the result is not always the same. The current best FPT algorithm for MAX LEAF is due to Bonsma [1] (following the extremal structure approach outlined by the authors) with a running time of $O^*(8.12^k)$ to determine whether a graph G on n vertices has a spanning tree with at least k leaves; however the authors present the FPT algorithm with the smallest kernel size.

The authors list five independent deliverables associated to the extremal structure theory, and illustrate all of the objectives for the MAX LEAF problem. The five objectives are:

(A) Better FPT algorithms as a result of deeper structure theory, more powerful reduction rules associated with that structure theory, and stronger inductive proofs of improved kernelization bounds.
(B) Powerful preprocessing (*data reduction/kernelization*) rules and *combinations* of rules that can be used regardless of whether the parameter is small and that can be combined with other approaches, such as approximation and heuristics. These are usually easy to program.
(C) Gradients and transformation rules for local search heuristics.
(D) Polynomial-time approximation algorithms and performance bounds proved in a systematic way.
(E) Structure to exploit for solving other problems.

Key Results

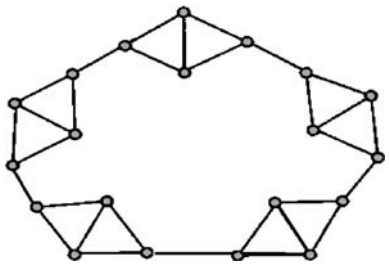
The key results are programmatic, providing a *method of extremal structure* as a systematic method for design-

ing FPT algorithms. The five interrelated objectives listed above are surveyed, and each is illustrated using the MAX LEAF SPANNING TREE problem.

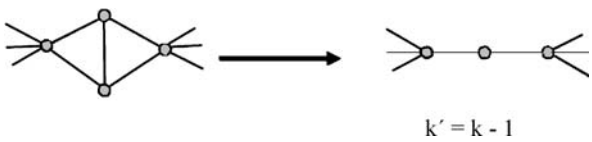
Objective A: FPT Algorithms

The objective here is to find polynomial-time preprocessing (*kernelization*) rules where $g(k)$ is as small as possible. This has a direct payoff in terms of program objective B.

Rephrased as a structure theory question, the crucial issue is: *What is the structure of graphs that do not have a subgraph with k leaves?* A graph theory result due to Kleitman and West shows that a graph of minimum degree at least 3, that excludes a k-leaf subgraph, has at most $4(k-3)$ vertices. Figure 1 shows that this is the best possible result for this hypothesis. However, investigating the structure using extremal methods reveals the need for the reduction rule of Fig. 2. About 20 different polynomial-time reduction rules (some much more complex and "global" in structure than the simple local reduction rule depicted) are sufficient to kernelize to a graph of minimum degree 2 having at most $3.5k$ vertices.



Max Leaf Spanning Tree, Figure 1
Reduction rules were developed in order to reduce this Kleitman–West graph structure



Max Leaf Spanning Tree, Figure 2
A reduction rule for the Kleitman–West graph

In general, an instance of a parameterized problem consists of a pair (x, k) and a "boundary" which is located by holding x fixed and varying k and regarding whether the outcome of the decision problem is *yes* or *no*. Of interest is the boundary when x is reduced. A typical boundary lemma looks like the following.

Lemma 2 *Suppose (G, k) is a reduced instance of* MAX LEAF, *with (G, k) a yes-instance and $(G, k+1)$ a no-instance. Then $|G| \leq ck$. (Here c is a small constant that becomes clarified during the investigation.)*

A proof of a boundary lemma is by minimum counterexample. A counterexample would be a graph such that (1) (G, k) is reduced, (2) (G, k) is a *yes*-instance of MAX LEAF, (3) $(G, k+1)$ is a *no*-instance, and (4) $|G| > ck$.

The proof of a boundary lemma unfolds gradually. Initially, it is not known what bound will eventually succeed and it is not known exactly what is meant by *reduced*. In the course of an attempted proof, these details are worked out. As the arguments unfold, structural situations will suggest new reduction rules. Strategic choices involved in a boundary lemma include:

(1) Determining the polarity of the boundary, and setting up the boundary lemma.
(2) Choosing a witness structure.
(3) Setting inductive priorities.
(4) Developing a series of structural claims that describe the situation at the boundary.
(5) Discovering reduction rules that can act in polynomial-time on relevant structural situations at the boundary.
(6) As the structure at the boundary becomes clear, filling in the blank regarding the kernelization bound.

The overall structure of the argument is "by minimum counterexample" according to the priorities established by choice 3, which generally make reference to choice 2. The proof proceeds by a series of small steps consisting of structural claims that lead to a detailed structural picture at the "boundary"—and thereby to the bound on the size of G that is the conclusion of the lemma. The complete proof assembles a series of claims made against the witness tree, various sets of vertices, and inductive priorities and sets up a master inequality leading to a proof by induction, and a $3.5k$ problem kernel.

Objective B: Polynomial-Time Preprocessing and Data-Reduction Routines

The authors have designed a table for tracing each possible *boundary state* for a possible solution. Examples are given that show the surprising power of cascading data-reduction rules on real input distributions and that describe a variety of mathematical phenomena relating to reduction rules. For example, some reduction rules, such as the *Kleitman–West dissolver rule* for MAX LEAF (Fig. 2), have a fixed "boundary size" (in this case 2), whereas crown-type reduction rules do not have a fixed boundary size.

Objective C: Gradients and Solution Transformations for Local Search

A generalization of the usual setup for local search is given, based on the mathematical power of the more complicated gradient in obtaining superior kernelization bounds. Idea 1 is that local search be conducted based on maintaining a "current witness structure" rather than a full solution (spanning tree). Idea 2 is to use the list of inductive priorities to define a "better solution" gradient for the local search.

Objective D: Polynomial-Time Approximation Algorithms

The polynomial-time extremal structure theory leads directly to a constant-factor p-time approximation algorithm for MAX LEAF. First, reduce G using the kernelization rules. The rules are approximation-preserving. Take *any* tree T (not necessarily spanning) in G. If all of the structural claims hold, then (by the boundary lemma arguments) the tree T must have at least n/c leaves for $c = 3.75$. Therefore, lifting T back along the reduction path, we obtain a c-approximation.

If at least one of the structural claims does not hold, then the tree T can be improved against one of the inductive priorities. Notice that each claim is proved by an argument that can be interpreted as a polynomial-time routine that improves T, when the claim is contradicted.

These consequences can be applied to the original T (and its successors) only a polynomial number of times (determined by the list of inductive priorities) until one arrives at a tree T' for which all of the various structural claims hold. At that point, we must have a c-approximate solution.

Objective E: Structure To Exploit in The Ecology of Complexity

The objective here is to understand how every input-governing problem parameter affects the complexity of every other problem. As a small example, consider Table 1 using the shorthand TW is TREEWIDTH, BW is BANDWIDTH, VC is VERTEX COVER, DS is DOMINATING SET, G is GENUS and ML is MAX LEAF. The entry in the second row and fourth column indicates that there is an *FPT* algorithm to optimally solve the DOMINATING SET problem for a graph G of bandwidth at most k. The entry in the fourth row and second column indicates that it is unknown whether BANDWIDTH can be solved optimally by an *FPT* algorithm when the parameter is a bound on the domination number of the input.

MAX LEAF applies to the last row of the table. For graphs of *max leaf number* bounded by k, the maximum size of an independent set can be computed in time $O^*(2.972^k)$ based on a reduction to a kernel of size at most $7k$. There is a practical payoff for using the output of one problem as the input to another.

Max Leaf Spanning Tree, Table 1
The complexity ecology of parameters

	TW	BW	VC	DS	G	ML
TW	*FPT*	*W*[1]-hard	*FPT*	*FPT*	?	*FPT*
BW	*FPT*	*W*[1]-hard	*FPT*	*FPT*	?	*FPT*
VC	*FPT*	?	*FPT*	*FPT*	?	*FPT*
DS	?	?	*W*[1]-hard	*W*[1]-hard	?	?
G	*W*[1]-hard	*W*[1]-hard	*W*[1]-hard	*W*[1]-hard	*FPT*	?
ML	*FPT*	?	*FPT*	*FPT*	*FPT*	?

Applications

The MAX LEAF SPANNING TREE problem has motivations in computer graphics for creating triangle strip representations for fast interactive rendering [5]. Other applications are found in the area of traffic grooming and network design, such as the design of optical networks and the utilization of wavelengths in order to minimize network cost, either in terms of the line-terminating equipment deployed or in terms of electronic switching [6]. The minimum-energy problem in wireless networks consists of finding a transmission radius vector for all stations in such a way that the total transmission power of the whole network is the least possible. A restricted version of this problem is equivalent to the MAX LEAF SPANNING TREE problem [7]. Finding spanning trees with many leaves is equivalent to finding small connected dominating sets and is also called the MINIMUM CONNECTED DOMINATING problem [13].

Open Problems

Branching Strategies

While extremal structure is in some sense *the right way* to design an FPT algorithm, this is not the only way. In particular, the recipe is silent on what to do with the kernel. An open problem is to find general strategies for employing "parameter-appropriate structure theory" in branching strategies for sophisticated problem kernel analysis.

Turing Kernelizability

The polynomial-time transformation of (x, k) to the simpler reduced instance (x', k') is a many:1 transformation. One can generalize the notion of many:1 reduction to Tur-

ing reduction. How should the quest for p-time extremal theory unfold under this "more generous" FPT?

Algorithmic Forms of The Boundary Lemma Approach

The hypothesis of the boundary lemma that (G, k) is a *yes*-instance implies that there exists a witness structure to this fact. There is no assumption that one has algorithmic access to this structure, and when reduction rules are discovered, these have to be transformations that can be applied to (G, k) and a structure that can be discovered in (G, k) in polynomial time. In other words, reduction rules cannot be *defined* with respect to the witness structure. Is it possible to describe more general approaches to kernelization where the witness structure used in the proof of the boundary lemma is polynomial-time computable, and this structure provides a conditional context for some reduction rules? How would this change the extremal method recipe?

Problem Annotation

One might consider a generalized MAX LEAF problem where vertices and edges have various annotations as to whether they *must* be leaves (or internal vertices) in a solution, etc. Such a generalized form of the problem would generally be expected to be "more difficult" than the vanilla form of the problem. However, several of the "best known" FPT algorithms for various problems, are based on these generalized, annotated forms of the problems. Examples include PLANAR DOMINATING SET and FEEDBACK VERTEX SET [4]. Should annotation be part of the recipe for the best possible polynomial-time kernelization?

Cross References

- ► Connected Dominating Set
- ► Data Reduction for Domination in Graphs

Recommended Reading

1. Bonsma, P.: Spanning trees with many leaves: new extremal results and an improved FPT algorithm. Memorandum Department of Applied Mathematics, vol. 1793, University of Twente, Enschede (2006)
2. Bonsma, P., Brueggemann, T., Woeginger, G.: A faster FPT algorithm for finding spanning trees with many leaves. Proceedings of MFCS 2003. Lecture Notes in Computer Science, vol. 2747, pp. 259–268. Springer, Berlin (2003)
3. Downey, R.G., Fellows, M.R.: Parameterized complexity. Monographs in Computer Science. Springer, New York (1999)
4. Dehne, F., Fellows, M., Langston, M., Rosamond, F., Stevens, K.: An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. Proceedings COCOON 2005. Lecture Notes in Computer Science, vol. 3595, pp. 859–869. Springer, Berlin (2005)
5. Diaz-Gutierrez, P., Bhushan, A., Gopi, M., Pajarola, R.: Single-strips for fast interactive rendering. J. Vis. Comput. **22**(6), 372–386 (2006)
6. Dutta, R., Savage, C.: A Note on the Complexity of Converter Placement Supporting Broadcast in WDM Optical Networks. In: Proceedings of the International Conference on Telecommunication Systems-Modeling and Analysis, Dallas, November 2005 ISBN: 0-9716253-3-6 pp. 23–31. American Telecommunication Systems Management Association, Nashville
7. Egecioglu, O., Gonzalez, T.: Minimum-energy Broadcast in Simple Graphs with Limited Node Power. In: Proc. IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2001), Anaheim, August 2001 pp. 334–338
8. Estivill-Castro, V., Fellows, M.R., Langston, M.A., Rosamond, F.A.: FPT is P-time extremal structure I. In: Algorithms and complexity in Durham 2005. Texts in Algorithmics, vol. 4, pp. 1–41. Kings College Publications, London (2005)
9. Fellows, M., Langston, M.: On well-partial-order theory and its applications to combinatorial problems of VLSI design. SIAM J. Discret. Math. **5**, 117–126 (1992)
10. Fellows, M.: Blow-ups, win/win's and crown rules: some new directions in FPT. In: Proceedings of the 29th Workshop on Graph Theoretic Concepts in Computer Science (WG 2003). Lecture Notes in Computer Science, vol. 2880, pp. 1–12. Springer, Berlin (2003)
11. Fellows, M., McCartin, C., Rosamond, F., Stege, U.: Coordinatized kernels and catalytic reductions: an improved FPT algorithm for max leaf spanning tree and other problems. In: Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science (FST-TCS 2000). Lecture Notes in Theoretical Computer Science 1974, pp. 240–251. Springer, Berlin (2000)
12. Kleitman, D.J., West, D.B.: Spanning trees with many leaves. SIAM J. Discret. Math. **4**, 99–106 (1991)
13. Kouider, M., Vestergaard, P.D.: Generalized connected domination in graphs. Discret. Math. Theor. Comput. Sci. (DMTCS) **8**, 57–64 (2006)
14. Lu, H.-I., Ravi, R.: Approximating maximum leaf spanning trees in almost linear time. J. Algorithm **29**, 132–141 (1998)
15. Niedermeier, R.: Invitation to Fixed Parameter Algorithms. Lecture Series in Mathematics and Its Applications, Oxford University Press, Oxford (2006)
16. Prieto-Rodriguez, E.: Systematic kernelization in FPT algorithm design. Dissertation, School of Electrical Engineering and Computer Science, University of Newcastle, Australia (2005)
17. Solis-Oba, R.: 2-approximation algorithm for finding a spanning tree with the maximum number of leaves. In: Proceedings of the 6th Annual European Symposium on Algorithms (ESA'98). Lecture Notes in Computer Science, vol. 1461, pp. 441–452. Springer, Berlin (1998)

Metrical Task Systems

1992; Borodin, Linial, Saks

MANOR MENDEL
Department of Mathematics and Computer Science, The Open University of Israel, Raanana, Israel

Keywords and Synonyms

MTS

Problem Definition

Metrical task systems (MTS), introduced by Borodin, Linial, and Saks [5], is a cost minimization problem defined on a metric space (X, d_X) and informally described as follows: A given *system* has a set of internal states X. The aim of the system is to serve a given sequence of tasks. The servicing of each task has a certain cost that depends on the task and the state of the system. The system may switch states before serving the task, and the total cost for servicing the task is the sum of the service cost of the task in the new state and the distance between the states in a metric space defined on the set of states. Following Manasse, McGeoch, and Sleator [11], an extended model is considered here, in which the set of allowable tasks may be restricted.

Notation

Let T^* denote the set of finite sequences of elements from a set T. For $x, y \in T^*$, $x \circ y$ is the concatenation of the sequences x and y, and $|x|$ is the length of the sequence x.

Definition 1 (Metrical Task System) Fix a metric space (X, d_X). Let $\Gamma = \{(r_x)_{x\in X} : \forall x \in X,\ r(x) \in [0, \infty]\}$ be the set of all possible tasks. Let $T \subseteq \Gamma$ be a subset of tasks, called *allowable tasks*.
MTS$((X, d_X), T, a_0 \in X)$:
INPUT: A finite sequence of tasks $\tau = (\tau_1, \ldots, \tau_m) \in T^*$.
OUTPUT: A sequence of points $a = (a_1, \ldots, a_m) \in X^*$, $|a| = |\tau|$.
OBJECTIVE: minimize

$$\mathsf{cost}(\tau, a) = \sum_{i=1}^{m} (d_X(a_{i-1}, a_i) + \tau_i(a_i)).$$

When $T = \Gamma$, the MTS problem is called *general*.

When X is finite and the task sequence $\tau \in T^*$ is given in advance, a dynamic programming algorithm can compute an optimal solution in space $O(|X|)$ and time $O(|\tau| \cdot |X|)$. MTS, however, is most interesting in an online setting, where the system must respond to a task τ_i with a state $a_i \in X$ without knowing the future tasks in τ. Formally,

Definition 2 (Online algorithms for MTS) A deterministic algorithm for a MTS$((X, d_X), T, a_0)$ is a mapping $S : T^* \to X^*$ such that for every $\tau \in T$, $|S(\tau)| = |\tau|$. A deterministic algorithm $S : T^* \to X^*$ is called *online* if for every $\tau, \sigma \in T^*$, there exists $a \in X^*$, $|a| = |\sigma|$ such that $S(\tau \circ \sigma) = S(\tau) \circ a$. A randomized online algorithm is a probability distribution over deterministic online algorithms.

Online algorithms for MTS are evaluated using *(asymptotic) competitive analysis*, which is, roughly speaking, the worst ratio of the algorithm's cost to the optimal cost taken over all possible task sequences.

Definition 3 A randomized online algorithm R for MTS$((X, d_X), a_0)$ is called c-competitive (against oblivious adversaries) if there exists $b = b(X) \in \mathbb{R}$ such that for any task sequence $\tau \in T^*$, and any point sequence $a \in X^*$, $|a| = |\tau|$,

$$\mathbb{E}[\mathsf{cost}(\tau, R(\tau))] \le c \cdot \mathsf{cost}(\tau, a) + b,$$

where the expectation is taken over the distribution R.

The competitive ratio of an online algorithm R is the infimum over $c \ge 1$ for which R is c-competitive. The deterministic [respectively, randomized] competitive ratio of MTS$((X, d_X), T, a_0)$ is the infimum over the competitive ratios of all deterministic [respectively, randomized] online algorithms for this problem. Note that because of the existential quantifier over b, the asymptotic competitive ratio (both randomized and deterministic) of a MTS$((X, d_X), T, a_0)$ is independent of a_0, and it can therefore be dropped from the notation.

Key Results

Theorem 1 ([5]) *The deterministic competitive ratio of the general MTS problem on any n-point metric space is $2n - 1$.*

In contrast to the deterministic case, the understanding of randomized algorithms for general MTS is not complete, and generally no sharp bounds such as Theorem 1 are known.

Theorem 2 ([5,10]) *The randomized competitive ratio of the general MTS problem on n-point uniform space (where all distances are equal) is at least $H_n = \sum_{i=1}^{n-1} i^{-1}$, and at most $(1 + o(1))H_n$.*

The best bounds currently known for general n-point metrics are proved in two steps: First the given metric is approximated by an *ultrametric*, and then a bound on the competitive ratio of general MTS on ultrametrics is proved.

Theorem 3 ([8,9]) *For any n-point metric space (X, d_X), there exists an $O(\log^2 n \log\log n)$ competitive randomized algorithm for the general MTS on (X, d_X).*

The metric approximation component in the proof of Theorem 3 is called *probabilistic embedding*. An optimal $O(\log n)$ probabilistic embedding is shown by

Fakcheroenphol, Rao and Talwar before [8] improving on results by Alon, Karp, Peleg, and West and by Bartal, where this notion was invented. A different type of metric approximation with better bounds for metrics of low *aspect ratio* is given in [3].

Fiat and Mendel [9] show a $O(\log n \log\log n)$ competitive algorithm for n-point ultrametrics, improving (and using) a result of Bartal, Blum, Burch, and Tomkins [1], where the first poly-logarithmic (or even sublinear) competitive randomized algorithm for general MTS on general metric spaces is presented.

Theorem 4 ([2,12]) *For any n-point metric space (X, d_X), the randomized competitive ratio of the general MTS on (X, d_X) is at least $\Omega(\log n/\log\log n)$.*

The metric approximation component in the proof of Theorem 4 is called *Ramsey subsets*. It was first used in this context by Karloff, Rabani, and Ravid, later improved by Blum, Karloff, Rabani and Saks, and Bartal, Bollobás, and Mendel [2]. A tight result on Ramsey subsets is proved by Bartal, Linial, Mendel, and Naor. For a simpler (and stronger) proof, see [12].

A lower bound of $\Omega(\log n/\log\log n)$ on the competitive ratio of any randomized algorithm for general MTS on n-point ultrametrics is proved in [2], improving previous results of Karloff, Rabani, and Ravid, and Blum, Karloff, Rabani and Saks.

The last theorem is the only one not concerning general MTSs.

Theorem 5 ([6]) *It is PSPACE hard to determine the competitive ratio of a given MTS instance $((X, d_X), a_0 \in X, T)$, even when d_X is the uniform metric. On the other hand, when d_X is uniform, there is a polynomial time deterministic online algorithm for MTS$((X, d_X), a_0 \in X, T)$ whose competitive ratio is $O(\log |X|)$ times the deterministic competitive ratio of the MTS$((X, d_X), a_0, T)$. Here it is assumed that the instance $((X, d_X), a_0, T)$ is given explicitly.*

Applications

Metrical task systems were introduced as an abstraction for online computation, they generalize many concrete online problems such as paging, weighted caching, k-server, and list update. Historically, it served as an indicator for a general theory of competitive online computation.

The main technical contribution of the MTS model is the development of the work function algorithm used to prove the upper bound in Theorem 1. This algorithm was later analyzed by Koutsoupias and Papadimitriou in the context of the k-server problem, and was shown to be $2k-1$ competitive. Furthermore, although the MTS model generalizes the k-server problem, the general MTS problem on the n-point metric is essentially equivalent to the $(n-1)$-server problem on the same metric [2]. Hence, lower bounds on the competitive ratio of general MTS imply lower bounds for the k-server problem, and algorithms for general MTS may constitute a first step in devising an algorithm for the k-server problem, as is the case with the work function algorithm.

The metric approximations used in Theorem 3, and Theorem 4 have found other algorithmic applications.

Open Problems

There is still an obvious gap between the upper bound and lower bound known on the randomized competitive ratio of general MTS on general finite metrics. It is known that, contrary to the deterministic case, the randomized competitive ratio is *not* constant across all metric spaces of the same size. However, in those cases where exact bounds are known, the competitive ratio is $\Theta(\log n)$. An obvious conjecture is that the randomized competitive is $\Theta(\log n)$ for any n-point metric. Arguably, the simplest classes of metric spaces for which no upper bound on the randomized competitive ratio better than $O(\log^2 n)$ is known, are paths and cycles.

Also lacking is a "middle theory" for MTS. On the one hand, general MTS are understood fairly well. On the other hand, specialized MTS such as list update, deterministic k-server algorithms, and deterministic weighted-caching, are also understood fairly well, and have a much better competitive ratio than the corresponding general MTS. What may be missing are "in between" models of MTS that can explain the low competitive ratios for some of the concrete online problems mentioned above.

It would be also nice to strengthen Theorem 5, and obtain a polynomial time deterministic online algorithm whose competitive ratio on any MTS instance on *any* n-point metric space is at most poly-log(n) times the deterministic competitive ratio of that MTS instance.

Cross References

- ▶ Algorithm DC-Tree for k Servers on Trees
- ▶ Approximating Metric Spaces by Tree Metrics
- ▶ Online List Update
- ▶ Online Paging and Caching
- ▶ Paging
- ▶ Ski Rental Problem
- ▶ Work-Function Algorithm for k Servers

Recommended Reading

1. Bartal, Y., Blum, A., Burch, C., Tomkins, A.: A polylog()-competitive algorithm for metrical task systems. In: Proceedings of the 29th annual ACM Symposium on the Theory of Computing, pp. 711–719. ACM, New York (1997)
2. Bartal, Y., Bollobás, B., Mendel, M.: Ramsey-type theorems for metric spaces with applications to online problems. J. Comput. Syst. Sci. **72**, 890–921 (2006)
3. Bartal, Y., Mendel, M.: Multiembedding of metric spaces. SIAM J. Comput. **34**, 248–259 (2004)
4. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press, Cambridge, UK (1998)
5. Borodin, A., Linial, N., Saks, M.E.: An optimal on-line algorithm for metrical task system. J. ACM **39**, 745–763 (1992)
6. Burley, W.R., Irani, S.: On algorithm design for metrical task systems. Algorithmica **18**, 461–485 (1997)
7. Chrobak, M., Larmore, L.L.: Metrical task systems, the server problem and the work function algorithm. In: Fiat, A., Woeginger, G.J. (eds.) Online Algorithms. The State of the Art. LNCS, vol. 1442, ch. 4, pp. 74–96. Springer, London (1998)
8. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. J. Comput. Syst. Sci. **69**, 485–497 (2004)
9. Fiat, A., Mendel, M.: Better algorithms for unfair metrical task systems and applications. SIAM J. Comput. **32**, 1403–1422 (2003)
10. Irani, S., Seiden, S.S.: Randomized algorithms for metrical task systems. Theor. Comput. Sci. **194**, 163–182 (1998)
11. Manasse, M.S., McGeoch, L.A., Sleator, D.D.: Competitive algorithms for server problems. J. Algorithms **11**, 208–230 (1990)
12. Mendel, M., Naor, A.: Ramsey partitions and proximity data structures. J. Eur. Math. Soc. **9**(2), 253–275 (2007)

Metric TSP

1976; Christofides

MARKUS BLÄSER
Department of Computer Science, Saarland University, Saarbrücken, Germany

Keywords and Synonyms

Metric traveling salesman problem; Metric traveling salesperson problem

Problem Definition

The *Traveling Salesman Problem (TSP)* is the following optimization problem:

Input: A complete loopless undirected graph $G = (V, E, w)$ with a weight function $w\colon E \to \mathbb{Q}_{\geq 0}$ that assigns to each edge a non-negative weight.

Feasible solutions: All Hamiltonian tours, i. e, the subgraphs H of G that are connected, and each node in them that has degree two.

Objective function: The weight function $w(H) = \sum_{e \in H} w(e)$ of the tour.

Goal: Minimization.

The TSP is an NP-hard optimization problem. This means that a polynomial time algorithm for the TSP does not exist unless P = NP. One way out of this dilemma is provided by *approximation algorithms*. A polynomial time algorithm for the TSP is called an α-approximation algorithm if the tour H produced by the algorithm fulfills $w(H) \leq \alpha \cdot \mathrm{OPT}(G)$. Here $\mathrm{OPT}(G)$ is the weight of a minimum weight tour of G. If G is clear from the context, one just writes OPT. An α-approximation algorithm always produces a feasible solution whose objective value is at most a factor of α away from the optimum value. α is also called the approximation factor or performance guarantee. α does not need to be a constant; it can be a function that depends on the size of the instance or the number of nodes n.

If there exists a polynomial time approximation algorithm for the TSP that achieves an exponential approximation factor in n, then P = NP [6]. Therefore, one has to look at restricted instances. The most natural restriction is the *triangle inequality*, that means,

$$w(u, v) \leq w(u, x) + w(x, v) \quad \text{for all } u, v, x \in V\,.$$

The corresponding problem is called the *Metric TSP*. For the Metric TSP, approximation algorithms that achieve a constant approximation factor exist. Note that for the Metric TSP, it is sufficient to find a tour that visits each vertex *at least* once: Given such a tour, we can find a Hamiltonian tour of no larger weight by skipping every vertex that we already visited. By the triangle inequality, the new tour cannot get heavier.

Key Results

A simple 2-approximation algorithm for the Metric TSP is the *tree doubling algorithm*. It uses minimum spanning trees to compute Hamiltonian tours. A *spanning tree* T of a graph $G = (V, E, w)$ is a connected acyclic subgraph of G that contains each node of V. The weight $w(T)$ of such a spanning tree is the sum of the weights of the edges in it, i. e., $w(T) = \sum_{e \in T} w(e)$. A spanning tree is called a minimum spanning tree if its weight is minimum among all spanning trees of G. One can efficiently compute a minimum spanning tree, for instance via Prim's or Kruskal's algorithm, see e. g. [5].

The tree doubling algorithm seems to be folklore. The next lemma is the key for proving the upper bound on the approximation performance of the tree doubling algorithm.

Input: a complete loopless edge weighted undirected graph $G = (V, E, w)$ with weight function $w : E \to \mathbb{Q}_{\geq 0}$ that fulfills the triangle inequality
Output: a Hamiltonian tour of G that is a 2" approximation
1: Compute a minimum spanning tree T of G.
2: Duplicate each edge of T and obtain a Eulerian multigraph T'.
3: Compute a Eulerian tour of T' (for instance via a depth first search in T). Whenever a node is visited in the Eulerian tour that was already visited, this node is skipped and one proceeds with the next unvisited node along the Eulerian cycle. (This process is called *shortcutting*.) Return the resulting Hamiltonian tour H.

Metric TSP, Algorithm 1
Tree doubling algorithm

Lemma 1 *Let T be a minimum spanning tree of $G = (V, E, w)$. Then $w(T) \leq$ OPT.*

Proof If one deletes any edge of a Hamiltonian tour of G, one gets a spanning tree of G. □

Theorem 2 *Algorithm 1 always returns a Hamiltonian tour whose weight is at most twice the weight of an optimum tour. Its running time is polynomial.*

Proof By Lemma 1, $w(T) \leq$ OPT. Since one duplicates each edge of T, the weight of T' equals $w(T') = 2w(T) \leq$ 2OPT. When taking shortcuts in step 3, a path in T' is replaced by a single edge. By the triangle inequality, the sum of the weights of the edges in such a path is at least the weight of the edge it is replaced by. (Here, the algorithm breaks down for arbitrary weight functions.) Thus $w(H) \leq w(T')$. This proves the claim about the approximation performance.

The running time is dominated by the time needed to compute a minimum spanning tree. This is clearly polynomial. □

Christofides' algorithm (Algorithm 2) is a clever refinement of the tree doubling algorithm. It first computes a minimum spanning tree. On the nodes that have an odd degree in T, it then computes a minimum weight perfect matching. A matching M of G is called a matching on $U \subseteq V$ if all edges of M consist of two nodes from U. Such a matching is called *perfect* if every node of U is incident with an edge of M.

Lemma 3 *Let $U \subseteq V, \#U$ even. Let M be a minimum weight perfect matching on U. Then $w(M) \leq$ OPT/2.*

Input: a complete loopless edge weighted undirected graph $G = (V, E, w)$ with weight function $w : E \to \mathbb{Q}_{\geq 0}$ that fulfills the triangle inequality
Output: a Hamiltonian tour of G that is a 3/2" approximation
1: Compute a minimum spanning tree T of G.
2: Let $U \subseteq V$ be the set of all nodes that have odd degree in T. In G, compute a minimum weight perfect matching M on U.
3: Compute a Eulerian tour of $T \cup M$ (considered as a multigraph).
4: Take shortcuts in this Eulerian tour to a Hamiltonian tour H.

Metric TSP, Algorithm 2
Christofides' algorithm

Proof Let H be an optimum Hamiltonian tour of G. One takes shortcuts in H to get a tour H' on $G|_U$ as follows: H induces a permutation of the nodes in U, namely the order in which the nodes are visited by H. One connects the nodes of U in the order given by the permutation. To every edge of H' corresponds a path in H connecting the two nodes of this edge. By the triangle inequality, $w(H') \leq w(H)$. Since $\#U$ is even, H' is the union of two matchings. The lighter one of these two has a weight of at most $w(H')/2 \leq$ OPT/2. □

One can compute a minimum weight perfect matching in time $O(n^3)$, see for instance [5].

Theorem 4 *Algorithm 2 is a 3/2-approximation algorithm with polynomial running time.*

Proof First observe that the number of odd degree nodes of the spanning tree is even, since the sum of the degrees of all nodes equals $2(n-1)$, which is even. Thus a perfect matching on U exists. The weight of the Eulerian tour is obviously $w(T) + w(M)$. By Lemma 1, $w(T) \leq$ OPT. By Lemma 3, $w(M) \leq$ OPT/2. The weight $w(H)$ of the computed tour H is at most the weight of the Eulerian tour by the triangle inequality, i. e., $w(H) \leq \frac{3}{2}$OPT. Thus the algorithm is a 3/2-approximation algorithm. Its running time is $O(n^3)$. □

Applications

Experimental analysis shows that Christofides' algorithm itself deviates by 10% to 15% from the optimum tour [3]. However, it can serve as a good starting tour for other heuristics like the Lin–Kernigham heuristic.

Metric TSP, Figure 1
A tight example for Christofides' algorithm. There are $2n+1$ nodes. *Solid edges* have a weight of one, *dashed ones* have a weight of $1+\epsilon$

Open Problems

The analysis of Algorithm 2 is tight; an example is the metric completion of the graph depicted in Fig. 1. The unique minimum spanning tree consists of all solid edges. It has only two nodes of odd degree. The edge between these two nodes has weight $(1+\epsilon)(n+1)$. No shortcuts are needed, and the weight of the tour produced by the algorithm is $\approx 3n$. An optimum tour consists of all dashed edges plus the leftmost and rightmost solid edge. The weight of this tour is $(2n-1)(1+\epsilon)+2 \approx 2n$.

The question whether there is an approximation algorithm with a better performance guarantee is a major open problem in the theory of approximation algorithms.

Held and Karp [2] design an LP based algorithm that computes a lower bound for the weight of an optimum TSP tour. It is conjectured that the weight of an optimum TSP tour is at most a factor of 4/3 larger than this lower bound, but this conjecture is unproven for more than three decades. An algorithmic proof of this conjecture would yield an 4/3-approximation algorithm for the Metric TSP.

Experimental Results

See e. g. [3], where a deviation of 10% to 15% of the optimum (more precisely of the Held–Karp bound) is reported for various sorts of instances.

Data Sets

The webpage of the 8th DIMACS implementation challenge, www.research.att.com/~dsj/chtsp/, contains a lot of instances.

Cross References

► Minimum Spanning Trees

Recommended Reading

Christofides never published his algorithm. It is usually cited as one of two technical reports from Carnegie Mellon University, TR 388 of the Graduate School of Industrial Administration (now Tepper School of Business) and CS-93-13. None of them seem to be available at Carnegie Mellon University anymore [Frank Balbach, personal communication, 2006]. A one-page abstract was published in a conference record. But his algorithm quickly found his way into standard textbooks on algorithm theory, see [7] for a recent one.

1. Christofides, N.: Worst case analysis of a new heuristic for the traveling salesman problem, Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, (1976). Also: Carnegie-Mellon University Technical Report CS-93-13, 1976. Abstract in Traub, J.F. (ed.) Symposium on new directions and recent results in algorithms and complexity, pp. 441. Academic Press, New York (1976)
2. Held, M., Karp, R.M.: The traveling salesman problem and minimum spanning trees. Oper. Res. **18**, 1138–1162 (1970)
3. Johnson, D.S., McGeoch, L.A.: Experimental analysis of heuristics for the STSP. In: Gutin, G., Punnen, A.P. (eds.) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht (2002)
4. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (eds.): The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization. Wiley, Chichester (1985)
5. Papadimitriou, C., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs (1982)
6. Sahni, S., Gonzalez, T.: P-complete approximation problems. J. ACM **23**, 555–565 (1976)
7. Vazirani, V.V.: Approximation Algorithms. Springer, Berlin (2001)
8. Traveling Salesman Problem. www.tsp.gatech.edu (2006). Accessed 28 Mar 2008

Minimum Bisection

1999; Feige, Krauthgamer

Robert Krauthgamer[1,2]
[1] Weizmann Institute of Science, Rehovot, Israel
[2] IBM Almaden Research Center, San Jose, CA, USA

Keywords and Synonyms

Graph bisection

Problem Definition

Overview

Minimum bisection is a basic representative of a family of discrete optimization problems dealing with partitioning the vertices of an input graph. Typically, one wishes to minimize the number of edges going across between the different pieces, while keeping some control on the partition, say by restricting the number of pieces and/or their size. (This description corresponds to an edge-cut of the graph; other variants correspond to a vertex-cut with similar restrictions.) In the minimum bisection problem, the

goal is to partition the vertices of an input graph into two equal-size sets, such that the number of edges connecting the two sets is as small as possible.

In a seminal paper in 1988, Leighton and Rao [14] devised for MINIMUM-BISECTION a logarithmic-factor bicriteria approximation algorithm.[1] Their algorithm has found numerous applications, but the question of finding a true approximation with a similar factor remained open for over a decade later. In 1999, Feige and Krauthgamer [6] devised the first polynomial-time algorithm that approximates this problem within a factor that is polylogarithmic (in the graph size).

Cuts and Bisections

Let $G = (V, E)$ be an undirected graph with $n = |V|$ vertices, and assume for simplicity that n is even. For a subset S of the vertices, let $\bar{S} = V \setminus S$. The *cut* (also known as *cutset*) $(S, \bar{S})$ is defined as the set of all edges with one endpoint in S and one endpoint in $\bar{S}$. These edges are said to *cross* the cut, and the two sets S and $\bar{S}$ are called the two *sides* of the cut.

Assume henceforth that G has nonnegative edge-weights. (In the unweighted version, every edge has a unit weight.) The *cost* of a cut $(S, \bar{S})$ is then defined to be the total edge-weight of all the edges crossing the cut.

A cut $(S, \bar{S})$ is called a *bisection* of G if its two sides have equal cardinality, namely $|S| = |\bar{S}| = n/2$. Let $b(G)$ denote the minimum cost of a bisection of G.

Problem 1 (MINIMUM-BISECTION)
Input: *An undirected graph G with nonnegative edge-weights.*
Output: *A bisection $(S, \bar{S})$ of G that has minimum cost.*

This definition has a crucial difference from the classical MINIMUM-CUT problem (see e. g. [10] and references therein), namely, there is a restriction on the sizes of the two sides of the cut. As it turns out, MINIMUM-BISECTION is NP-hard (see [9]), while MINIMUM-CUT can be solved in polynomial time.

Balanced Cuts and Edge Separators

The above rather basic definition of minimum bisection can be extended in several ways. Specifically, one may require only an upper bound on the size of each side. For $0 < \beta < 1$, a cut $(S, \bar{S})$ is called β-balanced if $\max\{|S|, |\bar{S}|\} \leq \beta n$. Note the latter requirement implies $\min\{|S|, |\bar{S}|\} \geq (1 - \beta)n$. In this terminology, a bisection is a 1/2-balanced cut.

Problem 2 (β-BALANCED-CUT)
Input: *An undirected graph G with nonnegative edge-weights.*
Output: *A β-balanced cut $(S, \bar{S})$ of G with $\max\{|S|, |\bar{S}|\} \leq \beta n$, that has cost as small as possible.*

The special case of $\beta = 2/3$ is commonly refered to as the EDGE-SEPARATOR problem.

In general, the sizes of the two sides may be specified in advance arbitrarily (rather than being equal); in this case the input contains a number k, and the goal is to find a cut $(S, \bar{S})$ such that $|S| = k$. One may also wish to divide the graph into more than two pieces of equal size and then the input contains a number $r \geq 2$, or alternatively, to divide the graph into r pieces of whose sizes are $k_1, \ldots, k_r$, where the numbers k_i are prescribed in the input; in either case, the goal is to minimize the number of edges crossing between different pieces.

Problem 3 (PRESCRIBED-PARTITION)
Input: *An undirected graph $G = (V, E)$ with nonnegative edge-weights, and integers $k_1, \ldots, k_r$ such that $\sum_i k_i = |V|$.*
Output: *A partition $V = V_1 \cup \cdots \cup V_r$ of G with $|V_i| = k_i$ for all i, such that the total edge-weight of edges whose endpoints lie in different sets V_i is as small as possible.*

Key Results

The main result of Feige and Krauthgamer [6] is an approximation algorithm for MINIMUM-BISECTION. The approximation factor they originally claimed is $O(\log^2 n)$, because it used the algorithm of Leighton and Rao [14]; however, by using instead the algorithm of [2], the factor immediately improves to $O(\log^{1.5} n)$.

Theorem 1 Minimum-Bisection *can be approximated in polynomial time within $O(\log^{1.5} n)$ factor. Specifically, the algorithm produces for an input graph G a bisection $(S, \bar{S})$ whose cost is at most $O(\log^{1.5} n) \cdot b(G)$.*

The algorithm immediately extends to similar results for related and/or more general problems that are defined above.

Theorem 2 β-Balanced-Cut *(and in particular* Edge-Separator*) can be approximated in polynomial time within $O(\log^{1.5} n)$ factor.*

Theorem 3 Prescribed-Partition *can be approximated in time $n^{O(r)}$ to within $O(\log^{1.5} n)$ factor.*

For all three problems above, the approximation ratio improves to $O(\log n)$ for the family of graphs excluding

[1] A bicriteria approximation algorithm partitions the vertices into two sets each containing at most 2/3 of the vertices, and its value, i. e. the number of edges connecting the two sets, is compared against that of the best partition into equal-size sets.

a fixed minor (which includes in particular planar graphs). For simplicity, this result is stated for Minimum-Bisection.

Theorem 4 *In graphs excluding a fixed graph as a minor (e. g., planar graphs), the problems (i)* Minimum-Bisection, *(ii)* β-Balanced-Cut, *and (iii)* Prescribed-Partition *with fixed r can all be approximated in polynomial time within factor* $O(\log n)$.

It should be noted that all these results can be generalized further, including vertex-weights and terminals-vertices ($s-t$ pairs), see [Sect. 5 in 6].

Related Work

A bicriteria approximation algorithm for β-balanced cut returns a cut that is β'-balanced for a predetermined $\beta' > \beta$. For bisection, for example, $\beta = 1/2$ and typically $\beta' = 2/3$.

The algorithms in the above theorems use (in a black-box manner) an approximation algorithm for a problem called minimum quotient-cuts (or equivalently, sparsest-cut with uniform-demands). For this problem, the best approximation currently known is $O(\sqrt{\log n})$ for general graphs due to Arora, Rao, and Vazirani [2], and $O(1)$ for graphs excluding a fixed minor due to Klein, Plotkin, and Rao [13]. These approximation algorithms for minimum quotient-cuts immediately give a polynomial time bicriteria approximation (sometimes called pseudo-approximation) for MINIMUM-BISECTION. For example, in general graphs the algorithm is guaranteed to produce a 2/3-balanced cut whose cost is at most $O(\sqrt{\log n}) \cdot b(G)$. Note however that a 2/3-balanced cut does not provide a good approximation for the value of $b(G)$. For instance, if G consists of three disjoint cliques of equal size, an optimal 2/3-balanced cut has no edges, whereas $b(G) = \Omega(n^2)$. For additional related work, including approximation algorithms for dense graphs, for directed graphs, and for other graph partitioning problems, see [Sect. 1 in 6] and the references therein.

Applications

One major motivation for MINIMUM-BISECTION, and graph partitioning in general, is a divide-and-conquer approach to solving a variety of optimization problems, especially in graphs, see e. g. [15,16]. In fact, these problems arise naturally in a wide range of practical settings such as VLSI design and image processing; sometimes, the motivation is described differently, e. g. as a clustering task.

Another application of MINIMUM-BISECTION is in assignment problems, of a form that is common in parallel systems and in scientific computing: jobs need to be assigned to machines in a balanced way, while assigning certain pairs of jobs the same machine, as much as possible. For example, consider assigning n jobs to 2 machines, when the amount of communication between every two jobs is known, and the goal is to have equal load (number of jobs) on each machine, and bring to minimum the total communication that goes between the machines. Clearly, this last problem can be restated as MINIMUM-BISECTION in a suitable graph.

It should be noted that in many of these settings, a true approximation is not absolutely necessary, and a bicriteria approximation may suffice. Nevertheless, the algorithms stated in Sect. "Key Results" have been used to design algorithms for other problems, such as (1) an approximation algorithm for minimum bisection in k-uniform hypergraphs [3]; (2) an approximation algorithm for a variant of the minimum multicut problem [17]; and (3) an algorithm that efficiently certifies the unsatisfiability of random $2k$-SAT with sufficiently many clauses [5].

From a practical perspective, numerous heuristics (algorithms without worst-case guarantees) for graph partitioning have been proposed and studied, see [1] for an extensive survey. For example, one of the most famous heuristics is Kerninghan and Lin's local search heuristic for minimum bisection [11].

Open Problems

Currently, there is a large gap between the $O(\log^{1.5} n)$ approximation ratio for MINIMUM-BISECTION achieved by Theorem 1 and the hardness of approximation results known for it. As mentioned above, MINIMUM-BISECTION is known to be NP-hard (see [9]).

The problem is not known to be APX-hard but several results provide evidence towards this possibility. Bui and Jones [4] show that for every fixed $\epsilon > 0$, it is NP-hard to approximate the minimum bisection within an *additive* term of $n^{2-\epsilon}$. Feige [7] showed that if refuting 3SAT is hard on average on a natural distribution of inputs, then for every fixed $\varepsilon > 0$ there is no $4/3 - \varepsilon$ approximation algorithm for minimum bisection. Khot [12] proved that minimum bisection does not admit a polynomial-time approximation scheme (PTAS) unless NP has randomized sub-exponential time algorithms.

Taking a broader perspective, currently there is a (multiplicative) gap of $O(\log n)$ between the approximation ratio for MINIMUM-BISECTION and that of minimum quotient-cuts (and thus also to the factor achieved by bicriteria approximation). It is interesting whether this gap can be reduced, e. g. by using the algorithm of [2] in a non-black box manner.

The vertex-cut version of MINIMUM-BISECTION is defined as follows: the goal is to partition the vertices of the input graph into $V = A \cup B \cup S$ with $|S|$ as small as possible, under the constraints that $\max\{|A|, |B|\} \leq n/2$ and no edge connects A with B. It is not known whether a polylogarithmic factor approximation can be attained for this problem. It should be noted that the same question regarding the directed version of MINIMUM-BISECTION was answered negatively by Feige and Yahalom [8].

Cross References

See entry on the paper by Arora, Rao, and Vazirani [2].

- ► Separators in Graphs
- ► Sparsest Cut

Recommended Reading

1. Alpert, C.J., Kahng, A.B.: Recent directions in netlist partitioning: a survey. Integr. VLSI J. **19**(1–2), 1–81 (1995)
2. Arora, S., Rao, S., Vazirani, U.: Expander flows, geometric embeddings, and graph partitionings. In: 36th Annual Symposium on the Theory of Computing, pp. 222–231, Chicago, June 2004
3. Berman, P., Karpinski, M.: Approximability of hypergraph minimum bisection. ECCC Report TR03-056, Electronic Colloquium on Computational Complexity, vol. 10 (2003)
4. Bui, T.N., Jones, C.: Finding good approximate vertex and edge partitions is NP-hard. Inform. Process. Lett. **42**(3), 153–159 (1992)
5. Coja-Oghlan, A., Goerdt, A., Lanka, A., Schädlich, F.: Techniques from combinatorial approximation algorithms yield efficient algorithms for random 2k-SAT. Theor. Comput. Sci. **329**(1–3), 1–45 (2004)
6. Feige, U., Krauthgamer, R.: A polylogarithmic approximation of the minimum bisection. SIAM Review **48**(1), 99–130 (2006) (Previous versions appeared in Proceedings of 41st FOCS, 1999; and in SIAM J. Comput. 2002)
8. Feige, U., Yahalom, O.: On the complexity of finding balanced oneway cuts. Inf. Process. Lett. **87**(1), 1–5 (2003)
7. Feige, U.: Relations between average case complexity and approximation complexity. In: 34th Annual ACM Symposium on the Theory of Computing, pp. 534–543, Montréal, May 19–21, 2002
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W.H. Freeman and Company (1979)
10. Karger, D.R.: Minimum cuts in near-linear time. J. ACM **47**(1), 46–76 (2000)
11. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. **49**(2), 291–307 (1970)
12. Khot, S.: Ruling out PTAS for graph Min-Bisection, Densest Subgraph and Bipartite Clique. In: 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 136–145, Georgia Inst. of Technol., Atlanta 17–19 Oct. 2004
13. Klein, P., Plotkin, S.A., Rao, S.: Excluded minors, network decomposition, and multicommodity flow. In: 25th Annual ACM Symposium on Theory of Computing, pp. 682–690, San Diego, 1993 May 16–18
14. Leighton, T., Rao, S.: Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. J. ACM **46**(6), 787–832, 29th FOCS, 1988 (1999)
15. Lipton, R.J., Tarjan, R.E.: Applications of a planar separator theorem. SIAM J. Comput. **9**(3), 615–627 (1980)
16. Rosenberg, A.L., Heath, L.S.: Graph separators, with applications. Frontiers of Computer Science. Kluwer Academic/Plenum Publishers, New York (2001)
17. Svitkina, Z., Tardos, É.: Min-Max multiway cut. In: 7th International workshop on Approximation algorithms for combinatorial optimization (APPROX), pp. 207–218, Cambridge, 2004 August 22–24

Minimum Congestion Redundant Assignments

2002; Fotakis, Spirakis

DIMITRIS FOTAKIS[1], PAUL SPIRAKIS[2]

[1] Department of Information and Communication Systems Engineering, University of the Aegean, Samos, Greece

[2] Computer Engineering and Informatics, Research and Academic Computer Technology Institute, Patras University, Patras, Greece

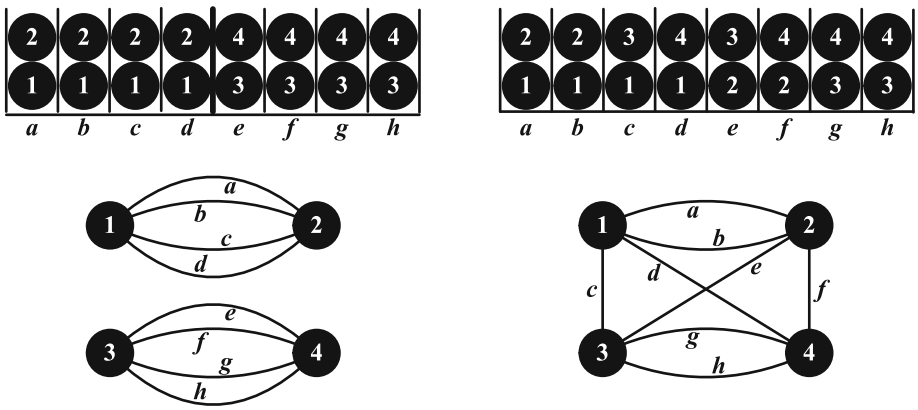
Keywords and Synonyms

Minimum fault-tolerant congestion; Maximum fault-tolerant partition

Problem Definition

This problem is concerned with the most efficient use of redundancy in load balancing on faulty parallel links. More specifically, this problem considers a setting where some messages need to be transmitted from a source to a destination through some faulty parallel links. Each link fails independently with a given probability, and in case of failure, none of the messages assigned to it reaches the destination[1]. An assignment of the messages to the links may use redundancy, i. e. assign multiple copies of some messages to different links. The reliability of a redundant assignment is the probability that every message has a copy

[1] This assumption is realistic if the messages are split into many small packets transmitted in a round-robin fashion. Then the successful delivery of a message requires that all its packets should reach the destination.



Minimum Congestion Redundant Assignments, Figure 1

Two redundant assignments of 4 unit size messages to 8 identical links. Both assign every message to 4 links and 2 messages to every link. The corresponding graph is depicted below each assignment. The assignment on the *left* is the most reliable 2-partitioning assignment ϕ_2. Lemma 3 implies that for every failure probability f, ϕ_2 is at least as reliable as any other assignment ϕ with $\text{Cong}(\phi) \leq 2$. For instance, ϕ_2 is at least as reliable as the assignment on the *right*. Indeed the reliability of the assignment on the right is $1 - 4f^4 + 2f^6 + 4f^7 - 3f^8$, which is bounded from above by $\text{Rel}(\phi_2) = 1 - 2f^4 + f^8$ for all $f \in [0, 1]$

on some active link, thus managing to reach the destination. Redundancy increases reliability, but also increases the message load assigned to the links. A good assignment should achieve high reliability and keep the maximum load of the links as small as possible.

The reliability of a redundant assignment depends on its structure. In particular, the reliability of different assignments putting the same load on every link and using the same number of copies for each message may vary substantially (e. g. compare the reliability of the assignments in Fig. 1). The crux of the problem is to find an efficient way of exploiting redundancy in order to achieve high reliability and low maximum load[2].

The work of Fotakis and Spirakis [1] formulates the scenario above as an optimization problem called *Minimum Fault-Tolerant Congestion* and suggests a simple and provably efficient approach of exploiting redundancy. This approach naturally leads to the formulation of another interesting optimization problem, namely that of computing an efficient fault-tolerant partition of a set of faulty parallel links. [1] presents polynomial-time approximation algorithms for computing a fault-tolerant partition of the links and proves that combining fault-tolerant partitions with standard load balancing algorithms results in a good approximation to Minimum Fault-Tolerant Congestion. To the best knowledge of the entry authors, this work is the first to consider the approximability of computing a redundant assignment that minimizes the maximum load of the links subject to the constraint that random faults should be tolerated with a given probability.

[2]If one does not insist on minimizing the maximum load, a reliable assignment is constructed by assigning every message to the most reliable links.

Notations and Definitions

Let M denote a set of m faulty parallel links connecting a source s to a destination t, and let J denote a set of n messages to be transmitted from s to t. Each link i has a rational capacity $c_i \geq 1$ and a rational failure probability $f_i \in (0, 1)$. Each message j has a rational size $s_j \geq 1$. Let $f_{\max} \equiv \max_{i \in M}\{f_i\}$ denote the failure probability of the most unreliable link. Particular attention is paid to the special case of identical capacity links, where all capacities are assumed to be equal to 1.

The reliability of a set of links M', denoted $\text{Rel}(M')$, is the probability that there is an active link in M'. Formally, $\text{Rel}(M') \equiv 1 - \prod_{i \in M'} f_i$. The reliability of a collection of disjoint link subsets $\mathcal{M} = \{M_1, \ldots, M_v\}$, denoted $\text{Rel}(\mathcal{M})$, is the probability that there is an active link in every subset of $\mathcal{M}$. Formally,

$$\text{Rel}(\mathcal{M}) \equiv \prod_{\ell=1}^{v} \text{Rel}(M_\ell) = \prod_{\ell=1}^{v} \left(1 - \prod_{i \in M_\ell} f_i\right) .$$

A redundant assignment $\phi : J \mapsto 2^M \setminus \emptyset$ is a function that assigns every message j to a non-empty set of links $\phi(j) \subseteq M$. An assignment ϕ is feasible for a set of links M' if for every message j, $\phi(j) \cap M' \neq \emptyset$. The *reliability* of an assignment ϕ, denoted $\text{Rel}(\phi)$, is the probability that ϕ is

feasible for the actual set of active links. Formally,

$$\mathrm{Rel}(\phi) \equiv \sum_{\substack{M' \subseteq M \\ \forall j \in J, \phi(j) \cap M' \neq \emptyset}} \left(\prod_{i \in M'} (1 - f_i) \prod_{i \in M \setminus M'} f_i \right)$$

The *congestion* of an assignment ϕ, denoted $\mathrm{Cong}(\phi)$, is the maximum load assigned by ϕ to a link in M. Formally,

$$\mathrm{Cong}(\phi) \equiv \max_{i \in M} \left\{ \sum_{j:\, i \in \phi(j)} \frac{s_j}{c_i} \right\} .$$

Problem 1 (Minimum Fault-Tolerant Congestion)
INPUT: *A set of faulty parallel links* $M = \{(c_1, f_1), \ldots, (c_m, f_m)\}$, *a set of messages* $J = \{s_1, \ldots, s_n\}$, *and a rational number* $\epsilon \in (0, 1)$.
OUTPUT: *A redundant assignment* $\phi : J \mapsto 2^M \setminus \emptyset$ *with* $Rel(\phi) \geq 1 - \epsilon$ *that minimizes* $Cong(\phi)$.

Minimum Fault-Tolerant Congestion is **NP**-hard because it is a generalization of minimizing makespan on (reliable) parallel machines. The decision version of Minimum Fault-Tolerant Congestion belongs to **PSPACE**, but it is not clear whether it belongs to **NP**. The reason is that computing the reliability of a redundant assignment and deciding whether it is a feasible solution is **#P**-complete.

The work of Fotakis and Spirakis [1] presents polynomial-time approximation algorithms for Minimum Fault-Tolerant Congestion based on a simple and natural class of redundant assignments whose reliability can be computed easily. The high level idea is to separate the reliability aspect from load balancing. Technically, the set of links is partitioned in a collection of disjoint subsets $\mathcal{M} = \{M_1, \ldots, M_\nu\}$ with $\mathrm{Rel}(\mathcal{M}) \geq 1 - \epsilon$. Every subset $M_\ell \in \mathcal{M}$ is regarded as a *reliable link* of *effective capacity* $c(M_\ell) \equiv \min_{i \in M_\ell}\{c_i\}$. Then any algorithm for load balancing on reliable parallel machines can be used for assigning the messages to the subsets of $\mathcal{M}$, thus computing a redundant assignment ϕ with $\mathrm{Rel}(\phi) \geq 1 - \epsilon$.

The assignments produced by this approach are called *partitioning assignments*. More precisely, an assignment $\phi : J \mapsto 2^M \setminus \emptyset$ is a *ν-partitioning assignment* if for every pair of messages j, j', either $\phi(j) = \phi(j')$ or $\phi(j) \cap \phi(j') = \emptyset$, and ϕ assigns the messages to ν different link subsets.

Computing an appropriate fault-tolerant collection of disjoint link subsets is an interesting optimization problem by itself. A feasible solution $\mathcal{M}$ satisfies the constraint that $\mathrm{Rel}(\mathcal{M}) \geq 1 - \epsilon$. For identical capacity links, the most natural objective is to maximize the number of subsets in $\mathcal{M}$ (equivalently, the number of reliable links used by the load balancing algorithm). For arbitrary capacities, this objective generalizes to maximizing the total effective capacity of $\mathcal{M}$.

Problem 2 (Maximum Fault-Tolerant Partition)
INPUT: *A set of faulty parallel links* $M = \{(c_1, f_1), \ldots, (c_m, f_m)\}$, *and a rational number* $\epsilon \in (0, 1)$.
OUTPUT: *A collection* $\mathcal{M} = \{M_1, \ldots, M_\nu\}$ *of disjoint subsets of* M *with* $Rel(\mathcal{M}) \geq 1 - \epsilon$ *that maximizes* $\sum_{\ell=1}^{\nu} c(M_\ell)$.

The problem of Maximum Fault-Tolerant Partition is **NP**-hard. More precisely, given m identical capacity links with rational failure probabilities and a rational number $\epsilon \in (0, 1)$, it is **NP**-complete to decide whether the links can be partitioned into sets M_1 and M_2 with $\mathrm{Rel}(M_1) \cdot \mathrm{Rel}(M_2) \geq 1 - \epsilon$.

Key Results

Theorem 1 *There is a 2-approximation algorithm for Maximum Fault-Tolerant Partition of identical capacity links. The time complexity of the algorithm is* $O((m - \sum_{i \in M} \ln f_i) \ln m)$.

Theorem 2 *For every constant* $\delta > 0$, *there is a* $(8 + \delta)$*-approximation algorithm for Maximum Fault-Tolerant Partition of capacitated links. The time complexity of the algorithm is polynomial in the input size and* $1/\delta$.

To demonstrate the efficiency of the partitioning approach for Maximum Fault-Tolerant Congestion, Fotakis and Spirakis prove that for certain instances, the reliability of the most reliable partitioning assignment bounds from above the reliability of any other assignment with the same congestion (see Fig. 1 for an example).

Lemma 3 *For any positive integers* Λ, ν, μ *and any rational* $f \in (0, 1)$, *let* ϕ *be a redundant assignment of* $\Lambda\nu$ *unit size messages to* $\nu\mu$ *identical capacity links with failure probability* f. *Let* ϕ_ν *be the* ν*-partitioning assignment that assigns* Λ *messages to each of* ν *disjoint subsets consisting of* μ *links each. If* $Cong(\phi) \leq \Lambda = Cong(\phi_\nu)$, *then* $Rel(\phi) \leq (1 - f^\mu)^\nu = Rel(\phi_\nu)$.

Based on the previous upper bound on the reliability of any redundant assignment, [1] presents polynomial-time approximation algorithms for Maximum Fault-Tolerant Congestion.

Theorem 4 *There is a quasi-linear-time 4-approximation algorithm for Maximum Fault-Tolerant Congestion on identical capacity links.*

Theorem 5 *There is a polynomial-time* $2\lceil \ln(m/\epsilon)/\ln(1/f_{\max})\rceil$*-approximation algorithm for Maximum Fault-Tolerant Congestion on instances with unit size messages and capacitated links.*

Applications

In many applications dealing with faulty components (e. g. fault-tolerant network design, fault-tolerant routing), a combinatorial structure (e. g. a graph, a hypergraph) should optimally tolerate random faults with respect to a given property (e. g. connectivity, non-existence of isolated points). For instance, Lomonosov [5] derived tight upper and lower bounds on the probability that a graph remains connected under random edge faults. Using the bounds of Lomonosov, Karger [3] obtained improved theoretical and practical results for the problem of estimating the reliability of a graph. In this work, Lemma 3 provides a tight upper bound on the probability that isolated nodes do not appear in a not necessarily connected hypergraph with $\Lambda\nu$ nodes and $\nu\mu$ "faulty" hyperedges of cardinality Λ.

More precisely, let ϕ be any assignment of $\Lambda\nu$ unit size messages to $\nu\mu$ identical links that assigns every message to μ links and Λ messages to every link. Then ϕ corresponds to a hypergraph H_ϕ, where the set of nodes consists of $\Lambda\nu$ elements corresponding to the unit size messages and the set of hyperedges consists of $\nu\mu$ elements corresponding to the identical links. Every hyperedge contains the messages assigned to the corresponding link and has cardinality Λ (see Fig. 1 for a simple example with $\Lambda = 2$, $\nu = 2$, and $\mu = 4$). Clearly, an assignment ϕ is feasible for a set of links $M' \subseteq M$ iff the removal of the hyperedges corresponding to the links in $M \setminus M'$ does not leave any isolated nodes[3] in H_ϕ. Lemma 3 implies that the hypergraph corresponding to the most reliable ν-partitioning assignment maximizes the probability that isolated nodes do not appear when hyperedges are removed equiprobably and independently.

The previous work on fault-tolerant network design and routing mostly focuses on the worst-case fault model, where a feasible solution must tolerate any configuration of a given number of faults. The work of Gasieniec et al. [2] studies the fault-tolerant version of minimizing congestion of virtual path layouts in a complete ATM network. In addition to several results for the worst-case fault model, [2] constructs a virtual path layout of logarithmic congestion that tolerates random faults with high probability. On the other hand, the work of Fotakis and Spirakis shows how to construct redundant assignments that tolerate random faults with a probability given as part of the input and achieve a congestion close to optimal.

[3]For a node v, let $\deg_H(v) \equiv |\{e \in E(H) : v \in e\}|$. A node v is isolated in H if $\deg_H(v) = 0$.

Open Problems

An interesting research direction is to determine the computational complexity of Minimum Fault-Tolerant Congestion and related problems. The decision version of Minimum Fault-Tolerant Congestion is included in the class of languages decided by a polynomial-time non-deterministic Turing machine that reduces the language to a single call of a **#P** oracle. After calling the oracle once, the Turing machine rejects if the oracle's outcome is less than a given threshold and accepts otherwise. This class is denoted $\mathbf{NP}^{\mathbf{\#P}[1,\mathrm{comp}]}$ in [1]. In addition to Minimum Fault-Tolerant Congestion, $\mathbf{NP}^{\mathbf{\#P}[1,\mathrm{comp}]}$ includes the decision version of Stochastic Knapsack considered in [4]. A result of Toda and Watanabe [6] implies that $\mathbf{NP}^{\mathbf{\#P}[1,\mathrm{comp}]}$ contains the entire Polynomial Hierarchy. A challenging open problem is to determine whether the decision version of Minimum Fault-Tolerant Congestion is complete for $\mathbf{NP}^{\mathbf{\#P}[1,\mathrm{comp}]}$.

A second direction for further research is to consider the generalizations of other fundamental optimization problems (e. g. shortest paths, minimum connected subgraph) under random faults. In the fault-tolerant version of minimum connected subgraph for example, the input consists of a graph whose edges fail independently with given probabilities, and a rational number $\epsilon \in (0, 1)$. The goal is to compute a spanning subgraph with a minimum number of edges whose reliability is at least $1 - \epsilon$.

Cross References

- Approximation Schemes for Bin Packing
- Bin Packing
- Connectivity and Fault-Tolerance in Random Regular Graphs
- List Scheduling

Recommended Reading

1. Fotakis, D., Spirakis, P.: Minimum Congestion Redundant Assignments to Tolerate Random Faults. Algorithmica **32**, 396–422 (2002)
2. Gasieniec, L., Kranakis, E., Krizanc, D., Pelc, A.: Minimizing Congestion of Layouts for ATM Networks with Faulty Links. In: Penczek, W., Szalas, A. (eds.) Proceedings of the 21st International Symposium on Mathematical Foundations of Computer Science. Lecture Notes in Computer Science, vol. 1113, pp. 372–381. Springer, Berlin (1996)

3. Karger, D.: A Randomized Fully Polynomial Time Approximation Scheme for the All-Terminal Network Reliability Problem. SIAM J. Comput. **29**, 492–514 (1999)
4. Kleinberg, J., Rabani, Y., Tardos, E.: Allocating Bandwidth for Bursty Connections. SIAM J. Comput. **30**, 191–217 (2000)
5. Lomonosov, M.: Bernoulli Scheme with Closure. Probl. Inf. Transm. **10**, 73–81 (1974)
6. Toda, S., Watanabe, O.: Polynomial-Time 1-Turing Reductions from #**PH** to #**P**. Theor. Comput. Sci. **100**, 205–221 (1992)

Minimum Energy Broadcasting in Wireless Geometric Networks

2005; Ambühl

CHRISTOPH AMBÜHL
Department of Computer Science,
University of Liverpool, Liverpool, UK

Keywords and Synonyms

Energy-efficient broadcasting in wireless networks

Problem Definition

In the most common model for wireless networks, stations are represented by points in $\mathbb{R}^d$. They are equipped with a omnidirectional transmitter and receiver which enables them to communicate with other stations. In order to send a message from a station s to a station t, station s needs to emit the message with enough power such that t can receive it. It is usually assumed that the power required by a station s to transmit data directly to station t is $\|st\|^\alpha$, for some constant $\alpha \geq 1$, where $\|st\|$ denotes the distance between s and t.

Because of the omnidirectional nature of the transmitters and receivers, a message sent by a station s with power r^α can be received by all stations within a disc of radius r around s. Hence the energy required to send a message from a station s directly to a set of stations S' is determined by $\max_{v \in S'} \|sv\|^\alpha$.

An instance of the *minimum energy broadcast routing problem in wireless networks* (MEBR) consists of a set of stations S and a constant $\alpha \geq 1$. One of the stations in S is designated as the source station s_0. The goal is to send a message at minimum energy cost from s_0 to all other stations in S. This operation is called *broadcast*.

In the case $\alpha = 1$, the optimal solution is to send the message directly from s_0 to all other stations. For $\alpha > 1$, sending the message via intermediate stations which forward it to other stations is often more energy efficient.

A solution of the MEBR instance can be described in terms of a so-called *broadcast tree*. That is, a directed spanning tree of S which contains directed paths from s_0 to all other vertices. The solution described by a broadcast tree T is the one in which every station forwards the message to all its out-neighbors in T.

Problem 1 (MEBR)
INSTANCE: *A set S of points in R^d, $s_0 \in S$ designated as the source, and a constant α.*
SOLUTION: *A broadcast tree T of S.*
MEASURE: *The objective is to minimize the total energy needed to broadcast a message from s_0 to all other nodes, which can be expressed by*

$$\sum_{u \in S} \max_{v \in \delta(u)} \|uv\|^\alpha , \tag{1}$$

where $\delta(u)$ denotes the set of out-neighbors of station u in T.

The MEBR problem is known to be NP-hard for $d \geq 2$ and $\alpha > 1$ [2]. APX-hardness is known for $d \geq 3$ [5].

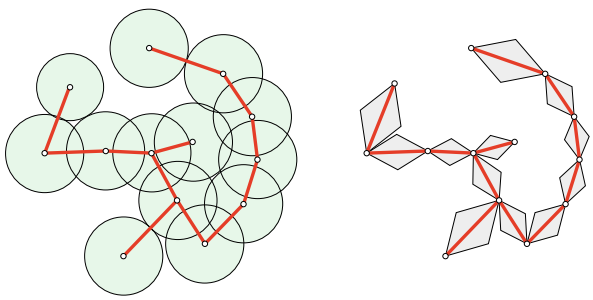
Key Results

Numerous heuristics have been proposed for this problem. Only a few of them have been analyzed theoretically. The one which attains the best approximation guarantee is the so-called MST-heuristic [12].

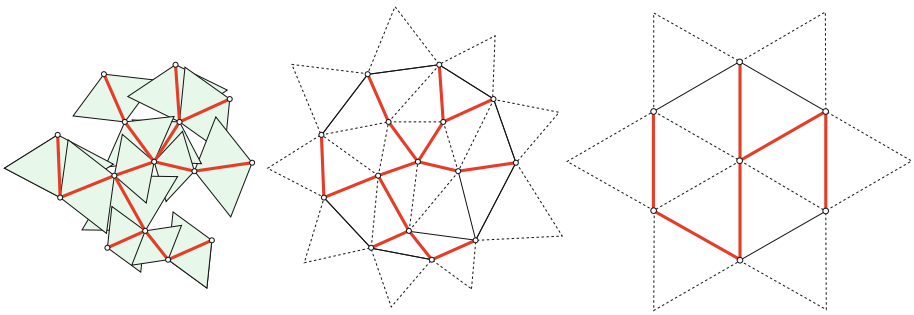
MST-HEURISTIC: Compute a minimum spanning tree of S (*mst(S)*) and turn it into an broadcast tree by directing the edges.

Theorem 1 [1] *In the Euclidean plane, the MST-heuristic is a 6 approximation algorithm for MEBR for all $\alpha \geq 2$.*

Theorem 2 [9] *In the Euclidean three-dimensional space, the MST-heuristic is a 18.8 approximation algorithm for MEBR for all $\alpha \geq 3$.*



Minimum Energy Broadcasting in Wireless Geometric Networks, Figure 1
Illustration of the first and second approach for bounding *w(S)*. In both approaches, *w(S)* is bounded in terms of the total area covered by the shapes



Minimum Energy Broadcasting in Wireless Geometric Networks, Figure 2
Illustration of the tight bound for $d = 2$. The total area of the equilateral triangles on the left is bounded by its extended convex hull shown in the middle. The point set that maximizes area of the extended convex hull is the star shown on the right

For $\alpha < d$, the MST-heuristic does not provide a constant approximation ratio. The d-dimensional kissing numbers represent lower bounds for the performance of the MST-heuristic. Hence the analysis for $d = 2$ is tight, whereas for $d = 3$ the lower bound is 12.

Analysis

The analysis of the MST-heuristic is based on good upper bounds for

$$w(S) := \sum_{e \in mst(S)} \|e\|^{\alpha}, \tag{2}$$

which obviously is an upper bound on (1). The radius of an instance of MEBR is the distance between s_0 to the station furthest from s_0. It turns out that the MST-heuristic performs worst on instances of radius 1 whose optimal solution is to broadcast the message directly from s_0 to all other stations. Since the optimal value for such instances is 1, the approximation ratio follows from good upper bounds on *w(S)* for instances with radius 1.

The rest of this section focuses on the case $d = \alpha = 2$. There are two main approaches for upper bounding *w(S)*. In both approaches, *w(S)* is upper bounded in terms of the area of particular kinds of shapes associated with either the stations or with the edges of the MST.

In the first approach, the shapes are disks of radius $m/2$ placed around every station of S, where m is the length of the longest edge of *mst(S)*. Let A denote the area covered by the disks. One can prove $w(S) \leq \frac{4}{\pi}\left(A - \pi(m/2)^2\right)$. Assuming that S has radius 1, one can prove $w(S) \leq 8$ quite easily [4]. This approach can even be extended to obtain $w(S) \leq 6.33$ [8], and it can be generalized for $d \geq 2$.

In the second approach [7,11], *w(S)* is expressed in terms of shapes associated with the edges of *mst(S)*, e. g., diamond shapes as shown on the right of Fig. 1. The area of a diamond for an edge e is equal to $\|e\|^2/(2\sqrt{3})$. Since one can prove that the diamonds never intersect, one obtains $w(S) = A/(2\sqrt{3})$. For instances with radius 1, one can get $w(S) \leq 12.15$.

For the 2-dimensional case, one can even obtain a matching upper bound [1]. The shapes used in this proof are equilateral triangles, arranged in pairs along every edge of the MST. As can be seen on the left of Fig. 2, these shapes do intersect. Still one can obtain a good upper bound on their total area by means of the convex hull of S:

Let the *extended convex hull of S* be the convex hull of S extended by equilateral triangles along the border of the convex hull. One can prove that the total area generated by the equilateral triangle shapes along the edges of *mst(S)* is upper bounded by the area of the extended convex hull of S. By showing that for instances of radius 1 the area of the extended convex hull is maximized by the point configuration shown on the right of Fig. 2, the matching upper bound of 6 can be established.

Applications

The MEBR problem is a special case of a large class of problems called *range assignment problems*. In all these problems, the goal is to assign a range to each station such that a certain type of communication operation such as broadcast, all-to-1 (gathering), all-to-all (gossiping), can be accomplished. See [3] for a survey on range assignment problems.

It is worth noting that the problem of upper bounding *w(S)* has already been considered in different contexts. The idea of using diamond shapes to upper bound the length of an MST has already been used by Gilbert and Pollak in [6]. Steele [10] makes use of space filling curves to bound *w(S)*.

Open Problems

An obvious open problem is to close the gap in the analysis of the MST-heuristic for the three dimensional case. This

might be very difficult, as the lower bound from the kissing number might not be tight.

Even in the plane, the approximation ratio of the MST-heuristic is quite large. It would be interesting to see a different approach for the problem, maybe based on LP-rounding. It is still not known whether MEBR is APX-hard for instances in the Euclidean plane. Hence there might exist a PTAS for this problem.

Cross References

- ▶ Broadcasting in Geometric Radio Networks
- ▶ Deterministic Broadcasting in Radio Networks
- ▶ Geometric Spanners
- ▶ Minimum Geometric Spanning Trees
- ▶ Minimum Spanning Trees
- ▶ Randomized Broadcasting in Radio Networks
- ▶ Randomized Gossiping in Radio Networks

Recommended Reading

1. C. Ambühl: An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks. In: Proceedings of 32th International Colloquium on Automata, Languages and Programming (ICALP). Lecture Notes in Computer Science, vol. 3580, pp. 1139–1150. Springer, Berlin (2005)
2. Clementi, A., Crescenzi, P., Penna, P., Rossi, G., Vocca, P.: On the Complexity of Computing Minimum Energy Consumption Broadcast Subgraphs. In: Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS), pp. 121–131 (2001)
3. Clementi, A., Huiban, G., Penna, P., Rossi, G., Verhoeven, Y.: Some Recent Theoretical Advances and Open Questions on Energy Consumption in Ad-Hoc Wireless Networks. In: Proceedings of the 3rd Workshop on Approximation and Randomization Algorithms in Communication Networks (ARACNE), pp. 23–38 (2002)
4. Flammini, M., Klasing, R., Navarra, A., Perennes, S.: Improved approximation results for the minimum energy broadcasting problem. In: Proceedings of the 2004 joint workshop on Foundations of mobile computing (2004)
5. Fuchs, B.: On the hardness of range assignment problems. In: Proceedings of the 6th Italian Conference on Algorithms and Complexity (CIAC), pp. 127–138 (2006)
6. Gilbert, E.N., Pollak, H.O.: Steiner minimal trees. SIAM J. Appl. Math. **16**, 1–29 (1968)
7. Klasing, R., Navarra, A., Papadopoulos, A., Perennes, S.: Adaptive broadcast consumption (ABC), a new heuristic and new bounds for the minimum energy broadcast routing problem. In: Proceeding of the 3rd IFIP-TC6 international networking conference (NETWORKING), pp. 866–877 (2004)
8. Navarra, A.: Tighter bounds for the minimum energy broadcasting problem. In: Proceedings of the 3rd International Symposium on Modeling and Optimization in Mobile, Ad-hoc and Wireless Networks (WiOpt), pp. 313–322 (2005)
9. Navarra, A.: 3-d minimum energy broadcasting. In: Proceedings of the 13th Colloquium on Structural Information and Communication Complexity (SIROCCO), pp. 240–252 (2006)
10. Steele, J.M.: Cost of sequential connection for points in space. Oper. Res. Lett. **8**, 137–142 (1989)
11. Wan, P.-J., Calinescu, G., Li, X.-Y., Frieder, O.: Minimum-energy broadcasting in static ad hoc wireless networks. Wirel. Netw. **8**, 607–617 (2002)
12. Wieselthier, J.E., Nguyen, G.D., Ephremides, A.: Energy-efficient broadcast and multicast trees in wireless networks. Mobile Netw. Appl. **7**, 481–492 (2002)

Minimum Energy Cost Broadcasting in Wireless Networks

2001; Wan, Calinescu, Li, Frieder

Peng-Jun Wan, Xiang-Yang Li, Ophir Frieder
Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA

Keywords and Synonyms

Minimum energy broadcast; MST; MEB

Problem Definition

Ad hoc wireless networks have received significant attention in recent years due to their potential applications in battlefield, emergency disaster relief and other applications [11,15]. Unlike wired networks or cellular networks, no wired backbone infrastructure is installed in ad hoc wireless networks. A communication session is achieved either through a single-hop transmission if the communication parties are close enough, or through relaying by intermediate nodes otherwise. Omni-directional antennas are used by all nodes to transmit and receive signals. They are attractive in their broadcast nature. A single transmission by a node can be received by many nodes within its vicinity. This feature is extremely useful for multicasting/broadcasting communications. For the purpose of energy conservation, each node can dynamically adjust its transmitting power based on the distance to the receiving node and the background noise. In the most common power-attenuation model [10], the signal power falls as $\frac{1}{r^\kappa}$, where r is the distance from the transmitter antenna and κ is a real *constant* between 2 and 4 dependent on the wireless environment. Assume that all receivers have the same power threshold for signal detection, which is typically normalized to one. With these assumptions, the power required to support a link between two nodes separated by a distance r is r^κ. A key observation here is that relaying a signal between two nodes may result in lower total transmission power than communicating over a large distance due to the nonlinear power attenuation. They as-

sume the network nodes are given as a finite point[1] set P in a two-dimensional plane. For any real number κ, they use $G^{(\kappa)}$ to denote the weighted complete graph over P in which the weight of an edge e is $\|\mathbf{e}\|^{\kappa}$.

The minimum-energy unicast routing is essentially a shortest-path problem in $G^{(\kappa)}$. Consider any unicast path from a node $\mathbf{p} = \mathbf{p}_0 \in P$ to another node $\mathbf{q} = \mathbf{p}_m \in P$: $\mathbf{p}_0\mathbf{p}_1 \cdots \mathbf{p}_{m-1}\mathbf{p}_m$. In this path, the transmission power of each node $\mathbf{p}_i$, $0 \le i \le m-1$, is $\|\mathbf{p}_i\mathbf{p}_{i+1}\|^{\kappa}$ and the transmission power of $\mathbf{p}_m$ is zero. Thus the total transmission energy required by this path is $\sum_{i=0}^{m-1} \|\mathbf{p}_i\mathbf{p}_{i+1}\|^{\kappa}$, which is the total weight of this path in G^{κ}. So by applying any shortest-path algorithm such as the Dijkstra's algorithm [5], one can solve the minimum-energy unicast routing problem.

However, for broadcast applications (in general multicast applications), Minimum-Energy Routing is far more challenging. Any broadcast routing is viewed as an arborescence (a directed tree) T, rooted at the source node of the broadcasting, that spans all nodes. Use $f_T(\mathbf{p})$ to denote the transmission power of the node $\mathbf{p}$ required by T. For any leaf node $\mathbf{p}$ of T, $f_T(\mathbf{p}) = 0$. For any internal node $\mathbf{p}$ of T,

$$f_T(\mathbf{p}) = \max_{\mathbf{pq} \in T} \|\mathbf{pq}\|^{\kappa},$$

in other words, the κ-th power of the longest distance between $\mathbf{p}$ and its children in T. The total energy required by T is $\sum_{\mathbf{p} \in P} f_T(\mathbf{p})$. Thus the minimum-energy broadcast routing problem is different from the conventional link-based minimum spanning tree (MST) problem. Indeed, while the MST can be solved in polynomial time by algorithms such as Prim's algorithm and Kruskal's algorithm [5], it is NP-hard [4] to find the minimum-energy broadcast routing tree for nodes placed in two-dimensional plane. In its general graph version, the minimum-energy broadcast routing can also be shown to be NP-hard [7], and even worse, it can not be approximated within a factor of $(1-\epsilon)\log \Delta$, unless $NP \subseteq DTIME\left[n^{O(\log\log n)}\right]$, by an approximation-preserving reduction from the Connected Dominating Set problem [8], where Δ is the maximal degree and ϵ is any arbitrary small positive constant.

Three greedy heuristics have been proposed for the minimum-energy broadcast routing problem by [15]. The MST heuristic first applies the Prim's algorithm to obtain a MST, and then orient it as an arborescence rooted at the source node. The SPT heuristic applies the Dijkstra's algorithm to obtain a SPT rooted at the source node. The BIP heuristic is the node version of Dijkstra's algorithm for SPT. It maintains, throughout its execution, a single arborescence rooted at the source node. The arborescence starts from the source node, and new nodes are added to the arborescence one at a time on the minimum incremental cost basis until all nodes are included in the arborescence. The incremental cost of adding a new node to the arborescence is the minimum additional power increased by some node in the current arborescence to reach this new node. The implementation of BIP is based on the standard Dijkstra's algorithm, with one fundamental difference on the operation whenever a new node q is added. Whereas the Dijkstra's algorithm updates the node weights (representing the current knowing distances to the source node), BIP updates the cost of each link (representing the incremental power to reach the head node of the directed link). This update is performed by subtracting the cost of the added link pq from the cost of every link qr that starts from q to a node r not in the new arborescence.

Key Results

The performance of these three greedy heuristics have been evaluated in [15] by simulation studies. However, their analytic performances in terms of the approximation ratio remained open until [13]. The work of Wan et al. [13] derived the bounds on their approximation ratios.

Let us begin with the SPT algorithm. Let ϵ be a sufficiently small positive number. Consider m nodes $\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_m$ evenly distributed on a cycle of radius 1 centered at a node $\mathbf{o}$. For $1 \le i \le m$, let $\mathbf{q}_i$ be the point in the line segment $\mathbf{op}_i$ with $\|\mathbf{oq}_i\| = \epsilon$. They consider a broadcasting from the node $\mathbf{o}$ to these $n = 2m$ nodes $\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_m, \mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_m$. The SPT is the superposition of paths $\mathbf{oq}_i\mathbf{p}_i$, $1 \le i \le m$. Its total energy consumption is $\epsilon^2 + m(1-\epsilon)^2$. On the other hand, if the transmission power of node $\mathbf{o}$ is set to 1, then the signal can reach all other points. Thus the minimum energy consumed by all broadcasting methods is at most 1. So the approximation ratio of SPT is at least $\epsilon^2 + m(1-\epsilon)^2$. As $\epsilon \longrightarrow 0$, this ratio converges to $\frac{n}{2} = m$.

They [13] also proved that

Theorem 1 *The approximation ratio of MST is at least 6 for any $\kappa \ge 2$.*

Theorem 2 *The approximation ratio of BIP is at least $\frac{13}{3}$ for any $\kappa = 2$.*

They then derived the upper bounds by extensively using the geometric structures of Euclidean MSTs (EMST). They

[1] The terms node, point and vertex are interchangeable here: node is a network term, point is a geometric term, and vertex is a graph term.

first observed that as long as the cost of a link is an increasing function of the Euclidean length of the link, the set of MSTs of any point set *coincides* with the set of Euclidean MSTs of the same point set. They proved a key result about an upper bound on the parameter $\sum_{e \in MST(P)} \|e\|^2$ for any finite point set P inside a disk with radius one.

Theorem 3 *Let c be the supreme of $\sum_{e \in MST(P)} \|e\|^2$ over all such point sets P. Then $6 \leq c \leq 12$.*

The following lemma proved in [13] is used to bound the energy cost for broadcast when each node can dynamically adjust its power.

Lemma 4 *For any point set P in the plane, the total energy required by any broadcasting among P is at least $\frac{1}{c} \sum_{e \in MST(P)} \|e\|^{\kappa}$.*

Lemma 5 *For any broadcasting among a point set P in a two-dimensional plane, the total energy required by the arborescence generated by the BIP algorithm is at most $\sum_{e \in MST(P)} \|e\|^{\kappa}$.*

Thus, they conclude the following two theorems.

Theorem 6 *The approximation ratio of EMST is at most c, and therefore is at most 12.*

Theorem 7 *The approximation ratio of BIP is at most c, and therefore is at most 12.*

Later, Wan et al. [14] studied the energy efficient multicast for wireless networks when each node can dynamically adjust its power. Given a set of receivers Q, the problem Min-Power Asymmetric Multicast seeks, for any given communication session, an arborescence T of minimum total power which is rooted at the source node s and reaches all nodes in Q. As a generalization of Min-Power Asymmetric Broadcast Routing, Min-Power Asymmetric Multicast Routing is also NP-hard. Wieselthier et al. [15] adapted their three broadcasting heuristics to three multicasting heuristics by a technique of pruning, which was called as pruned minimum spanning tree (P-MST), pruned shortest-path tree (P-SPT), and pruned broadcasting incremental power (P-BIP), respectively in [14]. The idea is as follows. They first obtain a spanning tree rooted at the source of a given multicast session by applying any of the three broadcasting heuristics. They then eliminate from the spanning arborescence all nodes which do not have any descendant in Q. They [14] show by constructing examples that all structures P-SPT, P-MST and P-BIP could have approximation ratio as large as $\Theta(n)$ in the worst case for multicast. They then further proposed a multicast scheme with a constant approximation ratio on the total energy consumption. Their protocol for Min-Power Asymmetric Multicast Routing is based on Takahashi-Matsuyama Steiner tree heuristic [12]. Initially, the multicast tree T contains only the source node. At each iterative step, the multicast tree T is grown by one path from some node in T to some destination node from Q that is not yet in the tree T. The path must have the least total power among all such paths from a node in T to a node in $Q - T$. This procedure is repeated until all required nodes are included in T. This heuristic is referred to as Shortest Path First (SPF).

Theorem 8 *For asymmetric multicast communication, the approximation ratio of SPF is between 6 and 2c, which is at most 24.*

Applications

Broadcasting and multicasting in wireless ad hoc networks are critical mechanisms in various applications such as information diffusion, wireless networks, and also for maintaining consistent global network information. Broadcasting is often necessary in MANET routing protocols. For example, many unicast routing protocols such as Dynamic Source Routing (DSR), Ad Hoc On Demand Distance Vector (AODV), Zone Routing Protocol (ZRP), and Location Aided Routing (LAR) use broadcasting or a derivation of it to establish routes. Currently, these protocols all rely on a simplistic form of broadcasting called *flooding*, in which each node (or all nodes in a localized area) retransmits each received unique packet exactly one time. The main problems with flooding are that it typically causes unproductive and often harmful bandwidth congestion, as well as inefficient use of node resources. Broadcasting is also more efficient than sending multiple copies the same packet through unicast. It is highly important to use power-efficient broadcast algorithms for such networks since wireless devises are often powered by batteries only.

Open Problems

There are some interesting questions left for further study. For example, the exact value of the constant c remains unsolved. A tighter upper bound on c can lead to tighter upper bounds on the approximation ratios of both the link-based MST heuristic and the BIP heuristic. They conjecture that the exact value for c is 6, which seems to be true based on their extensive simulations. The second question is what is the approximation lower bound for minimum energy broadcast? Is there a PTAS for this problem?

So far, all the known theoretically good algorithms either assume that the power needed to support a link uv is proportional to $\|uv\|^{\kappa}$ or is a fixed cost that is independent

of the neighboring nodes that it will communicate with. In practice, the energy consumption of a node is neither solely dependent on the distance to its farthest neighbor, nor totally independent of its communication neighbor. For example, a more general power consumption model for a node u would be $c_1 + c_2 \cdot \|uv\|^\kappa$ for some constants $c_1 \geq 0$ and $c_2 \geq 0$ where v is its farthest communication neighbor in a broadcast structure. No theoretical result is known about the approximation of the optimum broadcast or multicast structure under this model. When $c_2 = 0$, this is the case where all nodes have a fixed power for communication. Minimizing the total power used by a reliable broadcast tree is equivalent to the minimum connected dominating set problem (MCDS), i. e., minimize the number of nodes that relay the message, since all relaying nodes of a reliable broadcast form a connected dominating set (CDS). Notice that recently a PTAS [2] has been proposed for MCDS in UDG graph.

Another important question is how to find efficient broadcast/multicast structures such that the delay from the source node to the last node receiving message is bounded by a predetermined value while the total energy consumption is minimized. Notice that here the delay of a broadcast/multicast based on a tree is not simply the height of the tree: many nodes cannot transmit simultaneously due to the interference.

Cross References

- ► Broadcasting in Geometric Radio Networks
- ► Deterministic Broadcasting in Radio Networks

Recommended Reading

1. Ambühl, C.: An Optimal Bound for the MST Algorithm to Compute Energy Efficient Broadcast Trees in Wireless Networks. In: Proceedings of 32th International Colloquium on Automata, Languages and Programming (ICALP). LNCS, vol. 3580, pp. 1139–1150 (2005)
2. Cheng, X., Huang, X., Li, D., Du, D.-Z.: Polynomial-time approximation scheme for minimum connected dominating set in ad hoc wireless networks. Netw. **42**, 202–208 (2003)
3. Chvátal, V.: A Greedy Heuristic for the Set-Covering Problem. Math. Oper. Res. **4**(3), 233–235 (1979)
4. Clementi, A., Crescenzi, P., Penna, P., Rossi, G., Vocca, P.: On the complexity of computing minimum energy consumption broadcast subgraphs. In: 18th Annual Symposium on Theoretical Aspects of Computer Science. LNCS, vol. 2010, pp. 121–131 (2001)
5. Cormen, T.J., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. MIT Press and McGraw-Hill, Columbus (1990)
6. Flammini, M., Navarra, A., Klasing, R., Pérennes, A.: Improved approximation results for the minimum energy broadcasting problem. DIALM-POMC, pp. 85–91. ACM Press, New York (2004)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: a Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
8. Guha, S., Khuller, S.: Approximation Algorithms for Connected Dominating Sets. Algorithmica **20**, 347–387 (1998)
9. Preparata, F.P., Shamos, M.I.: Computational Geometry: an Introduction. Springer, New York (1985)
10. Rappaport, T.S.: Wireless Communications: Principles and Practices. Prentice Hall, IEEE Press, Piscataway (1996)
11. Singh, S., Raghavendra, C.S., Stepanek, J.: Power-Aware Broadcasting in Mobile Ad Hoc Networks. In: Proceedings of IEEE PIMRC'99, Osaka, September 1999
12. Takahashi, H., Matsuyama, A.: An approximate solution for steiner problem in graphs. Mathematica Japonica **24**(6), 573–577 (1980)
13. Wan, P.-J., Calinescu, G., Li, X.-Y., Frieder, O.: Minimum-energy broadcast routing in static ad hoc wireless networks. ACM Wirel. Netw. Preliminary version appeared in IEEE INFOCOM (2000)**8**(6), 607–617 (2002)
14. Wan, P.-J., Calinescu, G., Yi, C.-W.: Minimum-power multicast routing in static ad hoc wireless networks. IEEE/ACM Trans. Netw. **12**, 507–514 (2004)
15. Wieselthier, J.E., Nguyen, G.D., Ephremides, A.: On the Construction of energy-Efficient Broadcast and Multicast Trees in Wireless Networks. IEEE Infocom **2**, 585–594 (2000)

Minimum Flow Time

1997; Leonardi, Raz

Nikhil Bansal
IBM Research, IBM, Yorktown Heights, NY, USA

Keywords and Synonyms

Response time; Sojourn time

Problem Definition

The problem is concerned with efficiently scheduling jobs on a system with multiple resources to provide a good quality of service. In scheduling literature, several models have been considered to model the problem setting and several different measures of quality have been studied. This note considers the following model: There are several identical machines, and jobs are released over time. Each job is characterized by its size, which is the amount of processing it must receive to be completed, and its release time, before which it cannot be scheduled. In this model, Leonardi and Raz studied the objective of minimizing the average flow time of the jobs, where the flow time of a job is duration of time since it is released until its processing requirement is met. Flow time is also referred to as response time or sojourn time and is a very natural and commonly used measure of the quality of a schedule.

Notations Let $\mathcal{J} = \{1, 2, \ldots, n\}$ denote the set of jobs in the input instance. Each job j is characterized by its release time r_j and its processing requirement p_j. There is a collection of m identical machines, each having the same processing capability. A schedule specifies which job executes at what time on each machine. Given a schedule, the completion time c_j of a job is the earliest time at which job j receives p_j amount of service. The flow time f_j of j is defined as $c_j - r_j$. A schedule is said to be preemptive, if a job can be interrupted arbitrarily, and its execution can be resumed later from the point of interruption without any penalty. A schedule is non-preemptive if a job cannot be interrupted once it is started. In the context of multiple machines, a schedule is said to be migratory, if a job can be moved from one machine to another during its execution without any penalty. In the offline model, all the jobs J are given in advance. In scheduling algorithms, the online model is usually more realistic than the offline model.

Key Results

For a single machine, it is a folklore result that the Shortest Remaining Processing Time (SRPT) policy, that at any time works on the job with the least remaining processing time is optimal for minimizing the average flow time. Note that SRPT is an online algorithm, and is a preemptive scheduling policy.

If no preemption is allowed Kellerer, Tautenhahn, and Woeginger [6] gave an $O(n^{1/2})$ approximation algorithm for minimizing the flow time on a single machine, and also showed that no polynomial time algorithm can have an approximation ratio of $n^{1/2-\varepsilon}$ for any $\varepsilon > 0$ unless P=NP.

Leonardi and Raz [8] gave the first non-trivial results for minimizing the average flow time on multiple machines. Later, a simpler presentation of this result was given by Leonardi [7]. The main result of [8] is the following.

Theorem 1 ([8]) *On multiple machines, the SRPT algorithm is $O(\min(\log(n/m), \log P))$ competitive for minimizing average flow time, where P is the maximum to minimum job size ratio.*

They also gave a matching lower bound (up to constant factors) on the competitive ratio.

Theorem 2 ([8]) *For the problem of minimizing flow time on multiple machines, any online algorithm has a competitive ratio of $\Omega(\min(\log(n/m), \log P))$, even when randomization is allowed.*

Note that minimizing the average flow time is equivalent to minimizing the total flow time. Suppose each job pays \$1 at each time unit it is alive (i. e. unfinished), then the total payment received is equal to the total flow time. Summing up the payment over each time step, the total flow time can be expressed as the summation over the number of unfinished jobs at each time unit. As SRPT works on jobs that can be finished as soon as possible, it seems intuitively that it should have the least number of unfinished jobs at any time. While this is true for a single machine, it is not true for multiple machines (as shown in an example below). The main idea of [8] was to show that at any time, the number of unfinished jobs under SRPT is "essentially" no more than $O(\min(\log P))$ times that under any other algorithm. To do this, they developed a technique of grouping jobs into a logarithmic number of classes according to their remaining sizes and arguing about the total unfinished work in these classes. This technique has found a lot of uses since then to obtain other results. To obtain a guarantee in terms of n, some additional ideas are required.

The instance below shows how SRPT could deviate from optimum in the case of multiple machines. This instance is also the key component in the lower bound construction in Theorem 2 above. Suppose there are two machines, and three jobs of size 1, 1, and 2 arrive at time $t = 0$. SRPT would schedule the two jobs of size 1 at $t = 0$ and then work on size 2 job at time $t = 1$. Thus, it has one unit of unfinished work at $t = 2$. However, the optimum could schedule the size 2 job at time 0, and finish all these jobs by time 2. Now, at time $t = 2$ three more jobs with sizes 1/2, 1/2, and 1 arrive. Again, SRPT will work on size 1/2 jobs first, and it can be seen that it will have two unfinished jobs with remaining work 1/2 each at $t = 3$, whereas the optimum can finish all these jobs by time 3. This pattern is continued by giving three jobs of size 1/4, 1/4, and 1/2 at $t = 3$ and so on. After k steps, SRPT will have k jobs with sizes $1/2, 1/4, 1/8, \ldots, 1/2^{k-2}, 1/2^{k-1}, 1/2^{k-1}$, while the optimum has no jobs remaining. Now the adversary can give 2 jobs of size $1/2^k$ each every $1/2^k$ time units for a long time, which implies that SRPT could be $\Omega(\log P)$ worse than optimum.

Leonardi and Raz also considered offline algorithms for the non-preemptive setting in their paper.

Theorem 3 ([8]) *There is a polynomial time off-line algorithm that achieves an approximation ratio of $O(n^{1/2} \log n/m)$ for minimizing average flow time on m machines without preemption.*

To prove this result, they give a general technique to convert a preemptive schedule to a non-preemptive one at the loss of an $O(n^{1/2})$ factor in the approximation ratio. They also showed an almost matching lower bound. In particular,

Theorem 4 ([8]) *No polynomial time algorithm for minimizing the total flow time on multiple machines without preemption can have an approximation ratio of* $O(n^{1/3-\varepsilon})$ *for any* $\varepsilon > 0$*, unless P=NP.*

Extensions Since the publication of these results, they have been extended in several directions. Recall that SRPT is both preemptive and migratory. Awerbuch, Azar, Leonardi, and Regev [2] gave an online scheduling algorithm that is non-migratory and still achieves a competitive ratio of $O(\min(\log(n/m), \log P))$. Avrahami and Azar [1] gave an even more restricted $O(\min(\log P, \log(n/m)))$ competitive online algorithm. Their algorithm, in addition to being non-migratory, dispatches a job immediately to a machine upon its arrival. Recently, Garg and Kumar [4,5] have extended these results to a setting where machines have non-uniform speeds. Other related problems and settings such as stretch minimization (defined as the flow time divided by the size of a job), weighted flow time minimization, and the non-clairvoyant setting where the size of a job is not unknown upon its arrival have also been investigated. The reader is referred to a recent survey by Pruhs, Sgall, and Torng [9] for more details.

Applications

The flow time measure considered here is one of the most widely used measures of quality of service, as it corresponds to the amount of time one has to wait to get the job done. The scheduling model considered here arises very naturally when there are multiple resources and several agents that compete for service from these resources. For example, consider a computing system with multiple homogeneous processors where jobs are submitted by users arbitrarily over time. Keeping the average response time low also keeps the frustration levels of the users low. The model is not necessarily limited to computer systems. At a grocery store each cashier can be viewed as a machine, and the users lining up to checkout can be viewed as jobs. The flow time of a user is time spent waiting until she finishes her transaction with the cashier. Of course, in many applications there are additional constraints such as it may be infeasible to preempt jobs, or if customers expect a certain fairness such people might prefer to be serviced in a first come first served manner at a grocery store.

Open Problems

The online algorithm of Leonardi and Raz is also the best-known offline approximation algorithm for the problem. In particular, it is not known whether an $O(1)$ approximation exists even for the case of two machines. Settling this would be very interesting. In related work, Bansal [3] considered the problem of finding non-migratory schedules for a constant number of machines. He gave an algorithm that produces a $(1+\varepsilon)$-approximate solution for any $\varepsilon > 0$ in time $n^{O(\log n/\varepsilon^2)}$. This suggests the possibility of a polynomial time approximation scheme for the problem, at least for the case of a constant number of machines.

Cross References

- ▶ Flow Time Minimization
- ▶ Multi-level Feedback Queues
- ▶ Shortest Elapsed Time First Scheduling

Recommended Reading

1. Avrahami, N., Azar, Y.: Minimizing total flow time and completion time with immediate dispacthing. In: Proceedings of 15th SPAA, pp. 11–18. (2003)
2. Awerbuch, B., Azar, Y., Leonardi, S., Regev, O.: Minimizing the flow time without migration. SIAM J. Comput. **31**, 1370–1382 (2002)
3. Bansal, N.: Minimizing flow time on a constant number of machines with preemption. Oper. Res. Lett. **33**, 267–273 (2005)
4. Garg, N., Kumar, A.: Better algorithms for minimizing average flow-time on related machines. In: Proceesings of ICALP, pp. 181–190 (2006)
5. Garg, N., Kumar, A.: Minimizing average flow time on related machines. In: ACM Symposium on Theory of Compuring (STOC), pp. 730–738 (2006)
6. Kellerer, H., Tautenhahn, T., Woeginger, G.J.: Approximability and nonapproximability results for minimizing total flow time on a single machine. SIAM J. Comput. **28**, 1155–1166 (1999)
7. Leonardi, S.: A simpler proof of preemptive flow-time approximation. Approximation and On-line Algorithms. In: Bampis, E. (ed.) Lecture Notes in Computer Science. Springer, Berlin (2003)
8. Leonardi, S., Raz, D.: Approximating total flow time on parallel machines. In: ACM Symposium on Theory of Computing (STOC), pp. 110–119 (1997)
9. Pruhs, K., Sgall, J., Torng, E.: Online scheduling. In: Handbook on Scheduling: Algorithms, Models and Performance Analysis, CRC press (2004). Symposium on Theory of Computing (STOC), pp. 110–119. (1997)

Minimum Geometric Spanning Trees

1999; Krznaric, Levcopoulos, Nilsson

Christos Levcopoulos
Department of Computer Science, Lund University, Lund, Sweden

Keywords and Synonyms

Minimum length spanning trees; Minimum weight spanning trees; Euclidean minimum spanning trees; MST; EMST

Problem Definition

Let S be a set of n points in d-dimensional real space where $d \geq 1$ is an integer constant. A *minimum spanning tree* (MST) of S is a connected acyclic graph with vertex set S of minimum total edge length. The length of an edge equals the distance between its endpoints under some metric. Under the so-called L_p metric, the distance between two points x and y with coordinates $(x_1, x_2, \ldots, x_d)$ and $(y_1, y_2, \ldots, y_d)$, respectively, is defined as the pth root of the sum $\sum_{i=1}^{d} |x_i - y_i|^p$.

Key Results

Since there is a very large number of papers concerned with geometric MSTs, only a few of them will be mentioned here.

In the common Euclidean L_2 metric, which simply measures straight-line distances, the MST problem in two dimensions can be solved optimally in time $O(n \log n)$, by using the fact that the MST is a subgraph of the Delaunay triangulation of the input point set. The latter is in turn the dual of the Voronoi diagram of S, for which there exist several $O(n \log n)$-time algorithms. The term "optimally" here refers to the algebraic computation tree model. After computation of the Delaunay triangulation, the MST can be computed in only $O(n)$ additional time, by using a technique by Cheriton and Tarjan [5].

Even for higher dimensions, i. e., when $d > 2$, it holds that the MST is a subgraph of the dual of the Voronoi diagram; however, this fact cannot be exploited in the same way as in the two-dimensional case, because this dual may contain $\Omega(n^2)$ edges. Therefore, in higher dimensions other geometric properties are used to reduce the number of edges which have to be considered. The first subquadratic-time algorithm for higher dimensions was due to Yao [14]. A more efficient algorithm was later proposed by Agarwal et al. [1]. For $d = 3$, their algorithm runs in randomized expected time $O((n \log n)^{4/3})$, and for $d \geq 4$, in expected time $O(n^{2-2/(\lceil d/2 \rceil + 1) + \epsilon})$, where ε stands for an arbitrarily small positive constant.

The algorithm by Agarwal et al. builds on exploring the relationship between computing a MST and finding a closest pair between n red points and m blue points, which is called the *bichromatic closest pair* problem. They showed that if $T_d(n, m)$ denotes the time to solve the latter problem, then a MST can be computed in $O(T_d(n, n) \log^d n)$ time. Later, Callahan and Kosaraju [4] improved this bound to $O(T_d(n, n) \log n)$. Both methods achieve running time $O(T_d(n, n))$, if $T_d(n, n) = \Omega(n^{1+\alpha})$, for some $\alpha > 0$. Finally, Krznaric et al. [10] showed that the two problems, i. e., computing a MST and computing the bichromatic closest pair, have the same worst-case time complexity (up to constant factors) in the commonly used algebraic computation tree model, and for any fixed L_p metric. The hardest part to prove is that a MST can be computed in time $O(T_d(n, n))$. The other part, which is that the bichromatic closest pair problem is not harder than computing the MST, is easy to show: if one first computes a MST for the union of the $n + m$ red and blue points, one can then find a closest bichromatic pair in linear time, because at least one such pair has to be connected by some edge of the MST.

The algorithm proposed by Krznaric et al. [10] is based on the standard approach of joining trees in a forest with the shortest edge connecting two different trees, similar to the classical Kruskal's and Prim's MST algorithms for graphs. To reduce the number of candidates to be considered as edges of the MST, the algorithm works in a sequence of phases, where in each phase only edges of equal or similar length are considered, within a factor of 2.

The initial forest is the set S of points, that is, each point of the input constitutes an individual edgeless tree. Then, as long as there is more than one tree in the forest, two trees are merged by producing an edge connecting two nodes, one from each tree. After this procedure, the edges produced comprise a single tree that remains in the forest, and this tree constitutes the output of the algorithm.

Assume that the next edge that the algorithm is going to produce has length l. Each tree T in the forest is partitioned into groups of nodes, each group having a specific node representing the group. The representative node in such a group is called a *leader*. Furthermore, every node in a group including the leader has the property that it lies within distance $\epsilon \cdot l$ from its leader, where ε is a real constant close to zero.

Instead of considering all pairs of nodes which can be candidates for the next edge to produce, first only pairs of leaders are considered. Only if a pair of leaders belong to different trees and the distance between them is approximately l, then the closest pair of points between their two respective groups is computed, using the algorithm for the bichromatic closest pair problem.

Also, the following invariant is maintained: for any phase producing edges of length $\Theta(l)$, and for any leader, there is only a constant number of other leaders at distance $\Theta(l)$. Thus, the total number of pairs of leaders to consider is only linear in the number of leaders.

Nearby leaders for any given leader can be found efficiently by using bucketing techniques and data structures for dynamic closest pair queries [3], together with extra artificial points which can be inserted and removed for probing purposes at various small boxes at distance $\Theta(l)$ from

the leader. In order to maintain the invariant, when moving to subsequent phases, one reduces the number of leaders accordingly, as pairs of nearby groups merge into single groups. Another tool which is also needed to consider the right types of pairs is to organize the groups according to the various directions in which there can be new candidate MST edges adjacent to nodes in the group. For details, please see the original paper by Krznaric et al. [10].

There is a special version of the bichromatic closest point problem which was shown by Krznaric et al. [10] to have the same worst-case time complexity as computing a MST: namely, the problem for the special case when both the set of red points and the set of blue points have a very small diameter compared with the distance between the closest bichromatic pair. This ratio can be made arbitrarily small by choosing a suitable ε as the parameter for creating the groups and leaders mentioned above. This fact was exploited in order to derive more efficient algorithms for the three-dimensional case.

For example, in the L_1 metric it is possible to build in time $O(n \log n)$ a special kind of a planar Voronoi diagram for the blue points on a plane separating the blue from the red points having the following property: for each query point q in the half-space including the red points one can use this Voronoi diagram to find in time $O(\log n)$ the blue point which is closest to q under the L_1 metric. (This planar Voronoi diagram can be seen as defined by the vertical projections of the blue points onto the plane containing the diagram, and the size of a Voronoi cell depends on the distance between the corresponding blue point and the plane.) So, by using subsequently every red point as a query point for this data structure, one can solve the bichromatic closest pair problem for such well-separated red–blue sets in total $O(n \log n)$ time.

By exploiting and building upon this idea, Krznaric et al. [10] showed how to find a MST of S in optimal $O(n \log n)$ time under the L_1 and L_∞ metrics when $d = 3$. This is an improvement over previous bounds due to Gabow et al. [9] and Bespamyatnikh [2], who proved that, for $d = 3$, a MST can be computed in $O(n \log n \log \log n)$ time under the L_1 and L_∞ metrics.

The main results of Krznaric et al. [10] are summarized in the following theorem.

Theorem *In the algebraic computation tree model, for any fixed L_p metric, and for any fixed number of dimensions, computing the MST has the same worst-case complexity, within constant factors, as solving the bichromatic closest pair problem. Moreover, for three-dimensional space under the L_1 and L_∞ metrics, the MST (as well as the bichromatic closest pair) can be computed in optimal $O(n \log n)$ time.*

Approximate and Dynamic Solutions

Callahan and Kosaraju [4] showed that a spanning tree of length within a factor $1 + \epsilon$ from that of a MST can be computed in time $O(n(\log n + \epsilon^{-d/2} \log \epsilon^{-1}))$. Approximation algorithms with worse tradeoff between time and quality had earlier been developed by Clarkson [6], Vaidya [13] and Salowe [12]. In addition, if the input point set is supported by certain basic data structures, then the approximate length of a MST can be computed in randomized sublinear time [7]. Eppstein [8] gave fully dynamic algorithms that maintain a MST when points are inserted or deleted.

Applications

MSTs belong to the most basic structures in computational geometry and in graph theory, with a vast number of applications.

Open Problems

Although the complexity of computing MSTs is settled in relation to computing bichromatic closest pairs, this means also that it remains open for all cases where the complexity of computing bichromatic closest pairs remains open, e. g., when the number of dimensions is greater than 3.

Experimental Results

Narasimhan and Zachariasen [11] have reported experiments with computing geometric MSTs via well-separated pair decompositions.

Cross References

- ▶ Degree-Bounded Trees
- ▶ Fully Dynamic Minimum Spanning Trees
- ▶ Greedy Set-Cover Algorithms
- ▶ Max Leaf Spanning Tree
- ▶ Minimum Spanning Trees
- ▶ Parallel Connectivity and Minimum Spanning Trees
- ▶ Randomized Minimum Spanning Tree
- ▶ Rectilinear Spanning Tree
- ▶ Rectilinear Steiner Tree
- ▶ Steiner Forest
- ▶ Steiner Trees

Recommended Reading

1. Agarwal, P.K., Edelsbrunner, H., Schwarzkopf, O., Welzl, E.: Euclidean minimum spanning trees and bichromatic closest pairs. Discret. Comput. Geom. **6**, 407–422 (1991)

2. Bespamyatnikh, S.: On Constructing Minimum Spanning Trees in R_1^k. Algorithmica **18**(4), 524–529 (1997)
3. Bespamyatnikh, S.: An Optimal Algorithm for Closest-Pair Maintenance. Discret. Comput. Geom. **19**(2), 175–195 (1998)
4. Callahan, P.B., Kosaraju, S.R.: Faster Algorithms for Some Geometric Graph Problems in Higher Dimensions. In: SODA 1993, pp. 291–300
5. Cheriton, D.and Tarjan, R.E.: Finding Minimum Spanning Trees. SIAM J. Comput. **5**(4), 724–742 (1976)
6. Clarkson, K.L.: Fast Expected-Time and Approximation Algorithms for Geometric Minimum Spanning Trees. In: Proc. STOC 1984, pp. 342–348
7. Czumaj, A., Ergün, F., Fortnow, L., Magen, A., Newman, I., Rubinfeld, R., Sohler, C.: Approximating the Weight of the Euclidean Minimum Spanning Tree in Sublinear Time. SIAM J. Comput. **35**(1), 91–109 (2005)
8. Eppstein, D.: Dynamic Euclidean Minimum Spanning Trees and Extrema of Binary Functions. Discret. Comput. Geom. **13**, 111–122 (1995)
9. Gabow, H.N., Bentley, J.L., Tarjan, R.E.: Scaling and Related Techniques for Geometry Problems. In: STOC 1984, pp. 135–143
10. Krznaric, D., Levcopoulos, C., Nilsson, B.J.: Minimum Spanning Trees in d Dimensions. Nord. J. Comput. **6**(4), 446–461 (1999)
11. Narasimhan, G., Zachariasen, M.: Geometric Minimum Spanning Trees via Well-Separated Pair Decompositions. ACM J. Exp. Algorithms **6**, 6 (2001)
12. Salowe, J.S.: Constructing multidimensional spanner graphs. Int. J. Comput. Geom. Appl. **1**(2), 99–107 (1991)
13. Vaidya, P.M.: Minimum Spanning Trees in k-Dimensional Space. SIAM J. Comput. **17**(3), 572–582 (1988)
14. Yao, A.C.: On Constructing Minimum Spanning Trees in k-Dimensional Spaces and Related Problems. SIAM J. Comput. **11**(4), 721–736 (1982)

Minimum k-Connected Geometric Networks

2000; Czumaj, Lingas

Artur Czumaj[1], Andrzej Lingas[2]
[1] DIMAP and Computer Science, University of Warwick, Coventry, UK
[2] Department of Computer Science, Lund University, Lund, Sweden

Keywords and Synonyms

Geometric graphs; Euclidean graphs

Problem Definition

The following classical optimization problem is considered: for a given undirected weighted geometric network, find its minimum-cost sub-network that satisfies a priori given multi-connectivity requirements.

Notations

Let $G = (V, E)$ be a *geometric network*, whose vertex set V corresponds to a set of n points in $\mathbb{R}^d$ for certain integer d, $d \geq 2$, and whose edge set E corresponds to a set of straight-line segments connecting pairs of points in V. G is called complete if E connects all pairs of points in V.

The cost $\delta(x, y)$ of an edge connecting a pair of points $x, y \in \mathbb{R}^d$ is equal to the Euclidean distance between points x and y, that is, $\delta(x, y) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$, where $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$. More generally, the cost $\delta(x, y)$ could be defined using other norms, such as ℓ_p norms for any $p > 1$, i. e., $\delta(x, y) = \left(\sum_{i=1}^{p}(x_i - y_i)^p\right)^{1/p}$. The cost of the network is equal to the sum of the costs of the edges of the network, $\text{cost}(G) = \sum_{(x,y)\in E} \delta(x, y)$.

A network $G = (V, E)$ is *spanning* a set S of points if $V = S$. $G = (V, E)$ is *k-vertex-connected* if for any set $U \subseteq V$ of fewer than k vertices, the network $(V \setminus U, E \cap ((V \setminus U) \times (V \setminus U))$ is connected. Similarly, G is *k-edge-connected* if for any set $\mathcal{E} \subseteq E$ of fewer than k edges, the network $(V, E \setminus \mathcal{E})$ is connected.

The (Euclidean) Minimum-Cost k-Vertex Connected Spanning Network Problem For a given set S of n points in the Euclidean space $\mathbb{R}^d$, find a minimum-cost k-vertex connected Euclidean network spanning points in S.

The (Euclidean) Minimum-Cost k-Edge Connected Spanning Network Problem For a given set S of n points in the Euclidean space $\mathbb{R}^d$, find a minimum-cost k-edge connected Euclidean network spanning points in S.

A variant that allows *parallel edges* is also considered:

The (Euclidean) Minimum-Cost k-Edge Connected Spanning Multi-Network Problem For a given set S of n points in the Euclidean space $\mathbb{R}^d$, find a minimum-cost k-edge connected Euclidean multi-network spanning points in S (where the multi-network can have parallel edges).

The concept of minimum-cost k-connectivity naturally extends to include that of *Euclidean Steiner k-connectivity* by allowing the use of additional vertices, called *Steiner points*. For a given set S of points in $\mathbb{R}^d$, a geometric network G is a *Steiner k-vertex connected (or, Steiner k-edge connected) for S* if the vertex set of G is a *superset* of S and for every pair of points from S there are k internally vertex-disjoint (edge-disjoint, respectively) paths connecting them in G.

The (Euclidean) Minimum-Cost Steiner k-Vertex/Edge Connectivity Problem Find a minimum-cost network

on a superset of S that is Steiner k-vertex/edge connected for S.

Note that for $k = 1$, it is simply the *Steiner minimal tree* problem, which has been very extensively studied in the literature (see, e. g., [14]).

In a more general formulation of multi-connectivity graph problems, non-uniform connectivity constraints have to be satisfied.

The Survivable Network Design Problem For a given set S of points in $\mathbb{R}^d$ and a connectivity requirement function $r : S \times S \to \mathbb{N}$, find a minimum-cost geometric network spanning points in S such that for any pair of vertices $p, q \in S$ the sub-network has $r_{p,q}$ internally vertex-disjoint (or edge-disjoint, respectively) paths between p and q.

In many applications of this problem, often regarded as the most interesting ones [9,13], the connectivity requirement function is specified with the help of a one-argument function which assigns to each vertex p its connectivity type $r_v \in \mathbb{N}$. Then, for any pair of vertices $p, q \in S$, the connectivity requirement $r_{p,q}$ is simply given as $\min\{r_p, r_q\}$ [12,13,17,18]. This includes the *Steiner tree problem* (see, e. g., [2]), in which $r_p \in \{0, 1\}$ for any vertex $p \in S$.

A *polynomial-time approximation scheme (PTAS)* is a family of algorithms $\{\mathcal{A}_\varepsilon\}$ such that, for each fixed $\varepsilon > 0$, $\mathcal{A}_\varepsilon$ runs in time polynomial in the size of the input and produces a $(1 + \varepsilon)$-approximation.

Related Work

For a very extensive presentation of results concerning problems of finding minimum-cost k-vertex- and k-edge-connected spanning subgraphs, non-uniform connectivity, connectivity augmentation problems, and geometric problems, see [1,3,11,15].

Despite the practical relevance of the multi-connectivity problems for geometrical networks and the vast amount of practical heuristic results reported (see, e. g., [12,13,17,18]), very little theoretical research had been done towards developing efficient approximation algorithms for these problems until a few years ago. This contrasts with the very rich and successful theoretical investigations of the corresponding problems in general metric spaces and for general weighted graphs. And so, until 1998, even for the simplest and most fundamental multi-connectivity problem, that of finding a minimum-cost 2-vertex connected network spanning a given set of points in the Euclidean plane, obtaining approximations achieving better than a $\frac{3}{2}$ ratio had been elusive (the ratio $\frac{3}{2}$ is the best polynomial-time approximation ratio known for general networks whose weights satisfy the triangle inequality [8]; for other results, see e. g., [4,15]).

Key Results

The first result is an extension of the well-known $\mathcal{NP}$-hardness result of minimum-cost 2-connectivity in general graphs (see, e. g., [10]) to geometric networks.

Theorem 1 *The problem of finding a minimum-cost 2-vertex/edge connected geometric network spanning a set of n points in the plane is $\mathcal{NP}$-hard.*

Next result shows that if one considers the minimum-cost multi-connectivity problems in an enough high dimension, the problems become APX-hard.

Theorem 2 ([6]) *There exists a constant $\xi > 0$ such that it is $\mathcal{NP}$-hard to approximate within $1 + \xi$ the minimum-cost 2-connected geometric network spanning a set of n points in $\mathbb{R}^{\lceil \log_2 n \rceil}$.*

This result extends also to any ℓ_p norm.

Theorem 3 ([6]) *For and integer $d \geq \log n$ and for any fixed $p \geq 1$ there exists a constant $\xi > 0$ such that it is $\mathcal{NP}$-hard to approximate within $1 + \xi$ the minimum-cost 2-connected network spanning a set of n points in the ℓ_p metric in $\mathbb{R}^d$.*

Since the minimum-cost multi-connectivity problems are hard, the research turned into the study of approximation algorithms. By combining some of the ideas developed for the polynomial-time approximation algorithms for TSP due to Arora [2] (see also [16]) together with several new ideas developed specifically for the multi-connectivity problems in geometric networks, Czumaj and Lingas obtained the following results.

Theorem 4 ([5,6]) *Let k and d be any integers, $k, d \geq 2$, and let ε be any positive real. Let S be a set of n points in $\mathbb{R}^d$. There is a randomized algorithm that in time $n \cdot (\log n)^{(kd/\varepsilon)^{\mathcal{O}(d)}} \cdot 2^{2^{(kd/\varepsilon)^{\mathcal{O}(d)}}}$ with probability at least 0.99 finds a k-vertex-connected (or k-edge-connected) spanning network for S whose cost is at most $(1 + \varepsilon)$-time optimal.*

Furthermore, this algorithm can be derandomized in polynomial-time to return a k-vertex-connected (or k-edge-connected) spanning network for S whose cost is at most $(1 + \varepsilon)$ times the optimum.

Observe that when all d, k, and ε are constant, then the running-times are $n \cdot \log^{\mathcal{O}(1)} n$.

The results in Theorem 4 give a PTAS for small values of k and d.

Theorem 5 (PTAS for vertex/edge-connectivity [6,5]) *Let $d \geq 2$ be any constant integer. There is a certain positive constant $c < 1$ such that for all k such that $k \leq (\log\log n)^c$, the problems of finding a minimum-cost k-vertex-connected spanning network and a k-edge-connected spanning network for a set of points in $\mathbb{R}^d$ admit PTAS.*

The next theorem deals with multi-networks where feasible solutions are allowed to use parallel edges.

Theorem 6 ([5]) *Let k and d be any integers, $k, d \geq 2$, and let ε be any positive real. Let S be a set of n points in $\mathbb{R}^d$. There is a randomized algorithm that in time $n \cdot \log n \cdot (d/\varepsilon)^{\mathcal{O}(d)} + n \cdot 2^{2^{(k^{\mathcal{O}(1)} \cdot (d/\varepsilon)^{\mathcal{O}(d^2)})}}$, with probability at least 0.99 finds a k-edge-connected spanning multi-network for S whose cost is at most $(1+\varepsilon)$ times the optimum. The algorithm can be derandomized in polynomial-time.*

Combining this theorem with the fact that parallel edges can be eliminated in case $k = 2$, one obtains the following result for 2-connectivity in networks.

Theorem 7 (Approximation schemes for 2-connected graphs, [5]) *Let d be any integer, $d \geq 2$, and let ε be any positive real. Let S be a set of n points in $\mathbb{R}^d$. There is a randomized algorithm that in time $n \cdot \log n \cdot (d/\varepsilon)^{\mathcal{O}(d)} + n \cdot 2^{(d/\varepsilon)^{\mathcal{O}(d^2)}}$, with probability at least 0.99 finds a 2-vertex-connected (or 2-edge-connected) spanning network for S whose cost is at most $(1+\varepsilon)$ times the optimum. This algorithm can be derandomized in polynomial-time.*

For constant d the running time of the randomized algorithms is $n \log n \cdot (1/\varepsilon)^{\mathcal{O}(1)} + 2^{(1/\varepsilon)^{\mathcal{O}(1)}}$.

Theorem 8 ([7]) *Let d be any integer, $d \geq 2$, and let ε be any positive real. Let S be a set of n points in $\mathbb{R}^d$. There is a randomized algorithm that in time $n \cdot \log n \cdot (d/\varepsilon)^{\mathcal{O}(d)} + n \cdot 2^{(d/\varepsilon)^{\mathcal{O}(d^2)}} + n \cdot 2^{2^{d^{d^{\mathcal{O}(1)}}}}$, with probability at least 0.99 finds a Steiner 2-vertex-connected (or 2-edge-connected) spanning network for S whose cost is at most $(1+\varepsilon)$ times the optimum. This algorithm can be derandomized in polynomial-time.*

Theorem 9 ([7]) *Let d be any integer, $d \geq 2$, and let ε be any positive real. Let S be a set of n points in $\mathbb{R}^d$. There is a randomized algorithm that in time $n \cdot \log n \cdot (d/\varepsilon)^{\mathcal{O}(d)} + n \cdot 2^{(d/\varepsilon)^{\mathcal{O}(d^2)}} + n \cdot 2^{2^{d^{d^{\mathcal{O}(1)}}}}$, with probability at least 0.99 gives a $(1+\varepsilon)$-approximation for the geometric network survivability problem with $r_v \in \{0,1,2\}$ for any $v \in V$. This algorithm can be derandomized in polynomial-time.*

Applications

Multi-connectivity problems are central in algorithmic graph theory and have numerous applications in computer science and operation research, see, e. g., [1,13,11,18]. They also play very important role in the design of networks that arise in practical situations, see, e. g., [1,13]. Typical application areas include telecommunication, computer and road networks. Low degree connectivity problems for geometrical networks in the plane can often closely *approximate* such practical connectivity problems (see, e. g., the discussion in [13,17,18]). The survivable network design problem in geometric networks also arises in many applications, e. g., in telecommunication, communication network design, VLSI design, etc. [12,13,17,18].

Open Problems

The results discussed above lead to efficient algorithms only for small connectivity requirements k; the running-time is polynomial only for the value of k up to $(\log\log n)^c$ for certain positive constant $c < 1$. It is an interesting open problem if one can obtain polynomial-time approximation schemes algorithms also for large values of k.

It is also an interesting open problem if the multi-connectivity problems in geometric networks can have practically fast approximation schemes.

Cross References

- ► Euclidean Traveling Salesperson Problem
- ► Minimum Geometric Spanning Trees

Recommended Reading

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B., Reddy, M.R.: Applications of network optimization. In: Handbooks in Operations Research and Management Science, vol. 7, Network Models, chapter 1, pp. 1–83. North-Holland, Amsterdam (1995)
2. Arora, S.: Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. J. ACM **45**(5), 753–782 (1998)
3. Berger, A., Czumaj, A., Grigni, M., Zhao, H.: Approximation schemes for minimum 2-connected spanning subgraphs in weighted planar graphs. Proc. 13th Annual European Symposium on Algorithms, pp. 472–483. (2005)
4. Cheriyan, J., Vetta, A.: Approximation algorithms for network design with metric costs. Proc. 37th Annual ACM Symposium on Theory of Computing, Baltimore, 22–24 May 2005, pp. 167–175. (2005)
5. Czumaj, A., Lingas, A.: Fast approximation schemes for Euclidean multi-connectivity problems. Proc. 27th Annual International Colloquium on Automata, Languages and Programming, Geneva, 9–15 July 2000, pp. 856–868

6. Czumaj, A., Lingas, A.: On approximability of the minimum-cost *k*-connected spanning subgraph problem. Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, 17–19 January 1999, pp. 281–290
7. Czumaj, A., Lingas, A., Zhao, H.: Polynomial-time approximation schemes for the Euclidean survivable network design problem. Proc. 29th Annual International Colloquium on Automata, Languages and Programming, Malaga, 8–13 July 2002, pp. 973–984
8. Frederickson, G.N., JáJá, J.: On the relationship between the biconnectivity augmentation and Traveling Salesman Problem. Theor. Comput. Sci. **19**(2), 189–201 (1982)
9. Gabow, H.N., Goemans, M.X., Williamson, D.P.: An efficient approximation algorithm for the survivable network design problem. Math. Program. Ser. B **82**(1–2), 13–40 (1998)
10. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman, New York, NY (1979)
11. Goemans, M.X., Williamson, D.P.: The primal-dual method for approximation algorithms and its application to network design problems. In: Hochbaum, D. (ed.) Approximation Algorithms for $\mathcal{NP}$-Hard Problems, Chapter 4, pp. 144–191. PWS Publishing Company, Boston (1996)
12. Grötschel, M., Monma, C.L., Stoer, M.: Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. Oper. Res. **40**(2), 309–330 (1992)
13. Grötschel, M., Monma, C.L., Stoer, M.: Design of survivable networks. In: Handbooks in Operations Research and Management Science, vol. 7, Network Models, chapter 10, pp. 617–672. North-Holland, Amsterdam (1995)
14. Hwang, F.K., Richards, D.S., Winter, P.: The Steiner Tree Problem. North-Holland, Amsterdam (1992)
15. Khuller, S.: Approximation algorithms for finding highly connected subgraphs. In: Hochbaum, D. (ed.) Approximation Algorithms for $\mathcal{NP}$-Hard Problems, Chapter 6, pp. 236–265. PWS Publishing Company, Boston (1996)
16. Mitchell, J.S.B.: Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, *k*-MST, and related problems. SIAM J. Comput. **28**(4), 1298–1309 (1999)
17. Monma, C.L., Shallcross, D.F.: Methods for designing communications networks with certain two-connected survivability constraints. Operat. Res. **37**(4), 531–541 (1989)
18. Stoer, M.: Design of Survivable Networks. Springer, Berlin (1992)

Minimum Makespan on Unrelated Machines

1990; Lenstra, Shmoys, Tardos

MAXIM SVIRIDENKO
IBM Research, IBM, Yorktown Heights, NY, USA

Keywords and Synonyms

Schedule of minimum length on different machines

Problem Definition

Consider the following scheduling problem. There are m parallel machines and n independent jobs. Each job is to be assigned to one of the machines. The processing of job j on machine i requires p_{ij} units of time. The objective is to find a schedule that minimizes the makespan, defined to be the time by which all jobs are completed. This problem is denoted $R||C_{\max}$ using standard scheduling notation terminology [6].

There are few important special cases of the problem: the restricted assignment problem with $p_{ij} \in \{1, \infty\}$, the identical parallel machines with $p_{ij} = p_j$ and the uniform parallel machines $p_{ij} = p_j/s_i$ where $s_i > 0$ is a speed of machine i. These problems are denoted $R|p_{ij} \in \{1, \infty\}|C_{\max}$, $P||C_{\max}$ and $Q||C_{\max}$, respectively. Two later problems admit polynomial time approximation schemes [4,5].

Consider the following integer programming formulation of the feasibility problem that finds a feasible assignment of jobs to machines with makespan at most T

$$\sum_{i=1}^{m} x_{ij} = 1\,, \quad j = 1, \ldots, n\,, \tag{1}$$

$$\sum_{j=1}^{n} p_{ij} x_{ij} \le T\,, \quad i = 1, \ldots, m\,, \tag{2}$$

$$x_{ij} = 0\,, \quad \text{if} \quad p_{ij} > T\,, \tag{3}$$

$$x_{ij} \in \{0, 1\}\,, \quad \forall i, j\,. \tag{4}$$

The variable $x_{ij} = 1$ if job j is assigned to machine i and $x_{ij} = 0$, otherwise. The constraint (1) corresponds to job assignments. The constraint (2) bounds the total processing time of jobs assigned to one machine. The constraint (3) forbids an assignment of a job to a machine if its processing time is larger than the target makespan T.

Key Results

Theorem 1 (Rounding Theorem) *Consider the linear programming relaxation of the integer program (1)–(4) by relaxing the integrality constraint (4) with the constraint*

$$x_{ij} \ge 0\,, \quad \forall i, j\,. \tag{5}$$

If the linear program (1)–(3),(5) has a feasible solution for some value of parameter $T = T^*$ then there exists a feasible solution to an integer program (1)–(4) with parameter $T = T^* + p_{\max}$ where $p_{\max} = \max_{i,j} p_{ij}$ and such a solution can be found in polynomial time.

The idea of the proof is to start with a basic feasible solution of the linear program (1)–(3),(5). The properties of basic solutions imply the bound on the number of fractional variables which in turn implies that the bipartite graph defined between jobs and machines with edges corresponding to fractional variables has a very special structure. Lenstra, Shmoys and Tardos [7] show that it is possible to round fractional edges in such a way that each machine node has at most one edge (variable) rounded up which implies the bound on the makespan.

The Theorem 1 combined with binary search on parameter T implies

Corollary 1 *There is a 2-approximation algorithm for the makespan minimization problem on unrelated parallel machines that runs in time polynomial in the input size.*

Lenstra, Shmoys and Tardos [7] proved an inapproximability result that is valid even for the case of the restricted assignment problem

Theorem 2 *For every $\rho < 3/2$ there does not exist a polynomial ρ-approximation algorithm for the makespan minimization problem on unrelated parallel machines unless $P = NP$.*

Generalizations

A natural generalization of the scheduling problem is to add additional resource requirements $\sum_{i,j} c_{ij} x_{ij} \leq B$, i. e. there is a cost c_{ij} associated with assigning job i to machine j. The goal is to find an assignment of jobs to machines of total cost at most B minimizing the makespan. This problem is known under the name of the generalized assignment problem. Shmoys and Tardos [8] proved an analogous Rounding Theorem leading to a 2-approximation algorithm.

Even more general problem arises when each machine i has few modes $s = 1, \ldots, k$ to process job j. Each mode has the processing time p_{ijs} and the cost c_{ijs} associated with it. The goal is to find a an assignment of jobs to machines and modes of total cost at most B minimizing the makespan. Consider the following analogous integer programming formulation of the problem

$$\sum_{i=1}^{m} \sum_{s=0}^{k} x_{ijs} = 1 \,, \quad j = 1, \ldots, n \,, \tag{6}$$

$$\sum_{j=1}^{n} \sum_{s=0}^{k} x_{ijs}\, p_{ijs} \leq T \,, \quad i = 1, \ldots, m \,, \tag{7}$$

$$\sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{s=0}^{k} x_{ijs}\, c_{ijs} \leq B \,, \tag{8}$$

$$x_{ijs} = 0 \,, \quad \text{if } p_{ijs} > T \,, \tag{9}$$

$$x_{ijs} \in \{0, 1\} \,, \quad \forall\, i, j, s \,. \tag{10}$$

Theorem 3 (General Rounding Theorem) *Consider the linear programming relaxation of the integer program (6)–(10) by relaxing the integrality constraint (10) with the constraint*

$$x_{ijs} \geq 0 \,, \quad \forall i, j, s \,. \tag{11}$$

If linear program (6)–(9),(11) has a feasible solution for some value of parameter $T = T^*$ then there exists a feasible solution to the integer program (6)–(10) with parameter $T = T^* + p_{\max}$ where $p_{\max} = \max_{i,j,s} p_{ijs}$ and such a solution can be found in polynomial time.

The randomized version of this Theorem was originally proved by Gandhi, Khuller, Parthasarathy and Srinivasan [2]. The deterministic version appeared first in [3].

Applications

Unrelated parallel machine scheduling is one of the basic scheduling models with a lot of industrial applications, see for example [1, 9]. The rounding Theorem by Lenstra, Shmoys, Tardos and its generalizations have found numerous applications applications in design and analysis of approximation algorithms where quite often generalized assignment problem needs to be solved as a subroutine.

Open Problems

The most exciting open problem is to close the gap between positive (Corollary 1) and negative (Theorem 2) results for $R||C_{\max}$. A very simple example shows that the integrality gap of the linear programming relaxation (1)–(3),(5) is 2 and therefore there is a need for a stronger LP to improve upon 2-approximation.

This example consists of $m(T-1)$ jobs such that for each $i \in 1, \ldots, m$, processing time $p_{ij} = 1$ for $j = (T-1)(i-1)+1, \ldots, (T-1)i$ and $p_{ij} = \infty$ otherwise. In other words, each machine has $T-1$ jobs with unit processing time, that cannot be processed on any other machine. Additionally, there is one large job b with processing time $p_{ib} = T$ for $i = 1, \ldots, m$.

One way to define a stronger LP is to define variables x_{iS} for each possible set S of jobs. The variable $x_{iS} = 1$ if the set S of jobs is assigned to be processed on machine i. The set of jobs S is feasible for machine i if $\sum_{j \in S} p_{ij} \leq T$. Let C_i be the set of feasible sets for machine i. Consider the following linear programming relaxation:

$$\sum_{i, S \in C_i : j \in S} x_{iS} = 1 \,, \quad j = 1, \ldots, n \,, \tag{12}$$

$$\sum_{S \in C_i} x_{iS} = 1\,, \quad i = 1, \ldots, m\,, \tag{13}$$

$$x_{iS} \geq 0\,, \quad \forall\, i, S \in C_i\,. \tag{14}$$

The integrality gap of this linear program is also 2 for general unrelated parallel machine scheduling but it is open for the special case of restricted assignment problem.

Cross References

- ▶ Flow Time Minimization
- ▶ List Scheduling
- ▶ Load Balancing
- ▶ Minimum Flow Time
- ▶ Minimum Weighted Completion Time

Recommended Reading

1. Jeng-Fung, C.: Unrelated parallel machine scheduling with secondary resource constraints. Int. J. Adv. Manuf. Technol. **26**, 285–292 (2005)
2. Gandhi, R., Khuller, S., Parthasarathy, S., Srinivasan, A.: Dependent rounding and its applications to approximation algorithms. J. ACM **53**(3), 324–360 (2006)
3. Grigoriev, A., Sviridenko, M., Uetz, M.: Machine scheduling with resource dependent processing times. Math. Program. **110**(1B), 209–228 (2002)
4. Hochbaum, D.S., Shmoys, D.B.: Using dual approximation algorithms for scheduling problems: theoretical and practical results. J. Assoc. Comput. Mach. **34**(1), 144–162 (1987)
5. Hochbaum, D.S., Shmoys, D.B.: A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach. SIAM J. Comput. **17**(3), 539–551 (1988)
6. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: Sequencing and Scheduling: Algorithms and Complexity. In: Graves, S.C., Rinnooy Kan, A.H.G., Zipkin, P.H. (eds.) Logistics of Production and Inventory. Handbooks in Operations Research and Management Science, vol. 4, pp. 445–522. North–Holland, Amsterdam (1993)
7. Lenstra, J.K., Shmoys, D., Tardos, E.: Approximation algorithms for scheduling unrelated parallel machines. Math. Program. **46**(3A), 259–271 (1990)
8. Shmoys, D., Tardos, E.: An approximation algorithm for the generalized assignment problem. Math. Program. **62**(3A), 461–474 (1993)
9. Yu, L., Shih, H., Pfund, M., Carlyle, W., Fowler, J.: Scheduling of unrelated parallel machines: an application to PWB manufacturing. IIE Trans. **34**, 921–931 (2002)

Minimum Spanning Trees

2002; Pettie, Ramachandran

Seth Pettie
Department of Computer Science, University of Michigan, Ann Arbor, Ann Arbor, MI, USA

Keywords and Synonyms

Minimal spanning tree; Minimum weight spanning tree; Shortest spanning tree

Problem Definition

The *minimum spanning tree* (MST) problem is, given a connected, weighted, and undirected graph $G = (V, E, w)$, to find the tree with minimum total *weight* spanning all the vertices V. Here $w\colon E \to \mathbb{R}$ is the weight function. The problem is frequently defined in geometric terms, where V is a set of points in d-dimensional space and w corresponds to Euclidean distance. The main distinction between these two settings is the form of the input. In the graph setting the input has size $O(m + n)$ and consists of an enumeration of the $n = |V|$ vertices and $m = |E|$ edges and edge weights. In the geometric setting the input consists of an enumeration of the coordinates of each point ($O(dn)$ space): all $\binom{V}{2}$ edges are implicitly present and their weights implicit in the point coordinates. See [16] for a discussion of the Euclidean minimum spanning tree problem.

History

The MST problem is generally recognized [7,12] as one of the first combinatorial problems studied specifically from an algorithmic perspective. It was formally defined by Borůvka in 1926 [1] (predating the fields of computability theory and combinatorial optimization, and even much of graph theory) and since his initial algorithm there has been a sustained interest in the problem. The MST problem has motivated research in matroid optimization [3] and the development of efficient data structures, particularly priority queues (aka heaps) and disjoint set structures [2,18].

Related Problems

The MST problem is frequently contrasted with the *traveling salesman* and *minimum Steiner tree* problems [6]. A Steiner tree is a tree that may span any *superset* of the given points; that is, additional points may be introduced that reduce the weight of the minimum spanning tree. The traveling salesman problem asks for a tour (cycle) of the vertices with minimum total length. The generalization of the MST problem to directed graphs is sometimes called the *minimum branching* [5]. Whereas the undirected and directed versions of the MST problem are solvable in polynomial time, traveling salesman and minimum Steiner tree are NP-complete [6].

Optimality Conditions

A *cut* is a partition (V', V'') of the vertices V. An edge (u, v) *crosses* the cut (V', V'') if $u \in V'$ and $v \in V''$. A sequence $(v_0, v_1, \ldots, v_{k-1}, v_0)$ is a *cycle* if $(v_i, v_{i+1(\mathrm{mod}\ k)}) \in E$ for $0 \leq i < k$.
The correctness of all MST algorithms is established by appealing to the dual *cut* and *cycle* properties, also known as the blue rule and red rule [18].

Cut Property An edge is in some minimum spanning tree if and only if it is the lightest edge crossing some cut.

Cycle Property An edge is not in any minimum spanning tree if and only if it is the sole heaviest edge on some cycle.

It follows from the cut and cycle properties that if the edge weights are unique then there is a unique minimum spanning tree, denoted **MST**(G). Uniqueness can always be enforced by breaking ties in any consistent manner. MST algorithms frequently appeal to a useful corollary of the cut and cycle properties called the *contractibility* property. Let $G \setminus C$ denote the graph derived from G by contracting the subgraph C, that is, C is replaced by a single vertex c and all edges incident to exactly one vertex in C become incident to c; in general $G \setminus C$ may have more than one edge between two vertices.

Contractibility Property If C is a subgraph such that for all pairs of edges e and f with exactly one endpoint in C, there exists a path $P \subseteq C$ connecting ef with each edge in P lighter than either e or f, then C is *contractible*. For any contractible C it holds that **MST**(G) = **MST**$(C) \cup$ **MST**$(G \setminus C)$.

The Generic Greedy Algorithm

Until recently all MST algorithms could be viewed as mere variations on the following generic greedy MST algorithm. Let $\mathcal{T}$ consist initially of n trivial trees, each containing a single vertex of G. Repeat the following step $n - 1$ times. Choose any $T \in \mathcal{T}$ and find the minimum weight edge (u, v) with $u \in T$ and v in a different tree, say $T' \in \mathcal{T}$. Replace T and T' in $\mathcal{T}$ with the single tree $T \cup \{(u, v)\} \cup T'$. After $n - 1$ iterations $\mathcal{T} = \{$**MST**$(G)\}$. By the cut property every edge selected by this algorithm is in the MST.

Modeling MST Algorithms

Another corollary of the cut and cycle properties is that the set of minimum spanning trees of a graph is determined solely by the relative order of the edge weights—their specific numerical values are not relevant. Thus, it is natural to model MST algorithms as *binary decision trees*, where nodes of the decision tree are identified with edge weight comparisons and the children of a node correspond to the possible outcomes of the comparison. In this decision tree model a trivial lower bound on the *time* of the optimal MST algorithm is the *depth* of the optimal decision tree.

Key Results

The primary result of [14] is an explicit MST algorithm that is *provably* optimal even though its asymptotic running time is currently unknown.

Theorem 1 *There is an explicit, deterministic minimum spanning tree algorithm whose running time is on the order of $D_{MST}(m, n)$, where m is the number of edges, n the number of vertices, and $D_{MST}(m, n)$ the maximum depth of an optimal decision tree for any m-edge n-node graph.*

It follows that the Pettie–Ramachandran algorithm [14] is asymptotically no worse than *any* MST algorithm that deduces the solution through edge weight comparisons. The best known upper bound on $D_{MST}(m, n)$ is $O(m\alpha(m, n))$, due to Chazelle [2]. It is trivially $\Omega(m)$.
Let us briefly describe how the Pettie–Ramachandran algorithm works. An (m, n) instance is a graph with m edges and n vertices. Theorem 1 is proved by giving a linear time *decomposition* procedure that reduces any (m, n) instance of the MST problem to instances of size $(m^*, n^*), (m_1, n_1), \ldots, (m_s, n_s)$, where $m = m^* + \sum_i m_i$, $n = \sum_i n_i$, $n^* \leq n/\log\log\log n$ and each $n_i \leq \log\log\log n$. The (m^*, n^*) instance can be solved in $O(m + n)$ time with existing MST algorithms [2]. To solve the other instances the Pettie–Ramachandran algorithm performs a brute-force search to find the minimum depth decision tree for *every* graph with at most $\log\log\log n$ vertices. Once these decision trees are found the remaining instances are solved in $O(\sum_i D_{MST}(m_i, n_i)) = O(D_{MST}(m, n))$ time. Due to the restricted size of these instances ($n_i \leq \log\log\log n$) the time for a brute force search is a negligible $o(n)$. The decomposition procedure makes use of Chazelle's *soft heap* [2] (an approximate priority queue) and an extension of the contractibility property.

Approximate Contractibility Let G' be derived from G by *increasing* the weight of some edges. If C is contractible w.r.t. G' then **MST**(G) = **MST**(**MST**$(C) \cup$ **MST**$(G \setminus C) \cup E^*)$, where E^* is the set of edges with increased weights.

A secondary result of [14] is that the running time of the optimal algorithm is actually linear on nearly ev-

ery graph topology, under any permutation of the edge weights.

Theorem 2 *Let G be selected uniformly at random from the set of all n-vertex, m-edge graphs. Then regardless of the edge weights,* **MST**(G) *can be found in $O(m + n)$ time with probability $1 - 2^{-\Omega(m/\alpha^2)}$, where $\alpha = \alpha(m, n)$ is the slowly growing inverse-Ackermann function.*

Theorem 1 should be contrasted with the results of Karger, Klein, and Tarjan [9] and Chazelle [2] on the randomized and deterministic complexity of the MST problem.

Theorem 3 [9] *The minimum spanning forest of a graph with m edges can be computed by a randomized algorithm in $O(m)$ time with probability $1 - 2^{-\Omega(m)}$.*

Theorem 4 [2] *The minimum spanning tree of a graph can be computed in $O(m\alpha(m, n))$ time by a deterministic algorithm, where α is the inverse-Ackermann function.*

Applications

Borůvka [1] invented the MST problem while considering the practical problem of electrifying rural Moravia (present day Czech Republic) with the shortest electrical network. MSTs are used as a starting point for heuristic approximations to the optimal traveling salesman tour and optimal Steiner tree, as well as other network design problems. MSTs are a component in other graph optimization algorithms, notably the single-source shortest path algorithms of Thorup [19] and Pettie–Ramachandran [15]. MSTs are used as a tool for visualizing data that is presumed to have a tree structure; for example, if a matrix contains dissimilarity data for a set of species, the minimum spanning tree of the associated graph will presumably group closely related species; see [7]. Other modern uses of MSTs include modeling physical systems [17] and image segmentation [8]; see [4] for more applications.

Open Problems

The chief open problem is to determine the *deterministic* complexity of the minimum spanning tree problem. By Theorem 1 this is tantamount to determining the decision-tree complexity of the MST problem.

Experimental Results

Moret and Shapiro [11] evaluated the performance of greedy MST algorithms using a variety of priority queues. They concluded that the best MST algorithm is Jarník's [7] (also attributed to Prim and Dijkstra; see [3,7,12]) as implemented with a pairing heap [13]. Katriel, Sanders, and Träff [10] designed and implemented a non-greedy randomized MST algorithm based on that of Karger et al. [9]. They concluded that on moderately dense graphs it runs substantially faster than the greedy algorithms tested by Moret and Shapiro.

Cross References

► Randomized Minimum Spanning Tree

Recommended Reading

1. Borůvka, O.: O jistém problému minimálním. Práce Moravské Přírodovědecké Společnosti **3**, 37–58 (1926). In Czech
2. Chazelle, B.: A minimum spanning tree algorithm with inverse-Ackermann type complexity. J. ACM **47**(6), 1028–1047 (2000)
3. Cormen, TH., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2001)
4. Eppstein, D.: Geometry in action: minimum spanning trees. http://www.ics.uci.edu/~eppstein/gina/mst.html
5. Gabow, H.N., Galil, Z., Spencer, T.H., Tarjan, R.E.: Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica **6**, 109–122 (1986)
6. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to NP-completeness. Freeman, San Francisco (1979)
7. Graham, R.L., Hell, P.: On the history of the minimum spanning tree problem. Ann. Hist. Comput. **7**(1), 43–57 (1985)
8. Ion, A., Kropatsch, W.G., Haxhimusa, Y.: Considerations regarding the minimum spanning tree pyramid segmentation method. In: Proc. 11th Workshop Structural, Syntactic, and Statistical Pattern Recognition (SSPR). LNCS, vol. 4109, pp. 182–190. Springer, Berlin (2006)
9. Karger, D.R., Klein, P.N., Tarjan, R.E.: A randomized linear-time algorithm for finding minimum spanning trees. J. ACM **42**, 321–329 (1995)
10. Katriel, I., Sanders, P., Träff, J.L.: A practical minimum spanning tree algorithm using the cycle property. In: Proc. 11th Annual European Symposium on Algorithms. LNCS, vol. 2832, pp. 679–690. Springer, Berlin (2003)
11. Moret, B.M.E., Shapiro, H.D.: An empirical assessment of algorithms for constructing a minimum spanning tree. In: DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 15, Am. Math. Soc., Providence, RI (1994)
12. Pettie, S.: On the shortest path and minimum spanning tree problems. Ph.D. thesis, The University of Texas, Austin, August 2003
13. Pettie, S.: Towards a final analysis of pairing heaps. In: Proc. 46th Annual Symposium on Foundations of Computer Science (FOCS), 2005, pp. 174–183
14. Pettie, S., Ramachandran, V.: An optimal minimum spanning tree algorithm. J. ACM **49**(1), 16–34 (2002)
15. Pettie, S., Ramachandran, V.: A shortest path algorithm for real-weighted undirected graphs. SIAM J. Comput. **34**(6), 1398–1431 (2005)
16. Preparata, F.P., Shamos, M.I.: Computational geometry. Springer, New York (1985)
17. Subramaniam, S., Pope, S.B.: A mixing model for turbulent reactive flows based on euclidean minimum spanning trees. Combust. Flame **115**(4), 487–514 (1998)

18. Tarjan, R.E.: Data structures and network algorithms. In: CBMS-NSF Reg. Conf. Ser. Appl. Math., vol. 44. SIAM, Philadelphia (1983)
19. Thorup, M.: Undirected single-source shortest paths with positive integer weights in linear time. J. ACM **46**(3), 362–394 (1999)

Minimum Weighted Completion Time

1999; Afrati et al.

V.S. Anil Kumar[1], Madhav V. Marathe[2], Srinivasan Parthasarathy[3], Aravind Srinivasan[4]

[1] Network Dynamics and Simulation Science Laboratory, Bioinformatics Institute, Virginia Tech, Blacksburg, VA, USA
[2] Department of Computer Science and Virginia Bioinformatics Institute, Virginia Tech, Blacksburg, VA, USA
[3] IBM T.J. Watson Research Center, Hawthorne, NY, USA
[4] Department of Computer Science, University of Maryland, College Park, MD, USA

Keywords and Synonyms

Average weighted completion time

Problem Definition

The minimum weighted completion time problem involves (i) a set J of n jobs, a positive weight w_j for each job $j \in J$, and a release date r_j before which it cannot be scheduled; (ii) a set of m machines, each of which can process at most one job at any time; and (iii) an arbitrary set of positive values $\{p_{i,j}\}$, where $p_{i,j}$ denotes the time to process job j on machine i. A schedule involves assigning jobs to machines and choosing an order in which they are processed. Let C_j denote the completion time of job j for a given schedule. The *weighted completion time* of a schedule is defined as $\sum_{j \in J} w_j C_j$, and the goal is to compute a schedule that has the minimum weighted completion time.

In the scheduling notation introduced by Graham et al. [7], a scheduling problem is denoted by a 3-tuple $\alpha|\beta|\gamma$, where α denotes the machine environment, β denotes the additional constraints on jobs, and γ denotes the objective function. In this article, we will be concerned with the α-values 1, P, R, and Rm, which respectively denote one machine, identical parallel machines (i. e., for a fixed job j and for each machine i, $p_{i,j}$ equals a value p_j that is independent of i), unrelated machines (the $p_{i,j}$'s are dependent on both job i and machine j), and a fixed number m (not part of the input) of unrelated machines. The field β takes on the values r_j, which indicates that the jobs have release dates, and the value *pmtn*, which indicates that preemption of jobs is permitted. Further, the value *prec* in the field β indicates that the problem may involve precedence constraints between jobs, which poses further restrictions on the schedule. The field γ is either $\sum w_j C_j$ or $\sum C_j$, which denote total weighted and total (unweighted) completion times, respectively.

Some of the simpler classes of the weighted completion time scheduling problems admit optimal polynomial-time solutions. They include the problem $P||\sum C_j$, for which the *shortest-job-first* strategy is optimal, the problem $1||\sum w_j C_j$, for which Smith's rule [13] (scheduling jobs in their nondecreasing order of p_j/w_j values) is optimal, and the problem $R||\sum C_j$, which can be solved via matching techniques [2,9]. With the introduction of release dates, even the simplest classes of the weighted completion time minimization problem becomes strongly nondeterministic polynomial-time (NP)-hard. In this article, we focus on the work of Afrati et al. [1], whose main contribution is the design of polynomial-time approximation schemes (PTASs) for several classes of scheduling problems to minimize weighted completion time *with release dates*. Prior to this work, the best solutions for minimizing weighted completion time with release dates were all $O(1)$-approximation algorithms (e. g., [4,5,11]); the only known PTAS for a strongly NP-hard problem involving weighted completion time was due to Skutella and Woeginger [12], who developed a PTAS for the problem $P||\sum w_j C_j$. For an excellent survey on the minimum weighted completion time problem, we refer the reader to Chekuri and Khanna [3].

Key Results

Afrati et al. [1] were the first to develop PTASs for weighted completion time problems involving release dates. We summarize the running times of these PTASs in Table 1.

The results presented in Table 1 were obtained through a careful sequence of input transformations followed by dynamic programming. The input transformations ensure that the input becomes *well structured* at a slight loss in optimality, while dynamic programming allows efficient enumeration of all the near-optimal solutions to the well-structured instance.

The first step in the input transformation is *geometric rounding*, in which the processing times and release dates are converted to powers of $1+\epsilon$, with at most $1+\epsilon$ loss in the overall performance. More significantly, this step

Minimum Weighted Completion Time, Table 1
Summary of results of Afrati et al. [1]

Problem	Running time of polynomial-time approximation schemes
$1\|r_j\|\sum w_jC_j$	$O(2^{poly(\frac{1}{\epsilon})}n + n\log n)$
$P\|r_j\|\sum w_jC_j$	$O((m+1)^{poly(\frac{1}{\epsilon})}n + n\log n)$
$P\|r_j, pmtn\|\sum w_jC_j$	$O(2^{poly(\frac{1}{\epsilon})}n + n\log n)$
$Rm\|r_j\|\sum w_jC_j$	$O(f(m,\frac{1}{\epsilon})poly(n))$
$Rm\|r_j, pmtn\|\sum w_jC_j$	$O(f(m,\frac{1}{\epsilon})n + n\log n)$
$Rm\|\|\sum w_jC_j$	$O(f(m,\frac{1}{\epsilon})n + n\log n)$

(i) ensures that there are only a small number of distinct processing times and release dates to deal with, (ii) allows time to be broken into geometrically increasing intervals, and (iii) aligns release dates with start and end times of intervals. These are useful properties that can be exploited by dynamic programming.

The second step in the input transformation is *time stretching*, in which small amounts of idle time are added throughout the schedule. This step also changes completion times by a factor of at most $1 + O(\epsilon)$, but is useful for *cleaning up* the scheduling. Specifically, if a job is *large* (i. e., occupies a large portion of the interval where it executes), it can be pushed into the idle time of a later interval where it is small. This ensures that most jobs have small sizes compared with the length of the intervals where they execute, which greatly simplifies schedule computation. The next step is *job shifting*. Consider a partition of the time interval $[0, \infty)$ into intervals of the form $I_x = [(1+\epsilon)^x, (1+\epsilon)^{x+1})$, for integral values of x. The job-shifting step ensures that there is a slightly suboptimal schedule in which every job j gets completed within $O(\log_{1+\epsilon}(1+\frac{1}{\epsilon}))$ intervals after r_j. This has the following nice property: If we consider blocks of intervals $\mathcal{B}_0, \mathcal{B}_1, \ldots$, with each block $\mathcal{B}_i$ containing $O(\log_{1+\epsilon}(1+\frac{1}{\epsilon}))$ consecutive intervals, then a job j starting in block $\mathcal{B}_i$ completes within the next block. Further, the other steps in the job-shifting phase ensure that there are not too many *large* jobs which spill over to the next block; this allows the dynamic programming to be done efficiently.

The precise steps in the algorithms and their analysis are subtle, and the above description is clearly an oversimplification. We refer the reader to [1] or [3] for further details.

Applications

A number of optimization problems in parallel computing and operations research can be formulated as machine scheduling problems. When precedence constraints are introduced between jobs, the weighted completion time objective can generalize the more commonly studied makespan objective, and hence is important.

Open Problems

Some of the major open problems in this area are to improve the approximation ratios for scheduling on unrelated or related machines for jobs with precedence constraints. The following problems in particular merit special mention. The best known solution for the $1|prec|\sum w_jC_j$ problem is the 2-approximation algorithm due to Hall et al. [8]; improving upon this factor is a major open problem in scheduling theory. The problem $R|prec|\sum_j w_jC_j$ in which the precedence constraints form an arbitrary acyclic graph is especially open – the only known results in this direction are when the precedence constraints form chains [6] or trees [10].

The other open direction is inapproximability – there are significant gaps between the known approximation guarantees and hardness factors for various problem classes. For instance, the $R||\sum w_jC_j$ and $R|r_j|\sum w_jC_j$ are both known to be approximable-hard, but the best known algorithms for these problems (due to Skutella [11]) have approximation ratios of 3/2 and 2 respectively. Closing these gaps remain a significant challenge.

Cross References

- ► Flow Time Minimization
- ► List Scheduling
- ► Minimum Flow Time
- ► Minimum Makespan on Unrelated Machines

Acknowledgements

The Research of V.S. Anil Kumar and M.V. Marathe was supported in part by NSF Award CNS-0626964. A. Srinivasan's research was supported in part by NSF Award CNS-0626636.

Recommended Reading

1. Afrati, F.N., Bampis, E., Chekuri, C., Karger, D.R., Kenyon, C., Khanna, S., Milis, I., Queyranne, M., Skutella, M., Stein, C., Sviridenko, M.: Approximation schemes for minimizing average weighted completion time with release dates. In: Proc. of Foundations of Computer Science, pp. 32–44 (1999)
2. Bruno, J.L., Coffman, E.G., Sethi, R.: Scheduling independent tasks to reduce mean finishing time. Commun. ACM **17**, 382–387 (1974)
3. Chekuri, C., Khanna, S.: Approximation algorithms for minimizing weighted completion time. In: J. Y-T. Leung (eds.) Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, Boca Raton (2004)

4. Chekuri, C., Motwani, R., Natarajan, B., Stein, C.: Approximation techniques for average completion time scheduling. SIAM J. Comput. **31**(1), 146–166 (2001)
5. Goemans, M., Queyranne, M., Schulz, A., Skutella, M., Wang, Y.: Single machine scheduling with release dates. SIAM J. Discret. Math. **15**, 165–192 (2002)
6. Goldberg, L.A., Paterson, M., Srinivasan, A., Sweedyk, E.: Better approximation guarantees for job-shop scheduling. SIAM J. Discret. Math. **14**, 67–92 (2001)
7. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discret. Math. **5**, 287–326 (1979)
8. Hall, L.A., Schulz, A.S., Shmoys, D.B., Wein, J.: Scheduling to minimize average completion time: off-line and on-line approximation algorithms. Math. Oper. Res. **22**(3), 513–544 (1997)
9. Horn, W.: Minimizing average flow time with parallel machines. Oper. Res. **21**, 846–847 (1973)
10. Kumar, V.S.A., Marathe, M.V., Parthasarathy, S., Srinivasan, A.: Scheduling on unrelated machines under tree-like precedence constraints. In: APPROX-RANDOM, pp. 146–157 (2005)
11. Skutella, M.: Convex quadratic and semidefinite relaxations in scheduling. J. ACM **46**(2), 206–242 (2001)
12. Skutella, M., Woeginger, G.J.: A PTAS for minimizing the weighted sum of job completion times on parallel machines. In: Proc. of 31st Annual ACM Symposium on Theory of Computing (STOC '99), pp. 400–407 (1999)
13. Smith, W.E.: Various optimizers for single-stage production. Nav. Res. Log. Q. **3**, pp. 59–66 (1956)

Minimum Weight Triangulation

1998; Levcopoulos, Krznaric

Christos Levcopoulos
Department of Computer Science, Lund University, Lund, Sweden

Keywords and Synonyms

Minimum length triangulation

Problem Definition

Given a set S of n points in the Euclidean plane, a *triangulation* T of S is a maximal set of non-intersecting straight-line segments whose endpoints are in S. The *weight* of T is defined as the total Euclidean length of all edges in T. A triangulation that achieves minimum weight is called a *minimum weight triangulation*, often abbreviated MWT, of S.

Key Results

Since there is a very large number of papers and results dealing with minimum weight triangulation, only relatively very few of them can be mentioned here.

Mulzer and Rote have shown that MWT in NP-hard [11]. Their proof of NP-completeness is not given explicitly; it relies on extensive calculations which they performed with a computer. Also recently, Remy and Steger have shown a quasi-polynomial time approximation scheme for MWT [12]. These results are stated in the following theorem.

Theorem 1 *The problem of computing the MWT (minimum weight triangulation) of an input set S of n points in the plane is NP-hard. However, for any constant $\epsilon > 0$, a triangulation of S achieving the approximation ratio of $1 + \epsilon$, for an arbitrarily small positive constant ϵ, can be computed in time $n^{O(\log^8 n)}$.*

The Quasi-Greedy Triangulation Approximates the MWT

Levcopoulos and Krznaric showed that a triangulation of total length within a constant factor of MWT can be computed in polynomial time for arbitrary point sets [7]. The triangulation achieving this result is a modification of the so-called *greedy* triangulation. The greedy triangulation starts with the empty set of diagonals and keeps adding a shortest diagonal not intersecting the diagonals which have already been added, until a full triangulation is produced. The greedy triangulation has been shown to approximate the minimum weight triangulation within a constant factor, unless a special case arises where the greedy diagonals inserted are "climbing" in a special, very unbalanced way along a relatively long concave chain containing many vertices and with a large empty space in front of it, at the same time blocking visibility from another, opposite concave chain of many vertices. In such "bad" cases the worst case ratio between the length of the greedy and the length of the minimum weight triangulation is shown to be $\Theta(\sqrt{n})$. To obtain a triangulation which always approximates the MWT within a constant factor, it suffices to take care of this special bad case in order to avoid the unbalanced "climbing", and replace it by a more balanced climbing along these two opposite chains. Each edge inserted in this modified method is still almost as short as the shortest diagonal, within a factor smaller than 1.2. Therefore, the modified triangulation which always approximates the MWT is named the *quasi-greedy* triangulation. In a similar way as the original greedy triangulation, the quasi-greedy triangulation can be computed in time $O(n \log n)$ [8]. Gudmundsson and Levcopoulos [5] showed later that a variant of this method can also be parallelized, thus achieving a constant factor approximation of MWT in $O(\log n)$ time, using $O(n)$ processors in the

CRCW PRAM model. Another by-product of the quasi-greedy triangulation is that one can easily select in linear time a subset of its edges to obtain a convex partition which is within a constant factor of the minimum length convex partition of the input point set. This last property was crucial in the proof that the quasi-greedy triangulation approximates the MWT. The proof also uses an older result that the (original, unmodified) greedy triangulation of any convex polygon approximates the minimum weight triangulation [9]. Some of the results from [7] and from [8] can be summarized in the following theorem:

Theorem 2 *Let S be an input set of n points in the plane. The quasi-greedy triangulation of S, which is a slightly modified version of the greedy triangulation of S, has total length within a constant factor of the length of the MWT (minimum weight triangulation) of S, and can be computed in time $O(n \log n)$. Moreover, the (unmodified) greedy triangulation of S has length within $O(\sqrt{n})$ of the length of MWT of S, and this bound is asymptotically tight in the worst case.*

Computing the Exact Minimum Weight Triangulation

Below three approaches to compute the exact MWT are shortly discussed. These approaches assume that it is numerically possible to efficiently compare the total length of sets of line segments in order to select the set of smallest weight. This is a simplifying assumption, since this is an open problem per se. However, the problem of computing the exact MWT remains NP-hard even under this assumption [11]. The three approaches differ with respect to the creation and selection of subproblems, which are then solved by dynamic programming.

The first approach, sketched by Lingas [10], employs a general method for computing optimal subgraphs of the complete Euclidean graph. By developing this approach, it is possible to achieve subexponential time $2^{O(\sqrt{n}\log n)}$. The idea to create the subproblems which are solved by dynamic programming. This is done by trying all (suitable) planar separators of length $O(\sqrt{n})$, separating the input point set in a balanced way, and then to proceed recursively within the resulting subproblems.

The second approach uses fixed-parameter algorithms. So, for example, if there are only $O(\log n)$ points in the interior of the convex hull of S, then the MWT of S can be computed in polynomial time [4]. This approach extends also to compute the minimum weight triangulation under the constraint that the outer boundary is not necessarily the convex hull of the input vertices, it can be an arbitrary polygon. Some of these algorithms have been implemented, see Grantson et al. [2] for a comparison of some implementations. These dynamic programming approaches take typically cubic time with respect to the points of the boundary, but exponential time with respect to the number of remaining points. So, for example, if k is the number of hole points inside the boundary polygon, then an algorithm, which has also been implemented, can compute the exact MWT in time $O(n^3 \cdot 2^k \cdot k)$ [2].

In an attempt to solve larger problems, a different approach uses properties of MWT which usually help to identify, for random point sets, many edges that *must* be, respectively *cannot* be, in MWT. One can then use dynamic programming to fill in the remaining MWT-edges. For random sets consisting of tens of thousands of points from the uniform distribution, one can thus compute the exact MWT in minutes [1].

Applications

The problem of computing a triangulation arises, for example, in finite element analysis, terrain modeling, stock cutting and numerical approximation [3,6]. The *minimum weight* triangulation has attracted the attention of many researchers, mainly due to its natural definition of optimality, and because it has proved to be a challenging problem over the past thirty years, with unknown complexity status until the end of 2005.

Open Problems

All results mentioned leave open problems. For example, can one find a simpler proof of NP-completeness, which can be checked without running computer programs? It would be desirable to improve the approximation constant which can be achieved in polynomial time (to simplify the proof, the constant shown in [7] is not explicitly calculated and it would be relatively large, if the proof is not refined). The time bound for the approximation scheme could hopefully be improved. It could also be possible to refine the software which computes efficiently the exact MWT for large random point sets, so that it can handle efficiently a wider range of input, i. e., not only completely random point sets. This could perhaps be done by combining this software with implementations of fixed parameter algorithms, as the ones reported in [2,4], or with other approaches. It is also open whether or not the subexponential exact method can be further improved.

Experimental Results

Please see the last paragraph under the section about key results.

URL to Code

Link to code used to compare some dynamic programming approaches in [2]: http://fuzzy.cs.uni-magdeburg.de/~borgelt/pointgon.html

Cross References

- ▶ Fast Minimal Triangulation
- ▶ Greedy Set-Cover Algorithms
- ▶ Minimum Geometric Spanning Trees
- ▶ Minimum k-Connected Geometric Networks

Recommended Reading

1. Beirouti, R., Snoeyink, J.: Implementations of the LMT Heuristic for Minimum Weight Triangulation. Symposium on Computational Geometry, pp. 96–105, Minneapolis, Minnesota, June 7–10, 1998
2. Borgelt, C., Grantson, M., Levcopoulos, C.: Fixed-Parameter Algorithms for the Minimum Weight Triangulation Problem. Technical Report LU-CS-TR:2006-238, ISSN 1650-1276 Report 158. Lund University, Lund (An extended version has been submitted to IJCGA) (2006)
3. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry – Algorithms and Applications, 2nd edn. Springer, Heidelberg (2000)
4. Grantson, M., Borgelt, C., Levcopoulos, C.: Minimum Weight Triangulation by Cutting Out Triangles. In: Proceedings 16th Annual International Symposium on Algorithms and Computation, ISAAC 2005, Sanya, China, pp. 984–994. Lecture Notes in Computer Science, vol. 3827. Springer, Heidelberg (2005)
5. Gudmundsson, J., Levcopoulos, C.: A Parallel Approximation Algorithm for Minimum Weight Triangulation. Nordic J. Comput. **7**(1), 32–57 (2000)
6. Hjelle, Ø., Dæhlen, M.: Triangulations and Applications. In: Mathematics and Visualization, vol. IX. Springer, Heidelberg (2006). ISBN 978-3-540-33260-2
7. Levcopoulos, C., Krznaric, D.: Quasi-Greedy Triangulations Approximating the Minimum Weight Triangulation. J. Algorithms **27**(2), 303–338 (1998)
8. Levcopoulos, C., Krznaric, D.: The Greedy Griangulation can be Computed from the Delaunay Triangulation in Linear Time. Comput. Geom. **14**(4), 197–220 (1999)
9. Levcopoulos, C., Lingas, A.: On Approximation Behavior of the Greedy Triangulation for Convex Polygons. Algorithmica **2**, 15–193 (1987)
10. Lingas, A.: Subexponential-time algorithms for minimum weight triangulations and related problems. In: Proceedings 10th Canadian Conference on Computational Geometry (CCCG), McGill University, Montreal, Quebec, 10–12 August 1998
11. Mulzer, W., Rote, G.: Minimum-weight triangulation is NP-hard. In: Proceedings 22nd Annual ACM Symposium on Computational Geometry, SoCG'06, Sedona, AZ, USA. ACM Press, New York, NY, USA (2006)
12. Remy, J., Steger, A.: A Quasi-Polynomial Time Approximation Scheme for Minimum Weight Triangulation. In: Proceedings 38th ACM Symposium on Theory of Computing (STOC'06). ACM Press, New York, NY, USA (2006)

Mobile Agents and Exploration

1952; Shannon

Evangelos Kranakis[1], Danny Krizanc[2]

[1] Department of Computer Science, Carleton, Ottawa, ON, Canada

[2] Department of Computer Science, Wesleyan University, Middletown, CT, USA

Keywords and Synonyms

Distributed algorithms; Graph exploration; Mobile agent; Navigation; Rendezvous; Routing; Time/Memory tradeoffs

Problem Definition

How can a network be explored efficiently with the help of mobile agents? This is a very broad question and to answer it adequately it will be necessary to understand more precisely what mobile agents are, what kind of networked environment they need to probe, and what complexity measures are interesting to analyze.

Mobile Agents

Mobile agents are autonomous, intelligent computer software that can move within a network. They are modeled as automata with limited memory and computation capability and are usually employed by another entity (to which they must report their findings) for the purpose of collecting information. The actions executed by the mobile agents can be discrete or continuous and transitions from one state to the next can be either deterministic or nondeterministic, thus giving rise to various natural complexity measures depending on the assumptions being considered.

Network Model

The network model is inherited directly from the theory of distributed computing. It is a connected graph whose vertices comprise the computing nodes and edges correspond to communication links. It may be static or dynamic and its resources may have various levels of accessibility. Depending on the model being considered, nodes and links of the network may have distinct labels. A particularly useful abstraction is an anonymous network whereby the nodes have no identities, which means that an agent cannot distinguish two nodes except perhaps by their degree. The outgoing edges of a node are usually thought of as distinguishable but an important distinction can be made be-

tween a globally consistent edge-labeling versus a locally independent edge-labeling.

Efficiency Measures for Exploration

Efficiency measures being adopted involve the time required for completing the exploration task, usually measured either by the number of edge traversals or nodes visited by the mobile agent. The interplay between time required for exploration and memory used by the mobile agent (*time/memory tradeoffs*) are key parameters considered for evaluating algorithms. Several researchers impose no restrictions on the memory but rather seek algorithms minimizing exploration time. Others, investigate the minimum size of memory which allows for exploration of a given type of network (e. g., tree) of given (known or unknown) size, regardless of the exploration time. Finally, several researchers consider time/memory tradeoffs.

Main Problems

Given a model for both the agents and the network, the graph exploration problem is that of designing an algorithm for the agent that allows it to visit all of the nodes and/or edges of the network. A closely related problem is where the domain to be explored is presented as a region of the plane with obstacles and exploration becomes visiting all unobstructed portions of the region in the sense of visibility. Another related problem is that of rendezvous where two or more agents are required to gather at a single node of a network.

Key Results

Claude Shannon [17] is credited with the first finite automaton algorithm capable of exploring an arbitrary maze (which has a range of 5×5 squares) by trial and error means. Exploration problems for mobile agents have been extensively studied in the scientific literature and the reader will find a useful historical introduction in Fraigniaud et al.[11].

Exploration in General Graphs

The network is modeled as a graph and the agent can move from node to node only along the edges. The graph setting can be further specified in two different ways. In Deng and Papadimitriou [8] the agent explores strongly connected directed graphs and it can move only in the direction from head to tail of an edge, but not vice-versa. At each point, the agent has a map of all nodes and edges visited and can recognize if it sees them again. They minimize the ratio of the total number of edges traversed divided by the optimum number of traversals, had the agent known the graph. In Panaite and Pelc [15] the explored graph is undirected and the agent can traverse edges in both directions. In the graph setting it is often required that apart from completing exploration the agent has to draw a map of the graph, i. e., output an isomorphic copy of it. Exploration of directed graphs assuming the existence of labels is investigated in Albers and Henzinger [1] and Deng and Papadimitriou [8]. Also in Panaite and Pelc [15], an exploration algorithm is proposed working in time $e + O(n)$, where is n the number of nodes and e the number of links. Fraigniaud et al. [11] investigate memory requirements for exploring unknown graphs (of unknown size) with unlabeled nodes and locally labeled edges at each node. In order to explore all graphs of diameter D and max degree d a mobile agent needs $\Omega(D \log d)$ memory bits even when exploration is restricted to planar graphs. Several researchers also investigate exploration of anonymous graphs in which agents are allowed to drop and remove pebbles. For example in Bender et al. [4] it is shown that one pebble is enough for exploration, if the agent knows an upper bound on the size of the graph, and $\Theta(\log \log n)$ pebbles are necessary and sufficient otherwise.

Exploration in Trees

In this setting it is assumed the agent can distinguish ports at a node (locally), but there is no global orientation of the edges and no markers available. *Exploration with stop* is when the mobile agent has to traverse all edges and stop at some node. For *exploration with return* the mobile agent has to traverse all edges and stop at the starting node. In *perpetual exploration* the mobile agent has to traverse all edges of the tree but is not required to stop. The upper and lower bounds on memory for the exploration algorithms analyzed in Diks et al. [9] are summarized in the table, depending on the knowledge that the mobile agent has. Here, n is the number of nodes of the tree, $N \geq n$ is an upper bound known to the mobile agent, and d is the maximum degree of a node of the tree.

Exploration	Knowledge	Lower Bounds	Upper Bounds
Perpetual	$\emptyset$	None	$O(\log d)$
w/Stop	$n \leq N$	$\Omega(\log \log \log n)$	$O(\log N)$
w/Return	$\emptyset$	$\Omega(\log n)$	$O(\log^2 n)$

Exploration in a Geometric Setting

Exploration in a geometric setting with unknown terrain and convex obstacles is considered by Blum et al. [5]. They compare the distance walked by the agent (or robot) to the length of the shortest (obstacle-free) path in the scene and

describe and analyze robot strategies that minimize this ratio for different kinds of scenes. There is also related literature for exploration in more general settings with polygonal and rectangular obstacles by Deng et al. [7] and Bar-Eli et al. [3], respectively. A setting that is important in wireless networking is when nodes are aware of their location. In this case, Kranakis et al. [12] give efficient algorithms for navigation, namely compass routing and face routing that guarantee delivery in Delaunay and arbitrary planar geometric graphs, respectively, using only local information.

Rendezvous

The rendezvous search problem differs from the exploration problem in that it concerns two searchers placed at different nodes of a graph that want to minimize the time required to rendezvous (usually) at the same node. At any given time the mobile agents may occupy a vertex of the graph and can either stay still or move from vertex to vertex. It is of interest to minimize the time required to rendezvous. A natural extension of this problem is to study multi-agent mobile systems. More generally, given a particular agent model and network model, a set of agents distributed arbitrarily over the nodes of the network are said to rendezvous if executing their programs after some finite time they all occupy the same node of the network at the same time. Of special interest is the highly symmetric case of anonymous agents on an anonymous network and the simplest interesting case is that of two agents attempting to rendezvous on a ring network. In particular, in the model studied by Sawchuk [16] the agents cannot distinguish between the nodes, the computation proceeds in synchronous steps, and the edges of each node are oriented consistently. The table summarizes time/memory tradeoffs known for six algorithms (see Kranakis et al. [13] and Flocchini et al. [10]) when the k mobile agents use indistinguishable pebbles (one per mobile agent) to mark their position in an n node ring.

Memory	Time	Memory	Time
$O(k \log n)$	$O(n)$	$O(\log n)$	$O(n)$
$O(\log n)$	$O(kn)$	$O(\log k)$	$O(n)$
$O(k \log \log n)$	$O\left(\frac{n \log n}{\log \log n}\right)$	$O(\log k)$	$O(n \log k)$

Kranakis et al.[14] show a striking computational difference for rendezvous in an oriented, synchronous, $n \times n$ torus when the mobile agents may have more indistinguishable tokens. It is shown that two agents with a constant number of unmovable tokens, or with one movable token each cannot rendezvous if they have $o(\log n)$ memory, while they can perform rendezvous with detection as long as they have one unmovable token and $O(\log n)$ memory. In contrast, when two agents have two movable tokens each then rendezvous (respectively, rendezvous with detection) is possible with constant memory in a torus. Finally, two agents with three movable tokens each and constant memory can perform rendezvous with detection in a torus. If the condition on synchrony is dropped the rendezvous problem becomes very challenging. For a given initial location of agents in a graph, De Marco et al. [6] measure the performance of a rendezvous algorithm as the number of edge traversals of both agents until rendezvous is achieved. If the agents are initially situated at a distance D in an infinite line, they give a rendezvous algorithm with cost $O(D|L_{\min}|^2)$ when D is known and $O((D + |L_{\max}|)^3)$ if D is unknown, where $|L_{\min}|$ and $|L_{\max}|$ are the lengths of the shorter and longer label of the agents, respectively. These results still hold for the case of the ring of unknown size but then they also give an optimal algorithm of cost $O(n|L_{\min}|)$, if the size n of the ring is known, and of cost $O(n|L_{\max}|)$, if it is unknown. For arbitrary graphs, they show that rendezvous is feasible if an upper bound on the size of the graph is known and they give an optimal algorithm of cost $O(D|L_{\min}|)$ if the topology of the graph and the initial positions are known to the agents.

Applications

Interest in mobile agents has been fueled by two overriding concerns. First, to simplify the complexities of distributed computing, and second to overcome the limitations of user interface approaches. Today they find numerous applications in diverse fields such as distributed problem solving and planning (e. g., task sharing and coordination), network maintenance (e. g., daemons in networking systems for carrying out tasks like monitoring and surveillance), electronic commerce and intelligence search (e. g., data mining and surfing crawlers to find products and services from multiple sources), robotic exploration (e. g., rovers, and other mobile platforms that can explore potentially dangerous environments or even enhance planetary extravehicular activity), and distributed rational decision making (e. g., auction protocols, bargaining, decision making). The interested reader can find useful information in several articles in the volume edited by Weiss [18].

Open Problems

Specific directions for further research would include the study of time/memory tradeoffs in search game models (see Alpern and Gal [2]). Multi-agent systems are partic-

ularly useful for content-based searches and exploration, and further investigations in this area would be fruitful. Memory restricted mobile agents provide a rich model with applications in sensor systems. In the geometric setting, navigation and routing in a three dimensional environment using only local information is an area with many open problems.

Cross References

- ► Deterministic Searching on the Line
- ► Robotics
- ► Routing

Recommended Reading

1. Albers, S., Henzinger, M.R.: Exploring unknown environments. SIAM J. Comput. **29**, 1164–1188 (2000)
2. Alpern, S., Gal, S.: The Theory of Search Games and Rendezvous. Kluwer Academic Publishers, Norwell (2003)
3. Bar-Eli, E., Berman, P., Fiat, A., Yan, R.: On-line navigation in a room. J. Algorithms **17**, 319–341 (1994)
4. Bender, M.A., Fernandez, A., Ron, D., Sahai, A., Vadhan, S.: The power of a pebble: Exploring and mapping directed graphs. In: Proc. 30th Ann. Symp. on Theory of Computing, pp. 269–278. Dallas, 23–26 May 1998
5. Blum, A., Raghavan, P., Schieber, B.: Navigating in unfamiliar geometric terrain. SIAM J. Comput. **26**, 110–137 (1997)
6. De Marco, G., Gargano, L., Kranakis, E., Krizanc, D., Pelc, A., Vaccaro, U.: Asynchronous Deterministic Rendezvous in Graphs. Theoret. Comput. Sci. **355**, 315–326 (2006)
7. Deng, X., Kameda, T., Papadimitriou, C.H.: How to learn an unknown environment I: the rectilinear case. J. ACM **45**, 215–245 (1998)
8. Deng, X., Papadimitriou, C.H.: Exploring an unknown graph. J. Graph Theory **32**, 265–297 (1999)
9. Diks, K., Fraigniaud, P., Kranakis, E., Pelc, A.: Tree exploration with little memory. J. Algorithms **51**, 38–63 (2004)
10. Flocchini, P., Kranakis, E., Krizanc, D., Santoro, N., Sawchuk, C.: Multiple Mobile Agent Rendezvous in the Ring. In: Proc. LATIN 2004. LNCS, vol. 2976, pp. 599–608. Bueons Aires, 5–8 April 2004
11. Fraigniaud, P., Ilcinkas, D., Peer, G., Pelc, A., Peleg, D.: Graph exploration by a finite automaton. Theor. Comput. Sci. **345**, 331–344 (2005)
12. Kranakis, E., Singh, H., Urrutia, J.: Compass Routing in Geometric Graphs. In: Proceedings of 11th Canadian Conference on Computational Geometry, CCCG-99, pp. 51–54, Vancouver, 15–18 August 1999
13. Kranakis, E., Krizanc, D., Santoro, N., Sawchuk, C.: Mobile Agent Rendezvous Search Problem in the Ring. In: Proc. International Conference on Distributed Computing Systems (ICDCS), pp. 592–599. Providence, Rhode Island 19–22 May 2003
14. Kranakis, E., Krizanc, D., Markou, E.: Mobile Agent Rendezvous in a Synchronous Torus. In: Proceedings of LATIN 2006, 7th Latin American Symposium. Valdivia, March 20–24 2006. Correa, J., Hevia, A., Kiwi, M. SVLNCS **3887**, 653–664 (2006)
15. Panaite, P., Pelc, A.: Exploring unknown undirected graphs. J. Algorithms **33**, 281–295 (1999)
16. Sawchuk, C.: Mobile Agent Rendezvous in the Ring. Ph. D. thesis, Carleton University, Ottawa, Canada (2004)
17. Shannon, C.: Presentation of a Maze Solving Machine, in Cybernetics, Circular, Causal and Feedback Machines in Biological and Social Systems. In: von Feerster, H., Mead, M., Teuber, H.L. (eds.) Trans. 8th Conf, New York, March 15–16, 1951. pp. 169–181. Josiah Mary Jr. Foundation, New York (1952)
18. Weiss, G. (ed.): Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge, MA (1999)

MST

- ► Minimum Energy Broadcasting in Wireless Networks
- ► Minimum Geometric Spanning Trees

Multicommodity Flow, Well-linked Terminals and Routing Problems

2005; Chekuri, Khanna, Shepherd

Chandra Chekuri
Department of Computer Science, University of Illinois, Urbana-Champaign, Urbana, IL, USA

Keywords and Synonyms

Edge disjoint paths problem; Maximum edge disjoint paths problem; Node disjoint paths problem, All-or-nothing multicommodity flow problem

Problem Definition

Three related optimization problems derived from the classical edge disjoint paths problem (EDP) are described. An instance of EDP consists of an undirected graph $G = (V, E)$ and a multiset $\mathcal{T} = \{s_1t_1, s_2t_2, \ldots, s_kt_k\}$ of k node pairs. EDP is a decision problem: can the pairs in $\mathcal{T}$ be connected (alternatively routed) via edge-disjoint paths? In other words, are there paths $P_1, P_2, \ldots, P_k$ such that for $1 \leq i \leq k$, P_i is path from s_i to t_i, and no edge $e \in E$ is in more than one of these paths? EDP is known to be NP-Complete. This article considers there maximization problems related to EDP.

- **Maximum Edge-Disjoint Paths Problem (MEDP).** Input to MEDP is the same as for EDP. The objective is to *maximize* the number of pairs in $\mathcal{T}$ that can be routed via edge-disjoint paths. The output consists of a subset $S \subseteq \{1, 2, \ldots, k\}$ and for each $i \in S$ a path P_i connecting s_i to t_i such that the paths are edge-disjoint. The goal is to maximize $|S|$.

- **Maximum Edge-Disjoint Paths Problem with Congestion (MEDPwC).** MEDPwC is a relaxation of MEDP. The input, in addition to G and the node pairs, contains an integer congestion parameter c. The output is the same for MEDP; a subset $S \subseteq \{1, 2, \ldots, k\}$ and for each $i \in S$ a path P_i connecting s_i to t_i. However, the paths $P_i, 1 \leq i \leq k$ are not required to be edge-disjoint. The relaxed requirement is that for each edge $e \in E$, the number of paths for the routed pairs that contain e is at most c. Note that MEDPwC with $c = 1$ is the same as MEDP.
- **All-or-Nothing Multicommodity Flow Problem (ANF).** ANF is a different relaxation of MEDP obtained by relaxing the notion of routing. A pair $s_i t_i$ is now said to be routed if a unit flow is sent from s_i to t_i (potentially on multiple paths). The input is the same as for MEDP. The output consists of a subset $S \subseteq \{1, 2, \ldots, k\}$ such that there is a feasible multicommodity flow in G that routes one unit of flow for each pair in S. The goal is to maximize $|S|$.

In the rest of the article, graphs are assumed to be undirected multigraphs. Given a graph $G = (V, E)$ and $S \subset V$, let $\delta_G(S)$ denote the set of edges with exactly one end point in S. Let n denote the number of vertices in the input graph.

Key Results

A few results in the broader literature are reviewed in addition to the results from [6]. EDP is NP-Complete when k is part of the input. A highly non-trivial result of Robertson and Seymour yields a polynomial time algorithm when k is a fixed constant.

Theorem 1 ([16]) *There is a polynomial time algorithm for EDP when k is a fixed constant independent of the input size.*

Using Theorem 1 it is easy to see that MEDP and MEDPwC have polynomial time algorithms for fixed k . The same holds for ANF by simple enumeration since the decision version is polynomial-time solvable via linear programming.

The focus of this article is on the case when k is part of the input, and in this setting, all three problems considered are NP-hard. The starting point for most approximation algorithms is the natural multicommodity flow relaxation given below. This relaxation is valid for both MEDP and ANF. The end points of the input pairs are referred to as *terminals* and let X denote the set of terminals. To describe the relaxation as well as simplify further discussion, the following simple assumption is made without loss of generality; each node in the graph participates in at most one of the input pairs. This assumption implies that the input pairs induce a matching M on the terminal set X. Thus the input for the problem can alternatively given as a triple (G, X, M).

For the given instance (G, X, M), let $\mathcal{P}_i$ denote the set of paths joining s_i and t_i in G and let $\mathcal{P} = \cup_i \mathcal{P}_i$. The LP relaxation has the following variables. For each path $P \in \mathcal{P}$ there is a variable $f(P)$ which is the amount of flow sent on P. For each pair $s_i t_i$ there is a variable x_i to indicate the total flow that is routed for the pair.

$$\begin{aligned}
(\text{MCF} - \text{LP}) \max \sum_{i=1}^{k} x_i \quad & \text{s.t} \\
x_i - \sum_{P \in \mathcal{P}_i} f(P) = 0 \quad & 1 \leq i \leq k \\
\sum_{P:\, e \in P} f(P) \leq 1 \quad & \forall e \in E \\
x_i, f(P) \in [0, 1] \quad & 1 \leq i \leq k, P \in \mathcal{P}
\end{aligned}$$

The above path formulation has an exponential (in n) number of variables, however it can still be solved in polynomial time. There is also an equivalent compact formulation with a polynomial number of variables and constraints. Let OPT denote the value of an optimum solution to a given instance. Similarly, let OPT-LP denote the value of an optimum solution the LP relaxation for the given instance. It can be seen that OPT-LP $\geq$ OPT. It is known that the integrality gap of (MCF-LP) is $\Omega(\sqrt{n})$ [10]; that is, there is an infinite family of instances such that $OPT - LP/OPT = \Omega(\sqrt{n})$. The current best approximation algorithm for MEDP is given by the following theorem.

Theorem 2 ([4]) *The integrality gap of (MCF-LP) for MEDP is $\Theta(\sqrt{n})$ and there is an $O(\sqrt{n})$ approximation for MEDP.*

For MEDPwC the approximation ratio improves with the congestion parameter c.

Theorem 3 ([18])
There is an $O(n^{1/c})$ approximation for MEDPwC with congestion parameter c. In particular there is a polynomial time algorithm that routes $\Omega(\text{OPT-LP}/n^{1/c})$ pairs with congestion at most c.

The above theorem is established via randomized rounding of a solution to (MCF-LP). Similar results, but via simpler combinatorial algorithms, are obtained in [2,15].

In [6] a new framework was introduced to obtain approximation algorithm for routing problems in undirected graphs via (MCF-LP). A key part of the framework is the

Multicommodity Flow, Well-linked Terminals and Routing Problems, Table 1
Known bounds for MEDP, ANF and MEDPwC in general undirected graphs. The best upper bound on the approximation ratio is the same as the upper bound on the integrality gap of (MCF-LP)

	Integrality Gap of (MCF-LP)		*Approximation Ratio*
	Upper bound	*Lower bound*	*Lower bound*
MEDP	$O(\sqrt{n})$	$\Omega(\sqrt{n})$	$\Omega(\log^{1/2-\epsilon} n)$
MEDPwC	$O(n^{1/c})$	$\Omega(\log^{(1-\epsilon)/(c+1)} n)$	$\Omega(\log^{(1-\epsilon)/(c+1)} n)$
ANF	$O(\log^2 n)$	$\Omega(\log^{1/2-\epsilon} n)$	$\Omega(\log^{1/2-\epsilon} n)$

so-called well-linked decomposition that allows a reduction of an arbitrary instance to an instance in which the terminals satisfy a strong property.

Definition 1 Let $G = (V, E)$ be a graph. A subset $X \subseteq V$ is *cut-well-linked* in G if for every $S \subset V$, $|\delta_G(S)| \geq \min\{|S \cap X|, |(V \setminus S) \cap X|\}$. X is *flow-well-linked* if there exists a feasible fractional multicommodity flow in G for the instance in which there is a demand of $1/|X|$ for each unordered pair uv, $u, v \in X$.

The main result in [6] is the following.

Theorem 4 ([6]) *Let (G, X, M) be an instance of MEDP or ANF and let* OPT-LP *be the value of an optimum solution to (MCF-LP) on (G, X, M). There there is a polynomial time algorithm that obtains a collection of instances $(G_1, X_1, M_1), (G_2, X_2, M_2), \ldots, (G_h, X_h, M_h)$ with the following properties:*

- *The graphs $G_1, G_2, \ldots, G_h$ are node-disjoint induced subgraphs of G. For $1 \leq i \leq h$, $X_i \subseteq X$ and $M_i \subseteq M$.*
- *For $1 \leq i \leq h$, X_i is flow-well-linked in G_i.*
- $\sum_{i=1}^{h} |X_i| = \Omega(\text{OPT-LP}/\log^2 n)$.

For planar graphs and graphs that exclude a fixed minor, the above theorem gives a stronger guarantee: $\sum_{i=1}^{h} |X_i| = \Omega(\text{OPT-LP}/\log n)$. A well-linked instance satisfies a strong symmetry property based on the following observation. If A is flow-well-linked in G then for *any* matching J on X, OPT-LP on the instance (G, A, J) is $\Omega(|A|)$. Thus the particular matching M of a given well-linked instance (G, X, M) is essentially irrelevant. The second part of the framework in [6] consists of exploiting the well-linked property of the instances produced by the decomposition procedure. At a high level this is done by showing that if G has a well-linked set X, then it contains a "crossbar" (a routing structure) of size $\Omega(|X|/\text{poly}(\log n))$. See [6] for more precise definitions. Techniques for the second part vary based on the problem as well as the family of graphs in question. The following results are obtained using Theorem 4 and other non-trivial ideas for the second part [7,8,6,3].

Theorem 5 ([6]) *There is an $O(\log^2 n)$ approximation for ANF. This improves to an $O(\log n)$ approximation in planar graphs.*

Theorem 6 ([6]) *There is an $O(\log n)$ approximation for MEDPwC in planar graphs for $c \geq 2$. There is an $O(\log n)$ approximation for ANF in planar graphs.*

Theorem 7 ([3]) *There is an $O(r \log n \log r)$ approximation for MEDP in graphs of treewidth at most r.*

Generalizations and Variants

Some natural variants and generalizations of the problems mentioned in this article are obtained by considering three orthogonal aspects: (i) node disjointness instead of edge-disjointness, (ii) capacities on the edges and/or nodes, and (iii) demand values on the pairs (each pair $s_i t_i$ has an integer demand d_i and the objective is to route d_i units of flow between s_i and t_i). Results similar to those mentioned in the article are shown to hold for these generalizations and variants [6]. Capacities and demand values on pairs are somewhat easier to handle while node-disjoint problems often require additional non-trivial ideas. The reader is referred to [6] for more details.

For some special classes of graphs (trees, expanders and grids to name a few), constant factor or poly-logarithmic approximation ratios are known for MEDP.

Applications

Flow problems are at the core of combinatorial optimization and have numerous applications in optimization, computer science and operations research. Very special cases of EDP and MEDP include classical problems such as single-commodity flows, and matchings in general graphs, both of which have many applications. EDP and variants arise most directly in telecommunication networks and VLSI design. Since EDP captures difficult problems as special cases, there are only a few algorithmic tools that can address the numerous applications in a unified fashion. Consequently, empirical research tends to focus on application specific approaches to obtain satisfactory solutions.

The flip side of the difficulty of EDP is that it offers a rich source of problems, the study of which has led to important algorithmic advances of broad applicability, as well as fundamental insights in graph theory, combinatorial optimization, and related fields.

Open Problems

A number of very interesting open problems remain regarding the approximability of the problems discussed in this article. Table 1 gives the best known upper and lower bounds on the approximation ratio as well as integrality gap of (MCF-LP). All the inapproximability results in Table 1, and the integrality gap lower bounds for MEDPwC and ANF, are from [1]. The inapproximability results are based on the assumption that NP $\not\subseteq$ ZTIME($n^{\text{poly}(\log n)}$). Closing the gaps between the lower and upper bounds are the major open problems.

Cross References

- ► Randomized Rounding
- ► Separators in Graphs
- ► Treewidth of Graphs

Recommended Reading

The limited scope of this article does not do justice to the large literature on EDP and related problems. In addition to the articles cited in the main body of the article, the reader is referred to [5,9,11,12, 13,14,17] for further reading and pointers to existing literature.

1. Andrews, M., Chuzhoy, J., Khanna, S., Zhang, L.: Hardness of the Undirected Edge-Disjoint Paths Problem with Congestion. Proc. of IEEE FOCS, 2005, pp. 226–244
2. Azar, Y., Regev, O.: Combinatorial algorithms for the unsplittable flow problem. Algorithmica **44**(1), 49–66 (2006). Preliminary version in Proc. of IPCO 2001
3. Chekuri, C., Khanna, S., Shepherd, F.B.: A note on multiflows and treewidth. Algorithmica, published online (2007)
4. Chekuri, C., Khanna, S., Shepherd, F.B.: An $O(\sqrt{n})$ approximation and integrality gap for disjoint paths and UFP. Theor. Comput. **2**, 137–146 (2006)
5. Chekuri, C., Khanna, S., Shepherd, F.B.: Edge-Disjoint Paths in Planar Graphs with Constant Congestion. Proc. ACM STOC, pp. 757–766 (2006)
6. Chekuri, C., Khanna, S., Shepherd, F.B.: Multicommodity flow, well-linked terminals, and routing problems. Proc. ACM STOC, pp. 183–192 (2005)
7. Chekuri, C., Khanna, S., Shepherd, F.B.: The All-or-Nothing Multicommodity Flow Problem. Proc. ACM STOC, pp. 156–165 (2004)
8. Chekuri, C., Khanna, S., Shepherd, F.B.: Edge Disjoint Paths in Planar Graphs. Proc. of IEEE FOCS, 2004, pp. 71–80
9. Frank, A.: Packing paths, cuts, and circuits – a survey. In: Korte, B., Lovász, L., Prömel H.J., Schrijver A. (eds.) Paths, Flows and VLSI-Layout, pp. 49–100. Springer, Berlin (1990)
10. Garg, N., Vazirani, V., Yannakakis, M.: Primal-Dual Approximation Algorithms for Integral Flow and Multicut in Trees. Algorithmica **18**(1), 3–20 (1997). Preliminary version appeared in Proc. ICALP 1993
11. Guruswami, V., Khanna, S., Rajaraman, R., Shepherd, F.B., Yannakakis, M.: Near-Optimal Hardness Results and Approximation Algorithms for Edge-Disjoint Paths and Related Problems. J. CSS **67**, 473–496 (2003). Preliminary version in Proc. of ACM STOC 1999
12. Kleinberg, J.M.: Approximation algorithms for disjoint paths problems. Ph. D. thesis, MIT, Cambridge, MA (1996)
13. Kleinberg, J.M.: An Approximation Algorithm for the Disjoint Paths Problem in Even-Degree Planar Graphs. Proc. of IEEE FOCS, 2005, pp. 627–636
14. Kolliopoulos, S.G.: Edge Disjoint Paths and Unsplittable Flow. In: Handbook on Approximation Algorithms and Metaheuristics, Chapman & Hall/CRC Press Computer & Science Series, vol 13. Chapman Hall/CRC Press, May 2007
15. Kolliopoulos, S.G., Stein, C.: Approximating Disjoint-Path Problems Using Greedy Algorithms and Packing Integer Programs. Math. Program. A **99**, 63–87 (2004). Preliminary version in Proc. of IPCO 1998
16. Robertson, N., Seymour, P.D.: Graph Minors XIII.The Disjoint Paths Problem. J. Comb. Theor. B **63**(1), 65–110 (1995)
17. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin (2003)
18. Srinivasan, A.: Improved approximations for edge-disjoint paths, unsplittable flow, and related routing problems. Proc. IEEE FOCS, 1997, pp. 416–425

Multicut

1993; Garg, Vazirani, Yannakakis
1996; Garg, Vazirani, Yannakakis

Shuchi Chawla
Department of Computer Science,
University of Wisconsin–Madison, Madison, WI, USA

Problem Definition

The Multicut problem is a natural generalization of the s-t mincut problem—given an undirected capacitated graph $G = (V, E)$ with k pairs of vertices $\{s_i, t_i\}$; the goal is to find a subset of edges of the smallest total capacity whose removal from G disconnects s_i from t_i for every $i \in \{1, \cdots, k\}$. However, unlike the Mincut problem which is polynomial-time solvable, the Multicut problem is known to be NP-hard and APX-hard for $k \geq 3$ [6].

This problem is closely related to the Multi-Commodity Flow problem. The input to the latter is a capacitated network with k commodities (source-sink pairs); the goal is to route as much total flow between these source-sink

pairs as possible while satisfying capacity constraints. The maximum multi-commodity flow in a graph can be found in polynomial time via linear programming, and there are also several combinatorial FPTASes known for this problem [7,9,11].

It is immediate from the definition of Multicut that the multicommodity flow in a graph is bounded above by the capacity of a minimum multicut in the graph. When there is a single commodity to be routed, the **max-flow min-cut** theorem of Ford and Fulkerson [8] states that the converse also holds: the maximum *s*-*t* flow in a graph is exactly equal to the minimum *s*-*t* cut in the graph. This duality between flows and cuts in a graph has many applications and, in particular, leads to a simple algorithm for finding the minimum cut in a graph.

Given its simplicity and elegance, several attempts have been made to extend this duality to other classes of flow and partitioning problems. Hu showed, for example, that the min-multicut equals the maximum multicommodity flow when there are only two commodities in the graph [12]. Unfortunately, this property does not extend to graphs with more than two commodities. The focus has therefore been on obtaining approximate max-multicommodity flow min-multicut theorems. Such theorems would also imply a polynomial-time algorithm for approximately computing the capacity of the minimum multicut in a graph.

Key Results

Garg, Vazirani and Yannakakis [10] were the first to obtain an approximate max-multicommodity flow min-multicut theorem. They showed that the maximum multicommodity flow in a graph is always at least an $O(\log k)$ fraction of the minimum multicut in the graph. Moreover, their proof of this result is constructive. That is, they also provide an algorithm for computing a multicut for a given graph with capacity at most $O(\log k)$ times the maximum multicommodity flow in the graph. This is the best approximation algorithm known to date for the Multicut problem.

Theorem 1 *Let M denote the minimum multicut in a graph with k commodities and f denote the maximum multicommodity flow in the graph. Then*

$$\frac{M}{O(\log k)} \leq f \leq M .$$

Moreover, there is a polynomial time algorithm for finding an $O(\log k)$-approximate multicut in a graph.

Furthermore, they show that this theorem is tight to within constant factors. That is, there are families of graphs in which the gap between the maximum multicommodity flow and minimum multicut is $\Theta(\log k)$.

Theorem 2 *There exists a infinite family of multicut instances $\{(G_k, P_k)\}$ such that for all k, the graph $G_k = (V_k, E_k)$ contains k vertices and $P_k \subseteq V_k \times V_k$ is a set of $\Omega(k^2)$ source-sink pairs. Furthermore, the maximum multicommodity flow in the instance (G_k, P_k) is $O(k/\log k)$ and the minimum multicut is $\Omega(k)$.*

Garg et al. also consider the Sparsest Cut problem which is another partitioning problem closely related to Multicut, and provided an approximation algorithm for this problem. Their results for Sparsest Cut have subsequently been improved upon [3,15]. The reader is referred to the entry on ▶ Sparsest Cut for more details.

Applications

A key application of the Multicut problem is to the 2CNF ≡Deletion problem. The latter is a constraint satisfaction problem in which given a weighted set of clauses of the form $P \equiv Q$, where P and Q are literals, the goal is to delete a minimum weight set of clauses so that the remaining set is satisfiable. The 2CNF ≡Deletion problem models a number of partitioning problems, for example the Minimum Edge-Deletion Graph Bipartization problem—finding the minimum weight set of edges whose deletion makes a graph bipartite. Klein et al. [14] showed that the 2CNF ≡Deletion problem reduces in an approximation preserving way to Multicut. Therefore, a ρ-approximation to Multicut implies a ρ-approximation to 2CNF ≡Deletion. (See the survey by Shmoys [16] for more applications.)

Open Problems

There is a big gap between the best-known algorithm for Multicut and the best hardness result (APX-hardness) known for the problem. Improvements in either direction may be possible, although there are indications that the $O(\log k)$ approximation is the best possible. In particular, Theorem 2 implies that the integrality gap of the natural linear programming relaxation for Multicut is $\Theta(\log k)$. Although improved approximations have been obtained for other partitioning problems using semi-definite programming instead of linear programming, Agarwal et al. [1] showed that similar improvements cannot be achieved for Multicut—the integrality gap of the natural SDP-relaxation for Multicut is also $\Theta(\log k)$. On the other hand, there are indications that the APX-hardness is not tight. In particular, assuming the so-called Unique Games

conjecture, it has been shown that Multicut cannot be approximated to within any constant factor [4,13]. In light of these negative results, the main open problem related to this work is to obtain a super-constant hardness for the Multicut problem under a standard assumption such as $P \neq NP$.

The Multicut problem has also been studied in directed graphs. The best known approximation algorithm for this problem is an $O(n^{11/23}\log^{O(1)} n)$-approximation due to Aggarwal, Alon and Charikar [2], while on the hardness side, Chuzhoy and Khanna [5] show that there is no $2^{\Omega(\log^{1-\epsilon} n)}$ approximation, for any $\epsilon > 0$, unless NP$\subseteq$ZPP. Chuzhoy and Khanna also exhibit a family of instances for which the integrality gap of the natural LP relaxation of this problem (which is also the gap between the maximum directed multicommodity flow and the minimum directed multicut) is $\Omega(n^{1/7})$.

Cross References

► Sparsest Cut

Recommended Reading

1. Agarwal, A., Charikar, M., Makarychev, K., Makarychev, Y.: $O(\sqrt{\log n})$ approximation algorithms for Min UnCut, Min 2CNF Deletion, and directed cut problems. In: Proceedings of the 37th ACM Symposium on Theory of Computing (STOC), pp. 573–581, Baltimore, May 2005
2. Aggarwal, A., Alon, N., Charikar, M.: Improved approximations for directed cut problems. In: Proceedings of the 39th ACM Symposium on Theory of Computing (STOC), pp. 671–680, San Diego, June 2007
3. Arora, S., Satish, R., Vazirani, U.: Expander Flows, Geometric Embeddings, and Graph Partitionings. In: Proceedings of the 36th ACM Symposium on Theory of Computing (STOC), pp. 222–231, Chicago, June 2004
4. Chawla, S., Krauthgamer, R., Kumar, R., Rabani Y., Sivakumar, D.: On the Hardness of Approximating Sparsest Cut and Multicut. In: Proceedings of the 20th IEEE Conference on Computational Complexity (CCC), pp. 144–153, San Jose, June 2005
5. Chuzhoy, J., Khanna, S.: Polynomial flow-cut gaps and hardness of directed cut problems. In: Proceedings of the 39th ACM Symposium on Theory of Computing (STOC), pp. 179–188, San Diego, June 2007
6. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiterminal cuts. SIAM Comput. J. **23**(4), 864–894 (1994)
7. Fleischer, L.: Approximating fractional multicommodity flow independent of the number of commodities. In: Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 24–31, New York, October 1999
8. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Can. J. Math. **8**, 399–404. (1956)
9. Garg, N., Könemann., J.: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In: Proceedings of the 39th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 300–309. (1998)
10. Garg, N., Vazirani, V.V., Yannakakis, M.: Approximate max-flow min-(multi)cut theorems and their applications. SIAM Comput. J., **25**(2), 235–251. (1996)
11. Grigoriadis, M.D., Khachiyan, L.G.: Coordination complexity of parallel price-directive decomposition. Mathematics of Operations Research, **21**, 321–340. (1996)
12. Hu, T.C.: Multi-commodity network flows. Operations Research, **11**(3), 344–360. (1963)
13. Khot, S., Vishnoi, N.: The Unique Games Conjecture, Integrality Gap for Cut Problems and the Embeddability of Negative-Type Metrics into ℓ_1. In: Proceedings of the 46th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 53–62. (2005)
14. Klein, P., Agrawal, A., Ravi, R., Rao, S.: Approximation through multicommodity flow. In: Proceedings of the 31st IEEE Symposium on Foundations of Computer Science (FOCS), pp. 726–737 (1990)
15. Linial, N., London, E., Rabinovich, Y.: The geometry of graphs and some of its algorithmic applications. Combinatorica **15**(2), 215–245 (1995). Also in Proc. 35th FOCS, pp. 577–591 (1994)
16. Shmoys, D.B.: Cut problems and their application to divide-and-conquer. In: Hochbaum D.S., (ed.), Approximation Algorithms for NP-hard Problems, pp. 192–235. PWS Publishing, Boston (1997)

Multidimensional Compressed Pattern Matching

2003; Amir, Landau, Sokol

Amihood Amir[1,2]

[1] Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel

[2] Department of Computer Science, John Hopkins University, Baltimore, MD, USA

Keywords and Synonyms

Pattern matching in compressed images; Two-dimensional compressed matching; Multidimensional compressed search

Problem Definition

Let c be a given compression algorithm, and let $c(D)$ be the result of c compressing data D. The *compressed search problem with compression algorithm c* is defined as follows.

Input: Compressed text $c(T)$ and pattern P.

Output: All locations in T where pattern P occurs.

A compressed matching algorithm is *optimal* if its time complexity is $O(|c(T)|)$.

Although optimality in terms of time is always important, when dealing with compression, the criterion of **extra**

space is perhaps more important [20]. Applications employ compression techniques specifically because there is a limited amount of available space. Thus, it is not sufficient for a compressed matching algorithm to be optimal in terms of time, it must also satisfy the given space constraints. Space constraints may be due to limited amount of disk space (e. g., on a server), or they may be related to the size of the memory or cache. Note that if an algorithm uses as little extra space as the size of the cache, the runtime of the algorithm is also greatly reduced as no cache misses will occur [13]. It is also important to remember that in many applications, e. g., LZ compression on strings, the *compression ratio* – $|S|/|c(S)|$ – is a small constant. In a case where the compression ratio of the given text is a constant, an optimal compressed matching performs no better than the naive algorithm of decompressing the text. However, if the constants hidden in the *"big O"* are smaller than the compression ratio, then the compressed matching does offer a practical benefit. If those constants are larger than the optimal the compressed search algorithm may, in fact, be using more space than the uncompressed text.

Definition 1 (inplace) A compressed matching is said to be *inplace* if the extra space used is proportional to the input size of the pattern.

Note that this definition encompasses the compressed matching model (e. g., [2]) where the pattern is input in uncompressed form, as well as the *fully compressed* model [10], where the pattern is input in compressed form. The *inplace* requirement allows the extra space to be the input size of the pattern, whatever that size may be. However, in many applications the compression ratio is a constant; therefore, a stronger space constraint is defined.

Definition 2 Let $\mathcal{AP}$ be the set of all patterns of size m, and let $c(\mathcal{AP})$ be the set of all compressed images of $\mathcal{AP}$. Let m' be the length of the smallest pattern in $c(\mathcal{AP})$. A compressed matching algorithm with input pattern P of length m is called *strongly inplace* if the amount of extra space used is proportional to m'.

The problem as defined above is equally applicable to textual (one-dimensional), image (two-dimensional), or any type of data, such as bitmaps, concordances, tables, XML data, or any possible data structure.

The compressed matching problem is considered crucial in image databases, since they are highly compressible. The initial definition of the compressed matching paradigm was motivated by the *two dimensional run-length compression*. This is the compression used for fax transmissions. The run-length compression is defined as follows.

Let $S = s_1 s_2 \cdots s_n$ be a string over some alphabet Σ. The *run-length compression* of string S is the string $S' = \sigma_1^{r_1}\sigma_2^{r_2}\cdots\sigma_k^{r_k}$ such that (1) $\sigma_i \neq \sigma_{i+1}$ for $1 \leq i < k$ and (2) S can be described as the concatenation of k *segments*, the symbol σ_1 repeated r_1 times, the symbol σ_2 repeated r_2 times, . . ., and the symbol σ_k repeated r_k times. The *two-dimensional run-length compression* is the concatenation of the run-length compression of all the matrix rows (or columns).

The *two-dimensional run-length compressed matching problem* is defined as follows:
INPUT: Text array T of size $n \times n$, and pattern array P of size $m \times m$ *both* in two-dimensional run-length compressed form.
OUTPUT: All locations in T of occurrences of P. Formally, the output is the set of locations (i, j) such that $T[i + k, j + l] = P[k + 1, l + 1]$ $k, l = 0 \ldots m - 1$.

Another ubiquitous lossless two-dimensional compression is CompuServe's GIF standard, widely used on the World Wide Web. It uses LZW [19] (a variation of LZ78) on the image linearized row by row.

The *two-dimensional LZ compression* is formally defined as follows. Given an image $T[1 \ldots n, 1 \ldots n]$, create a string $T_{lin}[1 \ldots n^2]$ by concatenating all rows of T. Compressing T_{lin} with one-dimensional LZ78 yields the two-dimensional LZ compression of the image T.

The *two-dimensional LZ compressed matching problem* is defined as follows:
INPUT: Text array T of size $n \times n$, and pattern array P of size $m \times m$ *both* in two-dimensional LZ compressed form.
OUTPUT: All locations in T of occurrences of P. Formally, the output is the set of locations (i, j) such that $T[i + k, j + l] = P[k + 1, l + 1]$ $k, l = 0 \ldots m - 1$.

Key Results

The definition of compressed search first appeared in the context of searching for two dimensional run-length compression [1,2]. The following result was achieved there.

Theorem 1 (Amir and Benson [3]) *There exists an* $O(|c(T)| \log |c(T)|)$ *worst-case time solution to the compressed search problem with the two dimensional run-length compression algorithm.*

The abovementioned paper did not succeed in achieving either an optimal or an inplace algorithm. Nevertheless, it introduced the notion of *two-dimensional periodicity*. As in strings, periodicity plays a crucial rôle in two-dimensional string matching, and its advent has provided solutions to many longstanding open problems of two-dimensional string matching. In [5], it was used to achieve the

first linear-time, alphabet-independent, two-dimensional text scanning. Later, in [4,16] it was used in two different ways for a linear-time witness table construction. In [7] it was used to achieve the first parallel, time and work optimal, CREW algorithm for text scanning. A simpler variant of periodicity was used by [11] to obtain a constant-time CRCW algorithm for text scanning. A recent further attempt has been made [17] to generalize periodicity analysis to higher dimensions.

The first optimal two-dimensional compressed search algorithm was the following.

Theorem 2 (Amir et al. [6]) *There exists an $O(|c(T)|)$ worst-case time solution to the compressed search problem with the two-dimensional run-length compression algorithm.*

Optimality was achieved by a concept the authors called *witness-free dueling*. The paper proved new properties of two-dimensional periodicity. This enables duels to be performed in which no witness is required. At the heart of the dueling idea lies the concept that two overlapping occurrences of a pattern in a text can use the content of a predetermined text position or witness in the overlap to eliminate one of them. Finding witnesses is a costly operation in a compressed text; thus, the importance of witness-free dueling.

The original algorithm of Amir et al. [6] takes time $O(|c(T)| + |P| \log \sigma)$, where σ is $\min(|P|, |\Sigma|)$, and Σ is the alphabet. However with the witness table construction of Galil and Park [12] the time is reduced to $O(|c(T)| + |P|)$. Using known techniques, one can modify their algorithm so that its extra space is $O(|P|)$. This creates an optimal algorithm that is also inplace, provided the pattern is input in uncompressed form. With use of the run-length compression, the difference between $|P|$ and $|c(P)|$ can be quadratic. Therefore it is important to seek an inplace algorithm.

Theorem 3 (Amir et al. [9]) *There exists an $O(|c(T)| + |P| \log \sigma)$ worst-case time solution to the compressed search problem with the two-dimensional run-length compression algorithm, where σ is $\min(|P|, |\Sigma|)$, and Σ is the alphabet, for all patterns that have no trivial rows (rows consisting of a single repeating symbol). The amount of space used is $O(|c(P)|)$.*

This algorithm uses the framework of the noncompressed two dimensional pattern matching algorithm of [6]. The idea is to use the **dueling** mechanism defined by Vishkin [18]. Applying the dueling paradigm directly to run-length compressed matching has previously been considered impossible since the location of a witness in the compressed text cannot be accessed in constant time. In [9], a way was shown in which a witness *can* be accessed in (amortized) constant time, enabling a relatively straightforward application of the dueling paradigm to compressed matching.

A strongly inplace compressed matching algorithm exists for the two-dimensional LZ compression, but its preprocessing is not optimal.

Theorem 4 (Amir et al. [8]) *There exists an $O(|c(T)| + |P|^3 \log \sigma)$ worst-case time solution to the compressed search problem with the two-dimensional LZ compression algorithm, where σ is $\min(|P|, |\Sigma|)$, and Σ is the alphabet. The amount of space used is $O(m)$, for an $m \times m$ size pattern. $O(m)$ is the best compression achievable for any $m \times m$ sized pattern under the two-dimensional LZ compression.*

The algorithm of [8] can be applied to any two-dimensional compressed text, in which the compression technique allows sequential decompression in small space.

Applications

The problem has many applications since two-dimensional data appears in many different types of compression. The two compressions discussed here are the run-length compression, used by fax transmissions, and the LZ compression, used by GIF.

Open Problems

Any lossless two-dimensional compression used, especially one with a large compression ratio, presents the problem of enabling the search without uncompressing the data for saving of both time and space.

Searching in two-dimensional lossy compressions will be a major challenge. Initial steps in this direction can be found in [15,14], where JPEG compression is considered.

Cross References

- ▶ Compressed Pattern Matching
- ▶ Multidimensional String Matching

Recommended Reading

1. Amir, A., Benson, G.: Efficient two dimensional compressed matching. In: Proceeding of Data Compression Conference, Snow Bird, Utah, 1992, pp. 279–288
2. Amir, A., Benson, G.: Two-dimensional periodicity and its application. Proceeding of 3rd Symposium on Discrete Algorithms, Orlando, FL, 1992, pp. 440–452
3. Amir, A., Benson, G.: Two-dimensional periodicity and its application. SIAM J. Comput. **27**(1), 90–106 (1998)

4. Amir, A., Benson, G., Farach, M.: The truth, the whole truth, and nothing but the truth: Alphabet independent two dimensional witness table construction.Technical Report GIT-CC-92/52, Georgia Institute of Technology (1992)
5. Amir, A., Benson, G., Farach, M.: An alphabet independent approach to two dimensional pattern matching. SIAM J. Comput. **23**(2), 313–323 (1994)
6. Amir, A., Benson, G., Farach, M.: Optimal two-dimensional compressed matching. J. Algorithms **24**(2), 354–379 (1997)
7. Amir, A., Benson, G., Farach, M.: Optimal parallel two dimensional text searching on a crew pram. Inf. Comput. **144**(1), 1–17 (1998)
8. Amir, A., Landau, G., Sokol, D.: Inplace 2d matching in compressed images. J. Algorithms **49**(2), 240–261 (2003)
9. Amir, A., Landau, G., Sokol, D.: Inplace run-length 2d compressed search. Theor. Comput. Sci. **290**(3), 1361–1383 (2003)
10. Berman, P., Karpinski, M., Larmore, L., Plandowski, W., Rytter, W.: On the complexity of pattern matching for highly compressed two dimensional texts. Proceeding of 8th Annual Symposium on Combinatorial Pattern Matching (CPM 97). LNCS, vol. 1264, pp. 40–51. Springer, Berlin (1997)
11. Crochemore, M., Galil, Z., Gasieniec, L., Hariharan, R., Muthukrishnan, S., Park, K., Ramesh, H., Rytter, W.: Parallel two-dimensional pattern matching. In: Proceeding of 34th Annual IEEE FOCS, 1993, pp. 248–258
12. Galil, Z., Park, K.: Alphabet-independent two-dimensional witness computation. SIAM. J. Comput. **25**(5), 907–935 (1996)
13. Hennessy, J.L., Patterson, D.A.: Computer Architeture: A Quantitative Approach, 2nd edn. Morgan Kaufmann, San Francisco, CA (1996)
14. Klein, S.T., Shapira, D.: Compressed pattern matching in jpeg images. In: Proceeding Prague Stringology conference, 2005, pp. 125–134
15. Klein, S.T., Wiseman, Y.: Parallel huffman decoding with applications to jpeg files. Comput. J. **46**(5), 487–497 (2003)
16. Park, K., Galil, Z.: Truly alphabet-independent two-dimensional pattern matching. In: Proceeding 33rd IEEE FOCS, 1992, pp. 247–256
17. Régnier, M., Rostami, L.: A unifying look at d-dimensional periodicities and space coverings. In: 4th Symp. on Combinatorial Pattern Matching, 15, 1993
18. Vishkin, U.: Optimal parallel pattern matching in strings. In: Proceeding 12th ICALP, 1985, pp. 91–113
19. Welch, T.A.: A technique for high-performance data compression. IEEE Comput. **17**, 8–19 (1984)
20. Ziv, J.: Personal communication (1995)

Multidimensional String Matching

1999; Kärkkäinen, Ukkonen

Juha Kärkkäinen, Esko Ukkonen
Department of Computer Science, University of Helsinki, Helsinki, Finland

Keywords and Synonyms

Multidimensional array matching; Image matching; Template registration

Problem Definition

Given two two-dimensional arrays, the *text* $T[1 \ldots n, 1 \ldots n]$ and the *pattern* $P[1 \ldots m, 1 \ldots m]$, $m \leq n$, both with element values from *alphabet* Σ of size σ, the basic *two-dimensional string matching* (2DSM) problem is to find all *occurrences* of P in T, i. e., all $m \times m$ subarrays of T that are identical to P. In addition to the basic problem, several types of generalizations are considered: *approximate matching* (allow local errors), *invariant matching* (allow global transformations), *indexed matching* (preprocess the text), and *multidimensional matching*.

In approximate matching, an occurrence is a subarray S of the text, whose *distance* $d(S, P)$ from the pattern does not exceed a threshold k. Different distance measures lead to different variants of the problem. When no distance is explicitly mentioned, the *Hamming distance*, the number of mismatching elements, is assumed.

For one-dimensional strings, the most common distance is the Levenshtein distance, the minimum number of insertions, deletions and substitutions for transforming one string into the other. A simple generalization to two dimensions is the *Krithivasan–Sitalakshmi (KS) distance*, which is the sum of row-wise Levenshtein distances. Baeza-Yates and Navarro [6] introduced several other generalizations, one of which, the *RC distance*, is defined as follows. A two-dimensional array can be decomposed into a sequence of rows and columns by removing either the last row or the last column from the array until nothing is left. Different decompositions are possible depending on whether a row or a column is removed at each step. The RC distance is the minimum cost of transforming a decomposition of one array into a decomposition of the other, where the minimum is taken over all possible decompositions as well as all possible transformations. A transformation consists of insertions, deletions and modifications of rows and columns. The cost of inserting or deleting a row/column is the length of the row/column, and the cost of modification is the Levenshtein distance between the original and the modified row/column.

The invariant matching problems search for occurrences that match the pattern after some global transformation of the pattern. In the *scaling invariant matching* problem, an occurrence is a subarray that matches the pattern scaled by some factor. If only integral scaling factors are allowed, the definition of the problem is obvious. For real-valued scaling, a refined model is needed, where the text and pattern elements, called *pixels* in this case, are unit squares on a plane. Scaling the pattern means stretching the pixels. An occurrence is a matching M be-

tween text pixels and pattern pixels. The scaled pattern is placed on top of the text with one corner aligned, and each text pixel $T[r, s]$, whose center is covered by the pattern, is matched with the covering pattern pixel $P[r', s']$, i. e., $([r, s], [r', s']) \in M$.

In the *rotation invariant matching* problem, too, an occurrence is a matching between text pixels and pattern pixels. This time the center of the pattern is placed at the center of a text pixel and the pattern is rotated around the center. The matching is again defined by which pattern pixels cover which text pixel centers.

In the *indexed* form of the problems, the text can be preprocessed to speed up the matching. The preprocessing and matching complexities are reported separately.

All the problems can be generalized to more than two dimensions. In the d-dimensional problem, the text is an n^d array and the pattern an m^d array. The focus is on two dimensions, but multidimensional generalizations of the results are mentioned when they exist.

Many other variants of the problems are omitted here due to lack of space. Some of them as well as some of the results in this entry are surveyed by Amir [1]. A wider range of problems as well as traditional image processing techniques for solving them can be found in [9].

Key Results

The classical solution to the 2DSM problem by Bird [8] and independently by Baker [7] reduces the problem to one-dimensional string matching. It has two phases:

1. Find all occurrences of pattern rows on the text rows and mark them. This takes $\mathcal{O}(n^2 \log \min(m, \sigma))$ time using the Aho-Corasick algorithm. On an integer alphabet $\Sigma = \{0, 1, \ldots, \sigma - 1\}$, the time can be improved to $\mathcal{O}(n^2 + m^2 \min(m, \sigma) + \sigma)$ using $\mathcal{O}(m^2 \min(m, \sigma) + \sigma)$ space.
2. The pattern is considered a sequence of m rows and each $n \times m$ subarray of the text a sequence of n rows. The Knuth–Morris–Pratt string matching algorithm is used for finding the occurrences of the pattern in each subarray. The algorithm makes $\mathcal{O}(n)$ row comparisons for each of the $n - m + 1$ subarrays. With the markings from Step 1, a row comparison can be done in constant time, giving $\mathcal{O}(n^2)$ time complexity for Step 2.

The time complexity of the Bird–Baker algorithm is linear if the alphabet size σ is constant. The algorithm of Amir, Benson and Farach [2] (with improvements by Galil and Park [14]) achieves linear time independent of the alphabet size using a quite different kind of algorithm based on string matching by duels and two-dimensional periodicity.

Theorem 1 (Bird [8]; Baker [7]; Amir, Benson and Farach [2]) *The 2DSM problem can be solved in the optimal $\mathcal{O}(n^2)$ worst-case time.*

The Bird–Baker algorithm generalizes straightforwardly into higher dimensions by repeated application of Step 1 to reduce a problem in d dimensions into $n - m + 1$ problems in $d - 1$ dimensions. The time complexity is $\mathcal{O}(dn^d \log m^d)$. The Amir–Benson–Farach algorithm has been generalized to three dimensions with the time complexity $\mathcal{O}(n^3)$ [13].

The average-case complexity of the 2DSM problem was studied by Kärkkäinen and Ukkonen [15], who proved a lower bound and gave an algorithm matching the bound.

Theorem 2 (Kärkkäinen and Ukkonen [15]) *The 2DSM-problem can be solved in the optimal $\mathcal{O}(n^2(\log_\sigma m)/m^2)$ average-case time.*

The result (both lower and upper bound) generalizes to the d-dimensional case with the $\Theta(n^d \log_\sigma m/m^d)$ average-case time complexity.

Amir and Landau [5] give algorithms for approximate 2DSM problems for both the Hamming distance and the KS distance. The RC model was developed and studied by Baeza–Yates and Navarro [6].

Theorem 3 (Amir and Landau [5]; Baeza–Yates and Navarro [6]) *The approximate 2DSM problem can be solved in $\mathcal{O}(kn^2)$ worst-case time for the Hamming distance, in $\mathcal{O}(k^2 n^2)$ worst-case time for the KS distance, and in $\mathcal{O}(k^2 mn^2)$ worst-case time for the RC distance.*

The results for the KS and RC distances generalize to d dimensions with the time complexities $\mathcal{O}(k(k + d)n^d)$ and $\mathcal{O}(d! m^{2d} n^d)$, respectively.

Approximate matching algorithms with good average-case complexity are described by Kärkkäinen and Ukkonen [15] for the Hamming distance, and by Baeza-Yates and Navarro [6] for the KS and RC distances.

Theorem 4 (Kärkkäinen and Ukkonen [15]; Baeza–Yates and Navarro [6]) *The approximate 2DSM problem can be solved in $\mathcal{O}(kn^2(\log_\sigma m)/m^2)$ average-case time for the Hamming and KS distances, and in $\mathcal{O}(n^2/m)$ average-case time for the RC distance.*

The results for the Hamming and the RC distance have d-dimensional generalizations with the time complexities $\mathcal{O}(kn^d(\log_\sigma m^d)/m^d)$ and $\mathcal{O}(kn^d/m^{d-1})$, respectively.

The scaling and rotation invariant 2DSM problems involve a continuous valued parameter (scaling factor or rotation angle). However, the corresponding matching between text and pattern pixels changes only at certain points, and there are only $\mathcal{O}(nm)$ effectively distinct scales

and $\mathcal{O}(m^3)$ effectively distinct rotation angles. A separate search for each distinct scale or rotation would give algorithms with time complexities $\mathcal{O}(n^3 m)$ and $\mathcal{O}(n^2 m^3)$, but faster algorithms exist.

Theorem 5 (Amir and Chencinski [3]; Amir, Kapah and Tsur [4]) *The scaling invariant 2DSM problem can be solved in $\mathcal{O}(n^2 m)$ worst-case time, and the rotation invariant 2DSM problem in $\mathcal{O}(n^2 m^2)$ worst-case time.*

Fast average-case algorithms for the rotation invariant problem are described by Fredriksson, Navarro and Ukkonen [11]. They also consider approximate matching versions.

Theorem 6 (Fredriksson, Navarro and Ukkonen [11]) *The rotation invariant 2DSM problem can be solved in the optimal $\mathcal{O}(n^2(\log_\sigma m)/m^2)$ average-case time. The rotation invariant approximate 2DSM problem can be solved in the optimal $\mathcal{O}(n^2(k + \log_\sigma m)/m^2)$ average-case time.*

Fredriksson, Navarro and Ukkonen [11] also consider rotation invariant matching in d dimensions.

Indexed matching is based on two-dimensional suffix trees and arrays, which are the subject of another entry *2D-Pattern Indexing*. Their properties are similar to one-dimensional suffix trees and arrays.

Theorem 7 (Kim and Park [16]) *The text can be preprocessed in $\mathcal{O}(n^2)$ time so that subsequently a 2DSM query can be answered in $\mathcal{O}(m^2 \log \sigma)$ time or in $\mathcal{O}(m^2 + \log n)$ time.*

Fredriksson, Navarro and Ukkonen [11] describe an index suitable for rotation invariant matching.

Theorem 8 (Fredriksson, Navarro and Ukkonen [11]) *The text can be preprocessed in $\mathcal{O}(n^2)$ time so that subsequently a rotation invariant 2DSM query can be answered in $\mathcal{O}((\log_\sigma n)^{5/2})$ average-case time and a rotation invariant approximate 2DSM query can be answered in $\mathcal{O}((2\log_\sigma n)^{k+3/2}\sigma^k)$ average-case time.*

Applications

The main application area is pattern matching in images, particularly applications where the point of view in the image is well-defined, such as aerial and astronomical photography, optical character recognition, and biomedical imaging. Even three-dimensional problems arise in biomedical applications [12].

Open Problems

Many combinations of the different variants of the problem have not been studied. Combining scaling and rotation invariance is an example. With rotation invariant approximate matching under the RC distance even the problem needs further specification.

Experimental Results

No conclusive results exist though some experiments are reported in [10,12,15].

Cross References

Many of the problems and their solutions are related to one-dimensional string matching problems and techniques described in the entry ► Sequential Approximate String Matching. The construction of text index structures is described in ► Two-Dimensional Pattern Indexing. Matching on a compressed text without decompression is considered in ► Multidimensional Compressed Pattern Matching.

Recommended Reading

1. Amir, A.: Theoretical issues of searching aerial photographs: a bird's eye view. Int. J. Found. Comput. Sci. **16**, 1075–1097 (2005)
2. Amir, A., Benson, G., Farach, M.: An alphabet independent approach to two-dimensional pattern matching. SIAM J. Comput. **23**, 313–323 (1994)
3. Amir, A., Chencinski, E.: Faster two dimensional scaled matching. In: Proc. 17th Annual Symposium on Combinatorial Pattern Matching. LNCS, vol. 4009, pp. 200–210. Springer, Berlin (2006)
4. Amir, A., Kapah, O., Tsur, D.: Faster two dimensional pattern matching with rotations. In: Proc. 15th Annual Symposium on Combinatorial Pattern Matching. LNCS, vol. 3109, pp. 409–419. Springer, Berlin (2004)
5. Amir, A., Landau, G.M.: Fast parallel and serial multidimensional approximate array matching. Theoretical Comput. Sci. **81**, 97–115 (1991)
6. Baeza-Yates, R., Navarro, G.: New models and algorithms for multidimensional approximate pattern matching. J. Discret. Algorithms **1**, 21–49 (2000)
7. Baker, T.P.: A technique for extending rapid exact-match string matching to arrays of more than one dimension. SIAM J. Comput. **7**, 533–541 (1978)
8. Bird, R.S.: Two dimensional pattern matching. Inf. Process. Lett. **6**, 168–170 (1977)
9. Brown, L.G.: A survey of image registration techniques. ACM Computing Surveys **24**, 325–376 (1992)
10. Fredriksson, K., Navarro, G., Ukkonen, E.: Faster than FFT: Rotation invariant combinatorial template matching. In: Pandalai, S. (ed.) Recent Research Developments in Pattern Recognition, vol. II, pp. 75–112. Transworld Research Network, Trivandrum, India (2002)
11. Fredriksson, K., Navarro, G., Ukkonen, E.: Sequential and indexed two-dimensional combinatorial template matching allowing rotations. Theoretical Comput. Sci. **347**, 239–275 (2005)

12. Fredriksson, K., Ukkonen, E.: Combinatorial methods for approximate pattern matching under rotations and translations in 3D arrays. In: Proc. 7th International Symposium on String Processing and Information Retrieval, pp. 96–104. IEEE Computer Society, Washington, DC (2000)
13. Galil, Z., Park, J.G., Park, K.: Three-dimensional periodicity and its application to pattern matching. SIAM J. Discret. Math. **18**, 362–381 (2004)
14. Galil, Z., Park, K.: Alphabet-independent two-dimensional witness computation. SIAM J. Comput. **25**, 907–935 (1996)
15. Kärkkäinen, J., Ukkonen, E.: Two- and higher-dimensional pattern matching in optimal expected time. SIAM J. Comput. **29**, 571–589 (1999)
16. Kim, D.K., Park, K.: Linear-time construction of two-dimensional suffix trees. In: Proc. 26th International Colloquium on Automata Languages and Programming. LNCS, vol. 1644, pp. 463–472. Springer, Berlin (1999)

Multi-Hop Radio Networks, Ad Hoc Networks

► Randomized Broadcasting in Radio Networks

Multi-level Feedback Queues

1968; Coffman, Kleinrock

Nikhil Bansal
IBM Research, IBM, Yorktown Heights, NY, USA

Keywords and Synonyms

Fairness; Low sojourn times; Scheduling with unknown job sizes

Problem Definition

The problem is concerned with scheduling dynamically arriving jobs in the scenario when the processing requirements of jobs are unknown to the scheduler. This is a classic problem that arises for example in CPU scheduling, where users submit jobs (various commands to the operating system) over time. The scheduler is only aware of the existence of the job and does not know how long it will take to execute, and the goal is to schedule jobs to provide good quality of service to the users. Formally, this note considers the average flow time measure, defined as the average duration of time since a job is released until its processing requirement is met.

Notations

Let $\mathcal{J} = \{1, 2, \ldots, n\}$ denote the set of jobs in the input instance. Each job j is characterized by its release time r_j and its processing requirement p_j. In the online setting, job j is revealed to the scheduler only at time r_j. A further restriction is the *non-clairvoyant* setting, where only the existence of job j is revealed at r_j, in particular the scheduler does not know p_j until the job meets its processing requirement and leaves the system. Given a schedule, the completion time c_j of a job is the earliest time at which job j receives p_j amount of service. The flow time f_j of j is defined as $c_j - r_j$. A schedule is said to be preemptive, if a job can be interrupted arbitrarily, and its execution can be resumed later from the point of interruption without any penalty. It is well known that preemption is necessary to obtain reasonable guarantees even in the offline setting [4].

There are several natural non-clairvoyant algorithms such as First Come First Served, Processor Sharing (work on all current unfinished jobs at equal rate), Shortest Elapsed Time First (work on job that has received least amount of service thus far). Coffman and Kleinrock [2] proposed another natural algorithm known as the Multi-Level Feedback Queueing (MLF). MLF works as follows: There are queues $Q_0, Q_1, Q_2, \ldots$ and thresholds $0 < t_0 < t_1 < t_2 \ldots$. Initially upon arrival, a job is placed in Q_0. When a job in Q_i receives t_i amount of cumulative service, it is moved to Q_{i+1}. The algorithm at any time works on the lowest numbered non-empty queue. Coffman and Kleinrock analyzed MLF in a queuing theoretic setting, where the jobs arrive according to a Poisson process and the processing requirements are chosen identically and independently from a known probability distribution.

Recall that the online Shortest Remaining Processing Time (SRPT) algorithm, that at any time works on the job with the least remaining processing time, produces an optimum schedule. However, SRPT requires the knowledge of job sizes and hence is not non-clairvoyant. Since a non-clairvoyant algorithm only knows a lower bound on a jobs size (determined by the amount of service it has received thus far), MLF tries to mimic SRPT by favoring jobs that have received the least service thus far.

Key Results

While non-clairvoyant algorithms have been studied extensively in the queuing theoretic setting for many decades, this notion was considered relatively recently in the context of competitive analysis by Motwani, Phillips and Torng [5]. As in traditional competitive analysis, a non-clairvoyant algorithm is called c-competitive if for every input instance, its performance is no worse than c times than optimum offline solution for that in-

stance. Motwani, Phillips and Torng showed the following.

Theorem 1 ([5]) *For the problem of minimizing average flow time on a single machine, any deterministic nonclairvoyant algorithm must have a competitive ratio of at least $\Omega(n^{1/3})$ and any randomized algorithm must have a competitive ratio of at least $\Omega(\log n)$, where n is number of jobs in the instance.*

It is not too surprising that any deterministic algorithm must have a poor competitive ratio. For example, consider MLF where the thresholds are powers of 2, i. e. $1, 2, 4, \ldots$. Say $n = 2^k$ jobs of size $2^k + 1$ each arrive at times $0, 2^k, 2 \cdot 2^k, \ldots, (2^k - 1)2^k$ respectively. Then, it is easily verified that the average flow time under MLF is $\Omega(n^2)$, where as the average flow time is under the optimum algorithm is $\Omega(n)$.

Note that MLF performs poorly on the above instance since all jobs are stuck till the end with just one unit of work remaining. Interestingly, Kalyanasundaram and Pruhs [3] designed a randomized variant of MLF (known as RMLF) and proved that its competitive ratio is almost optimum. For each job j, and for each queue Q_i, the RMLF algorithm sets a threshold $t_{i,j}$ randomly and independently according to a truncated exponential distribution. Roughly speaking, setting a random threshold ensures that if a job is stuck in a queue, then its remaining processing is a reasonable fraction of its original processing time.

Theorem 2 ([3]) *The RMLF algorithm is $O(\log n \log\log n)$ competitive against an oblivious adversary. Moreover, the RMLF algorithm is $O(\log n \log\log n)$ competitive even against an adaptive adversary provided the adversary chooses all the job sizes in advance.*

Later, Becchetti and Leonardi [1] showed that in fact the RMLF is optimally competitive up to constant factors. They also analyzed RMLF on identical parallel machines.

Theorem 3 ([1]) *The RMLF algorithm is $O(\log n)$ competitive for a single machine. For multiple identical machines, RMLF achieves a competitive ratio of $O(\log n \log(\frac{n}{m}))$, where m is the number of machines.*

Applications

MLF and its variants are widely used in operating systems [6,7]. These algorithms are not only close to optimum with respect to flow time, but also have other attractive properties such as the amortized number of preemptions is logarithmic (preemptions occur only if a job arrives or departs or moves to another queue).

Open Problems

It is not known whether there exists a $o(n)$-competitive deterministic algorithm. It would be interesting to close the gap between the upper and lower bounds for this case. Often in real systems, even though the scheduler may not know the exact job size, it might have some information about its distribution based on historical data. An interesting direction of research could be to design and analyze algorithms that use this information.

Cross References

- ► Flow Time Minimization
- ► Minimum Flow Time
- ► Shortest Elapsed Time First Scheduling

Recommended Reading

1. Becchetti, L., Leonardi, S.: Nonclairvoyant scheduling to minimize the total flow time on single and parallel machines. J. ACM (JACM) **51**(4), 517–539 (2004)
2. Coffman, E.G., Kleinrock, L.: Feedback Queueing Models for Time-Shared Systems. J. ACM (JACM) **15**(4), 549–576 (1968)
3. Kalyanasundaram, B., Pruhs, K.: Minimizing flow time nonclairvoyantly. J. ACM (JACM) **50**(4), 551–567 (2003)
4. Kellerer, H., Tautenhahn, T., Woeginger, G.J.: Approximability and Nonapproximability Results for Minimizing Total Flow Time on a Single Machine. SIAM J. Comput. **28**(4), 1155–1166 (1999)
5. Motwani, R., Phillips, S., Torng, E.: Non-Clairvoyant Scheduling. Theor. Comput. Sci. **130**(1), 17–47 (1994)
6. Nutt, G.: Operating System Projects Using Windows NT. Addison Wesley, Reading (1999)
7. Tanenbaum, A.S.: Modern Operating Systems. Prentice-Hall Inc., Upper Saddle River (1992)

Multiple String Alignment

- ► Efficient Methods for Multiple Sequence Alignment with Guaranteed Error Bounds

Multiple Unit Auctions with Budget Constraint

2005; Borgs, Chayes, Immorlica, Mahdian, Saberi
2006; Abrams

Tian-Ming Bu
Department of Computer Science, Fudan University, Shanghai, China

Problem Definition

In this problem, an auctioneer would like to sell an idiosyncratic commodity with m copies to n bidders, de-

noted by $i = 1, 2, \ldots, n$. Each bidder i has two kinds of privately known information: $t_i^u \in \mathbb{R}^+$, $t_i^b \in \mathbb{R}^+$. $t_i^u \in \mathbb{R}^+$ represents the price buyer i is willing to pay per copy of the commodity and $t_i^b \in \mathbb{R}^+$ represents i's budget.

Then a *one-round sealed-bid* auction proceeds as follows. *Simultaneously* all the bidders submit their bids to the auctioneer. When receiving the reported unit value vector $\mathbf{u} = (u_1, \ldots, u_n)$ and the reported budget vector $\mathbf{b} = (b_1, \ldots, b_n)$ of bids, the auctioneer computes and outputs the allocation vector $\mathbf{x} = (x_1, \ldots, x_n)$ and the price vector $\mathbf{p} = (p_1, \ldots, p_n)$. Each element of the allocation vector indicates the number of copies allocated to the corresponding bidder. If bidder i receives x_i copies of the commodity, he pays the auctioneer $p_i x_i$. Then bidder i's total payoff is $(t_i^u - p_i)x_i$ if $x_i p_i \le t_i^b$ and $-\infty$ otherwise. Correspondingly, the revenue of the auctioneer is $\mathcal{A}(\mathbf{u}, \mathbf{b}, m) = \sum_i p_i x_i$.

If each bidder submits his privately *true* unit value t_i^u and budget t_i^b to the auctioneer, the auctioneer can determine the single price $p_{\mathcal{F}}$ (i. e., $\forall i$, $p_i = p_{\mathcal{F}}$) and the allocation vector which maximize the auctioneer's revenue. This optimal single price revenue is denoted by $\mathcal{F}(\mathbf{u}, \mathbf{b}, m)$.

Interestingly, in this problem, we assume bidders have free will, and have complete knowledge of the auction mechanism. Bidders would just report the bid (maybe different from his corresponding privately true values) which could maximize his payoff according to the auction mechanism.

So the objective of the problem is to design a *truthful* auction satisfying *voluntary participation* to raise the auctioneer's revenue as much as possible. An auction is *truthful* if for every bidder i, bidding his true valuation would maximize his payoff, regardless of the bids submitted by the other bidders [8,9]. An auction satisfies *voluntary participation* if each bidder's payoff is guaranteed to be non-negative if he reports his bid truthfully. The performance of the auction $\mathcal{A}$ is determined by competitive ratio β which is defined as the upper bound of $\frac{\mathcal{F}(\mathbf{u},\mathbf{b},m)}{\mathcal{A}(\mathbf{u},\mathbf{b},m)}$ [5]. Clearly, the smaller the competitive ratio β is, the better the auction $\mathcal{A}$ is.

Definition (Multiple Unit Auctions with Budget Constraint)
INPUT: the number of copies m, the submitted unit value vector $\mathbf{u}$, the submitted budget vector $\mathbf{b}$.
OUTPUT: the allocation vector $\mathbf{x}$ and the price vector $\mathbf{p}$.
CONSTRAINTS:
(a) Truthful
(b) Voluntary participation
(c) $\sum_i x_i \le m$.

Key Results

Let $b_{\max}$ denote the largest budget amongst the bidders receiving copies in the optimal solution and define $\alpha = \frac{\mathcal{F}}{b_{\max}}$.

Theorem 1 ([2]) *A truthful auction satisfying voluntary participation with competitive ratio* $1/\max_{0<\delta<1}\{(1-\delta)(1-2e^{-\frac{\alpha\delta^2}{36}})\}$ *can be designed.*

Theorem 2 ([1]) *A truthful auction satisfying voluntary participation with competitive ratio* $\frac{4\alpha}{\alpha-1}$ *can be designed.*

Theorem 3 ([1]) *If* α *is known in advance, then a truthful auction satisfying voluntary participation with competitive ratio* $\frac{(x\alpha+1)\alpha}{(x\alpha-1)^2}$ *can be designed, where* $x = \frac{\alpha-1+((\alpha-1)^2-4\alpha)^{1/2}}{2\alpha}$.

Theorem 4 ([1]) *For any truthful randomized auction* $\mathcal{A}$ *satisfying voluntary participation, the competitive ratio is at least* $2-\epsilon$ *when* $\alpha \ge 2$.

Applications

This problem is motivated by the development of the IT industry and the popularization of auctions, especially, auctions on the Internet. The multiple copy auction of relatively low-value goods, such as the auction of online ads for search terms to bidders with budget constraint, is assuming a very important role. Companies such as Google and Yahoo!'s revenue depends almost on certain types of auctions.

All previous work concerning auctions with budget constraint only focused on the traditional physical world where the object to sell is almost unique, such as antiques, paintings, land and nature resources. More specifically, [4] studied the problem of single unit, single bidder with a budget. [6] studied the model of single unit, multiple bidders with common public budget. [7] studied the problem of single unit, multiple bidders with flexible budgets.

Recently, [3] extended this problem so that the auctioneer could sell unlimited copies of goods and the bidders have both budget and copy constraints. This general model is especially suitable for digital goods, which can produce unlimited copies with marginal cost zero, such as license sales, mp3 copies, online advertisements, etc. Further, the auction mechanism designed in [3] could obtain a similar competitive ratio.

Cross References

► Competitive Auction

Recommended Reading

1. Abrams, Z.: Revenue maximization when bidders have budgets. In: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-06), Miami, Florida, 22–26 Jan 2006, pp. 1074–1082. ACM Press, New York
2. Borgs, C., Chayes, J.T., Immorlica, N., Mahdian, M., Saberi, A.: Multi-unit auctions with budget-constrained bidders. In: ACM Conference on Electronic Commerce (EC-05), 2005, pp. 44–51
3. Bu, T.-M., Qi, Q., Sun, A.W.: Unconditional competitive auctions with copy and budget constraints. In: Spirakis, P.G., Mavronicolas, M., Kontogiannis, S.C. (eds.) Internet and Network Economics, 2nd International Workshop, WINE 2006, Patras, Greece, 15–17 Dec 2006. Lecture Notes in Computer Science, vol. 4286, pp. 16–26. Springer, Berlin (2006)
4. Che, Y.-K., Gale, I.: Standard auctions with financially constrained bidders. Rev. Econ. Stud. **65**(1), 1–21 (1998)
5. Goldberg, A.V., Hartline, J.D., Karlin, A.R., Wright, A.: Competitive auctions. Games Econ. Behav. **55**(2), 242–269 (2006)
6. Laffont, J.-J., Robert, J.: Optimal auction with financially constrained buyers. Econ. Lett. **52**, 181–186 (1996)
7. Maskin, E.S.: Auctions, development, and privatization: Efficient auctions with liquidity-constrained buyers. Eur. Econ. Rev. **44**(4–6), 667–681 (2000)
8. Nisan, N. Ronen, A.: Algorithmic mechanism design. In: Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC-99), pp. 129–140. Association for Computing Machinery, New York (1999)
9. Parkes, D.C.: Chapter 2: Iterative Combinatorial Auctions. Ph. D. thesis, University of Pennsylvania (2004)

Multiplex PCR for Gap Closing (Whole-genome Assembly)

2002; Alon, Beigel, Kasif, Rudich, Sudakov

Vera Asodi
Center for the Mathematics of Information, California Institute of Technology, Pasadena, CA, USA

Keywords and Synonyms

Whole genome assemble; Multiplex PCR

Problem Definition

This problem is motivated by an important and timely application in computational biology that arises in whole-genome shotgun sequencing. Shotgun sequencing is a high throughput technique that has resulted in the sequencing of a large number of bacterial genomes as well as Drosophila (fruit fly) and Mouse and the celebrated Human genome (at Celera) (see, e. g. [8]). In all such projects, one is left with a collection of DNA fragments. These fragments are subsequently assembled, in-silico, by a computational algorithm. The typical assembly algorithm repeatedly merges overlapping fragments into longer fragments called contigs. For various biological and computational reasons some regions of the DNA cannot covered by the contigs. Thus, the contigs must be ordered and oriented and the gaps between them must be sequenced using slower, more tedious methods. For further details see, e. g., [3]. When the number of gaps is small (e. g., less than ten) biologists often use combinatorial PCR. This technique initiates a set of "bi-directional molecular walks" along the gaps in the sequence; these walks are facilitated by PCR. In order to initiate the molecular walks biologists use primers. Primers are designed so that they bind to unique (with respect to the entire DNA sequence) templates occurring at the end of each contig. A primer (at the right temperature and concentration) anneals to the designated unique DNA substring and promotes copying of the template starting from the primer binding site, initiating a one-directional walk along the gap in the DNA sequence. A PCR reaction occurs, and can be observed as a DNA ladder, when two primers that bind to positions on two ends of the same gap are placed in the same test tube.

If there are N contigs, the combinatorial (exhaustive) PCR technique tests all possible pairs (quadratically many) of $2N$ primers by placing two primers per tube with the original uncut DNA strand. PCR products can be detected using gels or they can be read using sequencing technology or DNA mass-spectometry. When the number of gaps is large, the quadratic number of PCR experiments is prohibitive, so primers are pooled using $K > 2$ primers per tube; this technique is called multiplex PCR [4]. This problem deals with finding optimal strategies for pooling the primers to minimize the number of biological experiments needed in the gap-closing process.

This problem can be modeled as the problem of identifying or learning a hidden matching given a vertex set V and an allowed query operation: for a subset $F \subseteq V$, the query Q_F is "does F contain at least one edge of the matching"? In this formulation each vertex represents a primer, an edge of the matching represents a reaction, and the query represents checking for a reaction when a set of primers are combined in a test tube. The objective is to identify the matching asking as few queries as possible, that is performing as few tests as possible. For further discussion of this model see [3,7].

This problem is of interest even in the deterministic, fully non-adaptive case. A family $\mathcal{F}$ of subsets of a vertex set V solves the matching problem on V if for any two distinct matchings M_1 and M_2 on V there is at least one $F \in \mathcal{F}$ that contains an edge of one of the matchings and does not contain any edge of the other. Obviously, any

such family enables learning an unknown matching deterministically and non-adaptively, by asking the questions Q_F for each $F \in \mathcal{F}$. The objective here is to determine the minimum possible cardinality of a family that solves the matching problem on a set of n vertices.

Other interesting variants of this problem are when the algorithm may be randomized, or when it is adaptive, that is when the queries are asked in k rounds, and the queries of each round may depend on the answers from the previous rounds.

Key Results

In [2], the authors study the number of queries needed to learn a hidden matching in several models. Following is a summary of the main results presented in this paper.

The trivial upper bound on the size of a family that solves the matching problem on n vertices is $\binom{n}{2}$, achieved by the family of all pairs of vertices. Theorem 1 shows that in the deterministic non-adaptive setting one cannot do much better than this, namely, that the trivial upper bound is tight up to a constant factor. Theorem 2 improves this upper bound by showing a family of approximately half that size that solves the matching problem.

Theorem 1 *For every $n > 2$, every family $\mathcal{F}$ that solves the matching problem on n vertices satisfies*

$$|\mathcal{F}| \geq \frac{49}{153}\binom{n}{2}.$$

Theorem 2 *For every n there exists a family of size*

$$\left(\frac{1}{2} + o(1)\right)\binom{n}{2}$$

that solves the matching problem on n vertices.

Theorem 3 shows that one can do much better using randomized algorithms. That is, one can learn a hidden matching asking only $O(n \log n)$ queries, rather than order of n^2. These randomized algorithms make no errors, however, they might ask more queries with some small probability.

Theorem 3 *The matching problem on n vertices can be solved by probabilistic algorithms with the following parameters:*

- *2 rounds and $(1/(2\ln 2))n \log n(1 + o(1)) \approx 0.72n \log n$ queries*
- *1 round and $(1/\ln 2)n \log n(1 + o(1)) \approx 1.44n \log n$ queries.*

Finally, Theorem 4 considers adaptive algorithms. In this case there is a tradeoff between the number of queries and the number of rounds. The more rounds one allows, the fewer tests are needed, however, as each round can start only after the previous one is completed, this increases the running time of the entire procedure.

Theorem 4 *For all $3 \leq k \leq \log n$, there is a deterministic k-round algorithm for the matching problem on n vertices that asks*

$$O\left(n^{1+\frac{1}{2(k-1)}}(\log n)^{1+\frac{1}{k-1}}\right)$$

queries per round.

Applications

As described in Sect. "Problem Definition", this problem was motivated by the application of gap closing in whole-genome sequencing, where the vertices correspond to primers, the edges to PCR reactions between pairs of primers that bind to the two ends of a gap, and the queries to tests in which a set of primers are combined in a test tube.

This gap-closing problem can be stated more generally as follows. Given a set of chemicals, a guarantee that each chemical reacts with at most one of the others, and an experimental mechanism to determine whether a reaction occurs when several chemicals are combined in a test tube, the objective is to determine which pairs of chemicals react with each other with a minimum number of experiments.

Another generalization which may have more applications in molecular biology is when the hidden subgraph is not a matching but some other fixed graph, or a family of graphs. The paper [2], as well as some other related works (e. g. [1,5,6]), consider this generalization for other graphs. Some of these generalizations have other specific applications in molecular biology.

Open Problems

- Determine the smallest possible constant c such that there is a deterministic non-adaptive algorithm for the matching problem on n vertices that performs $c\binom{n}{2}(1 + o(1))$ queries.
- Find more efficient deterministic k-round algorithms or prove lower bounds for the number of queries in such algorithms.
- Find efficient algorithms and prove lower bounds for the generalization of the problem to graphs other than matchings.

Recommended Reading

1. Alon, N., Asodi, V.: Learning a hidden subgraph, ICALP. LNCS **3142**, 110–121 (2004). Also: SIAM J. Discret. Math. **18**, 697–712 (2005)
2. Alon, N., Beigel, R., Kasif, S., Rudich, S., Sudakov, B.: Learning a Hidden Matching, Proceedings of the 43rd IEEE FOCS, 2002, 197–206. Also: SIAM J. Computing **33**, 487–501 (2004)
3. Beigel, R., Alon, N., Apaydin, M.S., Fortnow, L., Kasif, S.: An optimal procedure for gap closing in whole genome shotgun sequencing. Proc. RECOMB, ACM Press pp. 22–30. (2001)
4. Burgart, L.J., Robinson, R.A., Heller, M.J., Wilke, W.W., Iakoubova, O.K., Cheville, J.C.: Multiplex polymerase chain reaction. Mod. Pathol. **5**, 320–323 (1992)
5. Grebinski, V., Kucherov, G.: Optimal Query Bounds for Reconstructing a Hamiltonian Cycle in Complete Graphs. Proc. 5th Israeli Symposium on Theoretical Computer Science, pp. 166–173. (1997)
6. Grebinski, V., Kucherov, G.: Reconstructing a Hamiltonian Cycle by Querying the Graph: Application to DNA Physical Mapping. Discret. Appl. Math. **88**, 147–165 (1998)
7. Tettelin, H., Radune, D., Kasif, S., Khouri, H., Salzberg, S.: Pipette Optimal Multiplexed PCR: Efficiently Closing Whole Genome Shotgun Sequencing Project. Genomics **62**, 500–507 (1999)
8. Venter, J.C., Adams, M.D., Sutton, G.G., Kerlavage, A.R., Smith, H.O., Hunkapiller, M.: Shotgun sequencing of the human genome. Science **280**, 1540–1542 (1998)

Multiway Cut

1998; Calinescu, Karloff, Rabani

Gruia Calinescu
Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA

Keywords and Synonyms

Multiterminal cut

Problem Definition

Given an undirected graph with edge costs and a subset of k nodes called *terminals*, a *multiway cut* is a subset of edges whose removal disconnects each terminal from the rest. Multiway Cut is the problem of finding a multiway cut of minimum cost.

Previous Work

Dahlhaus, Johnson, Papadimitriou, Seymour, and Yannakakis [6] initiated the study of Multiway Cut and proved that Multiway Cut is MAX SNP-hard even when restricted to instances with three terminals and unit edge costs. Therefore, unless $P = NP$, there is no polynomial-time approximation scheme for Multiway Cut. For $k = 2$, the problem is identical to the undirected version of the extensively studied s-t min-cut problem of Ford and Fulkerson, and thus has polynomial-time algorithms (see, e. g., [1]). Prior to this paper, the best (and essentially the only) approximation algorithm for $k \geq 3$ was due to the above-mentioned paper of Dahlhaus et al. They give a very simple combinatorial *isolation heuristic* that achieves an approximation ratio of $2(1 - 1/k)$. Specifically, for each terminal i, find a minimum-cost cut separating i from the remaining terminals, and then output the union of the $k - 1$ cheapest of the k cuts. For $k = 4$ and for $k = 8$, Alon (see [6]) observed that the isolation heuristic can be modified to give improved ratios of 4/3 and 12/7, respectively.

In special cases, far better results are known. For fixed k in planar graphs, the problem is solvable in polynomial time [6]. For trees and 2-trees, there are linear-time algorithms [5]. For dense unweighted graphs, there is a polynomial-time approximation scheme [2,8].

Key Results

Theorem 1 ([3]) *There is a deterministic polynomial time algorithm that finds a multiway cut of cost at most $(1.5 - 1/k)$ times the optimum multiway cut.*

The approximation algorithm from Theorem 1 is based on a novel linear programming relaxation described later. On the basis of the same linear program, the approximation ratio was subsequently improved to 1.3438 by Karger, Klein, Stein, Thorup, and Young [10]. For three terminals, [10] and Cheung, Cunningham, and Tang [4] give very different 12/11-approximation algorithms.

Two variations of the problem have been considered in the literature: Garg, Vazirani, and Yannakakis [9] obtain a $(2 - 2/k)$-approximation ratio for the node-weighted version, and Naor and Zosin [11] obtain 2-approximation for the case of directed graphs. It is known that any approximation ratio for these variations translates immediately into the same approximation ratio for Vertex Cover, and thus it is hard to get any significant improvement over the approximation ratio of 2.

The algorithm from Theorem 1 appears next, giving a flavor of how this result is obtained. The complete proof of the approximation ratio is not long and appears in [3] or the book [12].

Notation

Let $G = (V, E)$ be an undirected graph on $V = \{1, 2, \ldots, n\}$ in which each edge $uv \in E$ has a non-negative cost $c(u, v) = c(v, u)$, and let $T = \{1, 2, \ldots, k\} \subseteq V$ be a set

of *terminals*. MULTIWAY CUT is the problem of finding a minimum cost set $C \subseteq E$ such that in $(V, E \setminus C)$, each of the terminals $1, 2, \ldots, k$ is in a different component. Let MWC = MWC(G) be the value of the optimal solution to MULTIWAY CUT.

Δ_k denotes the $(k-1)$-simplex, i. e., the $(k-1)$-dimensional convex polytope in $\mathbb{R}^k$ given by $\{x \in \mathbb{R}^k | (x \geq 0) \wedge (\sum_i x_i = 1)\}$.

For $x \in \mathbb{R}^k$, $\|x\|$ is its L_1 norm: $\|x\| = \sum_i |x_i|$. For $j = 1, 2, \ldots, k$, $e^j \in \mathbb{R}^k$ denotes the unit vector given by $(e^j)_j = 1$ and $(e^j)_i = 0$ for all $i \neq j$.

LP-Relaxation

The *simplex* relaxation for MULTIWAY CUT with edge costs has as variables k-dimensional real vectors x^u, defined for each vertex $u \in V$:

$$\text{Minimize } \frac{1}{2} \sum_{uv \in E} c(u,v) \cdot \|x^u - x^v\|$$

Subject to:

$$x^u \in \Delta_k \qquad \forall u \in V$$
$$x^t = e^t \qquad \forall t \in T.$$

In other words, the terminals stay at the vertices of the $(k-1)$-simplex, and the other nodes anywhere in the simplex, and measure an edge's length by the total variation distance between its endpoints. Clearly, placing all nodes at simplex vertices gives an integral solution: the lengths of edges are either 0 (if both endpoints are at the same vertex) or 1 (if the endpoints are at different vertices), and the removal of all unit length edges disconnects the graph into at least k components, each containing at most one terminal.

To solve this relaxation as a linear program, new variables are introduced: y^{uv}, defined for all $uv \in E$, and x_i^u, defined for all $u \in V$ and $i \in T$. Also new variables are y_i^{uv}, defined for all $i \in T$ and $uv \in E$. Then one writes the linear program:

$$\text{Minimize } \frac{1}{2} \sum_{uv \in E} c(u,v) y^{uv}$$

Subject to:

$$x^u \in \Delta_k \qquad \forall u \in V$$
$$x^t = e^t \qquad \forall t \in T$$
$$y^{uv} = \sum_{i \in T} y_i^{uv} \qquad \forall uv \in E$$
$$y_i^{uv} \geq x_i^u - x_i^v \qquad \forall uv \in E\,, \quad i \in T$$
$$y_i^{uv} \geq x_i^v - x_i^u \qquad \forall uv \in E\,, \quad i \in T\,.$$

It is easy to see that this linear program optimally solves the simplex relaxation above, by noticing that an optimal solution to the linear program can be assumed to put $y_i^{uv} = |x_i^u - x_i^v|$ and $y^{uv} = \|x^u - x^v\|$. Thus, solving the simplex relaxation can be done in polynomial time. This is the first step of the approximation algorithm. Clearly, the value Z^* of this solution is a lower bound on the cost of the minimum multiway cut MWC.

The second step of the algorithm is a rounding procedure which transforms a feasible solution of the simplex relaxation into an integral feasible solution. The rounding procedure below differs slightly from the one given in [3], but can be proven to give exactly the same solution. This variant is easier to present, although if one wants to prove the approximation ratio then the only way we know of is by showing that indeed this variant gives the same solution as the more complicated algorithm given in [3].

Rounding

Set $B(i, \rho) = \{u \in V \mid x_i^u > 1 - \rho\}$, the set of nodes suitably "close" to terminal i in the simplex. Choose a permutation $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_k \rangle$ to be either $\langle 1, 2, 3, \ldots, k-1, k \rangle$ or $\langle k-1, k-2, k-3, \ldots, 1, k \rangle$ with probability 1/2 each. Independently, choose $\rho \in (0, 1)$ uniformly at random. Then, process the terminals in the order $\sigma(1), \sigma(2), \sigma(3), \ldots, \sigma(k)$. For each j from 1 to $k-1$, place the nodes that remain in $B(\sigma_j, \rho)$ at e^{σ_j}. Place whatever nodes remain at the end at e^k. The following code specifies the rounding procedure more formally. $\bar{x}$ denotes the rounded (integral) solution.

```
1: Let σ = ⟨1, ..., k−3, k−2, k−1, k⟩ or ⟨k−1, k−2, k−3, ..., 1, k⟩, each with prob. 1/2
2: Let ρ be a random real in (0, 1) /* See the paragraph below. */
3: for j = 1 to k − 1 do
4:     for all u such that x^u ∈ B(σ_j, ρ) \ ∪_{i:i<j} B(σ_i, ρ) do
5:         x̄^u := e^{σ_j}  /* assign node u to terminal σ_j */
6:     end for
7: end for
8: for all u such that x^u ∉ ∪_{i:i<k} B(σ_i, ρ) do
9:     x̄^u := e^k
10: end for
```

Multiway Cut, Algorithm 1
The Rounding Procedure

To derandomize and implement this algorithm in polynomial time, one tries both permutations σ and at most

$k(n+1)$ values of ρ. Indeed, for any permutation σ, two different values of ρ, $\rho_1 < \rho_2$, produce combinatorially distinct solutions only if there is a terminal i and a node u such that $x_i^u \in (1-\rho_2, 1-\rho_1]$. Thus, there are at most $k(n+1)$ "interesting" values of ρ, which can be determined easily by sorting the nodes according to each coordinate separately. The resulting discrete sample space for (σ, ρ) has size at most $2k(n+1)$, so one can search it exhaustively.

The analysis of the algorithm, however, is based on the randomized algorithm above, as the proof shows that the expected total cost of edges whose endpoints are at different vertices of Δ_k in the rounded solution $\tilde{x}$ is at most $1.5\,Z^*$. To get an $(1.5 - 1/k)Z^*$ upper bound, one must rename the terminals such that terminal k maximizes a certain quantity given by the simplex relaxation, or alternatively randomly pick a terminal as the last element of the permutation (the order of the first $k-1$ terminals does not matter as long as both the increasing and the decreasing permutations are tried by the rounding procedure). Exhaustive search of the sample space produces one integral solution whose cost does not exceed the average.

Applications

MULTIWAY CUT is used in Computer Vision, but unless one can solve the instance exactly, algorithms for the generalization METRIC LABELING are needed. MULTIWAY CUT has applications in parallel and distributed computing, as well as in chip design.

Open Problems

The improvements of [10,4] are based on better rounding procedures and both compare the integral solution obtained to Z^*. This leads to the natural question: what is the supremum, over multiway cut instances G, of $Z^*(G)/\mathrm{MWC}(G)$. This supremum is called *integrality gap* or *integrality ratio*. For three terminals, [10] and [4] show that the integrality gap is exactly 12/11, while for general k, Freund and Karloff [7] give a lower bound of 8/7. The best-known upper bound is 1.3438, achieved by an approximation algorithm of [10].

Cross References

- ▶ Multicut
- ▶ Sparsest Cut

Recommended Reading

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice Hall, Englewood Cliffs (1993)
2. Arora, S., Karger, D., Karpinski, M.: Polynomial time approximation schemes for dense instances of NP-hard problems. J. Comput. Syst. Sci. **58**(1), 193–210 (1999). Preliminary version in STOC 1995
3. Calinescu, G., Karloff, H.J., Rabani, Y.: An Improved Approximation Algorithm for Multiway Cut. In: ACM Symposium on Theory of Computing 1998, pp. 48–52. Journal version in J. Comp. Syst. Sci. **60**, 564–574 (2000)
4. Cheung, K., Cunningham, W.H., Tang, L.: Optimal 3-Terminal Cuts and Linear Programming. Math. Program. **105**, 389–421 (2006), Preliminary version in IPCO 1999
5. Chopra, S., Rao, M.R.: On the Multiway Cut Polyhedron. Networks **21**, 51–89 (1991)
6. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The Complexity of Multiterminal Cuts. SIAM J. Comp. **23**, 864–894 (1994). Preliminary version in STOC 1992, An extended abstract was first announced in 1983
7. Freund, A., Karloff, H.: A lower bound of $8/(7 + \frac{1}{k-1})$ on the integrality ratio of the Calinescu–Karloff–Rabani relaxation for Multiway Cut. Inf. Process. Lett. **75**, 43–50 (2000)
8. Frieze, A., Kannan, R.: The Regularity Lemma and Approximation Schemes for Dense Problems. In: Proc. 37th IEEE FOCS 1996, pp. 12–20. IEEE Computer Society Press, Los Alamitos
9. Garg, N., Vazirani, V.V., Yannakakis, M.: Multiway cuts in node weighted graphs. J. Algorithms **50**(1), 49–61 (2004). Preliminary version in ICALP 1994
10. Karger, D.R., Klein, P., Stein, C., Thorup, M., Young, N.E.: Rounding algorithms for a geometric embedding of minimum multiway cut. Math. Oper. Res. **29**(3), 436–461 (2004). Preliminary version in STOC 1999
11. Naor, J.S., Zosin, L.: A 2-Approximation Algorithm for the Directed Multiway Cut Problem. SIAM J. Comput. **31**(2), 477–492 (2001). Preliminary version in FOCS 1997
12. Vazirani, V.V.: Approximation Algorithms. Springer, Berlin Heidelberg New York (2001)