G

Gate Sizing

2002; Sundararajan, Sapatnekar, Parhi

Vijay Sundararajan
Texas Instruments, Dallas, TX, USA

Keywords and Synonyms

Fast and exact transistor sizing

Problem Definition

For a detailed exposition of the solution approach presented in this article please refer to [15]. As evidenced by the successive announcement of ever faster computer systems in the past decade, increasing the speed of VLSI systems continues to be one of the major requirements for VLSI system designers today. Faster integrated circuits are making possible newer applications that were traditionally considered difficult to implement in hardware. In this scenario of increasing circuit complexity, reduction of circuit delay in integrated circuits is an important design objective. Transistor sizing is one such task that has been employed for speeding up circuits for quite some time now [6]. Given the circuit topology, the delay of a combinational circuit can be controlled by varying the sizes of transistors in the circuit. Here, the size of a transistor is measured in terms of its channel width, since the channel lengths of MOS transistors in a digital circuit are generally uniform. In any case, what really matters is the ratio of channel width to channel length, and if channel lengths are not uniform, this ratio can be considered as the size. In coarse terms, the circuit delay can usually be reduced by increasing the sizes of certain transistors in the circuit from the minimum size. Hence, making the circuit faster usually entails the penalty of increased circuit area relative to a minimum-sized circuit and the area-delay trade-off involved here is the problem of transistor size optimization. A related problem to transistor sizing is called gate sizing, where a logic gate in a circuit is modeled as an equivalent inverter and the sizing optimization is carried out on this modified circuit with equivalent inverters in place of more complex gates. There is, therefore, a reduction in the number of size parameters corresponding to every gate in the circuit. Needless to say, this is an easier problem to solve than the general transistor sizing problem. Note that gate sizing mentioned here is distinct from library specific gate sizing that is a discrete optimization problem targeted to selecting appropriate gate sizes from an underlying cell library. The gate sizing problem targeted here is one of continuous gate sizing where the gate sizes are allowed to vary in a continuous manner between a minimum and a maximum size. There has been a large amount of work done on transistor sizing [1,2,3,5,6,9,10,12,13], that underlines the importance of this optimization technique. Starting from a minimum-sized circuit, TILOS, [6], uses a greedy strategy for transistor sizing by iteratively sizing transistors in the critical path. A sensitivity factor is calculated for every transistor in the critical path to quantify the gain in circuit speed achieved by a unit upsizing of the transistor. The most sensitive transistor is then bumped up in size by a small constant factor to speed up the circuit. This process is repeated iteratively until the timing requirements are met. The technique is extremely simple to implement and has run-time behavior proportional to the size of the circuit. Its chief drawback is that it does not have guaranteed convergence properties and hence is not an exact optimization technique.

Key Results

The solution presented in the article heretofore referred to as MINFLOTRANSIT was a novel way of solving the transistor sizing problem exactly and in an extremely fast manner. Even though the article treats transistor sizing, in the description, the results apply as well to the less general problem of continuous gate sizing as described earlier. The proposed approach has some similarity in form to [2,5,8] which will be subsequently explained, but the similarity in

Gate Sizing, Table 1
Comparison of TILOS and MINFLOTRANSIT on a Sun Ultrasparc 10 workstation for ISCAS85 and MCNC91 benchmarks for 0.13 um technology. The delay specs. are with respect to a minimum-sized circuit. The optimization approach followed here was gate sizing

Circuit	# Gates	Area Saved over TILOS	Delay Specs.	CPU TIME (TILOS)	CPU TIME (OURS)
Adder32	480	$\leq$ 1%	0.5 D_{min}	2.2 s	5 s
Adder256	3840	$\leq$ 1%	0.5 D_{min}	262 s	608 s
Cm163a	65	2.1%	0.55 D_{min}	0.13 s	0.32 s
Cm162a	71	10.4%	0.5 D_{min}	0.23 s	0.96 s
Parity8	89	37%	0.45 D_{min}	0.68 s	2.15 s
Frg1	177	1.9%	0.7 D_{min}	0.55 s	1.49 s
population	518	6.7%	0.4 D_{min}	57 s	179 s
Pmult8	1431	5%	0.5 D_{min}	637 s	1476 s
Alu2	826	2.6%	0.6 D_{min}	28 s	71 s
C432	160	9.4%	0.4 D_{min}	0.5 s	4.8 s
C499	202	7.2%	0.57 D_{min}	1.47 s	11.26 s
C880	383	4%	0.4 D_{min}	2.7 s	8,2 s
C1355	546	9.5%	0.4 D_{min}	29 s	76 s
C1908	880	4.6%	0.4 D_{min}	36 s	84 s
C2670	1193	9.1%	0.4 D_{min}	27 s	69 s
C3540	1669	7.7%	0.4 D_{min}	226 s	651 s
C5315	2307	2%	0.4 D_{min}	90 s	201 s
C6288	2416	16.5%	0.4 D_{min}	1677 s	4138 s
C7552	3512	3.3%	0.4 D_{min}	320 s	683 s

content is minimal and the details of implementation are vastly different.

In essence, the proposed technique and the techniques in [2,5,8] are iterative relaxation approaches that involve a two-step optimization strategy. The first-step involves a delay budgeting step where optimal delays are computed for transistors/gates. The second step involves sizing transistors optimally under this "constant delay" model to achieve these delay budgets. The two steps are iteratively alternated until the solution converges, i. e., until the delay budgets calculated in the first step are exactly satisfied by the transistor sizes determined by the second step.

The primary features of the proposed approach are:

- It is computationally fast and is comparable to TILOS in its run-time behavior.
- It can be used for true transistor sizing as well as the relaxed problem of gate sizing. Additionally, the approach can easily incorporate wire-sizing [15].
- It can be adapted for more general delay models than the Elmore delay model [15].

The starting point for the proposed approach is a fast guess solution. This could be obtained, for example, from a circuit that has been optimized using TILOS to meet the given delay requirements. The proposed approach, as outlined earlier, is an iterative relaxation procedure that involves an alternating two-phase relaxed optimization sequence that is repeated iteratively until convergence is achieved. The two-phases in the proposed approach are:

- The **D-phase** where transistor sizes are assumed fixed and transistor delays are regarded as variable parameters. Irrespective of the delay model employed, this phase can be formulated as the dual of a min-cost network flow problem. Using $|V|$ to denote the number of transistors and $|E|$ the number of wires in the circuit, this step in our application has worst-case complexity of $O(|V|\,|E|\log(\log|V|))$ [7].
- The **W-phase** where transistor/gate delays are assumed fixed and their sizes are regarded as variable parameters. As long as the gate delay can be expressed as a separable function of the transistor sizes, this step can be solved as a Simple Monotonic Program (SMP) [11]. The complexity of SMP is similar to an all-pairs shortest path algorithm in a directed graph, [4,11], i. e., $O(|V|\,|E|)$.

The objective function for the problem is the minimization of circuit area. In the W-phase, this objective is addressed directly, and in the D-phase the objective is chosen to facilitate a move in the solution space in a direction that is known to lead to a reduction in the circuit area.

Applications

The primary application of the solution provided here is circuit and system optimization in automated VLSI design. The solution provided here can enable Electronic Design Automation (EDA) tools that take a holistic approach towards transistor sizing. This will in turn enable making custom circuit design flows more realizable in practice. The mechanics of some of the elements of the solution provided here especially the **D-phase** have been used to address other circuit optimization problems [14].

Open Problems

The related problem of Discrete gate sizing optimization matching gate sized to available gate sizes from a standard cell library is a provably hard optimization problem which could be aided by the development of efficient heuristics and probabilistic algorithms.

Experimental Results

A relative comparison of MINFLOTRANSIT with TILOS is provided in Table 1 for gate sizing of ISACS85 and mcnc91 benchmark circuits. As can be seen a significant performance improvement is observed with a tolerable loss in execution time.

Cross References

- ► Circuit Retiming
- ► Wire Sizing

Recommended Reading

1. Chen, C.P., Chu, C.N, Wong, D.F.: Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relax-ation. In: Proceedings of the 1998 IEEE/ACM International Conference on Computer-Aided Design, pp. 617–624. November 1998
2. Chen, H.Y., Kang, S.M.: icoach: A circuit optimiza-tion aid for cmos high-performance circuits. Intergr. VLSI. J. **10**(2), 185–212 (1991)
3. Conn, A.R., Coulman, P.K., Haring, R.A., Morrill, G.L., Visweshwariah, C., Wu, C.W.: JiffyTune: Circuit Optimization Using Time-Domain Sensitivities. IEEE Trans. Comput. Aided. Des. Integr. Circuits. Syst.**17**(12), 1292–1309 (1998)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to algorithms. McGraw-Hill, New York (1990)
5. Dai, Z., Asada, K.: MOSIZ: A Two-Step Transistor Sizing Algorithm based on Optimal Timing Assignment Method for Multi-Stage Complex Gates. In: Proceedings of the 1989 Custom Integrated Circuits Conference, pp. 17.3.1–17.3.4. May 1989
6. Fishburn, J.P., Dunlop, A. E.: TILOS: A Posynomial Programming Approach to Transistor Sizing. In: Proceedings of the 1985 International Conference on Computer-Aided Design, pp. 326–328. Santa Clara, CA, November 1985
7. Goldberg, A.V., Grigoriadis, M.D., Tarjan, R.E.: Use of Dynamic Trees in a Network Simplex Algorithm for the Maximum Flow Problem. Math. Program. **50**(3), 277–290 (1991)
8. Grodstein, J., Lehman, E., Harkness, H., Grundmann, B., Watanabe, Y.: A delay model for logic synthesis of continuously sized networks. In: Proceedings of the 1995 International Conference on Computer-Aided Design, pp. 458–462. November 1995
9. Marple, D.P.: Performance Optimization of Digital VLSI Circuits. Technical Report CSL-TR-86-308, Stanford University, October 1986
10. Marple, D.P.: Transistor Size Optimization in the Tailor Layout System. In: Proceedings of the 26th ACM/IEEE Design Automation Conference, pp. 43–48. June 1989
11. Papaefthymiou, M.C.: Asymptotically Efficient Retiming under Setup and Hold Constraints. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, pp. 288–295, November 1998
12. Sapatnekar, S.S., Rao, V.B., Vaidya, P.M., Kang, S.M.: An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization. IEEE Trans. Comput. Aided. Des. **12**(11), 1621–1634 (1993)
13. Shyu, J.M., Sangiovanni-Vincentelli, A.L., Fishburn, J.P., Dunlop, A.E.: Optimization-based Transistor Sizing. IEEE J. Solid. State. Circuits. **23**(2), 400–409 (1988)
14. Sundararajan, V., Parhi, K.: Low Power Synthesis of Dual Threshold Voltage CMOS VLSI Circuits. In: Proceedings of the International Symposium on Low Power Electronics and Design. pp. 139-144 (1999)
15. Sundararajan, V., Sapatnekar, S.S., Parhi, K.K.: Fast and exact transistor sizing based on iterative relaxation. Computer-Aided Design of Integrated Circuits and Systems, IEEE Trans. **21**(5),568–581 (2002)

General Equilibrium

2002; Deng, Papadimitriou, Safra

Li-Sha Huang
Department of Computer Science and Technology, Tsinghua University, Beijing, Beijing, China

Keywords and Synonyms

Competitive market equilibrium

Problem Definition

This problem is concerned with the computational complexity of finding an exchange market equilibrium. The exchange market model consists of a set of agents, each with an initial endowment of commodities, interacting through a market, trying to maximize each's utility function. The equilibrium prices are determined by a clearance condition. That is, all commodities are bought, col-

lectively, by all the utility maximizing agents, subject to their budget constraints (determined by the values of their initial endowments of commodities at the market price). The work of Deng, Papadimitriou and Safra [3] studies the complexity, approximability, inapproximability, and communication complexity of finding equilibrium prices. The work shows the NP-hardness of approximating the equilibrium in a market with indivisible goods. For markets with divisible goods and linear utility functions, it develops a pseudo-polynomial time algorithm for computing an ϵ-equilibrium. It also gives a communication complexity lower bound for computing Pareto allocations in markets with non-strictly concave utility functions.

Market Model

In a pure exchange economy, there are m traders, labeled by $i = 1, 2, ..., m$, and n types of commodities, labeled by $j = 1, 2, ..., n$. The commodities could be divisible or indivisible. Each trader i comes to the market with initial endowment of commodities, denoted by a vector $w_i \in \mathbb{R}_+^n$, whose j-th entry is the amount of commodity j held by trader i.

Associate each trader i a *consumption set* X_i to represents the set of possible commodity bundles for him. For example, when there are n_1 divisible commodities and $(n - n_1)$ indivisible commodities, X_i can be $\mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{n-n_1}$. Each trader has a utility function $X_i \mapsto \mathbb{R}_+$ to present his utility for a bundle of commodities. Usually, the utility function is required to be concave and nondecreasing.

In the market, each trader acts as both a buyer and a seller to maximize his utility. At a certain price $p \in \mathbb{R}_+^n$, trader i is is solving the following optimization problem, under his budget constraint:

$$\max \quad u_i(x_i) \quad s.t. \quad x_i \in X_i \text{ and } \langle p, x_i \rangle \le \langle p, w_i \rangle.$$

Definition 1 An equilibrium in a pure exchange economy is a price vector $\tilde{p} \in \mathbb{R}_+^n$ and bundles of commodities $\{\tilde{x}_i \in \mathbb{R}_+^n, i = 1, ..., m\}$, such that

$$\tilde{x}_i \in \text{argmax}\{u_i(x_i) | x_i \in X_i \text{ and } \langle x_i, \tilde{p} \rangle \le \langle w_i, \tilde{p} \rangle\}, \quad \forall 1 \le i \le m$$

$$\sum_{i=1}^{m} \tilde{x}_{ij} \le \sum_{i=1}^{m} w_{ij}, \forall 1 \le j \le n.$$

The concept of approximate equilibrium was introduced in [3]:

Definition 2 ([3]) An ϵ-approximate equilibrium in an exchange market is a price vector $\tilde{p} \in \mathbb{R}_+^n$ and bundles of goods $\{\tilde{x}_i \in \mathbb{R}_+^n, i = 1, ..., m\}$, such that

$$u_i(\tilde{x}_i) \ge \frac{1}{1+\epsilon} \max\{u_i(x_i) | x_i \in X_i, \langle x_i, \tilde{p} \rangle \le \langle w_i, \tilde{p} \rangle\}, \forall i \tag{1}$$

$$\langle \tilde{x}_i, \tilde{p} \rangle \le (1+\epsilon) \langle w_i, \tilde{p} \rangle, \forall i \tag{2}$$

$$\sum_{i=1}^{m} \tilde{x}_{ij} \le (1+\epsilon) \sum_{i=1}^{m} w_{ij}, \forall j\,. \tag{3}$$

Key Results

A linear market is a market in which all the agents have linear utility functions. The deficiency of a market is the smallest $\epsilon \ge 0$ for which an ϵ-approximate equilibrium exists.

Theorem 1 *The deficiency of a linear market with indivisible goods is NP-hard to compute, even if the number of agents is two. The deficiency is also NP-hard to approximate within 1/3.*

Theorem 2 *There is a polynomial-time algorithm for finding an equilibrium in linear markets with bounded number of divisible goods. Ditto for a polynomial number of agents.*

Theorem 3 *If the number of goods is bounded, there is a polynomial-time algorithm which, for any linear indivisible market for which a price equilibrium exists, and for any $\epsilon > 0$, finds an ϵ-approximate equilibrium.*

If the utility functions are strictly concave and the equilibrium prices are broadcasted to all agents, the equilibrium allocation can be computed distributely without any communication, since each agent's basket of goods is uniquely determined. However, if the utility functions are not strictly concave, e. g. linear functions, communications are needed to coordinate the agents' behaviors.

Theorem 4 *Any protocol with binary domains for computing Pareto allocations of m agents and n divisible commodities with concave utility functions (resp. ϵ-Pareto allocations for indivisible commodities, for any $\epsilon < 1$) must have market communication complexity $\Omega(m \log(m+n))$ bits.*

Applications

This concept of market equilibrium is the outcome of a sequence of efforts trying to fully understand the laws that govern human commercial activities, starting with the "invisible hand" of Adam Smith, and finally, the mathematical conclusion of Arrow and Debreu [1] that there exists a set of prices that bring supply and demand into equilibrium, under quite general conditions on the agent utility functions and their optimization behavior.

The work of Deng, Papadimitriou and Safra [3] explicitly called for an algorithmic complexity study of the problem, and developed interesting complexity results and approximation algorithms for several classes of utility functions. There has since been a surge of algorithmic study for the computation of the price equilibrium problem with continuous variables, discovering and rediscovering polynomial time algorithms for many classes of utility functions, see [2,4,5,6,7,8,9].

Significant progress has been made in the above directions but only as a first step. New ideas and methods have already been invented and applied in reality. The next significant step will soon manifest itself with many active studies in microeconomic behavior analysis for E-commercial markets. Nevertheless the algorithmic analytic foundation in [3] will be an indispensable tool for further development in this reincarnated exciting field.

Open Problems

The most important open problem is what is the computational complexity for finding the equilibrium price, as guaranteed by the Arrow–Debreu theorem. To the best of the author's knowledge, only the markets whose set of equilibria is convex can be solved in polynomial time with current techniques. And approximating equilibria in some markets with disconnected set of equilibria, e. g. Leontief economies, are shown to be PPAD-hard. Is the convexity or (weakly) gross substitutability a necessary condition for a market to be polynomial-time solvable?

Second, how to handle the dynamic case is especially interesting in theory, mathematical modeling, and algorithmic complexity as bounded rationality. Great progress must be made in those directions for any theoretical work to be meaningful in practice.

Third, incentive compatible mechanism design protocols for the auction models have been most actively studied recently, especially with the rise of E-Commerce. Especially at this level, a proper approximate version of the equilibrium concept handling price dynamics should be especially important.

Cross References

- ▶ Complexity of Core
- ▶ Leontief Economy Equilibrium
- ▶ Non-approximability of Bimatrix Nash Equilibria

Recommended Reading

1. Arrow, K.J., Debreu, G.: Existence of an equilibrium for a competitive economy. Econometrica **22**(3), 265–290 (1954)
2. Codenotti, B., McCune, B., Varadarajan, K.: Market equilibrium via the excess demand function. In: Proceedings STOC'05, pp. 74–83. ACM, Baltimore (2005)
3. Deng, X., Papadimitriou, C., Safra, S.: On the complexity of price equilibria. J. Comput. Syst. Sci. **67**(2), 311–324 (2002)
4. Devanur, N.R., Papadimitriou, C.H., Saberi, A., Vazirani, V.V.: Market equilibria via a primal-dual-type algorithm. In: Proceedings of FOCS'02, pp. 389–395. IEEE Computer Society, Vancouver (2002)
5. Eaves, B.C.: Finite solution for pure trade markets with Cobb-Douglas utilities, Math. Program. Study **23**, 226–239 (1985)
6. Garg, R., Kapoor, S.: Auction algorithms for market equilibrium, In: Proceedings of STOC'04, pp. 511–518. ACM, Chicago (2004)
7. Jain, K.: A polynomial time algorithm for computing the Arrow-Debreu market equilibrium for linear utilities. In: Proceeding of FOCS'04, pp. 286–294. IEEE Computer Society, Rome (2004)
8. Nenakhov, E., Primak, M.: About one algorithm for finding the solution of the Arrow-Debreu Model. Kibernetica **3**, 127–128 (1983)
9. Ye, Y.: A path to the Arrow-Debreu competitive market equilibrium, Math. Program. **111**(1–2), 315–348 (2008)

Generalized Steiner Network

2001; Jain

Julia Chuzhoy
Toyota Technological Institute, Chicago, IL, USA

Keywords and Synonyms

Survivable network design

Problem Definition

The generalized Steiner network problem is a network design problem, where the input consists of a graph together with a collection of connectivity requirements, and the goal is to find the cheapest subgraph meeting these requirements.

Formally, the input to the generalized Steiner network problem is an undirected multigraph $G = (V, E)$, where each edge $e \in E$ has a non-negative cost $c(e)$, and for each pair of vertices $i, j \in V$, there is a connectivity requirement $r_{i,j} \in \mathbb{Z}$. A feasible solution is a subset $E' \subseteq E$ of edges, such that every pair $i, j \in V$ of vertices is connected by at least $r_{i,j}$ edge-disjoint path in graph $G' = (V, E')$.

The generalized Steiner network problem asks to find a solution E' of minimum cost $\sum_{e \in E'} c(e)$.

This problem generalizes several classical network design problems. Some examples include minimum spanning tree, Steiner tree and Steiner forest. The most general special case for which a 2-approximation was previously known is the Steiner forest problem [1,4].

Williamson et al. [8] were the first to show a nontrivial approximation algorithm for the generalized Steiner network problem, achieving a $2k$-approximation, where $k = \max_{i,j \in V}\{r_{i,j}\}$. This result was improved to $O(\log k)$-approximation by Goemans et al. [3].

Key Results

The main result of [6] is a factor-2 approximation algorithm for the generalized Steiner network problem. The techniques used in the design and the analysis of the algorithm seem to be of independent interest.

The 2-approximation is achieved for a more general problem, defined as follows. The input is a multigraph $G = (V, E)$ with costs $c(\cdot)$ on edges, and connectivity requirement function $f : 2^V \to \mathbb{Z}$. Function f is weakly submodular, i. e., it has the following properties:

1. $f(V) = 0$.
2. For all $A, B \subseteq V$, at least one of the following two conditions holds:
   - $f(A) + f(B) \leq f(A \setminus B) + f(B \setminus A)$.
   - $f(A) + f(B) \leq f(A \cap B) + f(A \cup B)$.

For any subset $S \subseteq V$ of vertices, let $\delta(S)$ denote the set of edges with exactly one endpoint in S. The goal is to find a minimum-cost subset of edges $E' \subseteq E$, such that for every subset $S \subseteq V$ of vertices, $|\delta(S) \cap E'| \geq f(S)$.

This problem can be equivalently expressed as an integer program. For each edge $e \in E$, let x_e be the indicator variable of whether e belongs to the solution.

$$\text{(IP)} \qquad \min \sum_{e \in E} c(e) x_e$$

subject to:

$$\sum_{e \in \delta(S)} x_e \geq f(S) \qquad \forall S \subseteq V \qquad (1)$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \qquad (2)$$

It is easy to see that the generalized Steiner network problem is a special case of (IP), where for each $S \subseteq V$, $f(S) = \max_{i \in S, j \notin S}\{r_{i,j}\}$.

Techniques

The approximation algorithm uses the LP-rounding technique. The initial linear program (LP) is obtained from (IP) by replacing the integrality constraint (2) with:

$$0 \leq x_e \leq 1 \qquad \forall e \in E \qquad (3)$$

It is assumed that there is a separation oracle for (LP). It is easy to see that such an oracle exists if (LP) is obtained from the generalized Steiner network problem. The key result used in the design and the analysis of the algorithm is summarized in the following theorem.

Theorem 1 *In any basic solution of (LP), there is at least one edge $e \in E$ with $x_e \geq 1/2$.*

The approximation algorithm works by iterative LP-rounding. Given a basic optimal solution of (LP), let $E^* \subseteq E$ be the subset of edges e with $x_e \geq 1/2$. The edges of E^* are removed from the graph (and are eventually added to the solution), and the problem is then solved recursively on the residual graph, by solving (LP) on $G^* = (V, E \setminus E^*)$, where for each subset $S \subseteq V$, the new requirement is $f(S) - |\delta(S) \cap E^*|$. The main observation that leads to factor-2 approximation is the following: if E' is a 2-approximation for the residual problem, then $E' \cup E^*$ is a 2-approximation for the original problem.

Given any solution to (LP), set $S \subseteq V$ is called *tight* iff constraint (1) holds with equality for S. The proof of Theorem 1 involves constructing a large *laminar family* of tight sets (a family where for every pair of sets, either one set contains the other, or the two sets are disjoint). After that a clever accounting scheme that charges edges to the sets of the laminar family is used to show that there is at least one edge $e \in E$ with $x_e \geq 1/2$.

Applications

Generalized Steiner network is a very basic and natural network design problem that has many applications in different areas, including the design of communication networks, VLSI design and vehicle routing. One example is the design of survivable communication networks, which remain functional even after the failure of some network components (see [5] for more details).

Open Problems

The 2-approximation algorithm of Jain [6] for generalized Steiner network is based on LP-rounding, and it has high running time. It would be interesting to design a combinatorial approximation algorithm for this problem.

It is not known whether a better approximation is possible for generalized Steiner network. Very few hardness of approximation results are known for this type of problems. The best current hardness factor stands on 1.01063 [2],

and this result is valid even for the special case of Steiner tree.

Cross References

- ▶ Steiner Forest
- ▶ Steiner Trees

Recommended Reading

1. Agrawal A., Klein P., Ravi R.: When Trees Collide: An Approximation Algorithm for the Generalized Steiner Problem on Networks. J. SIAM Comput. **24**(3), 440–456 (1995)
2. Chlebik M., Chlebikova J.: Approximation Hardness of the Steiner Tree Problem on Graphs. In: 8th Scandinavian Workshop on Algorithm Theory. Number 2368 in LNCS, pp. 170–179, (2002)
3. Goemans M.X., Goldberg A.V., Plotkin S.A., Shmoys D.B., Tardos É., Williamson D.P.: Improved Approximation Algorithms for Network Design Problems. In: Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 223–232. (1994)
4. Goemans M.X., Williamson D.P.: A General Approximation Technique for Constrained Forest Problems. SIAM J. Comput. **24**(2), 296–317 (1995)
5. Grötschel M., Monma C.L., Stoer M.: Design of Survivable Networks. In: Network Models, Handbooks in Operations Research and Management Science. North Holland Press, Amsterdam, (1995)
6. Jain K.: A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem. Combinatorica **21**(1), 39–60 (2001)
7. Vazirani V.V.: Approximation Algorithms. Springer, Berlin (2001)
8. Williamson D.P., Goemans M.X., Mihail M., Vazirani V.V.: A Primal-Dual Approximation Algorithm for Generalized Steiner Network Problems. Combinatorica **15**(3), 435–454 (1995)

Generalized Two-Server Problem

2006; Sitters, Stougie

René A. Sitters
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

Keywords and Synonyms

CNN-problem

Problem Definition

In the *generalized two-server problem* we are given two servers: one moving in a metric space $\mathbb{X}$ and one moving in a metric space $\mathbb{Y}$. They are to serve requests $r \in \mathbb{X} \times \mathbb{Y}$ which arrive one by one. A request $r = (x, y)$ is served by moving either the $\mathbb{X}$-server to point x or the $\mathbb{Y}$-server to point y. The decision as to which server to move to the next request is irrevocable and has to be taken without any knowledge about future requests. The objective is to minimize the total distance traveled by the two servers.

On-line Routing Problems

The generalized two-server problem belongs to a class of routing problems called *metrical service systems* [5,10]. Such a system is defined by a metric space $\mathbb{M}$ of all possible system configurations, an initial configuration C_0, and a set $\mathcal{R}$ of possible requests, where each request $r \in \mathcal{R}$ is a subset of $\mathbb{M}$. Given a sequence, $r_1, r_2 \ldots, r_n$, of requests, a feasible solution is a sequence, $C_1, C_2, \ldots, C_n$, of configurations such that $C_i \in r_i$ for all $i \in \{1, \ldots, n\}$.

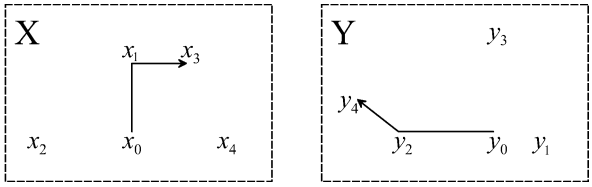
When we model the generalized two-server problem as a metrical service system we have $\mathbb{M} = \mathbb{X} \times \mathbb{Y}$ and $\mathcal{R} = \{\{x \times \mathbb{Y}\} \cup \{\mathbb{X} \times y\} | x \in \mathbb{X}, y \in \mathbb{Y}\}$. In the classical *two-server problem* both servers move in the same space and receive the same requests, i. e., $\mathbb{M} = \mathbb{X} \times \mathbb{X}$ and $\mathcal{R} = \{\{x \times \mathbb{X}\} \cup \{\mathbb{X} \times x\} | x \in \mathbb{X}\}$.

The performance of algorithms for on-line optimization problems is often measured using *competitive analysis*. We say that an algorithm is α*-competitive* ($\alpha \geq 1$) for some minimization problem if for every possible instance the cost of the algorithm's solution is at most α times the cost of an optimal solution for the instance.

A standard algorithm that performs provably well for several elementary routing problems is the so-called *work function algorithm* [2,6,8]; after each request the algorithm moves to a configuration with low cost and which is not too far from the current configuration. More precisely: If the system's configuration after serving a sequence σ is C and $r \subseteq \mathbb{M}$ is the next request, then the work function algorithm with parameter $\lambda \geq 1$ moves to a configuration $C' \in r$ that minimizes

$$\lambda W_{\sigma,r}(C') + d(C, C') ,$$

where $d(C, C')$ is the distance between configurations C and C', and $W_{\sigma,r}(C')$ is the cost of an optimal solution that



Generalized Two-Server Problem, Figure 1
In this example both servers move in the plane and start from the configuration (x_0, y_0). The $\mathbb{X}$-server moves through requests 1 and 3, and the $\mathbb{Y}$-server takes care of requests 2 and 4. The cost of this solution is the sum of the path-lengths

serves all requests (in order) in σ plus request r with the restriction that it ends in configuration C'.

Key Results

The main result in [11] is a sufficient condition for a metrical service system to have a constant-competitive algorithm. Additionally, the authors show that this condition holds for the generalized two-server problem.

For a fixed metrical service system S with metric space $\mathbb{M}$, denote by $A(C, \sigma)$ the cost of algorithm A on input sequence σ, starting in configuration C. Let $\textsc{Opt}(C, \sigma)$ be the cost of the corresponding optimal solution. We say that a path T in $\mathbb{M}$ *serves* a sequence σ if it visits all requests in order. Hence, a feasible path is a path that serves the sequence and starts in the initial configuration.

Paths T_1 and T_2 are said to be *independent* if they are far apart in the following way: $|T_1| + |T_2| < d(C_1^s, C_2^t) + d(C_2^s, C_1^t)$, where C_i^s and C_i^t are, respectively, the start and end point of path $T_i (i \in \{1, 2\})$. Notice, for example, that two intersecting paths are not independent.

Theorem 1 *Let S be a metrical service system with metric space $\mathbb{M}$. Suppose there exists an algorithm A and constants $\alpha \geq 1, \beta \geq 0$ and $m \geq 2$ such that for any point $C \in \mathbb{M}$, sequence σ and pairwise independent paths $T_1, T_2, \ldots, T_m$ that serve σ*

$$A(C, \sigma) \leq \alpha \textsc{Opt}(C, \sigma) + \beta \sum_{i=1}^{m} |T_i| \,. \tag{1}$$

Then there exists an algorithm B that is constant competitive for S.

The proof in [11] of the theorem above provides an explicit formulation of B. This algorithm combines algorithm A with the work function algorithm and operates in phases. In each phase, it applies algorithm A until its cost becomes too large compared to the optimal cost. Then, it makes one step of the work function algorithm and a new phase starts. In each phase algorithm A makes a restart, i. e., it takes the final configuration of the previous phase as the initial configuration, whereas the work function algorithm remembers the whole request sequence.

For the generalized two-server problem the so-called *balance algorithm* satisfies condition (1). This algorithm stores the cumulative costs of the two servers and with each request it moves the server that minimizes the maximum of the two new values. The balance algorithm itself is not constant competitive but Theorem 1 says that, if we combine it in a clever way with the work function algorithm, then we get an algorithm that is constant competitive.

Applications

A set of metrical service systems can be combined to get what is called in [9] the *sum system*. A request of the sum system consists of one request for each system and to serve it we need to serve at least one of the individual requests. The generalized two-server problem should be considered as one of the simplest sum systems since the two individual problems are completely trivial: there is one server and each request consists of a single point.

Sum systems are particularly interesting to model systems for information storage and retrieval. To increase stability or efficiency one may store copies of the same information in multiple systems (e. g. databases, hard disks). To retrieve one piece of information we may read it from any system. However, to read information it may be necessary to change the configuration of the system. For example, if the database is stored in a binary search tree, then it is efficient to make on-line changes to the structure of the tree, i. e., to use dynamic search trees [12].

Open Problems

A proof that the work function algorithm is competitive for the generalized two-server problem (as conjectured in [9] and [11]) is still lacking. Also, a randomized algorithm with a smaller competitive ratio than that of [11] is not known. No results (except for a lower bound) are known for the generalized problem with more than two servers. It is not even clear if the work function algorithm may be competitive here.

There are systems for which the work function algorithm is not competitive. It would be interesting to have a non-trivial property that implies competitiveness of the work function algorithm.

Cross References

- ▶ Algorithm DC-Tree for k Servers on Trees
- ▶ Metrical Task Systems
- ▶ Online Paging and Caching
- ▶ Work-Function Algorithm for k Servers

Recommended Reading

1. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press, Cambridge (1998)
2. Burley, W.R.: Traversing layered graphs using the work function algorithm. J. Algorithms **20**, 479–511 (1996)
3. Chrobak, M.: Sigact news online algorithms column 1. ACM SIGACT News **34**, 68–77 (2003)
4. Chrobak, M., Karloff, H., Payne, T.H., Vishwanathan, S.: New results on server problems. SIAM J. Discret. Math. **4**, 172–181 (1991)

5. Chrobak, M., Larmore, L.L.: Metrical service systems: Deterministic strategies. Tech. Rep. UCR-CS-93-1, Department of Computer Science, Univ. of California at Riverside (1992)
6. Chrobak, M., Sgall, J.: The weighted 2-server problem. Theor. Comput. Sci. **324**, 289–312 (2004)
7. Fiat, A., Ricklin, M.: Competitive algorithms for the weighted server problem. Theor. Comput. Sci. **130**, 85–99 (1994)
8. Koutsoupias, E., Papadimitriou, C.H.: On the *k*-server conjecture. J. ACM **42**, 971–983 (1995)
9. Koutsoupias, E., Taylor, D.S.: The CNN problem and other k-server variants. Theor. Comput. Sci. **324**, 347–359 (2004)
10. Manasse, M.S., McGeoch, L.A., Sleator, D.D.: Competitive algorithms for server problems. J. Algorithms **11**, 208–230 (1990)
11. Sitters, R.A., Stougie, L.: The generalized two-server problem. J. ACM **53**, 437–458 (2006)
12. Sleator, D.D., Tarjan, R.E.: Self-adjusting binary search trees. J. ACM **32**, 652–686 (1985)

Generalized Vickrey Auction

1995; Varian

MAKOTO YOKOO
Department of Information Science and Electrical Engineering, Kyushu University, Nishi-ku, Japan

Keywords and Synonyms

Generalized Vickrey auction; GVA;
Vickrey–Clarke–Groves mechanism; VCG

Problem Definition

Auctions are used for allocating goods, tasks, resources, etc. Participants in an auction include an auctioneer (usually a seller) and bidders (usually buyers). An auction has well-defined rules that enforce an agreement between the auctioneer and the winning bidder. Auctions are often used when a seller has difficulty in estimating the value of an auctioned good for buyers.

The Generalized Vickrey Auction protocol (GVA) [5] is an auction protocol that can be used for combinatorial auctions [3] in which multiple items/goods are sold simultaneously. Although conventional auctions sell a single item at a time, combinatorial auctions sell multiple items/goods. These goods may have interdependent values, e. g., these goods are complementary/substitutable and bidders can bid on any combination of goods. In a combinatorial auction, a bidder can express complementary/substitutable preferences over multiple bids. By taking into account complementary/substitutable preferences, the participants' utilities and the revenue of the seller can be increased. The GVA is one instance of the Clarke mechanism [2,4]. It is also called the Vickrey–Clarke–Groves mechanism (VCG). As its name suggests, it is a generalized version of the well-known Vickrey (or second-price) auction protocol [6], proposed by an American economist W. Vickrey, a 1996 Nobel Prize winner.

Assume there is a set of bidders $N = \{1, 2, \ldots, n\}$ and a set of goods $M = \{1, 2, \ldots, m\}$. Each bidder i has his/her preferences over a bundle, i. e., a subset of goods $B \subseteq M$. Formally, this can be modeled by supposing that bidder i privately observes a parameter, or signal, θ_i, which determines his/her preferences. The parameter θ_i is called the *type* of bidder i. A bidder is assumed to have a *quasilinear*, *private value* defined as follows.

Definition 1 (Utility of a Bidder) The utility of bidder i, when i obtains $B \subseteq M$ and pays p_i, is represented as $v(B, \theta_i) - p_i$.

Here, the valuation of a bidder is determined independently of other bidders' valuations. Also, the utility of a bidder is linear in terms of the payment. Thus, this model is called a quasilinear, private value model.

Definition 2 (Incentive Compatibility) An auction protocol is (dominant-strategy) *incentive compatible* (or *strategy-proof*) if declaring the true type/evaluation values is a dominant strategy for each bidder, i. e., an optimal strategy regardless of the actions of other bidders.

A combination of dominant strategies of all bidders is called a *dominant-strategy equilibrium*.

Definition 3 (Individual Rationality) An auction protocol is *individually rational* if no participant suffers any loss in a dominant-strategy equilibrium, i. e., the payment never exceeds the evaluation value of the obtained goods.

Definition 4 (Pareto Efficiency) An auction protocol is *Pareto efficient* when the sum of all participants' utilities (including that of the auctioneer), i. e., the social surplus, is maximized in a dominant-strategy equilibrium.

The goal is to design an auction protocol that is incentive compatible, individually rational, and Pareto efficient. It is clear that individual rationality and Pareto efficiency are desirable. Regarding the incentive compatibility, the *revelation principle* states that in the design of an auction protocol, it is possible to restrict attention only to incentive compatible protocols without loss of generality [4]. In other words, if a certain property (e. g., Pareto efficiency) can be achieved using some auction protocol in a dominant-strategy equilibrium, then the property can also be achieved using an incentive-compatible auction protocol.

Key Results

A *feasible* allocation is defined as a vector of n bundles $\vec{B} = \langle B_1, \ldots, B_n \rangle$, where $\bigcup_{j \in N} B_j \subseteq M$ and for all $j \neq j', B_j \cap B_{j'} = \emptyset$ hold.

The GVA protocol can be described as follows.

1. Each bidder i declares his/her type $\hat{\theta}_i$, which can be different from his/her true type θ_i.
2. The auctioneer chooses an optimal allocation $\vec{B}^*$ according to the declared types. More precisely, the auctioneer chooses $\vec{B}^*$ defined as follows:

$$\vec{B}^* = \arg\max_{\vec{B}} \sum_{j \in N} v\left(B_j, \hat{\theta}_j\right).$$

3. Each bidder i pays p_i, which is defined as follows ($B_j^{\sim i}$ and B_j^* are the jth element of $\vec{B}^{\sim i}$ and $\vec{B}^*$, respectively):

$$p_i = \sum_{j \in N \setminus \{i\}} v\left(B_j^{\sim i}, \hat{\theta}_j\right) - \sum_{j \in N \setminus \{i\}} v\left(B_j^*, \hat{\theta}_j\right), \quad \text{where } \vec{B}^{\sim i} = \arg\max_{\vec{B}} \sum_{j \in N \setminus \{i\}} v\left(B_j, \hat{\theta}_j\right). \tag{1}$$

The first term in Eq. (1) is the social surplus when bidder i does not participate. The second term is the social surplus except bidder i when i does participate. In the GVA, the payment of bidder i can be considered as the decreased amount of the other bidders' social surplus resulting from his/her participation.

A description of how this protocol works is given below.

Example 1 Assume there are two goods a and b, and three bidders, 1, 2, and 3, whose types are θ_1, θ_2, and θ_3, respectively. The evaluation value for a bundle $v(B, \theta_i)$ is determined as follows.

	$\{a\}$	$\{b\}$	$\{a, b\}$
θ_1	\$6	\$0	\$6
θ_2	\$0	\$0	\$8
θ_3	\$0	\$5	\$5

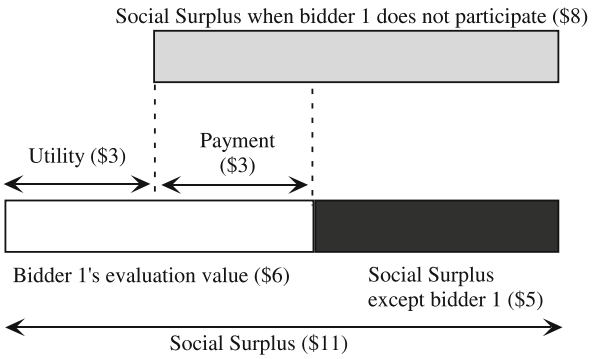
Here, bidder 1 wants good a only, and bidder 3 wants good b only. Bidder 2's utility is all-or-nothing, i. e., he/she wants both goods at the same time and having only one good is useless.

Assume each bidder i declares his/her true type θ_i. The optimal allocation is to allocate good a to bidder 1 and b to bidder 3, i. e., $\vec{B}^* = \langle \{a\}, \{\}, \{b\} \rangle$. The payment of bidder 1 is calculated as follows. If bidder 1 does not participate, the optimal allocation would have been allocating both items to bidder 2, i. e., $\vec{B}^{\sim 1} = \langle \{\}, \{a, b\}, \{\} \rangle$ and the social surplus, i. e., $\sum_{j \in N \setminus \{1\}} v\left(B_j^{\sim 1}, \hat{\theta}_j\right)$ is equal to \$8. When bidder 1 does participate, bidder 3 obtains $\{b\}$, and the social surplus except for bidder 1, i. e., $\sum_{j \in N \setminus \{1\}} v\left(B_j^*, \hat{\theta}_j\right)$, is 5. Therefore, bidder 1 pays the difference \$8 − \$5 = \$3. The obtained utility of bidder 1 is \$6 − \$3 = \$3. The payment of bidder 3 is calculated as \$8 − \$6 = \$2.

The intuitive explanation of why truth telling is the dominant strategy in the GVA is as follows. In the GVA, goods are allocated so that the social surplus is maximized. In general, the utility of society as a whole does not necessarily mean maximizing the utility of each participant. Therefore, each participant might have an incentive for lying if the group decision is made so that the social surplus is maximized.

However, the payment of each bidder in the GVA is cleverly determined so that the utility of each bidder is maximized when the social surplus is maximized. Figure 1 illustrates the relationship between the payment and utility of bidder 1 in Example 1. The payment of bidder 1 is defined as the difference between the social surplus when bidder 1 does not participate (i. e., the length of the upper shaded bar) and the social surplus except bidder 1 when bidder 1 does participate (the length of the lower black bar), i. e., \$8 − \$5 = \$3.

On the other hand, the utility of bidder 1 is the difference between the evaluation value of the obtained item and the payment, which equals \$6 − \$3 = \$3. This amount is equal to the difference between the total length of the lower bar and the upper bar. Since the length of the upper bar is determined independently of bidder 1's declaration, bidder 1 can maximize his/her utility by maximizing the length of the lower bar. However, the length of the lower bar represents the social surplus. Thus, bidder 1 can maximize his/her utility when the social surplus is maximized.



Generalized Vickrey Auction, Figure 1
Utilities and Payments in the GVA

Therefore, bidder 1 does not have an incentive for lying since the group decision is made so that the social surplus is maximized.

Theorem 1 *The GVA is incentive compatible.*

Proof Since the utility of bidder i is assumed to be quasi-linear, it can be represented as

$$\begin{aligned} v(B_i,\theta_i) - p_i &= v(B_i,\theta_i) \\ &\quad - \left[\sum_{j\in N\setminus\{i\}} v\left(B_j^{\sim i},\hat{\theta}_j\right) - \sum_{j\in N\setminus\{i\}} v\left(B_j^*,\hat{\theta}_j\right) \right] \\ &= \left[v(B_i,\theta_i) + \sum_{j\in N\setminus\{i\}} v\left(B_j^*,\hat{\theta}_j\right) \right] \\ &\quad - \sum_{j\in N\setminus\{i\}} v\left(B_j^{\sim i},\hat{\theta}_j\right) \end{aligned} \tag{2}$$

The second term in Eq. (2) is determined independently of bidder i's declaration. Thus, bidder 1 can maximize his/her utility by maximizing the first term. However, $\vec{B}^*$ is chosen so that $\sum_{j\in N} v\left(B_j,\hat{\theta}_j\right)$ is maximized. Therefore, bidder i can maximize his/her utility by declaring $\hat{\theta}_i = \theta_i$, i. e., by declaring his/her true type. □

Theorem 2 *The GVA is individually rational.*

Proof This is clear from Eq. (2), since the first term is always larger than (or at least equal to) the second term. □

Theorem 3 *The GVA is Pareto efficient.*

Proof From Theorem 1, truth telling is a dominant-strategy equilibrium. From the way of choosing the allocation, the social surplus is maximized if all bidders declare their true types. □

Applications

The GVA can be applied to combinatorial auctions, which have lately attracted considerable attention [3]. The US Federal Communications Commission has been conducting auctions for allocating spectrum rights. Clearly, there exist interdependencies among the values of spectrum rights. For example, a bidder may desire licenses for adjoining regions simultaneously, i. e., these licenses are complementary. Thus, the spectrum auctions is a promising application field of combinatorial auctions and have been a major driving force for activating the research on combinatorial auctions.

Open Problems

Although the GVA has these good characteristics (Pareto efficiency, incentive compatibility, and individual rationality), these characteristics cannot be guaranteed when bidders can submit *false-name* bids. Furthermore, [1] pointed out several other limitations such as vulnerability to the collusion of the auctioneer and/or losers.

Also, to execute the GVA, the auctioneer must solve a complicated optimization problem. Various studies have been conducted to introduce search techniques, which were developed in the artificial intelligence literature, for solving this optimization problem [3].

Cross References

► False-Name-Proof Auction

Recommended Reading

1. Ausubel, L.M., Milgrom, P.R.: Ascending auctions with package bidding. Front. Theor. Econ. **1**(1) Article 1 (2002)
2. Clarke, E.H., Multipart pricing of public goods. Publ. Choice **2**, 19–33 (1971)
3. Cramton, P., Steinberg, R., Shoham, Y. (eds.): Combinatorial Auctions. MIT Press, Cambridge (2005)
4. Mas-Colell, A., Whinston, M.D., Green, J.R.: Microeconomic Theory. Oxford University Press, Oxford (1995)
5. Varian, H.R.: Economic mechanism design for computerized agents. In: Proceedings of the 1st Usenix Workshop on Electronic Commerce, 1995
6. Vickrey, W.: Counter speculation, auctions, and competitive sealed tenders. J. Financ. **16**, 8–37 (1961)

Geographic Routing

2003; Kuhn, Wattenhofer, Zollinger

Aaron Zollinger
Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA, USA

Keywords and Synonyms

Directional routing; Geometric routing; Location-based routing; Position-based routing

Problem Definition

Geographic routing is a type of routing particularly well suited for dynamic ad hoc networks. Sometimes also called directional, geometric, location-based, or position-based routing, it is based on two principal assumptions. First, it is assumed that every node knows its own and its network

neighbors' positions. Second, the source of a message is assumed to be informed about the position of the destination. Geographic routing is defined on a Euclidean graph, that is a graph whose nodes are embedded in the Euclidean plane. Formally, geographic ad hoc routing algorithms can be defined as follows:

Definition 1 (Geographic Ad Hoc Routing Algorithm) Let $G = (V, E)$ be a Euclidean graph. The task of a geographic ad hoc routing algorithm $\mathcal{A}$ is to transmit a message from a source $s \in V$ to a destination $t \in V$ by sending packets over the edges of G while complying with the following conditions:

- All nodes $v \in V$ know their geographic positions as well as the geographic positions of all their neighbors in G.
- The source s is informed about the position of the destination t.
- The control information which can be stored in a packet is limited by $O(\log n)$ bits, that is, only information about a constant number of nodes is allowed.
- Except for the temporary storage of packets before forwarding, a node is not allowed to maintain any information.

Geographic routing is particularly interesting, as it operates without any routing tables whatsoever. Furthermore, once the position of the destination is known, all operations are strictly local, that is, every node is required to keep track only of its direct neighbors. These two factors—absence of necessity to keep routing tables up to date and independence of remotely occurring topology changes—are among the foremost reasons why geographic routing is exceptionally suitable for operation in ad hoc networks. Furthermore, in a sense, geographic routing can be considered a lean version of source routing appropriate for dynamic networks: While in source routing the complete hop-by-hop route to be followed by the message is specified by the source, in geographic routing the source simply addresses the message with the position of the destination. As the destination can generally be expected to move slowly compared to the frequency of topology changes between the source and the destination, it makes sense to keep track of the position of the destination instead of maintaining network topology information up to date; if the destination does not move too fast, the message is delivered regardless of possible topology changes among intermediate nodes.

The cost bounds presented in this entry are achieved on *unit disk graphs*. A unit disk graph is defined as follows:

Definition 2 (Unit Disk Graph) Let $V \subset \mathbb{R}^2$ be a set of points in the 2-dimensional plane. The graph with edges between all nodes with distance at most 1 is called the unit disk graph of V.

Unit disk graphs are often employed to model wireless ad hoc networks.

The routing algorithms considered in this entry operate on planar graphs, graphs that contain no two intersecting edges. There exist strictly local algorithms constructing such planar graphs given a unit disk graph. The edges of planar graphs partition the Euclidean plane into contiguous areas, so-called faces. The algorithms cited in this entry are based on these faces.

Key Results

The first geographic routing algorithm shown to always reach the destination was Face Routing introduced in [14].

Theorem 1 *If the source and the destination are connected, Face Routing executed on an arbitrary planar graph always finds a path to the destination. It thereby takes at most $O(n)$ steps, where n is the total number of nodes in the network.*

There exists however a geographic routing algorithm whose cost is bounded not only with respect to the total number of nodes, but in relation to the *shortest path* between the source and the destination: The GOAFR$^+$ algorithm [15,16,18,24] (pronounced as "gopher-plus") combines *greedy routing*—where every intermediate node relays the message to be routed to its neighbor located nearest to the destination—with face routing. Together with the locally computable *Gabriel Graph* planarization technique, the effort expended by the GOAFR$^+$ algorithm is bounded as follows:

Theorem 2 *Let c be the cost of an optimal path from s to t in a given unit disk graph. GOAFR$^+$ reaches t with cost $O(c^2)$ if s and t are connected. If s and t are not connected, GOAFR$^+$ reports so to the source.*

On the other hand it can be shown that—on certain worst-case graphs—no geographic routing algorithm operating in compliance with the above definition can perform asymptotically better than GOAFR$^+$:

Theorem 3 *There exist graphs where any deterministic (randomized) geographic ad hoc routing algorithm has (expected) cost $\Omega(c^2)$.*

This leads to the following conclusion:

Theorem 4 *The cost expended by GOAFR$^+$ to reach the destination on a unit disk graph is asymptotically optimal.*

In addition, it has been shown that the GOAFR$^+$ algorithm is not only guaranteed to have low worst-case cost but that

it also performs well in average-case networks with nodes randomly placed in the plane [15,24].

Applications

By its strictly local nature geographic routing is particularly well suited for application in potentially highly dynamic wireless ad hoc networks. However, also its employment in dynamic networks in general is conceivable.

Open Problems

A number of problems related to geographic routing remain open. This is true above all with respect to the dissemination within the network of information about the destination position and on the other hand in the context of node mobility as well as network dynamics. Various approaches to these problems have been described in [7] as well as in chapters 11 and 12 of [24]. More generally, taking geographic routing one step further towards its application in practical wireless ad hoc networks [12,13] is a field yet largely open. A more specific open problem is finally posed by the question whether geographic routing can be adapted to networks with nodes embedded in three-dimensional space.

Experimental Results

First experiences with geographic and in particular face routing in practical networks have been made [12,13]. More specifically, problems in connection with graph planarization that can occur in practice were observed, documented, and tackled.

Cross References

- ► Local Computation in Unstructured Radio Networks
- ► Planar Geometric Spanners
- ► Routing in Geometric Networks

Recommended Reading

1. Barrière, L., Fraigniaud, P., Narayanan, L.: Robust Position-Based Routing in Wireless Ad Hoc Networks with Unstable Transmission Ranges. In: Proc. of the 5th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M), pp 19–27. ACM Press, New York (2001)
2. Bose, P., Brodnik, A., Carlsson, S., Demaine, E., Fleischer R., López-Ortiz, A., Morin, P., Munro, J.: Online Routing in Convex Subdivisions. In: International Symposium on Algorithms and Computation (ISAAC). LNCS, vol. 1969, pp 47–59. Springer, Berlin/New York (2000)
3. Bose, P., Morin, P.: Online Routing in Triangulations. In: Proc. 10th Int. Symposium on Algorithms and Computation (ISAAC). LNCS, vol. 1741, pp 113–122. Springer, Berlin (1999)
4. Bose, P.,Morin, P., Stojmenovic, I., Urrutia J.: Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. In: Proc. of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M), 1999, pp 48–55
5. Datta, S., Stojmenovic, I., Wu J.: Internal Node and Shortcut Based Routing with Guaranteed Delivery in Wireless Networks. In: Cluster Computing 5, pp 169–178. Kluwer Academic Publishers, Dordrecht (2002)
6. Finn G.: Routing and Addressing Problems in Large Metropolitan-scale Internetworks. Tech. Report ISI/RR-87–180, USC/ISI, March (1987)
7. Flury, R., Wattenhofer, R.: MLS: An Efficient Location Service for Mobile Ad Hoc Networks. In: Proceedings of the 7th ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc), Florence, Italy, May 2006
8. Fonseca, R., Ratnasamy, S., Zhao, J., Ee, C.T., Culler, D., Shenker, S., Stoica, I.: Beacon Vector Routing: Scalable Point-to-Point Routing in Wireless Sensornets. In: 2nd Symposium on Networked Systems Design & Implementation (NSDI), Boston, Massachusetts, USA, May 2005
9. Gao, J., Guibas, L., Hershberger, J., Zhang, L., Zhu, A.: Geometric Spanner for Routing in Mobile Networks. In: Proc. 2nd ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc), Long Beach, CA, USA, October 2001
10. Hou, T., Li, V.: Transmission Range Control in Multihop Packet Radio Networks. IEEE Tran. Commun. **34**, 38–44 (1986)
11. Karp, B., Kung, H.: GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In: Proc. 6th Annual Int. Conf. on Mobile Computing and Networking (MobiCom), 2000, pp 243–254
12. Kim, Y.J., Govindan, R., Karp, B., Shenker, S.: Geographic Routing Made Practical. In: Proceedings of the Second USENIX/ACM Symposium on Networked System Design and Implementation (NSDI 2005), Boston, Massachusetts, USA, May 2005
13. Kim, Y.J., Govindan, R., Karp, B., Shenker, S.: On the Pitfalls of Geographic Face Routing. In: Proc. of the ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), Cologne, Germany, September 2005
14. Kranakis, E., Singh, H., Urrutia, J.: Compass Routing on Geometric Networks. In: Proc. 11th Canadian Conference on Computational Geometry, Vancouver, August 1999, pp 51–54
15. Kuhn, F., Wattenhofer, R., Zhang, Y., Zollinger, A.: Geometric Routing: Of Theory and Practice. In: Proc. of the 22nd ACM Symposium on the Principles of Distributed Computing (PODC), 2003
16. Kuhn, F., Wattenhofer, R., Zollinger, A.: Asymptotically Optimal Geometric Mobile Ad-Hoc Routing. In: Proc. 6th Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (Dial-M), pp 24–33. ACM Press, New York (2002)
17. Kuhn, F., Wattenhofer, R., Zollinger, A.: Ad-Hoc Networks Beyond Unit Disk Graphs. In: 1st ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), San Diego, California, USA, September 2003
18. Kuhn, F., Wattenhofer, R., Zollinger, A.: Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing. In: Proc. 4th ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc), 2003

19. Leong, B., Liskov, B., Morris, R.: Geographic Routing without Planarization. In: 3rd Symposium on Networked Systems Design & Implementation (NSDI), San Jose, California, USA, May 2006
20. Leong, B., Mitra, S., Liskov, B.: Path Vector Face Routing: Geographic Routing with Local Face Information. In: 13th IEEE International Conference on Network Protocols (ICNP), Boston, Massachusetts, USA, November 2005
21. Takagi, H., Kleinrock, L.: Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals. IEEE Trans. Commun. **32**, 246–257 (1984)
22. Urrutia, J.: Routing with Guaranteed Delivery in Geometric and Wireless Networks. In: Stojmenovic, I. (ed.) Handbook of Wireless Networks and Mobile Computing, ch. 18 pp. 393–406. Wiley, Hoboken (2002)
23. Wattenhofer, M., Wattenhofer, R., Widmayer, P.: Geometric Routing without Geometry. In: 12th Colloquium on Structural Information and Communication Complexity (SIROCCO), Le Mont Saint-Michel, France, May 2005
24. Zollinger, A.: Networking Unleashed: Geographic Routing and Topology Control in Ad Hoc and Sensor Networks, Ph. D. thesis, ETH Zurich, Switzerland Diss. ETH 16025 (2005)

Geometric Computing

Geometric Dilation of Geometric Networks

2006; Dumitrescu, Ebbers-Baumann, Grüne, Klein, Knauer, Rote

Rolf Klein
Institute for Computer Science I, University of Bonn, Bonn, Germany

Keywords and Synonyms

Detour; Spanning ratio; Stretch factor

Problem Definition

Urban street systems can be modeled by *plane geometric networks* $G = (V, E)$ whose edges $e \in E$ are piecewise smooth curves that connect the vertices $v \in V \subset \mathbb{R}^2$. Edges do not intersect, except at common endpoints in V. Since streets are lined with houses, the quality of such a network can be measured by the length of the connections it provides between two arbitrary points p and q on G.

Let $\xi_G(p, q)$ denote a shortest path from p to q in G. Then

$$\delta(p, q) := \frac{|\xi_G(p, q)|}{|pq|} \tag{1}$$

is the detour one encounters when using network G, in order to get from p to q, instead of walking straight. Here, $|.|$ denotes the Euclidean length. The *geometric dilation of network* G is defined by

$$\delta(G) := \sup_{p \neq q \in G} \delta(p, q). \tag{2}$$

This definition differs from the notion of stretch factor (or: spanning ratio) used in the context of spanners; see the monographs by Eppstein [6] or Narasimhan and Smid [11]. In the latter, only the paths between the vertices $p, q \in V$ are considered, whereas the geometric dilation involves all points on the edges as well. As a consequence, the stretch factor of a triangle T equals 1, but its geometric dilation is given by $\delta(T) = \sqrt{2/(1 - \cos\alpha)} \geq 2$, where $\alpha \leq 60°$ is the most acute angle of T.

Presented with a finite set S of points in the plane, one would like to find a finite geometric network containing S whose geometric dilation is as small as possible. The value of

$$\Delta(S) := \inf\{\delta(G); G \text{ finite plane geometric network containing } S\}$$

is called the *geometric dilation of point set* S. The problem is in computing, or bounding, $\Delta(S)$ for a given set S.

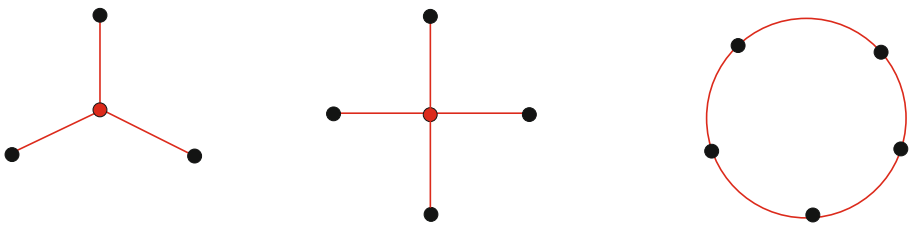
Key Results

Theorem 1 [4] *Let S_n denote the set of corners of a regular n-gon. Then, $\Delta(S_3) = 2/\sqrt{3}$, $\Delta(S_4) = \sqrt{2}$, and $\Delta(S_n) = \pi/2$ for all $n \geq 5$.*

The networks realizing these minimum values are shown in Fig. 1. The proof of minimality uses the following two lemmata that may be interesting in their own right. Lemma 1 was independently obtained by Aronov et al. [1].

Lemma 1 *Let T be a tree containing S_n. Then $\delta(T) \geq n/\pi$.*

Lemma 2 follows from a result of Gromov's [7]. It can more easily be proven by applying Cauchy's surface area formula, see [4].



Geometric Dilation of Geometric Networks, Figure 1
Minimum dilation embeddings of regular point sets

Lemma 2 *Let C denote a simple closed curve in the plane. Then $\delta(C) \geq \pi/2$.*

Clearly, Lemma 2 is tight for the circle. The next lemma implies that the circle is the only closed curve attaining the minimum geometric dilation of $\pi/2$.

Lemma 3 [3] *Let C be a simple closed curve of geometric dilation $< \pi/2 + \epsilon(\delta)$. Then C is contained in an annulus of width δ.*

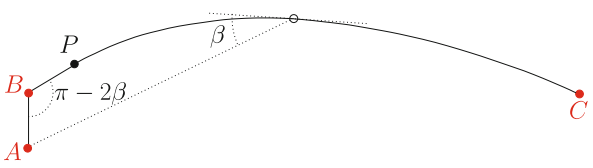
For points in general position, computing their geometric dilation seems quite complicated. Only for sets $S = \{A, B, C\}$ of size three is the solution completely known.

Theorem 2 [5] *The plane geometric network of minimum geometric dilation containing three given points $\{A, B, C\}$ is either a line segment, or a Steiner tree as depicted in Fig. 1, or a simple path consisting of two line segments and one segment of an exponential spiral; see Fig. 2.*

The optimum path shown in Fig. 2 contains a degree two Steiner vertex, P, situated at distance $|AB|$ from B. The path runs straight between A, B and B, P. From P to C it follows an exponential spiral centered at A.

The next results provide upper and lower bounds to $\Delta(S)$.

Theorem 3 [4] *For each finite point set S the estimate $\Delta(S) < 1.678$ holds.*

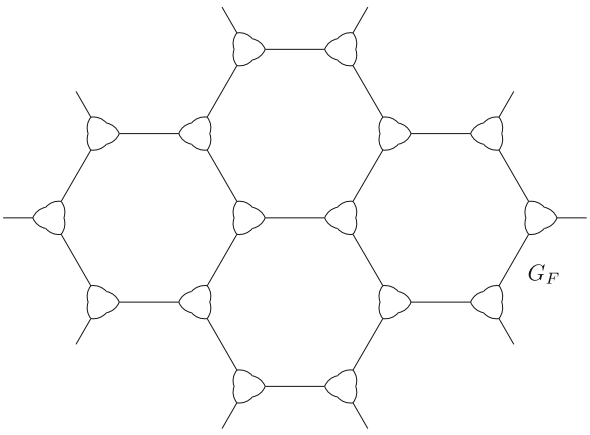


Geometric Dilation of Geometric Networks, Figure 2
The minimum dilation embedding of points A, B, and C

To prove this general upper bound one can replace each vertex of the hexagonal tiling of $\mathbb{R}^2$ with a certain closed Zindler curve (by definition, all point pairs bisecting the perimeter of a Zindler curve have identical distance). This results in a network G_F of geometric dilation ≈ 1.6778; see Fig. 3. Given a finite point set S, one applies a slight deformation to a scaled version of G_F, such that all points of S lie on a finite part, G, of the deformed net. By Dirichlet's result on simultaneous approximation of real numbers by rationals, a deformation small as compared to the cell size is sufficient, so that the dilation is not affected. See [8] for the history and properties of Zindler curves.

Theorem 4 [3] *There exists a finite point set S such that $\Delta(S) > (1 + 10^{-11})\pi/2$.*

Theorem 4 holds for the set S of 19×19 vertices of the integer grid. Roughly, if S were contained in a geometric network G of dilation close to $\pi/2$, the boundaries of the faces of G must be contained in small annuli, by Lemma 3. To the inner and outer circles of these annuli, one can now apply a result by Kuperberg et al. [9] stating that an enlarge-



Geometric Dilation of Geometric Networks, Figure 3
A network of geometric dilation ≈ 1,6778

ment, by a certain factor, of a packing of disks of radius ≤ 1 cannot cover a square of size 4.

Applications

The geometric dilation has applications in the theory of knots, see, e. g., Kusner and Sullivan [10] and Denne and Sullivan [2]. With respect to urban planning, the above results highlight principal dilation bounds for connecting given sites with plane geometric networks.

Open Problems

For practical applications, one would welcome upper bounds to the weight (= total edge length) of a geometric network, in addition to upper bounds on its geometric dilation. Some theoretical questions require further investigation, too. Is $\Delta(S)$ always attained by a finite network? How to compute, or approximate, $\Delta(S)$ for a given finite set S? What is the precise value of $\sup\{\Delta(S); S \text{ finite}\}$?

Cross References

► Dilation of Geometric Networks

Recommended Reading

1. Aronov, B., de Berg, M., Cheong, O., Gudmundsson, J., Haverkort, H., Vigneron, A.: Sparse Geometric Graphs with Small Dilation. 16th International Symposium ISAAC 2005, Sanya. In: Deng, X., Du, D. (eds.) Algorithms and Computation, Proceedings. LNCS, vol. 3827, pp. 50–59. Springer, Berlin (2005)
2. Denne, E., Sullivan, J.M.: The Distortion of a Knotted Curve. http://www.arxiv.org/abs/math.GT/0409438 (2004)
3. Dumitrescu, A., Ebbers-Baumann, A., Grüne, A., Klein, R., Rote, G.: On the Geometric Dilation of Closed Curves, Graphs, and Point Sets. Comput. Geom. Theory Appl. **36**(1), 16–38 (2006)
4. Ebbers-Baumann, A., Grüne, A., Klein, R.: On the Geometric Dilation of Finite Point Sets. Algorithmica **44**(2), 137–149 (2006)
5. Ebbers-Baumann, A., Klein, R., Knauer, C., Rote, G.: The Geometric Dilation of Three Points. Manuscript (2006)
6. Eppstein, D.: Spanning Trees and Spanners. In: Sack, J.-R., Urrutia, J. (eds.) Handbook of Computational Geometry, pp. 425–461. Elsevier, Amsterdam (1999)
7. Gromov, M.: Structures Métriques des Variétés Riemanniennes. Textes Math. CEDIX, vol. 1. F. Nathan, Paris (1981)
8. Grüne, A.: Geometric Dilation and Halving Distance. Ph. D. thesis, Institut für Informatik I, Universität Bonn (2006)
9. Kuperberg, K., Kuperberg, W., Matousek, J., Valtr, P.: Almost Tiling the Plane with Ellipses. Discrete Comput. Geom. **22**(3), 367–375 (1999)
10. Kusner, R.B., Sullivan, J.M.: On Distortion and Thickness of Knots. In: Whittington, S.G. et al. (eds.) Topology and Geometry in Polymer Science. IMA Volumes in Math. and its Applications, vol. 103, pp. 67–78. Springer, New York (1998)
11. Narasimhan, G., Smid, M.: Geometric Spanner Networks. Cambridge University Press(2007)

Geometric Spanners

2002; Gudmundsson, Levcopoulos, Narasimhan

Joachim Gudmundsson[1], Giri Narasimhan[2], Michiel Smid[3]

[1] DMiST, National ICT Australia Ltd, Alexandria, NSW, Australia

[2] Department of Computer Science, Florida International University, Miami, FL, USA

[3] School of Computer Science, Carleton University, Ottawa, ON, Canada

Keywords and Synonyms

Dilation; t-spanners

Problem Definition

Consider a set S of n points in d-dimensional Euclidean space. A network on S can be modeled as an undirected graph G with vertex set S of size n and an edge set E where every edge (u, v) has a weight. A geometric (Euclidean) network is a network where the weight of the edge (u, v) is the Euclidean distance $|uv|$ between its endpoints. Given a real number $t > 1$ we say that G is a *t-spanner* for S, if for each pair of points $u, v \in S$, there exists a path in G of weight at most t times the Euclidean distance between u and v. The minimum t such that G is a t-spanner for S is called the stretch factor, or dilation, of G. For a more detailed description of the construction of t-spanners see the book by Narasimhan and Smid [18]. The problem considered is the construction of t-spanners given a set S of n points in $\mathcal{R}^d$ and a positive real value $t > 1$, where d is a constant. The aim is to compute a good t-spanner for S with respect to the following quality measures:

size: the number of edges in the graph.

degree: the maximum number of edges incident on a vertex.

weight: the sum of the edge weights.

spanner diameter: the smallest integer k such that for any pair of vertices u and v in S, there is a path in the graph of length at most $t \cdot |uv|$ between u and v containing at most k edges.

fault-tolerance: the resilience of the graph to edge, vertex or region failures.

Thus, good t-spanners require large fault-tolerance and small size, degree, weight and spanner diameter.

Key Results

This section contains a description of the three most common approaches for constructing a t-spanner of a set of points in Euclidean space. It also contains a description of the construction of fault-tolerant spanners, spanners among polygonal obstacles and, finally, a short note on dynamic and kinetic spanners.

Spanners of Points in Euclidean Space

The most well-known classes of t-spanner networks for points in Euclidean space include: Θ-graphs, WSPD-graphs and Greedy-spanners. In the following sections the main idea of each of these classes is given, together with the known bounds on the quality measures.

The Θ-Graph The Θ-graph was discovered independently by Clarkson and Keil in the late 80's. The general idea is to process each point $p \in S$ independently as follows. Partition $\mathcal{R}^d$ into k simplicial cones of angular diameter at most θ and apex at p, where $k = O(1/\theta^{d-1})$. For each non-empty cone C, an edge is added between p and the point in C whose orthogonal projection onto some fixed ray in C emanating from p is closest to p, see Fig. 1a. The resulting graph is called the Θ-graph on S.

Theorem 1 *The Θ-graph is a t-spanner of S for $t = 1/(\cos\theta - \sin\theta)$ with $O(n/\theta^{d-1})$ edges, and can be computed in $O((n/\theta^{d-1})\log^{d-1} n)$ time using $O(n/\theta^{d-1} + n\log^{d-2} n)$ space.*

The following variants of the Θ-graph also give bounds on the degree, diameter and weight.

Skip-List Spanners The idea is to generalize skip-lists and apply them to the construction of spanners. Construct a sequence of h subsets, $S_1, \ldots, S_h$, where $S_1 = S$ and S_i is constructed from S_{i-1} as follows (reminiscent of the levels in a skip list). For each point in S_{i-1}, flip a fair coin. The set S_i is the set of all points of S_{i-1} whose coin flip produced heads. The construction stops if $S_i = \emptyset$. For each subset a Θ-graph is constructed. The union of the graphs is the skip-list spanner of S with dilation t, having $O(n/\theta^{d-1})$ edges and $O(\log n)$ spanner diameter with high probability [3].

Gap-Greedy A set of directed edges is said to satisfy the *gap* property if the sources of any two distinct edges in the set are separated by a distance that is at least proportional to the length of the shorter of the two edges. Arya and Smid [5] proposed an algorithm that uses the gap property to decide whether or not an edge should be added to the t-spanner graph. Using the gap property the constructed spanner can be shown to have degree $O(1/\theta^{d-1})$ and weight $O(\log n \cdot wt(\mathrm{MST}(S)))$, where $wt(\mathrm{MST}(S))$ is the weight of the minimum spanning tree of S.

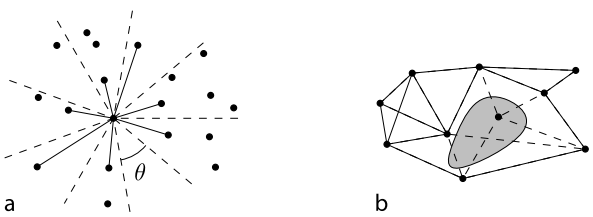
The WSPD-Graph Let A and B be two finite sets of points in $\mathcal{R}^d$. We say that A and B are *well-separated with respect to* a real value $s > 0$, if there are two disjoint balls C_A and C_B, having the same radius, such that C_A contains A, C_B contains B, and the distance between C_A and C_B is at least equal to s times the radius of C_A. The value s is denoted the *separation ratio*.

Definition 1 ([6]) Let S be a set of points in $\mathcal{R}^d$, and let $s > 0$ be a real number. A *well-separated pair decomposition* (WSPD) for S with respect to s is a sequence $\{A_i, B_i\}$, $1 \le i \le m$, of pairs of non-empty subsets of S, such that (1) $A_i \cap B_i = \emptyset$ for all $i = 1, 2, \ldots, m$, (2) for each unordered pair $\{p, q\}$ of distinct points of S, there is exactly one pair $\{A_i, B_i\}$ in the sequence, such that $p \in A_i$ and $q \in B_i$, or $p \in B_i$ and $q \in A_i$, and (3) A_i and B_i are well-separated with respect to s, for all $i = 1, 2, \ldots, m$.

The well-separated pair decomposition (WSPD) was developed by Callahan and Kosaraju [6]. The construction of a t-spanner using the well-separated pair decomposition is done by first constructing a WSPD of S with respect to a separation constant $s = (4(t+1))/(t-1)$. Initially set the spanner graph $G = (S, \emptyset)$ and add edges iteratively as follows. For each well-separated pair $\{A, B\}$ in the decomposition, an edge (a, b) is added to the graph, where a and b are arbitrary points in A and B, respectively. The resulting graph is called the WSPD-graph on S.

Theorem 2 *The WSPD-graph is a t-spanner for S with $O(s^d \cdot n)$ edges and can be constructed in time $O(s^d n + n\log n)$, where $s = 4(t+1)/(t-1)$.*

There are modifications that can be made to obtain bounded diameter or bounded degree.



Geometric Spanners, Figure 1
a Illustrating the Θ-graph. b A graph with a region-fault

Bounded Diameter Arya, Mount and Smid [3] showed how to modify the construction algorithm such that the diameter of the graph is bounded by $2 \log n$. Instead of selecting an arbitrary point in each well-separated set, their algorithm carefully chooses a specially selected point for each set.

Bounded Degree A single point v can be part of many well-separated pairs and each of these pairs may generate an edge with an endpoint at v. Arya et al. [2] suggested an algorithm that retains only the shortest edge for each cone direction, thus combining the Θ-graph approach with the WSPD-graph. By adding a post-processing step that handles all high-degree vertices, a t-spanner of degree $O(1/(t-1)^{2d-1})$ is obtained.

The Greedy-Spanner The greedy algorithm was first presented in 1989 by Bern (see also Levcopoulos and Lingas [15]) and since then the greedy algorithm has been subject to considerable research. The graph constructed using the greedy algorithm is called a Greedy-spanner, and the general idea is that the algorithm iteratively builds a graph G. The edges in the complete graph are processed in order of increasing edge length. Testing an edge (u, v) entails a shortest path query in the partial spanner graph G. If the shortest path in G between u and v is at most $t \cdot |uv|$ then the edge (u, v) is discarded, otherwise it is added to the partial spanner graph G.

Das, Narasimhan and Salowe [11] proved that the greedy-spanner fulfills the so-called *leapfrog property*. A set of undirected edges E is said to satisfy the t-leapfrog property, if for every $k \geq 2$, and for every possible sequence $\{(p_1, q_1), \ldots, (p_k, q_k)\}$ of pairwise distinct edges of E,

$$t \cdot |p_1 q_1| < \sum_{i=2}^{k} |p_i q_i| + t \cdot \Big(\sum_{i=1}^{k-1} |q_i p_{i+1}| + |p_k q_1| \Big) .$$

Using the leapfrog property it is possible to bound the weight of the graph. Das and Narasimhan [10] observed that the Greedy-spanner can be approximated while maintaining the leapfrog property. This observation allowed for faster construction algorithms.

Theorem 3 ([14]) *The approximate greedy-spanner is a t-spanner of S with maximum degree $O(1/(t-1)^{2d-1})$, weight $O((1/(t-1)^{2d-1} \cdot wt(MST(S))))$, and can be computed in time $O(n/((t-1)^{2d}) \log n)$.*

Fault-Tolerant Spanners

The concept of fault-tolerant spanners was first introduced by Levcopoulos et al. [16] in 1998, i. e., after one or more vertices or edges fail, the spanner should retain its good properties. In particular, there should still be a short path between any two vertices in what remains of the spanner after the fault. Czumaj and Zhao [8] showed that a greedy approach produces a k-vertex (or k-edge) fault tolerant geometric t-spanner with degree $O(k)$ and total weight $O(k^2 \cdot wt(MST(S)))$; these bounds are asymptotically optimal.

For geometric spanners it is natural to consider *region faults*, i. e., faults that destroy all vertices and edges intersecting some geometric fault region. For a fault region F let $G \ominus F$ be the part of G that remains after the points from S inside F and all edges that intersect F have been removed from the graph, see Fig. 1b. Abam et al. [1] showed how to construct region-fault tolerant t-spanners of size $O(n \log n)$ that are fault-tolerant to any convex region-fault. If one is allowed to use Steiner points then a linear size t-spanner can be achieved.

Spanners Among Obstacles

The visibility graph of a set of pairwise non-intersecting polygons is a graph of intervisible locations. Each polygonal vertex is a vertex in the graph, and each edge represents a visible connection between them; that is, if two vertices can see each other, an edge is drawn between them. This graph is useful since it contains the shortest obstacle avoiding path between any pair of vertices.

Das [9] showed that a t-spanner of the visibility graph of a point set in the Euclidean plane can be constructed by using the Θ-graph approach followed by a pruning step. The obtained graph has linear size and constant degree.

Dynamic and Kinetic Spanners

Not much is known in the areas of dynamic or kinetic spanners. Arya et al. [4] showed a data structure of size $O(n \log^d n)$ that maintains the skip-list spanner, described in Sect. "The Θ-Graph", in $O(\log^d n \log \log n)$ expected amortized time per insertion and deletion in the model of random updates.

Gao et al. [13] showed how to maintain a t-spanner of size $O(n/(t-1)^d)$ and maximum degree $O(1/(t-2)^d \log \alpha)$ in time $O((\log \alpha)/(t-1)^d)$ per insertion and deletion, where α denotes the aspect ratio of S, i. e., the ratio of the maximum pairwise distance to the minimum pairwise distance. The idea is to use an hierarchical structure T with $O(\log \alpha)$ levels, where each level contains a set of centers

(subset of S). Each vertex v on level i in T is connected by an edge to all other vertices on level i within distance $O(2^i/(t-1))$ of v. The resulting graph is a t-spanner of S and it can be maintained as stated above. The approach can be generalized to the kinetic case so that the total number of events in maintaining the spanner is $O(n^2 \log n)$ under pseudo-algebraic motion. Each event can be updated in $O((\log \alpha)/(t-1)^d)$ time.

Applications

The construction of sparse spanners has been shown to have numerous applications areas such as metric space searching [1], which includes query by content in multimedia objects, text retrieval, pattern recognition and function approximation. Another example is broadcasting in communication networks [17]. Several well-known theoretical results also use the construction of t-spanners as a building block, for example, Rao and Smith [19] made a breakthrough by showing an optimal $O(n \log n)$-time approximation scheme for the well-known Euclidean *traveling salesperson problem*, using t-spanners (or banyans). Similarly, Czumaj and Lingas [7] showed approximation schemes for minimum-cost multi-connectivity problems in geometric networks.

Open Problems

There are many open problems in this area. Only a few are mentioned here:

1. Design a dynamic t-spanner that can be updated in $O(\log^c n)$ time, for some constant c.
2. Determine if there exists a fault-tolerant t-spanner of linear size for convex region faults.
3. The k-vertex fault tolerant spanner by Czumaj and Zhao [8] produces a k-vertex fault tolerant t-spanner of degree $O(k)$ and weight $O(k^2 \cdot wt(MST(S)))$. However, it is not known how to implement it efficiently. Can such a spanner be computed in $O(n \log n + kn)$ time?
4. Bound the weight of skip-list spanners.

Experimental Results

The problem of constructing spanners has received considerable attention from a theoretical perspective but not much attention from a practical, or experimental perspective. Navarro and Paredes [1] presented four heuristics for point sets in high-dimensional space ($d = 20$) and showed by empirical methods that the running time was $O(n^{2.24})$ and the number of edges in the produced graphs was $O(n^{1.13})$. Recently Farshi and Gudmundsson [12] performed a thorough comparison of the construction algorithms discussed in Section "Spanners of Points in Euclidean Space".

Cross References

- ▶ Applications of Geometric Spanner Networks
- ▶ Approximating Metric Spaces by Tree Metrics
- ▶ Dilation of Geometric Networks
- ▶ Planar Geometric Spanners
- ▶ Single-Source Shortest Paths
- ▶ Sparse Graph Spanners
- ▶ Well Separated Pair Decomposition

Recommended Reading

1. Abam, M.A., de Berg, M., Farshi, M., Gudmundsson, J.: Region-fault tolerant geometric spanners. In: Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms, New Orleans, 7–9 January 2007
2. Arya, S., Das, G., Mount, D.M., Salowe, J.S., Smid, M.: Euclidean spanners: short, thin, and lanky. In: Proceedings of the 27th ACM Symposium on Theory of Computing, pp. 489–498. Las Vegas, 29 May–1 June 1995
3. Arya, S., Mount, D.M., Smid, M.: Randomized and deterministic algorithms for geometric spanners of small diameter. In: Proceedings of the 35th IEEE Symposium on Foundations of Computer Science, pp. 703–712. Santa Fe, 20–22 November 1994
4. Arya, S., Mount, D.M., Smid, M.: Dynamic algorithms for geometric spanners of small diameter: Randomized solutions. Comput. Geom. Theor. Appl. **13**(2), 91–107 (1999)
5. Arya, S., Smid, M.: Efficient construction of a bounded-degree spanner with low weight. Algorithmica **17**, 33–54 (1997)
6. Callahan, P.B., Kosaraju, S.R.: A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. J. ACM **42**, 67–90 (1995)
7. Czumaj, A., Lingas, A.: Fast approximation schemes for Euclidean multi-connectivity problems. In: Proceedings of the 27th International Colloquium on Automata, Languages and Programming. Lect. Notes Comput. Sci. **1853**, 856–868 (2000)
8. Czumaj, A., Zhao, H.: Fault-tolerant geometric spanners. Discret. Comput. Geom. **32**(2), 207–230 (2004)
9. Das, G.: The visibility graph contains a bounded-degree spanner. In: Proceedings of the 9th Canadian Conference on Computational Geometry, Kingston, 11–14 August 1997
10. Das, G., Narasimhan, G.: A fast algorithm for constructing sparse Euclidean spanners. Int. J. Comput. Geom. Appl. **7**, 297–315 (1997)
11. Das, G., Narasimhan, G., Salowe, J.: A new way to weigh malnourished Euclidean graphs. In: Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms, pp. 215–222. San Francisco, 22–24 January 1995
12. Farshi, M., Gudmundsson, J.: Experimental study of geometric t-spanners. In: Proceedings of the 13th Annual European Symposium on Algorithms. Lect. Notes Comput. Sci. **3669**, 556–567 (2005)
13. Gao, J., Guibas, L.J., Nguyen, A.: Deformable spanners and applications. In: Proceedings of the 20th ACM Symposium on Computational Geometry, pp. 190–199, New York, 9–11 June 2004

14. Gudmundsson, J., Levcopoulos, C., Narasimhan, G.: Improved greedy algorithms for constructing sparse geometric spanners. SIAM J. Comput. **31**(5), 1479–1500 (2002)
15. Levcopoulos, C., Lingas, A.: There are planar graphs almost as good as the complete graphs and almost as cheap as minimum spanning trees. Algorithmica **8**(3), 251–256 (1992)
16. Levcopoulos, C., Narasimhan, G., Smid, M.: Improved algorithms for constructing fault-tolerant spanners. Algorithmica **32**, 144–156 (2002)
17. Li, X.Y.: Applications of computational geometry in wireless ad hoc networks. In: Cheng, X.Z., Huang, X., Du, D.Z. (eds.) Ad Hoc Wireless Networking, pp. 197–264. Kluwer, Dordrecht (2003)
18. Narasimhan, G., Smid, M.: Geometric spanner networks. Cambridge University Press, New York (2006)
19. Navarro, G., Paredes, R.: Practical construction of metric *t*-spanners. In: Proceedings of the 5th Workshop on Algorithm Engineering and Experiments, pp. 69–81, 11 January 2003. SIAM Press, Baltimore
20. Rao, S., Smith, W.D.: Approximating geometrical graphs via spanners and banyans. In: Proceedings of the 30th ACM Symposium on Theory of Computing, pp. 540–550. Dallas, 23–26 May 1998

Gomory–Hu Trees

2007; Bhalgat, Hariharan, Kavitha, Panigrahi

Debmalya Panigrahi
Computer Science & Artificial Intelligence Laboratory, MIT, Cambridge, MA, USA

Keywords and Synonyms

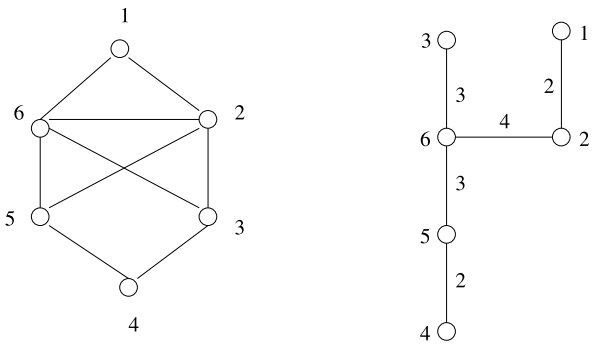
Cut trees

Problem Definition

Let $G = (V, E)$ be an undirected graph with $|V| = n$ and $|E| = m$. The edge connectivity of two vertices $s, t \in V$, denoted by $\lambda(s, t)$, is defined as the size of the smallest cut that separates s and t; such a cut is called a minimum s–t cut. Clearly, one can represent the $\lambda(s, t)$ values for all pairs of vertices s and t in a table of size $O(n^2)$. However, for reasons of efficiency, one would like to represent all the $\lambda(s, t)$ values in a more succinct manner. *Gomory–Hu trees* (also known as *cut trees*) offer one such succinct representation of linear (i. e., $O(n)$) space and constant (i. e., $O(1)$) lookup time. It has the additional advantage that apart from representing all the $\lambda(s, t)$ values, it also contains structural information from which a minimum s–t cut can be retrieved easily for any pair of vertices s and t.

Formally, a Gomory–Hu tree $T = (V, F)$ of an undirected graph $G = (V, E)$ is a weighted undirected tree defined on the vertices of the graph such that the following properties are satisfied:

- For any pair of vertices $s, t \in V$, $\lambda(s, t)$ is equal to the minimum weight on an edge in the unique path connecting s to t in T. Call this edge $e(s, t)$. If there are multiple edges with the minimum weight on the s to t path in T, any one of these edges is designated as $e(s, t)$.
- For any pair of vertices s and t, the bipartition of vertices into components produced by removing $e(s, t)$ (if there are multiple candidates for $e(s, t)$, this property holds for each candidate edge) from T corresponds to a minimum s–t cut in the original graph G.



Gomory–Hu Trees, Figure 1
An undirected graph (*left*) and a corresponding Gomory–Hu tree (*right*)

To understand this definition better, consider the following example. Figure 1 shows an undirected graph and a corresponding Gomory–Hu tree. Focus on a pair of vertices, for instance, 3 and 5. Clearly, the edge (6, 5) of weight 3 is a minimum-weight edge on the 3 to 5 path in the Gomory–Hu tree. It is easy to see that $\lambda(3, 5) = 3$ in the original graph. Now, removing this edge produces the vertex bipartition $(\{1, 2, 3, 6\}, \{4, 5\})$, which is a cut of size 3 in the original graph.

It is not immediate that such Gomory–Hu trees exist for all undirected graphs. In a classical result in 1961, Gomory and Hu [7] showed that not only do such trees exist for all undirected graphs, but that they can also be computed using $n - 1$ minimum s–t computations (which are equivalent to maximum flow computations, by the celebrated Menger's theorem). In fact, a graph can have multiple Gomory–Hu trees.

All previous algorithms for building Gomory–Hu trees in undirected graphs used maximum flow subroutines. Gomory and Hu showed how to compute the cut tree T using $n - 1$ maximum flow computations and graph contractions. Gusfield [8] proposed an algorithm that does not use graph contractions; all $n - 1$ maximum flow compu-

tations are performed on the input graph. Goldberg and Tsioutsiouliklis [6] did an experimental study of the algorithms due to Gomory and Hu and due to Gusfield for the cut tree problem and described efficient implementations of these algorithms. Examples were shown by Benczúr [1] that cut trees do not exist for directed graphs.

Any maximum flow based approach for constructing a Gomory–Hu tree would have a running time of $(n-1)$ times the time for computing a single maximum flow. Till now, faster algorithms for Gomory–Hu trees were by-products of faster algorithms for computing a maximum flow. The current fastest $\tilde{O}(m + n\lambda(s,t))$ (polylog n factors ignored in $\tilde{O}$ notation) maximum-flow algorithm, due to Karger and Levine [10], yields the current best expected running time of $\tilde{O}(n^3)$ for Gomory–Hu tree construction on simple unweighted graphs with n vertices. Bhalgat et al. [2] improved this time complexity to $\tilde{O}(mn)$. Note that both Karger and Levine's algorithm and Bhalgat et al.'s algorithm are randomized Las Vegas algorithms. The fastest deterministic algorithm for the Gomory–Hu tree construction problem is a by-product of Goldberg and Rao's maximum-flow algorithm [5] and has a running time of $\tilde{O}(nm^{1/2}\min(m, n^{3/2}))$.

Key Results

Bhalgat et al. [2] considered the problem of designing an efficient algorithm for constructing a Gomory–Hu tree on unweighted undirected graphs. The main theorem shown in this paper is the following.

Theorem 1 *Let $G = (V, E)$ be a simple unweighted graph with m edges and n vertices. Then a Gomory–Hu tree for G can be built in expected time $\tilde{O}(mn)$.*

Their algorithm is always faster by a factor of $\tilde{\Omega}(n^{2/9})$ (polylog n factors ignored in $\tilde{\Omega}$ notation) compared to the previous best algorithm.

Instead of using maximum flow subroutines, they use a Steiner connectivity algorithm. The *Steiner connectivity* of a set of vertices S (called the *Steiner set*) in an undirected graph is the minimum size of a cut which splits S into two parts; such a cut is called a *minimum Steiner cut*. Generalizing a tree-packing algorithm given by Gabow [4] for finding the edge connectivity of a graph, Cole and Hariharan [3] gave an algorithm for finding the Steiner connectivity k of a set of vertices in either undirected or directed Eulerian unweighted graphs in $\tilde{O}(mk^2)$ time. (For undirected graphs, their algorithm runs a little faster in time $\tilde{O}(m + nk^3)$.) Bhalgat et al. improved this result and gave the following theorem.

Theorem 2 *In an undirected or directed Eulerian unweighted graph, the Steiner connectivity k of a set of vertices can be determined in time $\tilde{O}(mk)$.*

The algorithm in [3] was used by Hariharan et al. [9] to design an algorithm with expected running time $\tilde{O}(m + nk^3)$ to compute a *partial* Gomory–Hu tree for representing the $\lambda(s,t)$ values for all pairs of vertices s, t that satisfied $\lambda(s,t) \leq k$. Replacing the algorithm in [3] by the new algorithm for computing Steiner connectivity yields an algorithm to compute a partial Gomory–Hu tree in expected running time $\tilde{O}(m + nk^2)$. Bhalgat et al. showed that using a more detailed analysis this result can be improved to give the following theorem.

Theorem 3 *The partial Gomory–Hu tree of an undirected unweighted graph to represent all $\lambda(s,t)$ values not exceeding k can be constructed in expected time $\tilde{O}(mk)$.*

Since $\lambda(s,t) < n$ for all s, t vertex pairs in an unweighted (and simple) graph, setting k to n in Theorem 3 implies Theorem 1.

Applications

Gomory–Hu trees have many applications in multiterminal network flows and are an important data structure in graph connectivity literature.

Open Problems

The problem of derandomizing the algorithm due to Bhalgat et al. [2] to produce an $\tilde{O}(mn)$ time deterministic algorithm for constructing Gomory–Hu trees for unweighted undirected graphs remains open. The other main challenge is to extend the results in [2] to weighted graphs.

Experimental Results

Goldberg and Tsioutsiouliklis [6] did an extensive experimental study of the cut tree algorithms due to Gomory and Hu [7] and that due to Gusfield [8]. They showed how to efficiently implement these algorithms and also introduced and evaluated heuristics for speeding up the algorithms. Their general observation was that while Gusfield's algorithm is faster in many situations, Gomory and Hu's algorithm is more robust. For more detailed results of their experiments, refer to [6].

No experimental results are reported for the algorithm due to Bhalgat et al. [2].

Cross References

► Approximate Maximum Flow Construction

Recommended Reading

1. Benczúr, A.A.: Counterexamples for Directed and Node Capacitated Cut-Trees. SIAM J. Comput. **24**(3), 505–510 (1995)
2. Bhalgat, A., Hariharan, R., Kavitha, T., Panigrahi, D.: An $\tilde{O}(mn)$ Gomory-Hu tree construction algorithm for unweighted graphs. In: Proc. of the 39th Annual ACM Symposium on Theory of Computing, San Diego 2007
3. Cole, R., Hariharan, R.: A Fast Algorithm for Computing Steiner Edge Connectivity. In: Proc. of the 35th Annual ACM Symposium on Theory of Computing, San Diego 2003, pp. 167–176
4. Gabow, H.N.: A matroid approach to finding edge connectivity and packing arborescences. J. Comput. Syst. Sci. **50**, 259–273 (1995)
5. Goldberg, A.V., Rao, S.: Beyond the Flow Decomposition Barrier. J. ACM **45**(5), 783–797 (1998)
6. Goldberg, A.V., Tsioutsiouliklis, K.: Cut Tree Algorithms: An Experimental Study. J. Algorithms **38**(1), 51–83 (2001)
7. Gomory, R.E., Hu, T.C.: Multi-terminal network flows. J. Soc. Indust. Appl. Math. **9**(4), 551–570 (1961)
8. Gusfield, D.: Very Simple Methods for All Pairs Network Flow Analysis. SIAM J. Comput. **19**(1), 143–155 (1990)
9. Hariharan, R., Kavitha, T., Panigrahi, D.: Efficient Algorithms for Computing All Low *s-t* Edge Connectivities and Related Problems. In: Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 127–136
10. Karger, D., Levine, M.: Random Sampling in Residual Graphs. In: Proc. of the 34th Annual ACM Symposium on Theory of Computing 2002, pp. 63–66

Graph Bandwidth

1998; Feige
2000; Feige

JAMES R. LEE
Department of Computer Science and Engineering, University of Washington, Seattle, WA, USA

Keywords and Synonyms

Graph bandwidth; Approximation algorithms; Metric embeddings

Problem Definition

The *graph bandwidth problem* concerns producing a linear ordering of the vertices of a graph $G = (V, E)$ so as to minimize the maximum "stretch" of any edge in the ordering. Formally, let $n = |V|$, and consider any one-to-one mapping $\pi : V \to \{1, 2, \ldots, n\}$. The *bandwidth* of this ordering is $\mathsf{bw}_\pi(G) = \max_{\{u,v\}\in E} |\pi(u) - \pi(v)|$. The *bandwidth of G* is given by the bandwidth of the best possible ordering: $\mathsf{bw}(G) = \min_\pi \mathsf{bw}_\pi(G)$.

The original motivation for this problem lies in the preprocessing of sparse symmetric square matrices. Let A be such an $n \times n$ matrix, and consider the problem of finding a permutation matrix P such that the non-zero entries of $P^{\mathrm{T}}AP$ all lie in as narrow a band as possible about the diagonal. This problem is equivalent to minimizing the bandwidth of the graph G whose vertex set is $\{1, 2, \ldots, n\}$ and which has an edge $\{u, v\}$ precisely when $A_{u,v} \neq 0$.

In lieu of this fact, one tries to efficiently compute a linear ordering π for which $\mathsf{bw}_\pi(G) \leq A \cdot \mathsf{bw}(G)$, with the *approximation factor* A is as small as possible. There is even evidence that achieving any value $A = O(1)$ is NP-hard [18]. Much of the difficulty of the bandwidth problem is due to the objective function being a maximum over all edges of the graph. This makes divide-and-conquer approaches ineffective for graph bandwidth, whereas they often succeed for related problems like Minimum Linear Arrangement [6] (here the objective is to minimize $\sum_{\{u,v\}\in E} |\pi(u) - \pi(v)|$). Instead, a more global algorithm is required. To this end, a good lower bound on the value of $\mathsf{bw}(G)$ has to be initially discussed.

The Local Density

For any pair of vertices $u, v \in V$, let $d(u, v)$ to be the shortest path distance between u and v in the graph G. Then, define $B(v, r) = \{u \in V : d(u, v) \leq r\}$ as the *ball of radius r* about a vertex $v \in V$. Finally, the *local density of G* is defined by $D(G) = \max_{v\in V, r\geq 1} |B(v, r)|/(2r)$. It is not difficult to see that $\mathsf{bw}(G) \geq D(G)$. Although it was conjectured that an upper bound of the form $\mathsf{bw}(G) \leq \mathrm{poly}(\log n) \cdot D(G)$ holds, it was not proven until the seminal work of Feige [7].

Key Results

Feige proved the following.

Theorem 1 *There is an efficient algorithm that, given a graph $G = (V, E)$ as input, produces a linear ordering $\pi : V \to \{1, 2, \ldots, n\}$ for which $\mathsf{bw}_\pi(G) \leq O\left((\log n)^3 \sqrt{\log n \log\log n}\right) \cdot D(G)$. In particular, this provides a $\mathrm{poly}(\log n)$-approximation algorithm for the bandwidth problem in general graphs.*

Feige's algorithmic framework can be described quite simply as follows.

1. Compute a representation $f : V \to \mathbb{R}^n$ of G in Euclidean space.
2. Let $u_1, u_2, \ldots, u_n$ be independent $N(0, 1)$[1] random variables, and for each vertex $v \in V$, compute $h(v) =$

[1] $N(0, 1)$ denotes a standard normal random variable with mean 0 and variance 1.

$\sum_{i=1}^{n} u_i f_i(v)$, where $f_i(v)$ is the ith coordinate of the vector $f(v)$.

3. Sort the vertices by the value $h(v)$, breaking ties arbitrarily, and output the induced linear ordering.

An equivalent characterization of steps (2) and (3) is to choose a uniformly random vector $\mathbf{a} \in S^{n-1}$ from the $(n-1)$-dimensional sphere $S^{n-1} \subseteq \mathbb{R}^n$ and output the linear ordering induced by the values $h(v) = \langle \mathbf{a}, f(v) \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the usual inner product on $\mathbb{R}^n$. In other words, the algorithm first computes a map $f: V \to \mathbb{R}^n$, projects the images of the vertices onto a randomly oriented line, and then outputs the induced ordering; step (2) is the standard way that such a random projection is implemented.

Volume-Respecting Embeddings

The only step left unspecified is (1); the function f has to somehow preserve the structure of the graph G in order for the algorithm to output a low-bandwidth ordering. The inspiration for the existence of such an f comes from the field of *low-distortion metric embeddings* (see, e. g. [2,14]). Feige introduced a generalization of low-distortion embeddings to mappings called *volume respecting embeddings*. Roughly, the map f should be non-expansive, in the sense that $\|f(u) - f(v)\| \leq 1$ for every edge $\{u, v\} \in E$, and should satisfy the following property: For any set of k vertices $v_1, \ldots, v_k$, the $(k-1)$-dimensional volume of the convex hull of the points $f(v_1), \ldots, f(v_k)$ should be as large as possible. The proper value of k is chosen to optimize the performance of the algorithm. Refer to [7,10,11] for precise definitions on volume-respecting embeddings, and a detailed discussion of their construction. Feige showed that a modification of Bourgain's embedding [2] yields a mapping $f: V \to \mathbb{R}^n$ which is good enough to obtain the results of Theorem 1.

The requirement $\|f(u) - f(v)\| \leq 1$ for every edge $\{u, v\}$ is natural since $f(u)$ and $f(v)$ need to have similar projections onto the random direction $\mathbf{a}$; intuitively, this suggests that u and v will not be mapped too far apart in the induced linear ordering. But even if $|h(u) - h(v)|$ is small, it may be that many vertices project between $h(u)$ and $h(v)$, causing u and v to incur a large stretch. To prevent this, the images of the vertices should be sufficiently "spread out," which corresponds to the volume requirement on the convex hull of the images.

Applications

As was mentioned previously, the graph bandwidth problem has applications to preprocessing sparse symmetric matrices. Minimizing the bandwidth of matrices helps in improving the efficiency of certain linear algebraic algorithms like Gaussian elimination; see [3,8,17]. Follow-up work has shown that Feige's techniques can be applied to VLSI layout problems [19].

Open Problems

First, state the *bandwidth conjecture* (see, e. g. [13]).

Conjecture: For any n-node graph $G = (V, E)$, one has $\mathsf{bw}(G) = O(\log n) \cdot D(G)$.

The conjecture is interesting and unresolved even in the special case when G is a tree (see [9] for the best results for trees). The best-known bound in the general case follows from [7,10], and is of the form $\mathsf{bw}(G) = O(\log n)^{3.5} \cdot D(G)$. It is known that the conjectured upper bound is best possible, even for trees [4]. One suspects that these combinatorial studies will lead to improved approximation algorithms.

However, the best approximation algorithms, which achieve ratio $O((\log n)^3 (\log \log n)^{1/4})$, are not based on the local density bound. Instead, they are a hybrid of a semi-definite programming approach of [1,5] with the arguments of Feige, and the volume-respecting embeddings constructed in [12,16]. Determining the approximability of graph bandwidth is an outstanding open problem, and likely requires improving both the upper and lower bounds.

Recommended Reading

1. Blum, A., Konjevod, G., Ravi, R., Vempala, S.: Semi-definite relaxations for minimum bandwidth and other vertex-ordering problems. Theor. Comput. Sci. **235**(1), 25–42 (2000), Selected papers in honor of Manuel Blum (Hong Kong, 1998)
2. Bourgain, J.: On Lipschitz embedding of finite metric spaces in Hilbert space. Israel J. Math. **52**(1–2), 46–52 (1985)
3. Chinn, P.Z., Chvátalová, J., Dewdney, A.K., Gibbs, N.E.: The bandwidth problem for graphs and matrices—a survey. J. Graph Theory **6**(3), 223–254 (1982)
4. Chung, F.R.K., Seymour, P.D.: Graphs with small bandwidth and cutwidth. Discret. Math. **75**(1–3), 113–119 (1989). Graph theory and combinatorics, Cambridge (1988)
5. Dunagan, J., Vempala, S.: On Euclidean embeddings and bandwidth minimization. In: Randomization, approximation, and combinatorial optimization, pp. 229–240. Springer (2001)
6. Even, G., Naor, J., Rao, S., Schieber, B.: Divide-and-conquer approximation algorithms via spreading metrics. J. ACM **47**(4), 585–616 (2000)
7. Feige, U.: Approximating the bandwidth via volume respecting embeddings. J. Comput. Syst. Sci. **60**(3), 510–539 (2000)
8. George, A., Liu, J.W.H.: Computer solution of large sparse positive definite systems. Prentice-Hall Series in Computational Mathematics, Prentice-Hall Inc. Englewood Cliffs (1981)

9. Gupta, A.: Improved bandwidth approximation for trees and chordal graphs. J. Algorithms **40**(1), 24–36 (2001)
10. Krauthgamer, R., Lee, J.R., Mendel, M., Naor, A.: Measured descent: A new embedding method for finite metrics. Geom. Funct. Anal. **15**(4), 839–858 (2005)
11. Krauthgamer, R., Linial, N., Magen, A.: Metric embeddings–beyond one-dimensional distortion. Discrete Comput. Geom. **31**(3), 339–356 (2004)
12. Lee, J.R.: Volume distortion for subsets of Euclidean spaces. In: Proceedings of the 22nd Annual Symposium on Computational Geometry, ACM, Sedona, AZ 2006, pp. 207–216.
13. Linial, N.: Finite metric-spaces—combinatorics, geometry and algorithms. In: Proceedings of the International Congress of Mathematicians, vol. III, Beijing, 2002, pp. 573–586. Higher Ed. Press, Beijing (2002)
14. Linial, N., London, E., Rabinovich, Y.: The geometry of graphs and some of its algorithmic applications. Combinatorica **15**(2), 215–245 (1995)
15. Papadimitriou, C.H.: The NP-completeness of the bandwidth minimization problem. Computing **16**(3), 263–270 (1976)
16. Rao, S.: Small distortion and volume preserving embeddings for planar and Euclidean metrics. In: Proceedings of the 15th Annual Symposium on Computational Geometry, pp. 300–306. ACM, New York (1999)
17. Strang, G.: Linear algebra and its applications, 2nd edn. Academic Press [Harcourt Brace Jovanovich Publishers], New York (1980)
18. Unger, W.: The complexity of the approximation of the bandwidth problem. In: 39th Annual Symposium on Foundations of Computer Science, IEEE, 8–11 Oct 1998, pp. 82–91.
19. Vempala, S.: Random projection: A new approach to VLSI layout. In: 39th Annual Symposium on Foundations of Computer Science, IEEE, 8–11 Oct 1998, pp. 389–398.

Graph Coloring

1994; Karger, Motwani, Sudan
1998; Karger, Motwani, Sudan

Michael Langberg
Department of Computer Science,
The Open University of Israel,
Raanana, Israel

Keywords and Synonyms

Clique cover

Problem Definition

An independent set in an undirected graph $G = (V, E)$ is a set of vertices that induce a subgraph which does not contain any edges. The size of the maximum independent set in G is denoted by $\alpha(G)$. For an integer k, a k-coloring of G is a function $\sigma : V \to [1 \ldots k]$ which assigns colors to the vertices of G. A valid k-coloring of G is a coloring in which each color class is an independent set. The chromatic number $\chi(G)$ of G is the smallest k for which there exists a valid k-coloring of G. Finding $\chi(G)$ is a fundamental NP-hard problem. Hence, when limited to polynomial time algorithms, one turns to the question of estimating the value of $\chi(G)$ or to the closely related problem of *approximate coloring*.

Problem 1 (Approximate coloring)
Input: *Undirected graph* $G = (V, E)$.
Output: *A valid coloring of* G *with* $r \cdot \chi(G)$ *colors, for some approximation ratio* $r \geq 1$.
Objective: *Minimize* r.

Let G be a graph of size n. The approximate coloring of G can be solved efficiently within an approximation ratio of $r = O\left(\frac{n(\log\log n)^2}{\log^3 n}\right)$ [12]. This holds also for the approximation of $\alpha(G)$ [8]. These results may seem rather weak, however it is NP-hard to approximate $\alpha(G)$ and $\chi(G)$ within a ratio of $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$ [9,14,21]. Under stronger complexity assumptions, there is some constant $0 < \delta < 1$ such that neither problem can be approximated within a ratio of $n/2^{\log^\delta n}$ [17,21]. This chapter will concentrate on the problem of coloring graphs G for which $\chi(G)$ is *small*. As will be seen, in this case the approximation ratio achievable significantly improves.

Vector Coloring of Graphs

The algorithms achieving the best ratios for approximate coloring when $\chi(G)$ is small [1,3,13,15] are all based on the idea of *vector coloring*, introduced by Karger, Motwani, and Sudan [15][1]

Definition 1 A vector k-coloring of a graph is an assignment of unit vectors to its vertices, such that for every edge, the inner product of the vectors assigned to its endpoints is at most (in the sense that it can only be more negative) $-1/(k-1)$.

The *vector chromatic number* $\overrightarrow{\chi}(G)$ of G is the smallest k for which there exists a vector k-coloring of G. The vector chromatic number can be formulated as follows:

$$\begin{array}{lll} \overrightarrow{\chi}(G) \quad \text{Minimize} & k & \\ \text{subject to :} & \langle v_i, v_j \rangle \leq -\dfrac{1}{k-1} & \forall (i,j) \in E \\ & \langle v_i, v_i \rangle = 1 & \forall i \in V. \end{array}$$

Here, assume that $V = [1, \ldots, n]$ and that the vectors $\{v_i\}_{i=1}^n$ are in R^n. Every k-colorable graph is also vector

[1] Vector coloring as presented in [15] is closely related to the Lovász θ function [19]. This connection will be discussed shortly.

k-colorable. This can be seen by identifying each color class with one vertex of a perfect $(k-1)$-dimensional simplex centered at the origin. Moreover, unlike the chromatic number, a vector k-coloring (when it exists) can be found in polynomial time using semidefinite programming (up to an arbitrarily small error in the inner products).

Claim 1 (Complexity of vector coloring [15]) *Let $\varepsilon > 0$. If a graph G has a vector k-coloring then a vector $(k+\varepsilon)$-coloring of the graph can be constructed in time polynomial in n and $\log(1/\varepsilon)$.*

One can strengthen Definition 1 to obtain a different notion of vector coloring and the vector chromatic number.

$$\begin{aligned} \vec{\chi}_2(G) \quad & \text{Minimize} \quad k \\ & \text{subject to:} \quad \langle v_i, v_j\rangle = -\frac{1}{k-1} \quad \forall (i,j) \in E \\ & \qquad\qquad\quad \langle v_i, v_i\rangle = 1 \qquad\quad \forall i \in V \end{aligned}$$

$$\begin{aligned} \vec{\chi}_3(G) \quad & \text{Minimize} \quad k \\ & \text{subject to:} \quad \langle v_i, v_j\rangle = -\frac{1}{k-1} \quad \forall (i,j) \in E \\ & \qquad\qquad\quad \langle v_i, v_j\rangle \geq -\frac{1}{k-1} \quad \forall i,j \in V \\ & \qquad\qquad\quad \langle v_i, v_i\rangle = 1 \qquad\quad \forall i \in V. \end{aligned}$$

The function $\vec{\chi}_2(G)$ is referred to as the *strict* vector chromatic number of G and is equal to the Lovász θ function on $\bar{G}$ [15,19], where $\bar{G}$ is the *complement* graph of G. The function $\vec{\chi}_3(G)$ is referred to as the *strong* vector chromatic number. An analog to Claim 1 holds for both $\vec{\chi}_2(G)$ and $\vec{\chi}_3(G)$. Let $\omega(G)$ denote the size of the maximum clique in G, it holds that: $\omega(G) \leq \vec{\chi}(G) \leq \vec{\chi}_2(G) \leq \vec{\chi}_3(G) \leq \chi(G)$.

Key Results

In what follows, assume that G has n vertices and maximal degree Δ. The $\tilde{O}(\cdot)$ and $\tilde{\Omega}(\cdot)$ notation are used to suppress polylogarithmic factors. The key result of Karger, Motwani, and Sudan [15] is stated below:

Theorem 1 ([15]) *If $\vec{\chi}(G) = k$ then G can be colored in polynomial time using $\min\{\tilde{O}(\Delta^{1-2/k}), \tilde{O}(n^{1-3/(k+1)})\}$ colors.*

As mentioned above, the use of vector coloring in the context of approximate coloring was initiated in [15]. Roughly speaking, once given a vector coloring of G, the heart of the algorithm in [15] finds a large independent set in G. In a nutshell, this independent set corresponds to a set of vectors in the vector coloring which are *close* to one another (and thus by definition cannot share an edge). Combining this with the ideas of Wigderson [20] mentioned below yields Theorem 1.

A description of related work is given below. The first two theorems below appeared prior to the work of Karger, Motwani, and Sudan [15].

Theorem 2 ([20]) *If $\chi(G) = k$ then G can be colored in polynomial time using $O(kn^{1-1/(k-1)})$ colors.*

Theorem 3 ([2]) *If $\chi(G) = 3$ then G can be colored in polynomial time using $\tilde{O}(n^{3/8})$ colors. If $\chi(G) = k \geq 4$ then G can be colored in polynomial time using at most $\tilde{O}(n^{1-1/(k-3/2)})$ colors.*

Combining the techniques of [15] and [2] the following results were obtained for graphs G with $\chi(G) = 3, 4$ (these results were also extended for higher values of $\chi(G)$).

Theorem 4 ([3]) *If $\chi(G) = 3$ then G can be colored in polynomial time using $\tilde{O}(n^{3/14})$ colors.*

Theorem 5 ([13]) *If $\chi(G) = 4$ then G can be colored in polynomial time using $\tilde{O}(n^{7/19})$ colors.*

The currently best-known result for coloring a 3-colorable graph is presented in [1]. In their algorithm, [1] use the strict vector coloring relaxation (*i. e.* $\vec{\chi}_2$) enhanced with certain *odd cycle* constraints.

Theorem 6 ([1]) *If $\chi(G) = 3$ then G can be colored in polynomial time using $O(n^{0.2111})$ colors.*

To put the above theorems in perspective, it is NP-hard to color a 3-colorable graph G with 4 colors [11,16] and a k-colorable graph (for sufficiently large k) with $k^{(\log k)/25}$ colors [17]. Under stronger complexity assumptions (related to the Unique Games Conjecture [18]) for any constant k it is hard to color a k-colorable graph with any constant number of colors [6]. The wide gap between these hardness results and the approximation ratios presented in this section has been a major initiative in the study of approximate coloring.

Finally, the limitations of vector coloring are addressed. Namely, are there graphs for which $\vec{\chi}(G)$ is a poor estimate of $\chi(G)$? One would expect the answer to be "yes" as estimating $\chi(G)$ beyond a factor of $n^{1-\varepsilon}$ is a hard problem. As will be stated below, this is indeed the case (even when $\vec{\chi}(G)$ is small). Some of the results that follow are stated in terms of the maximum independent set $\alpha(G)$ in G. As $\chi(G) \geq n/\alpha(G)$, these results im-

ply a lower bound on $\chi(G)$. Theorem 1 (i) states that the original analysis of [15] is essentially tight. Theorem 1 (ii) presents bounds for the case of $\overrightarrow{\chi}(G) = 3$. Theorem 1 (iii) and Theorem 2 present graphs G in which there is an extremely large gap between $\chi(G)$ and the relaxations $\overrightarrow{\chi}(G)$ and $\overrightarrow{\chi}_2(G)$.

Theorem 7 ([10]) *(i) For every constant $\varepsilon > 0$ and constant $k > 2$, there are infinitely many graphs G with $\overrightarrow{\chi}(G) = k$ and $\alpha(G) \le n/\Delta^{1-2/k-\varepsilon}$ (here $\Delta > n^{\delta}$ for some constant $\delta > 0$). (ii) There are infinitely many graphs G with $\overrightarrow{\chi}(G) = 3$ and $\alpha(G) \le n^{0.843}$. (iii) For some constant c, there are infinitely many graphs G with $\overrightarrow{\chi}(G) = O(\log n/\log\log n)$ and $\alpha(G) \le \log^c n$.*

Theorem 8 ([7]) *For some constant c, there are infinitely many graphs G with $\overrightarrow{\chi}_2(G) \le 2^{\sqrt{\log n}}$ and $\chi(G) \ge n/2^{c\sqrt{\log n}}$.*

Vector colorings, including the Lovász θ function and its variants, have been extensively studied in the context of approximation algorithms for problems other than Problem 1. These include approximating $\alpha(G)$, approximating the Minimum Vertex Cover problem, and combinatorial optimization in the context of random graphs.

Applications

Besides its theoretical significance, graph coloring has several concrete applications that fall under the model of *conflict free* allocation of resources (see for example [4,5]).

Open Problems

By far the major open problem in the context of approximate coloring addresses the wide gap between what is known to be hard and what can be obtained in polynomial time. The case of constant $\chi(G)$ is especially intriguing, as the best-known upper bounds (on the approximation ratio) are polynomial while the lower bounds are of constant nature. Regarding the vector coloring paradigm, a majority of the results stated in Sect. "Key Results" use the weakest form of vector coloring $\overrightarrow{\chi}(G)$ in their proof, while stronger relaxations such as $\overrightarrow{\chi}_2(G)$ and $\overrightarrow{\chi}_3(G)$ may also be considered. It would be very interesting to improve upon the algorithmic results stated above using stronger relaxations, as would a matching analysis of the limitations of these relaxations.

Cross References

- Channel Assignment and Routing in Multi-Radio Wireless Mesh Networks
- Max Cut
- Randomized Rounding
- Sparsest Cut

Recommended Reading

1. Arora, S., Chlamtac, E., Charikar, M.: New approximation guarantee for chromatic number. In: Proceedings of the 38th annual ACM Symposium on Theory of Computing (2006) pp. 215–224.
2. Blum, A.: New approximations for graph coloring. J. ACM **41**(3), 470–516 (1994)
3. Blum, A., Karger, D.: An $\tilde{O}(n^{3/14})$-coloring for 3-colorable graphs. Inf. Process. Lett. **61**(6), 49–53 (1997)
4. Chaitin, G.J.: Register allocation & spilling via graph coloring. In: Proceedings of the 1982 SIGPLAN Symposium on Compiler Construction (1982) pp. 98–105.
5. Chaitin, G.J., Auslander, M.A., Chandra, A.K., Cocke, J., Hopkins, M.E., Markstein, P.W.: Register allocation via coloring. Comp. Lang. **6**, 47–57 (1981)
6. Dinur, I., Mossel, E., Regev, O.: Conditional hardness for approximate coloring. In: Proceedings of the 38th annual ACM Symposium on Theory of Computing (2006) pp. 344–353.
7. Feige, U.: Randomized graph products, chromatic numbers, and the Lovász theta function. Combinatorica **17**(1), 79–90 (1997)
8. Feige, U.: Approximating maximum clique by removing subgraphs. SIAM J. Discret. Math. **18**(2), 219–225 (2004)
9. Feige, U., Kilian, J.: Zero knowledge and the chromatic number. J. Comput. Syst. Sci. **57**, 187–199 (1998)
10. Feige, U., Langberg, M., Schechtman, G.: Graphs with tiny vector chromatic numbers and huge chromatic numbers. SIAM J. Comput. **33**(6), 1338–1368 (2004)
11. Guruswami, V., Khanna, S.: On the hardness of 4-coloring a 3-colorable graph. In: Proceedings of the 15th annual IEEE Conference on Computational Complexity (2000) pp. 188–197.
12. Halldorsson, M.: A still better performance guarantee for approximate graph coloring. Inf. Process. Lett. **45**, 19–23 (1993)
13. Halperin, E., Nathaniel, R., Zwick, U.: Coloring k-colorable graphs using smaller palettes. J. Algorithms **45**, 72–90 (2002)
14. Håstad, J.: Clique is hard to approximate within $n^{1-\varepsilon}$. Acta Math. **182**(1), 105–142 (1999)
15. Karger, D., Motwani, R., Sudan, M.: Approximate graph coloring by semidefinite programming. J. ACM **45**(2), 246–265 (1998)
16. Khanna, S., Linial, N., Safra, S.: On the hardness of approximating the chromatic number. Combinatorica **20**, 393–415 (2000)
17. Khot, S.: Improved inapproximability results for max clique, chromatic number and approximate graph coloring. In: Proceedings of the 42nd annual IEEE Symposium on Foundations of Computer Science (2001) pp. 600–609.
18. Khot, S.: On the power of unique 2-prover 1-round games. In: Proceedings of the 34th annual ACM symposium on Theory of Computing (2002) pp. 767–775.
19. Lovász, L.: On the Shannon capacity of a graph. IEEE Trans. Inf. Theor. **25**, 2–13 (1979)
20. Wigderson, A.: Improving the performance guarantee for approximate graph coloring. J. ACM **30**(4), 729–735 (1983)

21. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. In: Proceedings of the 38th annual ACM symposium on Theory of Computing (2006) pp. 681–690.

Graph Connectivity

1994; Khuller, Vishkin

Samir Khuller[1], Balaji Raghavachari[2]
[1] Computer Science Department, University of Maryland, College Park, MD, USA
[2] Computer Science Department, University of Texas at Dallas, Richardson, TX, USA

Keywords and Synonyms

Highly connected subgraphs; Sparse certificates

Problem Definition

An undirected graph is said to be k-connected (specifically, k-vertex-connected) if the removal of any set of $k-1$ or fewer vertices (with their incident edges) does not disconnect G. Analogously, it is k-edge-connected if the removal of any set of $k-1$ edges does not disconnect G. Menger's theorem states that a k-vertex-connected graph has at least k openly vertex-disjoint paths connecting every pair of vertices. For k-edge-connected graphs there are k edge-disjoint paths connecting every pair of vertices. The connectivity of a graph is the largest value of k for which it is k-connected. Finding the connectivity of a graph, and finding k disjoint paths between a given pair of vertices can be found using algorithms for maximum flow. An edge is said to be *critical* in a k-connected graph if upon its removal the graph is no longer k-connected.

The problem of finding a minimum-cardinality k-vertex-connected (k-edge-connected) subgraph that spans all vertices of a given graph is called k-VCSS (k-ECSS) and is known to be nondeterministic polynomial-time hard for $k \geq 2$. We review some results in finding approximately minimum solutions to k-VCSS and k-ECSS. We focus primarily on simple graphs. A simple approximation algorithm is one that considers the edges in some order and removes edges that are not critical. It thus outputs a k-connected subgraph in which all edges are critical and it can be shown that it is a 2-approximation algorithm (that outputs a solution with at most kn edges in an n-vertex graph, and since each vertex has to have degree at least k, we can claim that $kn/2$ edges are necessary).

Approximation algorithms that do better than the simple algorithm mentioned above can be classified into two categories: depth first search (DFS) based, and matching based.

Key Results

Lower Bounds for k-Connected Spanning Subgraphs

Each node of a k-connected graph has at least k edges incident to it. Therefore, the sum of the degrees of all its nodes is at least kn, where n is the number of its nodes. Since each edge is counted twice in this degree-sum, the cardinality of its edges is at least $kn/2$. This is called the *degree lower bound*. Expanding on this idea yields a stronger lower bound on the cardinality of a k-connected spanning subgraph of a given graph. Let D_k be a subgraph in which the degree of each node is at least k. Unlike a k-connected subgraph, D_k has no connectivity constraints. The counting argument above shows that any D_k has at least $kn/2$ edges. A minimum cardinality D_k can be computed in polynomial time by reducing the problem to matching, and it is called the *matching lower bound*.

DFS-Based Approaches

The following natural algorithm finds a 3/2 approximation for 2-ECSS. Root the tree at some node r and run DFS. All edges of the graph are now either tree edges or back edges. Process the DFS tree in postorder. For each subtree, if the removal of the edge from its root to its parent separates the graph into two components, then add a farthest-back edge from this subtree, whose other end is closest to r. It can be shown that the number of back edges added by the algorithm is at most half the size of *Opt*.

This algorithm has been generalized to solve the 2-VCSS problem with the same approximation ratio, by adding carefully chosen back edges that allow the deletion of tree edges. Wherever it is unable to delete a tree edge, it adds a vertex to an independent set I. In the final analysis, the number of edges used is less than $n + |I|$. Since *Opt* is at least $\max(n, 2|I|)$, it obtains a 3/2-approximation ratio.

The algorithm can also be extended to the k-ECSS problem by repeating these ideas $k/2$ times, augmenting the connectivity by 2 in each round. It has been shown that this algorithm achieves a performance of about 1.61.

Matching-Based Approaches

Several approximation algorithms for k-ECSS and k-VCSS problems have used a minimum cardinality D_k as a starting solution, which is then augmented with additional edges to satisfy the connectivity constraints. This approach yields better ratios than the DFS-based approaches.

$1 + \frac{1}{k}$ Algorithm for k-VCSS Find a minimum cardinality D_{k-1}. Add just enough additional edges to it to make the subgraph k-connected. In this step, it is ensured that the edges added are critical. It is known by a theorem of Mader that in a k-connected graph, a cycle of critical edges contains at least one node of degree k. Since the edges added by the algorithm in the second step are all critical, there can be no cycle induced by these edges because the degree of all the nodes on such a cycle would be at least $k + 1$. Therefore, at most $n - 1$ edges are added in this step. The number of edges added in the first step, in the minimum D_{k-1} is at most $Opt - n/2$. The total number of edges in the solution thus computed is at most $(1 + 1/k)$ times the number of edges in an optimal k-VCSS.

$1 + \frac{2}{k+1}$ Algorithm for k-ECSS Mader's theorem about cycles induced by critical edges is valid only for vertex connectivity and not edge connectivity, Therefore, a different algorithm is proposed for k-ECSS in graphs that are k-edge-connected, but not k-connected. This algorithm finds a minimum cardinality D_k and augments it with a minimal set of edges to make the subgraph k-edge-connected. The number of edges added in the last step is at most $\frac{k}{k+1}(n-1)$. Since the number of edges added in the first step is at most Opt, the total number of edges is at most $(1 + \frac{2}{k+1})Opt$.

Better Algorithms for Small k For $k \in \{2, 3\}$, better algorithms have been obtained by implementing the abovementioned algorithms carefully, deleting unnecessary edges, and by getting better lower bounds. For $k = 2$, a 4/3 approximation can be obtained by generating a path/cycle cover from a minimum cardinality D_2 and 2-connecting them one at a time to a "core" component. Small cycles/paths allow an edge to be deleted when they are 2-connected to the core, which allows a simple amortized analysis. This method also generalizes to the 3-ECSS problem, yielding a 4/3 ratio.

Hybrid approaches have been proposed which use the path/cycle cover to generate a specific DFS tree of the original graph and then 2-connect the tree, trying to delete edges wherever possible. The best ratios achieved using this approach are 5/4 for 2-ECSS, 9/7 for 2-VCSS, and 5/4 for 2-VCSS in 3-connected graphs.

Applications

Network design is one of the main application areas for this work. This involves the construction of low-cost highly connected networks.

Recommended Reading

For additional information on DFS, matchings and path/cycle covers, see [3]. Fast 2-approximation algorithms for k-ECSS and k-VCSS were studied by Nagamochi and Ibaraki [13]. DFS-based algorithms for 2-connectivity were introduced by Khuller and Vishkin [11]. They obtained 3/2 for 2-ECSS, 5/3 for 2-VCSS, and 2 for weighted k-ECSS. The ratio for 2-VCSS was improved to 3/2 by Garg et al. [6], 4/3 by Vempala and Vetta [14], and 9/7 by Gubbala and Raghavachari [7]. Khuller and Raghavachari [10] gave an algorithm for k-ECSS, which was later improved by Gabow [4], who showed that the algorithm obtains a ratio of about 1.61. Cheriyan et al. [2] studied the k-VCSS problem with edge weights and designed an $O(\log k)$ approximation algorithm in graphs with at least $6k^2$ vertices.

The matching-based algorithms were introduced by Cheriyan and Thurimella [1]. They proposed algorithms with ratios of $1 + \frac{1}{k}$ for k-VCSS, $1 + \frac{2}{k+1}$ for k-ECSS, $1 + \frac{1}{k}$ for k-VCSS in directed graphs, and $1 + \frac{4}{\sqrt{k}}$ for k-ECSS in directed graphs. Vempala and Vetta [14] obtained a ratio of 4/3 for 2-VCSS. The ratios were further improved by Krysta and Kumar [12], who introduced the hybrid approach, which was used to derive a 5/4 algorithm by Jothi et al. [9]. A 3/2-approximation algorithm for 3-ECSS has been proposed by Gabow [5] that works on multigraphs, whereas the earlier algorithm of Cheriyan and Thurimella gets the same ratio in simple graphs only. This ratio has been improved to 4/3 by Gubbala and Raghavachari [8].

1. Cheriyan, J., Thurimella, R.: Approximating minimum-size k-connected spanning subgraphs via matching. SIAM J. Comput. **30**(2), 528–560 (2000)
2. Cheriyan, J., Vempala, S., Vetta, A.: An approximation algorithm for the minimum-cost k-vertex connected subgraph. SIAM J. Comput. **32**(4), 1050–1055 (2003)
3. Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A.: Combinatorial optimization. Wiley, New York (1998)
4. Gabow, H.N.: Better performance bounds for finding the smallest k-edge connected spanning subgraph of a multigraph. In: SODA, 2003, pp. 460–469
5. Gabow, H.N.: An ear decomposition approach to approximating the smallest 3-edge connected spanning subgraph of a multigraph. SIAM J. Discret. Math. **18**(1), 41–70 (2004)
6. Garg, N., Vempala, S., Singla, A.: Improved approximation algorithms for biconnected subgraphs via better lower bounding techniques. In: SODA, 1993, pp. 103–111
7. Gubbala, P., Raghavachari, B.: Approximation algorithms for the minimum cardinality two-connected spanning subgraph problem. In: Jünger, M., Kaibel, V. (eds.) IPCO. Lecture Notes in Computer Science, vol. 3509, pp. 422–436. Springer, Berlin (2005)
8. Gubbala, P., Raghavachari, B.: A 4/3-approximation algorithm for minimum 3-edge-connectivity. In: Proceedings of the

Workshop on Algorithms and Data Structures (WADS) August 2007, pp. 39–51. Halifax (2007)
9. Jothi, R., Raghavachari, B., Varadarajan, S.: A 5/4-approximation algorithm for minimum 2-edge-connectivity. In: SODA, 2003, pp. 725–734
10. Khuller, S., Raghavachari, B.: Improved approximation algorithms for uniform connectivity problems. J. Algorithms **21**(2), 434–450 (1996)
11. Khuller, S., Vishkin, U.: Biconnectivity approximations and graph carvings. J. ACM **41**(2), 214–235 (1994)
12. Krysta, P., Kumar, V.S.A.: Approximation algorithms for minimum size 2-connectivity problems. In: Ferreira, A., Reichel, H. (eds.) STACS. Lecture Notes in Computer Science, vol. 2010, pp. 431–442. Springer, Berlin (2001)
13. Nagamochi, H., Ibaraki, T.: A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. Algorithmica **7**(5–6), 583–596 (1992)
14. Vempala, S., Vetta, A.: Factor 4/3 approximations for minimum 2-connected subgraphs. In: Jansen, K., Khuller, S. (eds.) APPROX. Lecture Notes in Computer Science, vol. 1913, pp. 262–273. Springer, Berlin (2000)

Graph Isomorphism

1980; McKay

Brendan D. McKay
Department of Computer Science, Australian National University, Canberra, ACT, Australia

Keywords and Synonyms

Graph matching; Symmetry group

Problem Definition

The problem of determining isomorphism of two combinatorial structures is a ubiquitous one, with applications in many areas. The paradigm case of concern in this chapter is isomorphism of two graphs. In this case, an isomorphism consists of a bijection between the vertex sets of the graphs which induces a bijection between the edge sets of the graphs. One can also take the second graph to be a copy of the first, so that isomorphisms map a graph onto themselves. Such isomorphisms are called *automorphisms* or, less formally, *symmetries*. The set of all automorphisms forms a group under function composition called the *automorphism group*. Computing the automorphism group is a problem rather similar to that of determining isomorphisms.

Graph isomorphism is closely related to many other types of isomorphism of combinatorial structures. In the section entitled "Applications", several examples are given.

Formal Description

A *graph* is a pair $G = (V, E)$ of finite sets, with E being a set of 2-tuples (v, w) of elements of V. The elements of V are called *vertices* (also *points*, *nodes*), while the elements of E are called *directed edges* (also *arcs*). A complementary pair $(v, w), (w, v)$ of directed edges ($v \neq w$) will be called an *undirected edge* and denoted $\{v, w\}$. A directed edge of the form (v, v) will also be considered an undirected edge, called a *loop* (also *self-loop*). The word "edges" without qualification will indicate undirected edges, directed edges, or both.

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, an *isomorphism* from G_1 to G_2 is a bijection from V_1 to V_2 such that the induced action on E_1 is a bijection onto E_2. If $G_1 = G_2$, then the isomorphism is an *automorphism* of G_1. The set of all *automorphisms* of G_1 is a group under function composition, called the *automorphism group* of G_1, and denoted $\mathrm{Aut}(G_1)$.

In Fig. 1 two isomorphic graphs are shown, together with an isomorphism between them and the automorphism group of the first.

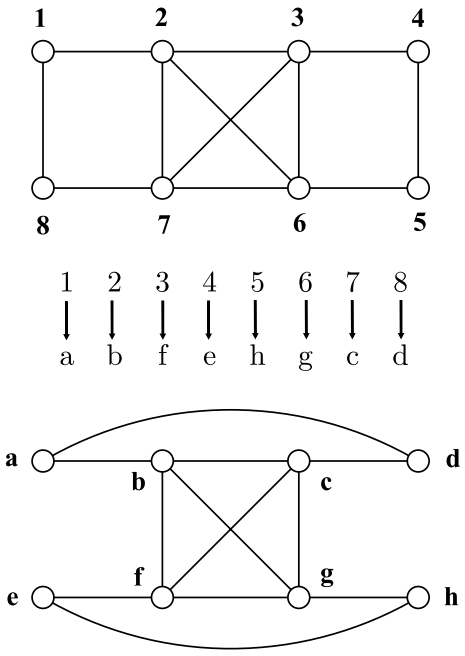
Canonical Labeling

Practical applications of graph isomorphism testing do not usually involve individual pairs of graphs. More commonly, one must decide whether a certain graph is isomorphic to any of a collection of graphs (the database lookup problem) or one has a collection of graphs and needs to identify the isomorphism classes in it (the graph sorting problem). Such applications are not well served by an algorithm that can only test graphs in pairs.

An alternative is a *canonical labeling* algorithm. The essential idea is that in each isomorphism class there is a unique, *canonical* graph which the algorithm can find, given as input any graph in the isomorphism class. The canonical graph might be, for example, the least graph in the isomorphism class according to some ordering (such as lexicographic) of the graphs in the class. Practical algorithms usually compute a canonical form designed for efficiency rather than ease of description.

Key Results

The graph isomorphism problem plays a key role in modern complexity theory. It is not known to be solvable in polynomial time, nor to be NP-complete, nor is it known to be in the class co-NP. See [3,8] for details. Polynomial-time algorithms are known for many special classes, notably graphs with bounded genus, bounded degree, bounded tree-width, and bounded eigenvalue multi-



Automorphisms:

(1)
(3 6)(4 5)
(1 4)(2 3)(5 8)(6 7)
(1 5 8 4)(2 6 7 3)
(1 8)(2 7)(3 6)(4 5)
(1 8)(2 7)
(1 5)(4 8)(2 6)(3 7)
(1 4 8 5)(2 3 7 6)

Graph Isomorphism, Figure 1
Example of an isomorphism and an automorphism group

plicity. The fastest theoretical algorithm for general graphs requires $\exp(n^{1/2+o(1)})$ time [1], but it is not known to be practical.

In this entry, the focus is on the program `nauty`, which is generally regarded as the most successful for practical use. McKay wrote the first version of `nauty` in 1976 and described its method of operation in [5]. It is known [7] to have exponential worst-case time, but in practice the worst case is rarely encountered.

The input to `nauty` is a graph with colored vertices. Two outputs are produced. The first is a set of generators for the color-preserving automorphism group. Though it is rarely necessary, the full group can also be developed element by element. The second, optional, output is a canonical graph. The canonical graph has the following property: two input graphs with the same number of vertices of each color have the same canonical graph if and only if they are isomorphic by a color-preserving isomorphism.

Two graph data structures are supported: a packed adjacency matrix suitable for small dense graphs and a linked list suitable for large sparse graphs.

Applications

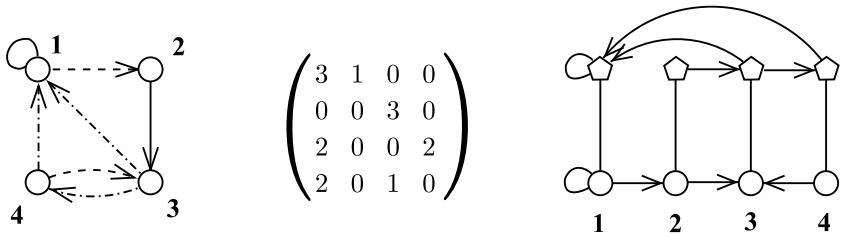
As mentioned, `nauty` can handle graphs with colored vertices. In this section, it is described how several other types of isomorphism problems can be solved by mapping them onto a problem for vertex-colored graphs.

Isomorphism of Edge-Colored Graphs

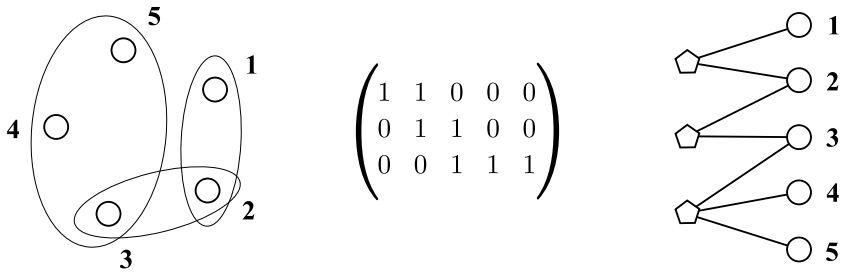
An isomorphism of two graphs, each with both vertices and edges colored, is defined in the obvious way. An example of such a graph appears at the left of Fig. 2.

In the center of the figure the colors are identified with the integers 1, 2, 3. At the right of the figure an equivalent vertex-colored graph is shown. In this case there are two layers, each with its own color. Edges of color 1 are represented as an edge in the first (lowest) layer, edges of color 2 are represented as an edge in the second layer, and edges of color 3 are represented as edges in both layers. It is now easy to see that the automorphism group of the new graph (specifically, its action on the first layer) is the automorphism group of the original graph. Moreover, the order in which a canonical labeling of the new graph labels the vertices of the first layer can be taken to be a canonical labeling of the original graph.

More generally, if the edge colors are integers in $\{1, 2, \ldots, 2^d - 1\}$, there are d layers, and the binary expansion of each color number dictates which layers contain edges. The vertical threads (each corresponding to one vertex of the original graph) can be connected using either paths or cliques. If the original graph has n vertices and k colors, the new graph has $O(n \log k)$ vertices. This can be improved to $O(n\sqrt{\log k})$ vertices by also using edges that are not horizontal.



Graph Isomorphism, Figure 2
Graph isomorphism with colored edges



Graph Isomorphism, Figure 3
Hypergraph/design isomorphism as graph isomorphism

Isomorphism of Hypergraphs and Designs

A *hypergraph* is similar to an undirected graph except that the edges can be vertex sets of any size, not just of size 2. Such a structure is also called a *design*.

On the left of Fig. 3 there is a hypergraph with five vertices, two edges of size 2, and one edge of size 3. On the right is an equivalent vertex-colored graph. The vertices on the left, colored with one color, represent the hypergraph edges, while the edges on the right, colored with a different color, represent the hypergraph vertices. The edges of the graph indicate the hypergraph incidence (containment) relationship.

The edge-vertex incidence matrix appears in the center of the figure. This can be any binary matrix at all, which correctly suggests that the problem under consideration is just that of determining the 0-1 matrix equivalence under independent permutation of the rows and columns. By combining this idea with the previous construction, such an equivalence relation on the set of matrices with arbitrary entries can be handled.

Other Examples

For several applications to equivalence operations such as isotopy, important for Latin squares and quasigroups, see [6].

Another important type of equivalence relates matrices over $\{-1, +1\}$. As well as permuting rows and columns, it allows multiplication of rows and columns by -1. A method of converting this *Hadamard equivalence* problem to a graph isomorphism problem is given in [4].

Experimental Results

`Nauty` gives a choice of sparse and dense data structures, and some special code for difficult graph classes. For the following timing examples, the best of the various options are used for a single CPU of a 2.4 GHz Intel Core-duo processor.

1. Random graph with 10,000 vertices, $p = \frac{1}{2}$: 0.014 s for group only, 0.4 s for canonical labeling as well.
2. Random cubic graph with 100,000 vertices: 8 s.
3. 1-skeleton of 20-dimensional cube (1,048,576 vertices, group size 2.5×10^{24}): 92 s.
4. 3-dimensional mesh of size 50 (125,000 vertices): 0.7 s.
5. 1027-vertex strongly regular graph from random Steiner triple system: 0.6 s.

Examples of more difficult graphs can be found in the `nauty` documentation.

URL to Code

The source code of `nauty` is available at http://cs.anu.edu.au/~bdm/nauty/. Another implementation of the automorphism group portion of nauty, highly optimized for large sparse graphs, is available as `saucy` [2]. `Nauty` is

also incorporated into a number of general-purpose packages, including GAP, Magma, and MuPad.

Cross References

- ▶ Abelian Hidden Subgroup Problem
- ▶ Parameterized Algorithms for Drawing Graphs

Recommended Reading

1. Babai, L., Luks, E.: Canonical labelling of graphs. In: Proceedings of the 15th Annual ACM Symposium on Theory of Computing, pp. 171–183. ACM, New York (1983)
2. Darga, P.T., Liffiton, M.H., Sakallah, K.A., Markov, I.L.: Exploiting Structure in Symmetry Generation for CNF. In: Proceedings of the 41st Design Automation Conference, 2004, pp. 530–534. Source code at http://vlsicad.eecs.umich.edu/BK/SAUCY/
3. Köbler, J., Schöning, U., Torán, J.: The Graph Isomorphism Problem: its structural complexity. Birkhäuser, Boston (1993)
4. McKay, B.D.: Hadamard equivalence via graph isomorphism. Discret. Math. **27**, 213–214 (1979)
5. McKay, B.D.: Practical graph isomorphism. Congr. Numer. **30**, 45–87 (1981)
6. McKay, B.D., Meynert, A., Myrvold, W.: Small Latin squares, quasigroups and loops. J. Comb. Des. **15**, 98–119 (2007)
7. Miyazaki, T.: The complexity of McKay's canonical labelling algorithm. In: Groups and Computation, II. DIMACS Ser. Discret. Math. Theor. Comput. Sci., vol. 28, pp. 239–256. American Mathematical Society, Providence, RI (1997)
8. Toran, J.: On the hardness of graph isomorphism. SIAM J. Comput. **33**, 1093–1108 (2004)

Graphs

- ▶ Algorithms for Spanners in Weighted Graphs
- ▶ Minimum Bisection
- ▶ Mobile Agents and Exploration

Greedy Approximation Algorithms

2004; Ruan, Du, Jia, Wu, Li, Ko

Feng Wang[1], Weili Wu[2]

[1] Department of Mathmatical Science and Applied Computing, Arizona State University, Phoenix, AZ, USA

[2] Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA

Keywords and Synonyms

Technique for analysis of greedy approximation

Problem Definition

Consider a graph $G = (V, E)$. A subset C of V is called a *dominating set* if every vertex is either in C or adjacent to a vertex in C. If, furthermore, the subgraph induced by C is connected, then C is called a *connected dominating set*.

Given a connected graph G, find a connecting dominating set of minimum cardinality. This problem is denoted by MCDS and is NP-hard. Its optimal solution is called a *minimum connected dominating set*. The following is a greedy approximation with potential function f.

Greedy Algorithm A:
$C \leftarrow \emptyset$;
while $f(C) > 2$ **do**
choose a vertex x to maximize $f(C) - f(C \cup \{x\})$ and $C \leftarrow C \cup \{x\}$; output C.

Here, f is defined as $f(C) = p(C) + q(C)$ where $p(C)$ is the number of connected components of subgraph induced by C and $q(C)$ is the number of connected components of subgraph with vertex set V and edge set $\{(u, v) \in E \mid u \in C \text{ or } v \in C\}$. f has an important property that C is a connected dominating set if and only if $f(C) = 2$.

If C is a connected dominating set, then $p(C) = q(C) = 1$ and hence $f(C) = 2$. Conversely, suppose $f(C \cup \{x\}) = 2$. Since $p(C) \geq 1$ and $q(C) \geq 1$, one has $p(C) = q(C) = 1$ which implies that C is a connected dominating set. f has another property, for G with at least three vertices, that if $f(C) > 2$, then there exists $x \in V$ such that $f(C) - f(C \cup \{x\}) > 0$. In fact, for $C = \emptyset$, since G is a connected graph with at least three vertices, there must exist a vertex x with degree at least two and for such a vertex x, $f(C \cup \{x\}) < f(C)$. For $C \neq \emptyset$, consider a connected component of the subgraph induced by C. Let B denote its vertex set which is a subset of C. For every vertex y adjacent to B, if y is adjacent to a vertex not adjacent to B and not in C, then $p(C \cup \{y\}) < p(C)$ and $q(C \cup \{y\}) \leq q(C)$; if y is adjacent to a vertex in $C - B$, then $p(C \cup \{y\}) \leq p(C)$ and $q(C \cup \{y\}) < q(C)$.

Now, look at a possible analysis for the above greedy algorithm: Let $x_1, \ldots, X_g$ be vertices chosen by the greedy algorithm in the ordering of their appearance in the algorithm. Denote $C_i = \{x_1, \ldots, x_i\}$. Let $C^* = \{y_1, \ldots, y_{opt}\}$ be a minimum connected dominating set. Since adding C^* to C_i will reduce the potential function value from $f(C_i)$ to 2, the value of f reduced by a vertex in C^* would be $(f(C_i) - 2)/opt$ in average. By the greedy rule for choosing $x_i + 1$, one has

$$f(C_i) - f(C_{i+1}) \geq \frac{f(C_i) - 2}{opt}.$$

Hence,

$$f(C_{i+1}) - 2 \le (f(C_i) - 2)(1 - \frac{1}{opt})$$
$$\le (f(\emptyset) - 2)(1 - \frac{1}{opt})^{i+1} = (n-2)(1 - \frac{1}{opt})^{i+1} ,$$

where $n = |V|$. Note that $1 - 1/opt \le e^{-1/opt}$. Hence,

$$f(C_i) - 2 \le (n-2)e^{-i/opt} .$$

Choose i such that $f(C_i) \ge opt + 2 > f(C_{i+1})$. Then

$$opt \le (n-2)e^{-i/opt}$$

and

$$g - i \le opt .$$

Therefore,

$$g \le opt + i \le opt\left(1 + \ln\frac{n-2}{opt}\right) .$$

Is this analysis correct? The answer is NO. Why? How could one give a correct analysis. This article will answer those questions and introduce a new general technique, analysis of greedy approximation with nonsubmodular potential function.

Key Results

The Role of Submodularity

Consider a set X and a function f defined on the power set 2^X, i. e., the family of all subsets of X. f is said to be *submodular* if for any two subsets A and B in 2^X,

$$f(A) + f(B) \ge f(A \cap B) + f(A \cup B) .$$

For example, consider a connected graph G. Let X be the vertex set of G. The function $-q(C)$ defined in last section is submodular. To see this, first mention a property of submodular functions.

A submodular function f is *normalized* if $f(\emptyset) = 0$. Every submodular function f can be normalized by setting $g(A) = f(A) - f(\emptyset)$. A function f is *monotone increasing* if $f(A) \le f(B)$ for $A \subset B$. Denote $\Delta_x f(A) = f(A \cup \{x\}) - f(A)$.

Lemma 1 *A function $f: 2^X \to R$ is submodular if and only if $\Delta_x f(A) \le \Delta_x f(B)$ for any $x \in X - B$ and $A \subseteq B$. Moreover, f is monotone increasing if and only if $\Delta_x f(A) \le \Delta_x f(B)$ for any $x \in B$ and $A \subseteq B$.*

Proof If f is submodular, then for $x \in X - B$ and $A \subseteq B$, one has

$$f(A \cup \{x\}) + f(B)$$
$$\ge f((A \cup \{x\}) \cup B) + f(A \cup \{x\}) \cap B)$$
$$= f(B \cup \{x\}) + f(A) ,$$

that is,

$$\Delta_x f(A) \ge \Delta_x f(B) . \quad (1)$$

Conversely, suppose (1) holds for any $x \in B$ and $A \subseteq B$. Let C and D be two set and $C \setminus D = \{x_1, \ldots, x_k\}$. Then

$$f(C \cup D) - f(D) = \sum_{i=1}^{k} \Delta_{x_i} f(D \cup \{x_1, \ldots, x_{i-1})$$
$$\le \sum_{i=1}^{k} \Delta_{x_i} f((C \cap D) \cup \{x_1, \ldots, x_{i-1})$$
$$= f(C) - f(C \cap D) .$$

If f is monotone increasing, then for $A \subseteq B$, $f(A) \le f(B)$. Hence, for $x \in B$,

$$\Delta) x f(A) \ge 0 = \Delta_x f(B) .$$

Conversely, if $\Delta_x f(A) \ge \Delta_x f(B)$ for any $x \in B$ and $A \subseteq B$, then for any x and A, $\Delta_x f(A) \ge \Delta_x f(A \cup \{x\}) = 0$, that is $f(A) \le f(A \cup \{x\})$. Let $B - A = \{x_1, \ldots, x_k\}$. Then

$$f(A) \le f(A \cup \{x_1\}) \le f(A \cup \{x_1, x_2\}) \le \cdots \le f(B) .$$

Next, the submodularity of $-q(A)$ is studied.

Lemma 2 *If $A \subset B$, then $\Delta_y q(A) \ge \Delta_y q(B)$.*

Proof Note that each connected component of graph $(V, D(B))$ is constituted by one or more connected components of graph $(V, D(A))$ since $A \subset B$. Thus, the number of connected components of $(V, D(B))$ dominated by y is no more than the number of connected components of $(V, D(A))$ dominated by y. Therefore, the lemma holds.

The relationship between submodular functions and greedy algorithms have been established for a long time [3].

Let f be a normalized, monotone increasing, submodular integer function. Consider the minimization problem

$$\begin{aligned} \min \quad & c(A) \\ \text{subject to} \quad & A \in C_f . \end{aligned}$$

where c is a nonnegative cost function defined on 2^X and $C_f = \{C \mid f(C \cup \{x\}) - f(C) = 0 \text{ for all } x \in X\}$. The following is a greedy algorithm to produce approximation solution for this problem.

Greedy Algorithm B
input submodular function f and cost function c;
$A \leftarrow \emptyset$;
while there exists $x \in E$ such that $\Delta_x f(A) > 0$
do select a vertex x that maximizes $\Delta_x f(A)/c(x)$ and set
$A \leftarrow A \cup \{x\}$;
return A.

The following two results are well-known.

Theorem 1 *If f is a normalized, monotone increasing, submodular integer function, then Greedy Algorithm B produces an approximation solution within a factor of $H(\gamma)$ from optimal, where $\gamma = \max_{x \in E} f(\{x\})$.*

Theorem 2 *Let f be a normalized, monotone increasing, submodular function and c a nonnegative cost function. If in Greedy Algorithm B, selected x always satisfies $\Delta_x f(A_{i-1})/c(x) \geq 1$, then it produces an approximation solution within a factor of $1 + \ln(f^*/opt)$ from optimal for above minimization problem where $f^* = f(A^*)$ and $opt = c(A^*)$ for optimal solution A^*.*

Now, come back to the analysis of Greedy Algorithm A for the MCDS. It looks like that the submodularity of f is not used. Actually, the submodularity was implicitly used in the following statement:
"Since adding C^* to C_i will reduce the potential function value from $f(C_i)$ to 2, the value of f reduced by a vertex in C^* would be $(f(C_i) - 2)/opt$ in average. By the greedy rule for choosing $x_i + 1$, one has

$$f(C_i) - f(C_{i+1}) \geq \frac{f(C_i) - 2}{opt} \; ."$$

To see this, write this argument more carefully.

Let $C^* = \{y_1, \ldots, y_{opt}\}$ and denote $C_j^* = \{y_1, \ldots, y_j\}$. Then

$$\begin{aligned} f(C_i) - 2 &= f(C_i) - f(C_i \cup C^*) \\ &= \sum_{j=1}^{opt} [f(C_i \cup C_{j-1}^*) - f(C_i \cup C_j^*)] \end{aligned}$$

where $C_0^* = \emptyset$. By the greedy rule for choosing $x_i + 1$, one has

$$f(C_i) - f(C_{i+1}) \geq f(C_i) - f(C_i \cup \{y_j\})$$

for $j = 1, \ldots, opt$. Therefore, it needs to have

$$\begin{aligned} -\Delta_{y_j} f(C_i) &= f(C_i) - f(C_i \cup \{y_j\}) \\ &\geq f(C_i \cup C_{j-1}^*) - f(C_i \cup C_j^*) \\ &= -\Delta_{y_j} f(C_i \cup C_{j-1}^*) \end{aligned} \tag{2}$$

in order to have

$$f(C_i) - f(C_{i+1}) \geq \frac{f(C_i) - 2}{opt} \; .$$

(2) asks the submodularity of $-f$. Unfortunately, $-f$ is not submodular. A counterexample can be found in [3]. This is why the analysis of Greedy Algorithm A in Sect. "Problem Definition" is incorrect.

Giving up Submodularity

Giving up submodularity is a challenge task since it is open for a long time. But, it is possible based on the following observation on (2) by Du et al. [1]: **The submodularity of $-f$ is applied to increment of a vertex y_j belonging to optimal solution C^*.**

Since the ordering of y_j's is flexible, one may arrange it to make $\Delta_{y_j} f(C_i) - \Delta_{y_j} f(C_i \cup C_{j-1}^*)$ under control. This is a successful idea for the MCDS.

Lemma 3 *Let y_j's be ordered in the way that for any $j = 1, \ldots, opt$, $\{y_1, \ldots, y_j\}$ induces a connected subgraph. Then*

$$\Delta_{y_j} f(C_i) - \Delta_{y_j} f(C_i \cup C_{j-1}^*) \leq 1 \; .$$

Proof Since all $y_1, \ldots, y_{j-1}$ are connected, y_j can dominate at most one additional connected component in the subgraph induced by $C_{i-1} \cup C_{j-1}^*$ than in the subgraph induced by $c_i - 1$. Hence

$$\Delta_{y_j} p(C_i) - \Delta_{y_j} f(C_i \cup C_{j-1}^*) \leq 1 \; .$$

Moreover, since $-q$ is submodular,

$$\Delta_{y_j} q(C_i) - \Delta_{y_j} q(C_i \cup C_{j-1}^*) \leq 0 \; .$$

Therefore,

$$\Delta_{y_j} f(C_i) - \Delta_{y_j} f(C_i \cup C_{j-1}^*) \leq 1 \; .$$

Now, one can give a correct analysis for the greedy algorithm for the MCDS [4].

By Lemma 3,

$$f(C_i) - f(C_{i+1}) \geq \frac{f(C_i) - 2}{opt} - 1 \; .$$

Hence,

$$f(C_{i+1}) - 2 - opt \le (f(C_i) - 2 + opt)\left(1 - \frac{1}{opt}\right) \le (f(\emptyset) - 2 - opt)\left(1 - \frac{1}{opt}\right)^{i+1} = (n - 2 - opt)\left(1 - \frac{1}{opt}\right)^{i+1},$$

where $n = |V|$. Note that $1 - 1/opt \le e^{-1/opt}$. Hence,

$$f(C_i) - 2 - opt \le (n-2)e^{-i/opt}.$$

Choose i such that $f(C_i) \ge 2 \cdot opt + 2 > f(C_{i+1})$. Then

$$opt \le (n-2)e^{-i/opt}$$

and

$$g - i \le 2 \cdot opt.$$

Therefore,

$$g \le 2 \cdot opt + i \le opt\left(2 + \ln\frac{n-2}{opt}\right) \le opt(2 + \ln\delta)$$

where δ is the maximum degree of input graph G.

Applications

The technique introduced in previous section has many applications, including analysis of iterated 1-Steiner trees for minimum Steiner tree problem and analysis of greedy approximations for optimization problems in optical networks [4] and wireless networks [3].

Open Problems

Can one show the performance ratio $1 + H(\delta)$ for Greedy Algorithm B for the MCDS? The answer is unknown. More generally, it is unknown how to get a clean generalization of Theorem 1.

Cross References

- ► Connected Dominating Set
- ► Local Search Algorithms for kSAT
- ► Steiner Trees

Acknowledgments

Weili Wu is partially supported by NSF grant ACI-0305567.

Recommended Reading

1. Du, D.-Z., Graham, R.L., Pardalos, P.M., Wan, P.-J., Wu, W., Zhao, W.: Analysis of greedy approximations with nonsubmodular potential functions. ACM-SIAM Symposium on Discrete Algorithms (SODA), 2008
3. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, Hoboken (1999)
4. Ruan, L., Du, H., Jia, X., Wu, W., Li, Y., Ko, K.-I.: A greedy approximation for minimum connected dominating set. Theor. Comput. Sci. **329**, 325–330 (2004)
5. Ruan, L., Wu, W.: Broadcast routing with minimum wavelength conversion in WDM optical networks. J. Comb. Optim. **9** 223–235 (2005)

Greedy Set-Cover Algorithms

1974–1979; Chvátal, Johnson, Lovász, Stein

Neal E. Young
Department of Computer Science, University of California at Riverside, Riverside, CA, USA

Keywords and Synonyms

Dominating set; Greedy algorithm; Hitting set; Set cover; Minimizing a linear function subject to a submodular constraint

Problem Definition

Given a collection S of sets over a universe U, a *set cover* $C \subseteq S$ is a subcollection of the sets whose union is U. The *set-cover problem* is, given S, to find a minimum-cardinality set cover. In the *weighted set-cover problem*, for each set $s \in S$ a weight $w_s \ge 0$ is also specified, and the goal is to find a set cover C of minimum total weight $\sum_{s\in C} w_s$.

Weighted set cover is a special case of *minimizing a linear function subject to a submodular constraint*, defined as follows. Given a collection S of objects, for each object s a non-negative weight w_s, and a non-decreasing submodular function $f : 2^S \to \mathbb{R}$, the goal is to find a subcollection $C \subseteq S$ such that $f(C) = f(S)$ minimizing $\sum_{s\in C} w_s$. (Taking $f(C) = |\cup_{s\in C} s|$ gives weighted set cover.)

Key Results

The *greedy algorithm* for weighted set cover builds a cover by repeatedly choosing a set s that minimize the weight w_s divided by number of elements in s not yet covered by chosen sets. It stops and returns the chosen sets when they form a cover:

greedy-set-cover(S, w)
1. Initialize $C \leftarrow \emptyset$. Define $f(C) \doteq |\cup_{s\in C} s|$.
2. Repeat until $f(C) = f(S)$:
3. Choose $s \in S$ minimizing the price per element $w_s/[f(C \cup \{s\}) - f(C)]$.
4. Let $C \leftarrow C \cup \{s\}$.
5. Return C.

Let H_k denote $\sum_{i=1}^{k} 1/i \approx \ln k$, where k is the largest set size.

Theorem 1 *The greedy algorithm returns a set cover of weight at most H_k times the minimum weight of any cover.*

Proof When the greedy algorithm chooses a set s, imagine that it charges the price per element for that iteration to each element newly covered by s. Then the total weight of the sets chosen by the algorithm equals the total amount charged, and each element is charged once.

Consider any set $s = \{x_k, x_{k-1}, \ldots, x_1\}$ in the optimal set cover C^*. Without loss of generality, suppose that the greedy algorithm covers the elements of s in the order given: $x_k, x_{k-1}, \ldots, x_1$. At the start of the iteration in which the algorithm covers element x_i of s, at least i elements of s remain uncovered. Thus, if the greedy algorithm were to choose s in that iteration, it would pay a cost per element of at most w_s/i. Thus, in this iteration, the greedy algorithm pays at most w_s/i per element covered. Thus, it charges element x_i at most w_s/i to be covered. Summing over i, the total amount charged to elements in s is at most $w_s H_k$. Summing over $s \in C^*$ and noting that every element is in some set in C^*, the total amount charged to elements overall is at most $\sum_{s\in C^*} w_s H_k = H_k \mathrm{OPT}$. □

The theorem was shown first for the unweighted case (each $w_s = 1$) by Johnson [6], Lovász [9], and Stein [14], then extended to the weighted case by Chvátal [2].

Since then a few refinements and improvements have been shown, including the following:

Theorem 2 *Let S be a set system over a universe with n elements and weights $w_s \le 1$. The total weight of the cover C returned by the greedy algorithm is at most $[1 + \ln(n/\mathrm{OPT})]\mathrm{OPT} + 1$ (compare to [13]).*

Proof Assume without loss of generality that the algorithm covers the elements in order $x_n, x_{n-1}, \ldots, x_1$. At the start of the iteration in which the algorithm covers x_i, there are at least i elements left to cover, and all of them could be covered using multiple sets of total cost OPT. Thus, there is some set that covers not-yet-covered elements at a cost of at most OPT/i per element.

Recall the charging scheme from the previous proof. By the preceding observation, element x_i is charged at most OPT/i. Thus, the total charge to elements $x_n, \ldots, x_i$ is at most $(H_n - H_{i-1})\mathrm{OPT}$. Using the assumption that each $w_s \le 1$, the charge to each of the remaining elements is at most 1 per element. Thus, the total charge to all elements is at most $i - 1 + (H_n - H_{i-1})\mathrm{OPT}$. Taking $i = 1 + \lceil \mathrm{OPT} \rceil$, the total charge is at most $\lceil \mathrm{OPT} \rceil + (H_n - H_{\lceil \mathrm{OPT} \rceil})\mathrm{OPT} \le 1 + \mathrm{OPT}(1 + \ln(n/\mathrm{OPT}))$. □

Each of the above proofs implicitly constructs a linear-programming primal-dual pair to show the approximation ratio. The same approximation ratios can be shown with respect to any fractional optimum (solution to the fractional set-cover linear program).

Other Results

The greedy algorithm has been shown to have an approximation ratio of $\ln n - \ln \ln n + O(1)$ [12]. For the special case of set systems whose duals have finite Vapnik-Chervonenkis (VC) dimension, other algorithms have substantially better approximation ratio [1]. Constant-factor approximation algorithms are known for geometric variants of the closely related k-median and facility location problems.

The greedy algorithm generalizes naturally to many problems. For example, for minimizing a linear function subject to a submodular constraint (defined above), the natural extension of the greedy algorithm gives an H_k-approximate solution, where $k = \max_{s\in S} f(\{s\}) - f(\emptyset)$, assuming f is integer-valued [10].

The set-cover problem generalizes to allow each element x to require an arbitrary number r_x of sets containing it to be in the cover. This generalization admits a polynomial-time $O(\log n)$-approximation algorithm [8].

The special case when each element belongs to at most r sets has a simple r-approximation algorithm ([15] § 15.2). When the sets have uniform weights ($w_s = 1$), the algorithm reduces to the following: select any maximal collection of elements, no two of which are contained in the same set; return all sets that contain a selected element.

The variant "Max k-coverage" asks for a set collection of total weight at most k covering as many of the elements as possible. This variant has a $(1 - 1/e)$-approximation algorithm ([15] Problem 2.18) (see [7] for sets with non-uniform weights).

For a general discussion of greedy methods for approximate combinatorial optimization, see ([5] Ch. 4).

Finally, under likely complexity-theoretic assumptions, the $\ln n$ approximation ratio is essentially the best possible for any polynomial-time algorithm [3,4].

Applications

Set Cover and its generalizations and variants are fundamental problems with numerous applications. Examples include:

- selecting a small number of nodes in a network to store a file so that all nodes have a nearby copy,
- selecting a small number of sentences to be uttered to tune all features in a speech-recognition model [11],
- selecting a small number of telescope snapshots to be taken to capture light from all galaxies in the night sky,
- finding a short string having each string in a given set as a contiguous sub-string.

Cross References

► Local Search for K-medians and Facility Location

Recommended Reading

1. Brönnimann, H., Goodrich, M.T.: Almost optimal set covers in finite VC-dimension. Discret. Comput. Geom. **14**(4), 463–479 (1995)
2. Chvátal, V.: A greedy heuristic for the set-covering problem. Math. Oper. Res. **4**(3), 233–235 (1979)
3. Lund, C., Yannakakis, M.: On the hardness of approximating minimization problems. J. ACM **41**(5), 960–981 (1994)
4. Feige, U.: A threshold of ln n for approximating set cover. J. ACM **45**(4), 634–652 (1998)
5. Gonzalez, T.F.: Handbook of Approximation Algorithms and Metaheuristics. Chapman & Hall/CRC Computer & Information Science Series (2007)
6. Johnson, D.S.: Approximation algorithms for combinatorial problems. J. Comput. Syst. Sci. **9**, 256–278 (1974)
7. Khuller, S., Moss, A., Naor, J.: The budgeted maximum coverage problem. Inform. Process. Lett. **70**(1), 39–45 (1999)
8. Kolliopoulos, S.G., Young, N.E.: Tight approximation results for general covering integer programs. In: Proceedings of the forty-second annual IEEE Symposium on Foundations of Computer Science, pp. 522–528 (2001)
9. Lovász, L.: On the ratio of optimal integral and fractional covers. Discret. Math. **13**, 383–390 (1975)
10. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988)
11. van Santen, J.P.H., Buchsbaum, A.L.: Methods for optimal text selection. In: Proceedings of the European Conference on Speech Communication and Technology (Rhodos, Greece) **2**, 553–556 (1997)
12. Slavik, P.: A tight analysis of the greedy algorithm for set cover. J. Algorithms **25**(2), 237–254 (1997)
13. Srinivasan, A.: Improved approximations of packing and covering problems. In: Proceedings of the twenty-seventh annual ACM Symposium on Theory of Computing, pp. 268–276 (1995)
14. Stein, S.K.: Two combinatorial covering theorems. J. Comb. Theor. A **16**, 391–397 (1974)
15. Vazirani, V.V.: Approximation Algorithms. Springer, Berlin Heidelberg (2001)