H

Hamilton Cycles in Random Intersection Graphs
2005; Efthymiou, Spirakis

Charilaos Efthymiou[1], Paul Spirakis[2]
[1] Department of Computer Engineering and Informatics, University of Patras, Patras, Greece
[2] Computer Engineering and Informatics, Research and Academic Computer Technology Institute, Patras University, Patras, Greece

Keywords and Synonyms

Threshold for appearance of Hamilton cycles in random intersection graphs; Stochastic order relations between Erdös–Rényi random graph model and random intersection graphs

Problem Definition

E. Marczewski proved that every graph can be represented by a list of sets where each vertex corresponds to a set and the edges to nonempty intersections of sets. It is natural to ask what sort of graphs would be most likely to arise if the list of sets is generated randomly.

Consider the model of random graphs where each vertex chooses randomly from a universal set the members of its corresponding set, each independently of the others. The probability space that is created is the space of random intersection graphs, $G_{n,m,p}$, where n is the number of vertices, m is the cardinality of a universal set of elements and p is the probability for each vertex to choose an element of the universal set. The model of random intersection graphs was first introduced by M. Karońsky, E. Scheinerman, and K. Singer-Cohen in [4]. A rigorous definition of the model of random intersection graphs follows:

Definition 1 Let n, m be positive integers and $0 \le p \le 1$. The random intersection graph $G_{n,m,p}$ is a probability space over the set of graphs on the vertex set $\{1, \ldots, n\}$ where each vertex is assigned a random subset from a fixed set of m elements. An edge arises between two vertices when their sets have at least a common element. Each random subset assigned to a vertex is determined by

$$\mathbf{Pr}\left[\text{vertex } i \text{ chooses element } j\right] = p$$

with these events mutually independent.

A common question for a graph is whether it has a cycle, a set of edges that form a path so that the first and the last vertex is the same, that visits *all* the vertices of the graph exactly once. We call this kind of cycle the *Hamilton cycle* and the graph that contains such a cycle is called a *Hamiltonian graph*.

Definition 2 Consider an undirected graph $G = (V, E)$ where V is the set of vertices and E the set of edges. This graph contains a Hamilton cycle if and only if there is a simple cycle that contains each vertex in V.

Consider an instance of $G_{n,m,p}$, for specific values of its parameters n, m, and p, what is the probability of that instance to be Hamiltonian? Taking the parameter p, of the model, to be a function of n and m, in [2], a threshold function $P(n, m)$ has been found for the graph property "Contains a Hamilton cycle"; i. e. a function $P(n, m)$ is derived such that

$$\text{if } p(n,m) \ll P(n,m)$$
$$\lim_{n,m\to\infty} \mathbf{Pr}\left[G_{n,m,p} \text{ Contains Hamilton cycle}\right] = 0$$
$$\text{if } p(n,m) \gg P(n,m)$$
$$\lim_{n,m\to\infty} \mathbf{Pr}\left[G_{n,m,p} \text{ Contains Hamilton cycle}\right] = 1$$

When a graph property, such as "Contains a Hamilton cycle," holds with probability that tends to 1 (or 0) as n, m tend to infinity, then it is said that this property holds (does not hold), "almost surely" or "almost certainly."

If in $G_{n,m,p}$ the parameter m is very small compared to n, the model is not particularly interesting and when m is exceedingly large (compared to n) the behavior of $G_{n,m,p}$ is essentially the same as the Erdös–Rényi model

of random graphs (see [3]). If someone takes $m = \lceil n^\alpha \rceil$, for fixed real $\alpha > 0$, then there is some deviation from the standard models, while allowing for a natural progression from sparse to dense graphs. Thus, the parameter m is assumed to be of the form $m = \lceil n^\alpha \rceil$ for some fixed positive real α.

The proof of existence of a Hamilton cycle in $G_{n,m,p}$ is mainly based on the establishment of a *stochastic order relation* between the model $G_{n,m,p}$ and the Erdös–Rényi random graph model $G_{n,\hat{p}}$.

Definition 3 Let n be a positive integer, $0 \le \hat{p} \le 1$. The random graph $G(n, \hat{p})$ is a probability space over the set of graphs on the vertex set $\{1, \ldots, n\}$ determined by

$$\mathbf{Pr}\left[i, j\right] = \hat{p}$$

with these events mutually independent.

The stochastic order relation between the two models of random graphs is established in the sense that if $\mathcal{A}$ is an increasing graph property, then it holds that

$$\mathbf{Pr}\left[G_{n,\hat{p}} \in \mathcal{A}\right] \le \mathbf{Pr}\left[G_{n,m,p} \in \mathcal{A}\right]$$

where $\hat{p} = f(p)$. A graph property $\mathcal{A}$ is increasing if and only if given that $\mathcal{A}$ holds for a graph $G(V, E)$ then $\mathcal{A}$ holds for any $G(V, E')$: $E' \supseteq E$.

Key Results

Theorem 1 *Let $m = \lceil n^\alpha \rceil$, where α is a fixed real positive, and C_1, C_2 be sufficiently large constants. If*

$$p \ge C_1 \frac{\log n}{m} \quad \textit{for} \quad 0 < \alpha < 1 \quad \textit{or}$$

$$p \ge C_2 \sqrt{\frac{\log n}{nm}} \quad \textit{for} \quad \alpha > 1$$

then almost all $G_{n,m,p}$ are Hamiltonian. Our bounds are asymptotically tight.

Note that the theorem above says nothing when $m = n$, i. e. $\alpha = 1$.

Applications

The Erdös–Rényi model of random graphs, $G_{n,p}$, is exhaustively studied in computer science because it provides a framework for studying practical problems such as "reliable network computing" or it provides a "typical instance" of a graph and thus it is used for average case analysis of graph algorithms. However, the simplicity of $G_{n,p}$ means it is not able to capture satisfactorily many practical problems in computer science. Basically, this is because of the fact that in many problems independent edge-events are not well justified. For example, consider a graph whose vertices represent a set of objects that either are placed or move in a specific geographical region, and the edges are radio communication links. In such a graph, we expect that, any two vertices u, w are more likely to be adjacent to each other, than any other, arbitrary, pair of vertices, if both are adjecent to a third vertex v. Even epidemiological phenomena (like the spread of disease) tend to be more accurately captured by this proximity-sensitive random intersection graph model. Other applications may include oblivious resource sharing in a distributive setting, interaction of mobile agents traversing the web etc.

The model of random intersection graphs $G_{n,m,p}$ was first introduced by M. Karońsky, E. Scheinerman, and K. Singer-Cohen in [4] where they explored the evolution of random intersection graphs by studying the thresholds for the appearance and disappearance of small induced subgraphs. Also, J.A. Fill, E.R. Scheinerman, and K. Singer Cohen in [3] proved an equivalence theorem relating the evolution of $G_{n,m,p}$ and $G_{n,p}$, in particular they proved that when $m = n^\alpha$ where $\alpha > 6$, the total variation distance between the graph random variables has limit 0. S. Nikoletseas, C. Raptopoulos, and P. Spirakis in [8] studied the existence and the efficient algorithmic construction of close to optimal independent sets in random intersection graphs. D. Stark in [12] studied the degree of the vertices of the random intersection graphs. However, after [2], Spirakis and Raptopoulos, in [11], provide algorithms that construct Hamilton cycles in instances of $G_{n,m,p}$, for p above the Hamiltonicity threshold. Finally, Nikoletseas et.al in [7] study the mixing time and cover time as the parameter p of the model varies.

Open Problems

As in many other random structures, e. g. $G_{n,p}$ and random formulae, properties of random intersection graphs also appear to have threshold behavior. So far threshold behavior has been studied for the induced subgraph appearance and hamiltonicity.

Other fields of research for random intersection graphs may include the study of connectivity behavior, of the model i. e. the path formation, the formation of giant components. Additionally, a very interesting research question is how cover and mixing times vary with the parameter p, of the model.

Cross References

► Independent Sets in Random Intersection Graphs

Recommended Reading

1. Alon, N., Spencer, J.H.: The Probabilistic Method. 2nd edn. Wiley, New York (2000)
2. Efthymiou, C., Spirakis, P.G.: On the Existence of Hamilton Cycles in Random Intersection Graphs. In: Proc. of the 32nd ICALP. LNCS, vol. 3580, pp. 690–701. Springer, Berlin/Heidelberg (2005)
3. Fill, J.A., Scheinerman, E.R., Singer-Cohen, K.B.: Random intersection graphs when $m = \omega(n)$: an equivalence theorem relating the evolution of the $G(n, m, p)$ and $G(n, p)$ models. Random Struct. Algorithms **16**, 156–176 (2000)
4. Karoński, M., Scheinerman, E.R., Singer-Cohen, K.: On Random Intersection Graphs: The Subgraph Problem. Comb. Probab. Comput. **8**, 131–159 (1999)
5. Komlós, J., Szemerédi, E.: Limit Distributions for the existence of Hamilton cycles in a random graph. Discret. Math. **43**, 55–63 (1983)
6. Korshunov, A.D.: Solution of a problem of P. Erdös and A. Rényi on Hamilton Cycles in non-oriented graphs. Metody Diskr. Anal. Teoriy Upr. Syst. Sb. Trubov Novosibrirsk **31**, 17–56 (1977)
7. Nikoletseas, S., Raptopoulos, C., Spirakis, P.: Expander Properties and the Cover Time of Random Intersection Graphs. In: Proc of the 32nd MFCS, pp. 44–55. Springer, Berlin/Heidelberg (2007)
8. Nikoletseas, S., Raptopoulos, C., Spirakis, P.: The existence and Efficient construction of Large Independent Sets in General Random Intersection Graphs. In: Proc. of the 31st ICALP. LNCS, vol. 3142, pp. 1029–1040. Springer, Berlin/Heidelberg (2004)
10. Singer, K.: Random Intersection Graphs. Ph. D. thesis, The Johns Hopkins University, Baltimore (1995)
11. Spirakis, P.G. Raptopoulos, C.: Simple and Efficient Greedy Algorithms for Hamilton Cycles in Random Intersection Graphs. In: Proc. of the 16th ISAAC. LNCS, vol. 3827, pp. 493–504. Springer, Berlin/Heidelberg (2005)
12. Stark, D.: The Vertex Degree Distribution of Random Intersection Graphs. Random Struct. Algorithms **24**, 249–258 (2004)

Hardness of Proper Learning

1988; Pitt, Valiant

Vitaly Feldman
Department of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

Keywords and Synonyms

Representation-based hardness of learning

Problem Definition

The work of Pitt and Valiant [16] deals with learning Boolean functions in the Probably Approximately Correct (PAC) learning model introduced by Valiant [17]. A learning algorithm in Valiant's original model is given random examples of a function $f : \{0,1\}^n \to \{0,1\}$ from a representation class $\mathcal{F}$ and produces a hypothesis $h \in \mathcal{F}$ that closely approximates f. Here a *representation class* is a set of functions and a language for describing the functions in the set. The authors give examples of natural representation classes that are NP-hard to learn in this model whereas they can be learned if the learning algorithm is allowed to produce hypotheses from a richer representation class $\mathcal{H}$. Such an algorithm is said to learn $\mathcal{F}$ by $\mathcal{H}$; learning $\mathcal{F}$ by $\mathcal{F}$ is called *proper* learning.

The results of Pitt and Valiant were the first to demonstrate that the choice of representation of hypotheses can have a dramatic impact on the computational complexity of a learning problem. Their specific reductions from NP-hard problems are the basis of several other follow-up works on the hardness of proper learning [1,3,6].

Notation

Learning in the PAC model is based on the assumption that the unknown function (or *concept*) belongs to a certain class of concepts C. In order to discuss algorithms that learn and output functions one needs to define how these functions are represented. Informally, a representation for a concept class C is a way to describe concepts from C that defines a procedure to evaluate a concept in C on any input. For example, one can represent a conjunction of input variables by listing the variables in the conjunction. More formally, a representation class can be defined as follows.

Definition 1 A *representation class* $\mathcal{F}$ is a pair $(L, \mathcal{R})$ where

- L is a language over some fixed finite alphabet (e. g. $\{0,1\}$);
- $\mathcal{R}$ is an algorithm that for $\sigma \in L$, on input $(\sigma, 1^n)$ returns a Boolean circuit over $\{0,1\}^n$.

In the context of efficient learning, only efficient representations are considered, or, representations for which $\mathcal{R}$ is a polynomial-time algorithm. The concept class represented by $\mathcal{F}$ is set of functions over $\{0,1\}^n$ defined by the circuits in $\{\mathcal{R}(\sigma, 1^n) \mid \sigma \in L\}$. For most of the representations discussed in the context of learning it is straightforward to construct a language L and the corresponding translating function $\mathcal{R}$, and therefore they are not specified explicitly.

Associated with each representation is the complexity of describing a Boolean function using this representation. More formally, for a Boolean function $f \in C$, $\mathcal{F}\text{-}\mathtt{size}(f)$ is the length of the shortest way to represent f using $\mathcal{F}$, or $\min\{|\sigma| \mid \sigma \in L,\ \mathcal{R}(\sigma, 1^n) \equiv f\}$.

In Valiant's PAC model of learning, for a function f and a distribution $\mathcal{D}$ over X, an *example oracle* $\mathrm{EX}(f, \mathcal{D})$ is an oracle that, when invoked, returns an example

$\langle x, f(x)\rangle$, where x is chosen randomly with respect to $\mathcal{D}$, independently of any previous examples. For $\epsilon \geq 0$, a function g ϵ-approximates a function f with respect to distribution $\mathcal{D}$ if $\mathbf{Pr}_{\mathcal{D}}[f(x) \neq g(x)] \leq \epsilon$.

Definition 2 A representation class $\mathcal{F}$ is *PAC learnable* by representation class $\mathcal{H}$ if there exist an algorithm that for every $\epsilon > 0$, $\delta > 0$, n, $f \in \mathcal{F}$, and distribution $\mathcal{D}$ over X, given ϵ, δ, and access to $\mathrm{EX}(f, \mathcal{D})$, runs in time polynomial in $n, s = \mathcal{F}\text{-}\mathtt{size}(c), 1/\epsilon$ and $1/\delta$, and outputs, with probability at least $1 - \delta$, a hypothesis $h \in \mathcal{H}$ that ϵ-approximates f.

A DNF expression is defined as an OR of ANDs of literals, where a *literal* is a possibly negated input variable. The ANDs of a DNF formula are referred to as its *terms*. Let DNF(k) denote the representation class of k-term DNF expressions. Similarly a CNF expression is an OR of ANDs of literals. Let k-CNF denote the representation class of CNF expressions with each AND having at most k literals.

For a real-valued vector $c \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$, a *linear threshold function* (also called a *halfspace*) $T_{c,\theta}(x)$ is the function that equals 1 if and only if $\sum_{i \leq n} c_i x_i \geq \theta$. The representation class of Boolean threshold functions consists of all linear threshold functions with $c \in \{0, 1\}^n$ and θ an integer.

Key Results

Theorem 3 ([16]) *For every $k \geq 2$, the representation class of* DNF(k) *is not properly learnable unless* RP = NP.

More specifically, Pitt and Valiant show that learning DNF(k) by DNF(ℓ) is at least as hard as coloring a k-colorable graph using ℓ colors. For the case $k = 2$ they obtain the result by reducing from Set Splitting (see [8] for details on the problems). Theorem 3 is in sharp contrast with the fact that DNF(k) is learnable by k-CNF [17].

Theorem 4 ([16]) *The representation class of Boolean threshold functions is not properly learnable unless* RP = NP.

This result is obtained via a reduction from the NP-complete Zero-One Integer Programming problem (see [8](p. 245) for details on the problem). The result is contrasted by the fact that general linear thresholds are properly learnable [4].

These results show that using a specific representation of hypotheses forces the learning algorithm to solve a combinatorial problem that can be NP-hard. In most machine learning applications it is not important which representation of hypotheses is used as long as the value of the unknown function is predicted correctly. Therefore learning in the PAC model is now defined without any restrictions on the output hypothesis (other than it being efficiently evaluatable). Hardness results in this setting are usually based on cryptographic assumptions (*cf.* [14]).

Hardness results for proper learning based on assumption NP $\neq$ RP are now known for several other representation classes and for other variants and extensions of the PAC learning model. Blum and Rivest show that for any $k \geq 3$, unions of k halfspaces are not properly learnable [3]. Hancock et al. prove that decision trees (*cf.* [15] for the definition of this representation) are not learnable by decision trees of somewhat larger size [10]. This result was strengthened by Alekhnovich et al. who also prove that intersections of two halfspaces are not learnable by intersections of k halfspaces for any constant k, general DNF expressions are not learnable by unions of halfspaces (and in particular are not properly learnable), and *k-juntas* are not properly learnable [1]. Feldman shows that DNF expressions are NP-hard to learn properly even if *membership queries*, or the ability to query the unknown function at any point, are allowed [6]. No efficient algorithms or hardness results are known for any of the above learning problems if no restriction is placed on the representation of hypotheses.

The choice of representation is very important even in powerful learning models. Feldman proved that n^c-term DNF are not properly learnable for any constant c even when the distribution of examples is assumed to be uniform and membership queries are available [6]. This contrasts with Jackson's celebrated algorithm for learning DNF in this setting [12], which is not proper.

In the *agnostic learning* model of Haussler [11] and Kearns et al. [13] even the representation classes of conjunctions, halfspaces, and parity functions are NP-hard to learn properly (*cf.* [2,7,9] and references therein). Here again the status of these problems in the representation-independent setting is largely unknown.

Applications

A large number of practical algorithms use representations for which hardness results are known (most notably decision trees, halfspaces, and neural networks). Hardness of learning $\mathcal{F}$ by $\mathcal{H}$ implies that an algorithm that uses $\mathcal{H}$ to represent its hypotheses will not be able to learn $\mathcal{F}$ in the PAC sense. Therefore such hardness results elucidate the limitations of algorithms used in practice. In particular, the reduction from an NP-hard problem used to prove the hardness of learning $\mathcal{F}$ by $\mathcal{H}$ can be used to generate hard instances of the learning problem.

Open Problems

A number of problems related to proper learning in the PAC model and its extensions are open. Almost all hardness of proper learning results are for learning with respect to unrestricted distributions. For most of the problems mentioned in Sect. "Key Results" it is unknown whether the result is true if the distribution is restricted to belong to some natural class of distributions (e. g. product distributions). It is unknown whether decision trees are learnable properly in the PAC model or in the PAC model with membership queries. This question is open even in the PAC model restricted to the uniform distribution only. Note that decision trees are learnable (non-properly) if membership queries are available [5] and are learnable properly in time $O(n^{\log s})$, where s is the number of leaves in the decision tree [1].

An even more interesting direction of research would be to obtain hardness results for learning by richer representations classes, such as AC^0 circuits, classes of neural networks and, ultimately, unrestricted circuits.

Cross References

- ► Cryptographic Hardness of Learning
- ► Graph Coloring
- ► Learning DNF Formulas
- ► PAC Learning

Recommended Reading

1. Alekhnovich, M., Braverman, M., Feldman, V., Klivans, A., Pitassi, T.: Learnability and automizability. In: Proceeding of FOCS, pp. 621–630 (2004)
2. Ben-David, S., Eiron, N., Long, P. M.: On the difficulty of approximately maximizing agreements. In: Proceedings of COLT, pp. 266–274 (2000)
3. Blum, A.L., Rivest, R.L.: Training a 3-node neural network is NP-complete. Neural Netw. **5**(1), 117–127 (1992)
4. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.: Learnability and the Vapnik-Chervonenkis dimension. J. ACM **36**(4), 929–965 (1989)
5. Bshouty, N.: Exact learning via the monotone theory. Inf. Comput. **123**(1), 146–153 (1995)
6. Feldman, V.: Hardness of Approximate Two-level Logic Minimization and PAC Learning with Membership Queries. In: Proceedings of STOC, pp. 363–372 (2006)
7. Feldman, V.: Optimal hardness results for maximizing agreements with monomials. In: Proceedings of Conference on Computational Complexity (CCC), pp. 226–236 (2006)
8. Garey, M., Johnson, D.S.: Computers and Intractability. W. H. Freeman, San Francisco (1979)
9. Guruswami, V., Raghavendra, P.: Hardness of Learning Halfspaces with Noise. In: Proceedings of FOCS, pp. 543–552 (2006)
10. Hancock, T., Jiang, T., Li, M., Tromp, J.: Lower bounds on learning decision lists and trees. In: 12th Annual Symposium on Theoretical Aspects of Computer Science, pp. 527–538 (1995)
11. Haussler, D.: Decision theoretic generalizations of the PAC model for neural net and other learning applications. Inf. Comput. **100**(1), 78–150 (1992)
12. Jackson, J.: An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. J. Comput. Syst. Sci. **55**, 414–440 (1997)
13. Kearns, M., Schapire, R., Sellie, L.: Toward efficient agnostic learning. Mach. Learn. **17**(2–3), 115–141 (1994)
14. Kearns, M., Valiant, L.: Cryptographic limitations on learning boolean formulae and finite automata. J. ACM **41**(1), 67–95 (1994)
15. Kearns, M., Vazirani, U.: An introduction to computational learning theory. MIT Press, Cambridge, MA (1994)
16. Pitt, L., Valiant, L.: Computational limitations on learning from examples. J. ACM **35**(4), 965–984 (1988)
17. Valiant, L.: A theory of the learnable. Commun. ACM **27**(11), 1134–1142 (1984)

High Performance Algorithm Engineering for Large-scale Problems

2005; Bader

David A. Bader
College of Computing, Georgia Institute of Technology, Atlanta, GA, USA

Keywords and Synonyms

Experimental algorithmics

Problem Definition

Algorithm engineering refers to the process required to transform a pencil-and-paper algorithm into a robust, efficient, well tested, and easily usable implementation. Thus it encompasses a number of topics, from modeling cache behavior to the principles of good software engineering; its main focus, however, is experimentation. In that sense, it may be viewed as a recent outgrowth of *Experimental Algorithmics* [14], which is specifically devoted to the development of methods, tools, and practices for assessing and refining algorithms through experimentation. The *ACM Journal of Experimental Algorithmics (JEA)*, at URL `www.jea.acm.org`, is devoted to this area.

High-performance algorithm engineering [2] focuses on one of the many facets of algorithm engineering: speed. The high-performance aspect does not immediately imply parallelism; in fact, in any highly parallel task, most of the impact of high-performance algorithm engineering tends to come from refining the serial part of the code.

The term *algorithm engineering* was first used with specificity in 1997, with the organization of the first *Workshop on Algorithm Engineering (WAE 97)*. Since then, this

workshop has taken place every summer in Europe. The 1998 *Workshop on Algorithms and Experiments (ALEX98)* was held in Italy and provided a discussion forum for researchers and practitioners interested in the design, analyzes and experimental testing of exact and heuristic algorithms. A sibling workshop was started in the Unites States in 1999, the *Workshop on Algorithm Engineering and Experiments (ALENEX99)*, which has taken place every winter, colocated with the *ACM/SIAM Symposium on Discrete Algorithms (SODA)*.

Key Results

Parallel computing has two closely related main uses. First, with more memory and storage resources than available on a single workstation, a parallel computer can solve correspondingly larger instances of the same problems. This increase in size can translate into running higher-fidelity simulations, handling higher volumes of information in data-intensive applications, and answering larger numbers of queries and datamining requests in corporate databases. Secondly, with more processors and larger aggregate memory subsystems than available on a single workstation, a parallel computer can often solve problems faster. This increase in speed can also translate into all of the advantages listed above, but perhaps its crucial advantage is in turnaround time. When the computation is part of a real-time system, such as weather forecasting, financial investment decision-making, or tracking and guidance systems, turnaround time is obviously the critical issue. A less obvious benefit of shortened turnaround time is higher-quality work: when a computational experiment takes less than an hour, the researcher can afford the luxury of exploration—running several different scenarios in order to gain a better understanding of the phenomena being studied.

In algorithm engineering, the aim is to present repeatable results through experiments that apply to a broader class of computers than the specific make of computer system used during the experiment. For sequential computing, empirical results are often fairly machine-independent. While machine characteristics such as word size, cache and main memory sizes, and processor and bus speeds differ, comparisons across different uniprocessor machines show the same trends. In particular, the number of memory accesses and processor operations remains fairly constant (or within a small constant factor). In high-performance algorithm engineering with parallel computers, on the other hand, this portability is usually absent: each machine and environment is its own special case. One obvious reason is major differences in hardware that affect the balance of communication and computation costs—a true shared-memory machine exhibits very different behavior from that of a cluster based on commodity networks.

Another reason is that the communication libraries and parallel programming environments (e. g., MPI [12], OpenMP [16], and High-Performance Fortran [10]), as well as the parallel algorithm packages (e. g., fast Fourier transforms using FFTW [6] or parallelized linear algebra routines in ScaLAPACK [4]), often exhibit differing performance on different types of parallel platforms. When multiple library packages exist for the same task, a user may observe different running times for each library version even on the same platform. Thus a running-time analysis should clearly separate the time spent in the user code from that spent in various library calls. Indeed, if particular library calls contribute significantly to the running time, the number of such calls and running time for each call should be recorded and used in the analysis, thereby helping library developers focus on the most cost-effective improvements. For example, in a simple message-passing program, one can characterize the work done by keeping track of sequential work, communication volume, and number of communications. A more general program using the collective communication routines of MPI could also count the number of calls to these routines. Several packages are available to instrument MPI codes in order to capture such data (e. g., MPICH's nupshot [8], Pablo [17], and Vampir [15]). The SKaMPI benchmark [18] allows running-time predictions based on such measurements even if the target machine is not available for program development. SKaMPI was designed for robustness, accuracy, portability, and efficiency; For example, SKaMPI adaptively controls how often measurements are repeated, adaptively refines message-length and step-width at "interesting" points, recovers from crashes, and automatically generates reports.

Applications

The following are several examples of algorithm engineering studies for high-performance and parallel computing.

1. Bader's prior publications (see [2] and http://www.cc.gatech.edu/~bader) contain many empirical studies of parallel algorithms for combinatorial problems like sorting, selection, graph algorithms, and image processing.
2. In a recent demonstration of the power of high-performance algorithm engineering, a million-fold speedup was achieved through a combination of a 2,000-fold speedup in the serial execution of the code and a 512-

fold speedup due to parallelism (a speed-up, however, that will scale to any number of processors) [13]. (In a further demonstration of algorithm engineering, additional refinements in the search and bounding strategies have added another speedup to the serial part of about 1,000, for an overall speedup in excess of 2 billion)

3. JáJá and Helman conducted empirical studies for prefix computations, sorting, and list-ranking, on symmetric multiprocessors. The sorting research (see [9]) extends Vitter's external Parallel Disk Model to the internal memory hierarchy of SMPs and uses this new computational model to analyze a general-purpose sample sort that operates efficiently in shared-memory. The performance evaluation uses 9 well-defined benchmarks. The benchmarks include input distributions commonly used for sorting benchmarks (such as keys selected uniformly and at random), but also benchmarks designed to challenge the implementation through load imbalance and memory contention and to circumvent algorithmic design choices based on specific input properties (such as data distribution, presence of duplicate keys, pre-sorted inputs, etc.).
4. In [3] Blelloch et al. compare through analysis and implementation three sorting algorithms on the Thinking Machines CM-2. Despite the use of an outdated (and no longer available) platform, this paper is a gem and should be required reading for every parallel algorithm designer. In one of the first studies of its kind, the authors estimate running times of four of the machine's primitives, then analyze the steps of the three sorting algorithms in terms of these parameters. The experimental studies of the performance are normalized to provide clear comparison of how the algorithms scale with input size on a 32*K*-processor CM-2.
5. Vitter et al. provide the canonical theoretic foundation for I/O-intensive experimental algorithmics using external parallel disks (e. g., see [1,19,20]). Examples from sorting, FFT, permuting, and matrix transposition problems are used to demonstrate the parallel disk model.
6. Juurlink and Wijshoff [11] perform one of the first detailed experimental accounts on the preciseness of several parallel computation models on five parallel platforms. The authors discuss the predictive capabilities of the models, compare the models to find out which allows for the design of the most efficient parallel algorithms, and experimentally compare the performance of algorithms designed with the model versus those designed with machine-specific characteristics in mind. The authors derive model parameters for each platform, analyses for a variety of algorithms (matrix multiplication, bitonic sort, sample sort, all-pairs shortest path), and detailed performance comparisons.
7. The LogP model of Culler et al. [5] provides a realistic model for designing parallel algorithms for message-passing platforms. Its use is demonstrated for a number of problems, including sorting.
8. Several research groups have performed extensive algorithm engineering for high-performance numerical computing. One of the most prominent efforts is that led by Dongarra for ScaLAPACK [4], a scalable linear algebra library for parallel computers. ScaLAPACK encapsulates much of the high-performance algorithm engineering with significant impact to its users who require efficient parallel versions of matrix–matrix linear algebra routines. New approaches for automatically tuning the sequential library (e. g., LAPACK) are now available as the ATLAS package [21].

Open Problems

All of the tools and techniques developed over the last several years for algorithm engineering are applicable to high-performance algorithm engineering. However, many of these tools need further refinement. For example, cache-efficient programming is a key to performance but it is not yet well understood, mainly because of complex machine-dependent issues like limited associativity, virtual address translation, and increasingly deep hierarchies of high-performance machines. A key question is whether one can find simple models as a basis for algorithm development. For example, cache-oblivious algorithms [7] are efficient at all levels of the memory hierarchy in theory, but so far only few work well in practice. As another example, profiling a running program offers serious challenges in a serial environment (any profiling tool affects the behavior of what is being observed), but these challenges pale in comparison with those arising in a parallel or distributed environment (for instance, measuring communication bottlenecks may require hardware assistance from the network switches or at least reprogramming them, which is sure to affect their behavior). Designing efficient and portable algorithms for commodity multicore and many-core processors is an open challenge.

Cross References

- ► Analyzing Cache Misses
- ► Cache-Oblivious B-Tree
- ► Cache-Oblivious Model
- ► Cache-Oblivious Sorting
- ► Engineering Algorithms for Computational Biology

- ▶ Engineering Algorithms for Large Network Applications
- ▶ Engineering Geometric Algorithms
- ▶ Experimental Methods for Algorithm Analysis
- ▶ External Sorting and Permuting
- ▶ Implementation Challenge for Shortest Paths
- ▶ Implementation Challenge for TSP Heuristics
- ▶ I/O-model
- ▶ Visualization Techniques for Algorithm Engineering

Recommended Reading

1. Aggarwal, A., Vitter, J.: The input/output complexity of sorting and related problems. Commun. ACM **31**, 1116–1127 (1988)
2. Bader, D.A., Moret, B.M.E., Sanders, P.: Algorithm engineering for parallel computation. In: Fleischer, R., Meineche-Schmidt, E., Moret, B.M.E. (ed) Experimental Algorithmics. Lecture Notes in Computer Science, vol. 2547, pp. 1–23. Springer, Berlin (2002)
3. Blelloch, G.E., Leiserson, C.E., Maggs, B.M., Plaxton, C.G., Smith, S.J., Zagha, M.: An experimental analysis of parallel sorting algorithms. Theor. Comput. Syst. **31**(2), 135–167 (1998)
4. Choi, J., Dongarra, J.J., Pozo, R., Walker, D.W.: ScaLAPACK: A scalable linear algebra library for distributed memory concurrent computers. In: The 4th Symp. the Frontiers of Massively Parallel Computations, pp. 120–127, McLean, VA (1992)
5. Culler, D.E., Karp, R.M., Patterson, D.A., Sahay, A., Schauser, K.E., Santos, E., Subramonian, R., von Eicken,T.: LogP: Towards a realistic model of parallel computation. In: 4th Symp. Principles and Practice of Parallel Programming, pp. 1–12. ACM SIGPLAN (1993)
6. Frigo, M., Johnson, S. G.: FFTW: An adaptive software architecture for the FFT. In: Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing, vol. 3, pp. 1381–1384, Seattle, WA (1998)
7. Frigo, M., Leiserson, C.E., Prokop, H., Ramachandran, S.: Cache-oblivious algorithms. In: Proc. 40th Ann. Symp. Foundations of Computer Science (FOCS-99), pp. 285–297, New York, NY, 1999. IEEE Press
8. Gropp, W., Lusk, E., Doss, N., Skjellum, A.: A high-performance, portable implementation of the MPI message passing interface standard. Technical report, Argonne National Laboratory, Argonne, IL, (1996) www.mcs.anl.gov/mpi/mpich/
9. Helman, D.R., JáJá, J.: Sorting on clusters of SMP's. In: Proc. 12th Int'l Parallel Processing Symp., pp. 1–7, Orlando, FL, March/April 1998
10. High Performance Fortran Forum. High Performance Fortran Language Specification, 1.0 edition, May 1993
11. Juurlink, B.H.H., Wijshoff, H.A.G.: A quantitative comparison of parallel computation models. ACM Trans. Comput. Syst. **13**(3), 271–318 (1998)
12. Message Passing Interface Forum. MPI: A message-passing interface standard. Technical report, University of Tennessee, Knoxville, TN, June 1995. Version 1.1
13. Moret, B.M.E., Bader, D.A., Warnow, T.: High-performance algorithm engineering for computational phylogenetics. J. Supercomput. **22**, 99–111 (2002) Special issue on the best papers from ICCS'01
14. Moret, B.M.E., Shapiro, H.D.: Algorithms and experiments: The new (and old) methodology. J. Univers. Comput. Sci. **7**(5), 434–446 (2001)
15. Nagel, W.E., Arnold, A., Weber, M., Hoppe, H.C., Solchenbach, K.: VAMPIR: visualization and analysis of MPI resources. Supercomputer 63. **12**(1), 69–80 (1996)
16. OpenMP Architecture Review Board. OpenMP: A proposed industry standard API for shared memory programming. www.openmp.org, October 1997
17. Reed, D.A., Aydt, R.A., Noe, R.J., Roth, P.C., Shields, K.A., Schwartz, B., Tavera, L.F.: Scalable performance analysis: The Pablo performance analysis environment. In: Skjellum, A., (ed) Proc. Scalable Parallel Libraries Conf., pp. 104–113, Mississippi State University, October 1993. IEEE Computer Society Press
18. Reussner, R., Sanders, P., Träff, J.: SKaMPI: A comprehensive benchmark for public benchmarking of MPI. Scientific Programming, 2001. accepted, conference version with Prechelt, L., Müller, M. In: Proc. EuroPVM/MPI (1998)
19. Vitter, J. S., Shriver, E.A.M.: Algorithms for parallel memory. I: Two-level memories. Algorithmica. **12**(2/3), 110–147 (1994)
20. Vitter, J. S., Shriver, E.A.M.: Algorithms for parallel memory II: Hierarchical multilevel memories. Algorithmica **12**(2/3), 148–169 (1994)
21. Whaley, R., Dongarra, J.: Automatically tuned linear algebra software (ATLAS). In: Proc. Supercomputing 98, Orlando, FL, November 1998. www.netlib.org/utk/people/JackDongarra/PAPERS/atlas-sc98.ps

Hitting Set

- ▶ Greedy Set-Cover Algorithms
- ▶ Set Cover with Almost Consecutive Ones

Hospitals/Residents Problem

1962; Gale, Shapley

David F. Manlove
Department of Computing Science,
University of Glasgow, Glasgow, UK

Keywords and Synonyms

College admissions problem; University admissions problem; Stable admissions problem; Stable assignment problem; Stable b-matching problem

Problem Definition

An instance I of the Hospitals/Residents problem (HR) [5,6,14] involves a set $R = \{r_1, \ldots, r_n\}$ of *residents* and a set $H = \{h_1, \ldots, h_m\}$ of *hospitals*. Each hospital $h_j \in H$ has a positive integral *capacity*, denoted by c_j. Also, each resident $r_i \in R$ has a *preference list* in which he ranks in strict order a subset of H. A pair $(r_i, h_j) \in R \times H$ is said

to be *acceptable* if h_j appears in r_i's preference list; in this case r_i is said to *find* h_j *acceptable*. Similarly each hospital $h_j \in H$ has a preference list in which it ranks in strict order those residents who find h_j acceptable. Given any three agents $x, y, z \in R \cup H$, x is said to *prefer* y to z if x finds each of y and z acceptable, and y precedes z on x's preference list. Let $C = \sum_{h_j \in H} c_j$.

Let A denote the set of acceptable pairs in I, and let $L = |A|$. An *assignment* M is a subset of A. If $(r_i, h_j) \in M$, r_i is said to be *assigned to* h_j, and h_j is *assigned* r_i. For each $q \in R \cup H$, the set of assignees of q in M is denoted by $M(q)$. If $r_i \in R$ and $M(r_i) = \emptyset$, r_i is said to be *unassigned*, otherwise r_i is *assigned*. Similarly, any hospital $h_j \in H$ is *under-subscribed*, *full* or *over-subscribed* according as $|M(h_j)|$ is less than, equal to, or greater than c_j, respectively.

A *matching* M is an assignment such that $|M(r_i)| \leq 1$ for each $r_i \in R$ and $|M(h_j)| \leq c_j$ for each $h_j \in H$ (i. e., no resident is assigned to an unacceptable hospital, each resident is assigned to at most one hospital, and no hospital is over-subscribed). For notational convenience, given a matching M and a resident $r_i \in R$ such that $M(r_i) \neq \emptyset$, where there is no ambiguity the notation $M(r_i)$ is also used to refer to the single member of $M(r_i)$.

A pair $(r_i, h_j) \in A \backslash M$ *blocks* a matching M, or is a *blocking pair* for M, if the following conditions are satisfied relative to M:

1. r_i is unassigned or prefers h_j to $M(r_i)$;
2. h_j is under-subscribed or prefers r_i to at least one member of $M(h_j)$ (or both).

A matching M is said to be *stable* if it admits no blocking pair. Given an instance I of HR, the problem is to find a stable matching in I.

Key Results

HR was first defined by Gale and Shapley [5] under the name "College Admissions Problem". In their seminal paper, the authors' primary consideration is the classical Stable Marriage problem (SM; see ▶ Stable Marriage and ▶ Optimal Stable Marriage), which is a special case of HR in which $n = m$, $A = R \times H$, and $c_j = 1$ for all $h_j \in H$ – in this case, the residents and hospitals are more commonly referred to as the men and women respectively. Gale and Shapley show that every instance I of HR admits at least one stable matching. Their proof of this result is constructive, i. e., an algorithm for finding a stable matching in I is described. This algorithm has become known as the *Gale/Shapley algorithm*.

An extended version of the Gale/Shapley algorithm for HR is shown in Fig. 1. The algorithm involves a sequence of *apply* and *delete* operations. At each iteration of the while loop, some unassigned resident r_i with a non-empty preference list applies to the first hospital h_j on his list, and becomes provisionally assigned to h_j (this assignment could subsequently be broken). If h_j becomes over-subscribed as a result of this assignment, then h_j rejects its worst assigned resident r_k. Next, if h_j is full (irrespective of whether h_j was over-subscribed earlier in the same loop iteration), then for each resident r_l that h_j finds less de-

```
M := ∅;
while (some resident r_i is unassigned and r_i has a non-empty list) {
    h_j := first hospital on r_i's list;
    /* r_i applies to h_j */
    M := M ∪ {(r_i, h_j)};
    if (h_j is over-subscribed) {
        r_k := worst resident in M(h_j) according to h_j's list;
        M := M\{(r_k, h_j)};
    }
    if (h_j is full) {
        r_k := worst resident in M(h_j) according to h_j's list;
        for (each successor r_l of r_k on h_j's list)
            delete the pair (r_l, h_j);
    }
}
```

Hospitals/Residents Problem, Figure 1
Gale/Shapley algorithm for HR

sirable than its worst resident r_k, the algorithm *deletes the pair* (r_l, h_j), which comprises deleting h_j from r_l's preference list and vice versa.

Given that the above algorithm involves residents applying to hospitals, it has become known as the *Resident-oriented* Gale/Shapley algorithm, or RGS algorithm for short [6, Sect. 1.6.3]. The RGS algorithm terminates with a stable matching, given an instance of HR [5,6, Theorem 1.6.2]. Using a suitable choice of data structures (extending those described in [6, Sect. 1.2.3]), the RGS algorithm can be implemented to run in $O(L)$ time. This algorithm produces the stable matching that is simultaneously best-possible for all residents [5,6, Theorem 1.6.2]. These observations may be summarized as follows:

Theorem 1 *Given an instance of HR, the RGS algorithm constructs, in O(L) time, the unique stable matching in which each assigned resident obtains the best hospital that he could obtain in any stable matching, whilst each unassigned resident is unassigned in every stable matching.*

A counterpart of the RGS algorithm, known as the *Hospital-oriented* Gale/Shapley algorithm, or HGS algorithm for short [6, Sect. 1.6.2], gives the unique stable matching that similarly satisfies an optimality property for the hospitals [6, Theorem 1.6.1].

Although there may be many stable matchings for a given instance I of HR, some key structural properties hold regarding unassigned residents and under-subscribed hospitals with respect to all stable matchings in I, as follows.

Theorem 2 *For a given instance of HR,*

- *the same residents are assigned in all stable matchings;*
- *each hospital is assigned the same number of residents in all stable matchings;*
- *any hospital that is under-subscribed in one stable matching is assigned exactly the same set of residents in all stable matchings.*

These results are collectively known as the "Rural Hospitals Theorem" (see [6, Sect. 1.6.4] for further details). Furthermore, the set of stable matchings in I forms a distributive lattice under a natural dominance relation [6, Sect. 1.6.5].

Applications

Practical applications of HR are widespread, most notably arising in the context of centralized automated matching schemes that assign applicants to posts (for example medical students to hospitals, school-leavers to universities, and primary school pupils to secondary schools). Perhaps the best-known example of such a scheme is the National Resident Matching Program (NRMP) in the US [16], which annually assigns around 31,000 graduating medical students (known as residents) to their first hospital posts, taking into account the preferences of residents over hospitals and vice versa, and the hospital capacities. Counterparts of the NRMP are in existence in other countries, including Canada [17], Scotland [18] and Japan [19]. These matching schemes essentially employ extensions of the RGS algorithm for HR.

Centralized matching schemes based largely on HR also occur in other practical contexts, such as school placement in New York [1], university faculty recruitment in France [3] and university admission in Spain [12].

Extensions of HR

One key extension of HR that has considerable practical importance arises when an instance may involve a set of couples, each of which submits a joint preference list over pairs of hospitals (typically in order that the members of a given couple can be located geographically close to one another, for example). The extension of HR in which couples may be involved is denoted by HRC; the stability definition in HRC is a natural extension of that in HR (see [6, Sect. 1.6.6] for a formal definition of HRC). It is known that an instance of HRC need not admit a stable matching (see [6, Section 1.6.6] and [14, Sect. 5.4.3]). Moreover, the problem of deciding whether an HRC instance admits a stable matching is NP-complete [13].

HR may be regarded as a many-one generalization of SM. A further generalization of SM is to a many-many stable matching problem, in which both residents and hospitals may be multiply assigned subject to capacity constraints. In this case, residents and hospitals are more commonly referred to as workers and firms respectively. There are two basic variations of the many-many stable matching problem according to whether (i) workers rank acceptable firms in order of preference and vice versa, or (ii) workers rank acceptable *subsets* of firms in order of preference and vice versa. Previous work relating to both models is surveyed in [4].

Other variants of HR may be obtained if preference lists include ties. This extension is again important from a practical perspective, since it may be unrealistic to expect a popular hospital to rank a large number of applicants in strict order, particularly if it is indifferent among groups of applicants. The extension of HR in which pref-

erence lists may include ties is denoted by HRT. In this context three natural stability definitions arise, so-called *weak stability*, *strong stability* and *super-stability* (see [8] for formal definitions of these concepts). Given an instance I of HRT, it is known that weakly stable matchings may have different sizes, and the problem of finding a maximum cardinality weakly stable matching is NP-hard (see ► Stable Marriage with Ties and Incomplete Lists for further details). On the other hand, in contrast to the case for weak stability, a super-stable matching in I need not exist, though there is an $O(L)$ algorithm to find a such a matching if one does [7]. Analogous results hold in the case of strong stability – in this case an $O(L^2)$ algorithm [8] was improved by an $O(CL)$ algorithm [10] and extended to the many-many case [11]. Furthermore, counterparts of the Rural Hospitals Theorem hold for HRT under each of the super-stability and strong stability criteria [7,15].

A further generalization of HR arises when each hospital may be split into several departments, where each department has a capacity, and residents rank individual departments in order of preference. This variant is modeled by the Student-Project Allocation problem [2]. Finally, the Stable Fixtures problem [9] is a non-bipartite extension of HR in which there is a single set of agents, each of whom has a capacity and ranks a subset of the others in order of preference.

Open Problems

Several approximation algorithms for finding a maximum cardinality weakly stable matching have been formulated, given an instance of HRT where each hospital has capacity 1 (see ► Stable Marriage with Ties and Incomplete Lists for further details). It remains open to extend these algorithms or to formulate effective heuristics for the case of HRT with arbitrary capacities. This problem is particularly relevant from the practical perspective, since as already noted in Sect. "Applications", hospitals may wish to include ties in their preference lists. In this case weak stability is the most commonly-used stability criterion, due to the guaranteed existence of such a matching. Attempting to match as many residents as possible motivates the search for large weakly stable matchings.

URL to Code

Ada implementations of the RGS and HGS algorithms for HR may be found via the following URL: http://www.dcs.gla.ac.uk/research/algorithms/stable.

Cross References

- ► Optimal Stable Marriage
- ► Ranked Matching
- ► Stable Marriage
- ► Stable Marriage and Discrete Convex Analysis
- ► Stable Marriage with Ties and Incomplete Lists
- ► Stable Partition Problem

Recommended Reading

1. Abdulkadiroğlu, A., Pathak, P.A., Roth, A.E.: The New York City high school match. Am. Economic. Rev. **95**(2), 364–367 (2006)
2. Abraham, D.J., Irving, R.W., Manlove, D.F.: Two algorithms for the Student-Project allocation problem. J. Discret. Algorithms **5**(1), 73–90 (2007)
3. Baïou, M., Balinski, M.: Student admissions and faculty recruitment. Theor. Comput. Sci. **322**(2), 245–265 (2004)
4. Bansal, V., Agrawal, A., Malhotra, V.S.: Stable marriages with multiple partners: efficient search for an optimal solution. In: Proceedings of ICALP '03: the 30th International Colloquium on Automata, Languages and Programming. Lecture Notes in Computer Science, vol. 2719, pp. 527–542. Springer, Berlin (2003)
5. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. Am. Math. Month. **69**, 9–15 (1962)
6. Gusfield, D., Irving, R.W.: The Stable Marriage Problem: Structure and Algorithms. MIT Press, Cambridge (1989)
7. Irving, R.W., Manlove, D.F., Scott, S.: The Hospitals/Residents problem with Ties. In: Proceedings of SWAT 2000: the 7th Scandinavian Workshop on Algorithm Theory. Lecture Notes in Computer Science, vol. 1851, pp. 259–271. Springer, Berlin (2000)
8. Irving, R.W., Manlove, D.F., Scott, S.: Strong stability in the Hospitals/Residents problem. In: Proceedings of STACS 2003: the 20th Annual Symposium on Theoretical Aspects of Computer Science. Lecture Notes in Computer Science, vol. 2607, pp. 439–450. Springer, Berlin (2003)
9. Irving, R.W., Scott, S.: The stable fixtures problem – a many-to-many extension of stable roommates. Discret. Appl. Math. **155**, 2118–2129 (2007)
10. Kavitha, T., Mehlhorn, K., Michail, D., Paluch, K.: Strongly stable matchings in time $O(nm)$ and extension to the Hospitals-Residents problem. In: Proceedings of STACS 2004: the 21st International Symposium on Theoretical Aspects of Computer Science. Lecture Notes in Computer Science, vol. 2996, pp. 222–233. Springer, Berlin (2004)
11. Malhotra, V.S.: On the stability of multiple partner stable marriages with ties. In: Proceedings of ESA '04: the 12th Annual European Symposium on Algorithms. Lecture Notes in Computer Science, vol. 3221, pp. 508–519. Springer, Berlin (2004)
12. Romero-Medina, A.: Implementation of stable solutions in a restricted matching market. Rev. Economic. Des. **3**(2), 137–147 (1998)
13. Ronn, E.: NP-complete stable matching problems. J. Algorithms **11**, 285–304 (1990)
14. Roth, A.E., Sotomayor, M.A.O.: Two-sided matching: a study in game-theoretic modeling and analysis. Econometric Society

Monographs, vol. 18. Cambridge University Press, Cambridge, UK (1990)
15. Scott, S.: A study of stable marriage problems with ties. Ph. D. thesis, University of Glasgow, Dept. Comput. Sci. (2005)
16. http://www.nrmp.org/ (National Resident Matching Program website)
17. http://www.carms.ca (Canadian Resident Matching Service website)
18. http://www.nes.scot.nhs.uk/sfas/ (Scottish Foundation Allocation Scheme website)
19. http://www.jrmp.jp (Japan Resident Matching Program website)