1-1-1995

Approximation Algorithms: Good Solutions to Hard Problems

Ran Libeskind-Hadas
Harvey Mudd College

THE COMPUTER SCIENCE SAMPLER
Edited by: **Catherine C. McGeoch**

Approximation Algorithms: Good Solutions to Hard Problems

Ran Libeskind-Hadas

1. INTRODUCTION. Consider a computer network represented by an undirected graph where the vertices represent computer nodes and the edges represent links between the nodes. Since some of the links in the network may become faulty, link testing devices are placed at some of the nodes. A tester at a particular node can test all links incident to that node. Since the testers are expensive, however, we wish to deploy the minimum number of these devices such that every link is incident to at least one node containing a tester. In graph theoretic terms, a *vertex cover* is a subset of the vertices such that every edge is incident to at least one vertex in this set. Our objective then is to find a minimum vertex cover. This is known as the *vertex cover problem*.

The vertex cover problem is one of many computational problems known to be NP-complete (see "Turing Machines and Computational Complexity" in the January 1994 issue of the *Monthly*). NP-complete problems can be solved in a number of steps that grows exponentially in the size of the problem, but no "efficient" algorithms are known for these problems. By "efficient" we mean that the number of steps, or *time*, is bounded by some polynomial in the size of the problem. In fact, not only are no polynomial time algorithms known for NP-complete problems, but the theory of NP-completeness tells us that if a polynomial time algorithm is found for *any* single NP-complete problem, then *all* NP-complete problems are solvable in polynomial time. Theoretical computer scientists generally believe, but have so far been unable to prove, that there do not exist polynomial time algorithms for NP-complete problems.

Let us reconsider the vertex cover problem. A simple algorithm enumerates all possible subsets of the vertices in increasing order of cardinality, and tests each set to see if it is a vertex cover for the graph. This process terminates when the first vertex cover is discovered. In the worst case, this algorithm will terminate at the very last set, since the set of all vertices is certainly a vertex cover. For a graph with n vertices, essentially 2^n subsets must be considered by the algorithm in the worst case. For example, if this algorithm was applied to deploy link testers in a network with 100 nodes and the algorithm was executed on a supercomputer capable of considering 10^{12} subsets per second, the computer would require over 40 billion years to consider all possible subsets! Since the vertex cover problem is NP-complete, it is unlikely that a dramatically faster algorithm can be found for this problem.

What, then, can we do when confronted with an NP-complete problem such as the vertex cover problem? One approach is to use *heuristic algorithms*. These algorithms employ simple rules of thumb and, consequently, tend to be very fast. However, heuristics do not guarantee that an optimal solution, or even anything close to an optimal solution, will be found. A natural heuristic for the vertex problem, for example, begins by selecting a vertex of highest degree (that is, the vertex with the maximum number of edges incident to it), in this way "covering" as many edges as possible. This step is repeated until every edge is covered. Unfortunately, there are many graphs for which this heuristic performs quite poorly. In fact, for any positive value of α, it is possible to construct a graph such that the solution found by the heuristic on this graph is α times larger than the optimal solution [**8**]! It would certainly be much more desirable to have an algorithm that finds a vertex cover that is guaranteed to be at most some fixed constant times larger than optimal. Such an algorithm is called an *approximation algorithm* and an approximation algorithm that runs in polynomial time is called a *polynomial time approximation algorithm*. The mere existence of polynomial time approximation algorithms is somewhat surprising, since we have no efficient way of determining optimal solutions to NP-complete problems. Using a number of clever techniques, however, researchers have discovered approximation algorithms for many important NP-complete problems.

2. APPROXIMATION ALGORITHMS. We begin by describing a surprisingly simple polynomial time approximation algorithm for the vertex cover problem. Let $G = (V, E)$ denote a given graph. The algorithm comprises the following steps:

1. S is initially the empty set.
2. While edges remain in the graph, select an edge (u, v) arbitrarily. Add u and v to the set S and remove u, v, and all edges incident to these vertices from G.

We claim that when this algorithm terminates, S is a vertex cover for graph G and the cardinality of S is at most twice that of a vertex cover of minimum size.

The first part of this claim is easily established, since at any step the edges remaining in G are exactly those edges that are not yet covered by vertices in S. To verify the second part of this claim, let $E' = \{e_1, \ldots, e_k\}$ denote the set of edges selected by the algorithm. By definition, every vertex cover must include at least one of the two endpoints of each of these edges. Observe also that these edges have no vertices in common, since once an edge is selected, both of its endpoints and all incident edges are removed from G. Therefore, every vertex cover, and in particular a minimum vertex cover, must have size at least k. However, the vertex cover constructed by this algorithm has size exactly $2k$ since S consists of both endpoints of each edge in E'. Thus, this algorithm obtains a vertex cover that is at most twice as large as a minimum vertex cover. Finally, it is not difficult to show that this algorithm runs in a number of steps that grows polynomially (in fact linearly) in the number of vertices and edges in the graph.

We have demonstrated a polynomial time approximation algorithm that finds vertex covers that are at most twice as large as optimal. In fact, our analysis is tight: It is not difficult to construct graphs for which this algorithm finds vertex covers that are exactly twice as large as optimal. In general, let A denote an approximation algorithm and let $A(I)$ denote the size of the solution obtained by this algorithm for a particular instance I of the problem. Similarly, let $\mathrm{OPT}(I)$ denote the size of an optimal solution for instance I of the problem. We define the

ratio $R_A(I)$ by

$$R_A(I) = \frac{A(I)}{\mathrm{OPT}(I)}$$

and the *absolute performance ratio* R_A of algorithm A is defined by

$$\inf\{r | R_A(I) \leq r, \text{ for all instances } I \text{ of the problem}\}.$$

Is it possible that more sophisticated approximation algorithms for the vertex cover problem achieve absolute performance ratios better than 2? The answer is indeed "yes", although surprisingly the best algorithm currently known improves this ratio only slightly to $2 - (\log\log n/2\log n)$ where n is the number of vertices in the graph [2]. Thus, asymptotically, this algorithm is no better than our simple approximation algorithm. Generalizing the notion of the absolute performance ratio, the *asymptotic performance ratio* R_A^∞ of algorithm A is defined by

$$R_A^\infty = \inf\{r | \exists N_0, \text{ s.t. } R_A(I) \leq r, \text{ for all instances } I \text{ of the problem s.t. } \mathrm{OPT}(I) \geq N_0\}.$$

We now turn to another problem, known as the *bin packing problem*, for which approximation algorithms with much better absolute and asymptotic performance ratios are known. In the bin packing problem we are given a finite set of items, each with size between 0 and 1. Our objective is to pack these items into unit capacity bins, minimizing the total number of bins used. More formally, let $I = \{s_1, s_2, \ldots, s_n\}, \forall I, s_i \in [0, 1]$ denote the set of items. We wish to partition I into disjoint subsets (bins) $B_1, B_2, \ldots, B_k$ such that $\forall i, \Sigma_{s_j \in B_i} s_j \leq 1$ and k is as small as possible.

Like the vertex cover problem, the bin packing problem is NP-complete. Like the vertex cover problem as well, a very simple polynomial time approximation algorithm for bin packing finds solutions that are at most twice as large as optimal. This approximation algorithm, known as the *first fit algorithm*, operates as follows: Select one item at a time in arbitrary order and place this item in the first bin which can accommodate it.

The ratio of 2 for the first fit algorithm follows from two observations. First, we show that when the algorithm terminates, at most one of the used bins is half full or less. Assume that this is not the case. Then when the algorithm terminates, there are two bins B_i and $B_j, i < j$, that are each at most half full. Then the last item placed in B_j clearly has size at most $\frac{1}{2}$. Since bin B_i has capacity at least $\frac{1}{2}$ throughout execution of the algorithm, the first fit algorithm would have placed this item in B_i rather than in B_j, a contradiction. Now, letting $\mathrm{FF}(I)$ denote the number of bins used by the first fit algorithm on a given problem instance I, this observation implies that

$$\mathrm{FF}(I) < \left\lceil 2 \sum_{s_i \in I} s_i \right\rceil.$$

Our second observation is that the total number of bins used in any solution is at least the sum of the sizes of all the items. In particular, letting $\mathrm{OPT}(I)$ denote the number of bins used in an optimal solution, we have

$$\left\lceil \sum_{s_i \in I} s_i \right\rceil \leq \mathrm{OPT}(I).$$

Combining these two observations, we now have

$$\mathrm{FF}(I) < 2 \cdot \mathrm{OPT}(I)$$

and thus $R_{FF} < 2$.

The above analysis shows that the absolute performance ratio of the first fit algorithm is less than 2. In fact, more careful analysis shows that for all instances I of the bin packing problem

$$\mathrm{FF}(I) \leq \frac{17}{10}\mathrm{OPT}(I) + 2$$

and that there exist instances I with arbitrarily large values of $\mathrm{OPT}(I)$ such that

$$\mathrm{FF}(I) \geq \frac{17}{10}(\mathrm{OPT}(I) - 1).$$

Therefore, the asymptotic performance ratio of the first fit algorithm R_{FF}^{∞} is in fact 1.7. Moreover, a minor modification of the first fit algorithm achieves an even better performance ratio. The modified algorithm, known as the *first fit decreasing* algorithm is identical to the first fit algorithm except that items are selected for insertion in the bins in decreasing order of size. The analysis of this algorithm, which is quite long and complicated, shows that this modification results in an asymptotic performance ratio of $11/9$ [5].

3. APPROXIMATION SCHEMES. Do approximation algorithms exist for all NP-complete problems? Unfortunately, it appears that the answer to this question is probably "no". For many NP-complete problems, including the infamous traveling salesperson problem, it can be shown that the existence of a polynomial time approximation algorithm with any fixed performance ratio would imply that P = NP, that is, all NP-complete problems could be solved exactly in polynomial time.

On the other hand, for some NP-complete problems we can do even better than finding approximation algorithms with fixed performance ratios. For many important problems there exist families of approximation algorithms that allow us to obtain performance ratios arbitrarily close to 1 in exchange for increasingly larger polynomial time bounds. A *polynomial time approximation scheme (PTAS)* is a family of approximation algorithms $\{A_\epsilon | \epsilon > 0\}$ where for each $\epsilon > 0$, A_ϵ is a polynomial time approximation algorithm with absolute ratio bound R_{A_ϵ} at most $1 + \epsilon$.

Although it is unlikely that PTAS can be found for all NP-complete problems (since this would imply approximation algorithms for all NP-complete problems and thus that P = NP), it is natural to ask whether they at least exist for all problems with approximation algorithms. In a result hailed by many theoretical computer scientists as one of the most important in the field in over two decades, a group of researchers from Berkeley, Stanford, and Bell Labs showed in 1992 [1] that this too would imply that P = NP. Specifically, it was shown that if a PTAS exists for any problem in a rich subset of the NP-complete problems known as the MAXSNP-complete problems, then P = NP. Among the many important problems known to be MAXSNP-complete is the vertex cover problem.

4. FURTHER READING. Garey and Johnson's [4] classic text offers an eminently readable introduction to NP-completeness, including a discussion of approximation algorithms and schemes. Texts by Papadimitriou and Steiglitz [8] and Cormen, Leiserson, and Rivest [3] have very good discussions and a number of illustrative examples. Motwani's technical report on approximation algorithm [7] is also excellent. Finally, the recent result on the intractability on the hardness of

MAXSNP-complete appeared in [1] accompanied by an entertaining story in the New York Times [6].

REFERENCES

1. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, 1992.
2. R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–45, 1985.
3. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw-Hill and MIT Press, 1990.
4. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
5. D. Johnson. *Near-Optimal Bin Packing Algorithms*. PhD thesis, Dept. of Mathematics, Massachusetts Institute of Technology, 1973.
6. G. Kolata. New short cut found for long math proofs. *New York Times*, April 7, 1992.
7. R. Motwani. Lecture notes on approximation algorithms. Technical report, Dept. of Computer Science, Stanford University, 1992.
8. C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.

Department of Computer Science
Harvey Mudd College
301 E. 12th Street
Claremont, CA 91711
hadas@cs.hmc.edu

Mathematics, while giving no quick remuneration, like the art of stenography or the craft of bricklaying, does furnish the power of deliberate thought and accurate statement, and to speak the truth is one of the most social qualities a person can possess. Gossip, flattery, slander, deceit, all spring from a slovenly mind that has not been trained in the power of truthful statement, which is one of the highest utilities.

—*S. T. Dutton*