APPROXIMATION ALGORITHMS

P. Crescenzi
Università di Firenze, Italy

Keywords: approximation algorithm, approximation scheme, complexity class, gap technique, optimization, performance ratio, reducibility

Contents

Summary

In many practical applications of computers (for example in the area of logistics, scheduling, optimal vehicle routing and facility location problems) we need to solve problems that are known to be computationally intractable due to their inherent complexity. In order to approach these problems a large body of approximation algorithms has been provided by the specialists in the field. In particular, various heuristic techniques have been proposed, such as local search, Greedy techniques, dynamic programming based heuristics, approximation schemes. More recently, other heuristic techniques based on randomization and mathematical programming, have been proposed. Moreover, several techniques (such as the gap technique, the PCP theorem and the approximation preserving reductions) have been introduced in order to prove limits on the approximability of specific combinatorial optimization problems. The goal of this chapter is to provide the reader with a review of these basic techniques useful to design approximation algorithms or to prove that no such algorithms exist.

1. Introduction

In real life and in many application areas, computationally hard optimization problems arise, that is, problems for which no efficient algorithm computing an optimal solution is known. This is true for areas like logistics, scheduling, optimal vehicle routing, facility location and so on. But since these problems arise in practical contexts, they have to be solved anyway: To this aim, we have several alternatives.

The first and the foremost approach consists of using exponential-time algorithms, which are not efficient: In this case we can hope that, when these algorithms are faced with real instances, the running time is not exponential but is more reasonable. This is a feasible approach and most people probably follow it. But, we should realize that, if we follow this approach it can so happen that for some instances the chosen algorithm will require years (if not centuries) in order to compute an optimal solution.

Another approach is to try to analyze our algorithm from an average-case point of view. In this case, we hope that the real distribution of the input instances is similar to the distribution we use to perform the probabilistic analysis. The main problem with this approach is that very often we do not know what the probability distribution of the real instances is. Usually, people assume that instances are distributed according to the uniform distribution but there is no evidence that this is a realistic assumption.

A third alternative, instead, consists of relaxing the optimality constraint of the computed solution in order to reduce the time complexity. This leads to the development of polynomial-time approximation algorithms. This review will deal with this kind of algorithms and its main goal is to provide the reader with some basic techniques useful to design approximation algorithms or to prove that no such algorithms exist, for specific combinatorial optimization problems.

In particular, we will first introduce the formal definition of a combinatorial optimization problem and the notion of performance ratio as a quality measure of a solution with respect to a specific instance. We will then describe several approximation algorithms based on well-known techniques which are widely used in the development of computer algorithms (such as local search or dynamic programming). While describing the algorithmic techniques, we will also introduce some of the most studied combinatorial optimization problems in the field of approximation algorithms (such as the minimum bin packing problem and the maximum satisfiability problem). Once we have shown these positive results, it will seem natural to ask ourselves whether there are some limits to the approximability of combinatorial optimization problems. In order to answer this question, we will introduce the gap technique which is the main technique to obtain such kind of results. Finally, we will briefly describe more advanced techniques both for developing approximation algorithms and for proving negative results.

2. Combinatorial Optimization Problems

The main ingredients of a combinatorial optimization problem are the instances, the solutions of an instance and the measure of these solutions. In particular, a combinatorial optimization problem consists of:

- The set I of *instances*.
- For each instance $x \in I$, the finite set $sol(x)$ of *feasible solutions* of x.

  For each instance $x \in I$ and for each feasible solution $y \in sol(x)$, the *measure* $m(x, y)$ that is a real number.

- The *goal* of the problem that can be either *maximizing* or *minimizing* the measure of the solution: In the first case, we say that the problem is a *maximization problem* while in the second case we say it is a *minimization problem*.

Solving a combinatorial optimization problem thus consists of computing, for each instance x, an optimal solution, that is, a solution whose measure is either maximal or minimal (depending on the goal of the problem) among all feasible solutions of x: The measure of an optimal solution is denoted by $opt(x)$.

Unfortunately, for many combinatorial optimization problems no polynomial-time algorithm is known that, for any instance x, computes an optimal solution of x: Indeed, it is known that the existence of such an algorithm for just one of these problems would imply that all of them are efficiently solvable (which is considered an unlikely event). For a thorough discussion of the concept of the complexity of algorithms see *Complexity Theory*. This is true for problems arising in different application areas such as logistics, scheduling, optimal vehicle routing, facility location and so on. For this reason, in order to deal with these computationally hard problems we have to give up the optimality constraint and to consider the possibility of computing approximate solutions.

2.1. Performance Ratio

Several quality measures of a solution of an instance of a combinatorial optimization problem have been introduced in the literature. In the following, we will focus on one of them that is, in our opinion, the most widely used: This measure is called the *performance ratio* of a solution y with respect to an instance x and is defined as follows:

$$R(x, y) = \max\left(\frac{\mathrm{opt}(x)}{\mathrm{m}(x, y)}, \frac{\mathrm{m}(x, y)}{\mathrm{opt}(x)}\right). \qquad (1)$$

Observe that, by using the above definition, we do not have to distinguish between maximization problems and minimization problems. In both cases, in fact, the performance ratio will be a number greater than 1, and the closer this number is to 1, the better is the quality of the solution computed by our algorithm.

2.2. Approximation Algorithms

We are now in a position to formally define the notion of approximation algorithm. In particular, for any real number r, we say that a polynomial-time algorithm $\mathcal{A}$ is an r-

approximation algorithm for a combinatorial optimization problem $\mathcal{P}$ if, for any instance x of $\mathcal{P}$, $\mathcal{A}$ with input x returns an *r-approximate solution* y whose performance ratio is bounded by r, that is,

$$R(x, \mathcal{A}(x)) \leq r\,. \qquad (2)$$

-
-
-

Bibliography

Ausiello G., Crescenzi P., Gambosi G., Kann V., Marchetti-Spaccamela A., and Protasi M. (2000). *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*, 524 pp. New York: Springer, Heidelberg. [Introduction to the theory of approximation algorithms]

Garey M.R. and Johnson D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*, 338 pp. San Francisco: Freeman. [Introduction to the theory of computationally hard problems]

Hochbaum D.S. ed. (1997). *Approximation Algorithms for NP-hard Problems*. Boston, Massachusetts: PWS. [Series of papers about basic and advanced topics in the field of approximation algorithms]

Biographical Sketch

Pierluigi Crescenzi was born in Rome, Italy, in 1961. He received a degree in Mathematics from the University of Rome, La Sapienza, in 1984 and the Ph.D. in Computer Science from the University of Rome, La Sapienza, in 1991. From 1989 to 1992, he joined the University of L'Aquila as an Assistant Professor. In 1992 he became Associate Professor at the Department of Computer Science of the University of Rome, La Sapienza. In 1997, he moved to the Department of Systems and Computer Science of the University of Florence, where he teaches courses in computer architectures and networks. He became a full professor in 2001. Professor Crescenzi has more than seventy publications, including articles in scientific journals such as *SIAM Journal on Computing, Journal of Computer Systems Sciences, Information and Computation, Theory of Computer Systems*, and *Theoretical Computer Science*. He is co-author of the books *Introduction to the Theory of Complexity* and *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*. He is on the editorial board of *Electronic Notes in Theoretical Computer Science*, and has been a member of the program committee of several international conferences and workshops.