Optimization and approximation on systems of geometric objects

van Leeuwen, E.J.

Chapter 2

Primer on Optimization and Approximation

A key ingredient of this thesis is approximation algorithms for certain optimization problems. Hence it is useful to formally define what an optimization problem is and what types of approximation algorithms we distinguish.

In addition to defining the classic types of approximation schemes, we propose a new type of approximation scheme, the asymptotic approximation scheme. Many of the approximation algorithms that we will encounter later are essentially an asymptotic approximation scheme. We prove however that two classes of optimization problems having such a scheme, namely FPTAS$^\omega$ and FIPTAS$^\omega$, both coincide with the well-known class EPTAS. Hence instead of placing problems in a class of problems with an asymptotic approximation scheme, we can place these problems in the more familiar EPTAS class as well.

2.1 Classic Notions

To make formal statements about (equivalences among) classes of approximation schemes, we have to be precise about the machine model that we use, the type of problems that are considered, and the definitions of the studied classes. Throughout, we assume the random access machine model with logarithmic costs and representations in bits. This machine model is polynomially equivalent to the classic Turing machine and thus defines equivalent complexity classes up to polynomial factors. Furthermore, all numbers in this chapter are assumed to be rationals, unless otherwise specified.

Using this model, we study optimization problems following the definitions as can be found for example in Ausiello et al. [19].

Definition 2.1.1 *An* optimization problem P *is characterized by four properties:*

- *a set of* instances *(bitstrings)* I_P*;*
- *a function* S_P *that maps instances of* P *to (nonempty) sets of* feasible solutions *(bitstrings) for these instances;*

- *an* objective function m_P *that gives for each pair* (x, y) *consisting of instance* $x \in I_P$ *and solution* $y \in S_P(x)$ *a positive integer* $m_P(x, y)$*, the* objective value*;*
- *a* goal $\text{goal}_P \in \{\min, \max\}$ *depending on whether* P *is a minimization or a maximization problem.*

We denote by $S^*_P(x) \subseteq S_P(x)$ the set of optimal solutions for an instance $x \in I_P$, i.e. for every $y^* \in S^*_P(x)$,

$$m_P(x, y^*) = \text{goal}_P\{m_P(x, y) \mid y \in S_P(x)\}.$$

The objective value attained by an optimal solution for an instance x is denoted by $m^*_P(x)$.

Definition 2.1.2 *An optimization problem* P *is in the class* NPO *if*

- *the set of instances* I_P *can be recognized in polynomial time;*
- *there is a (monotone nondecreasing) polynomial (say* q_P*) such that* $|y| \leq q_P(|x|)$ *for any instance* $x \in I_P$ *and any feasible solution* $y \in S_P(x)$*;*
- *for any instance* $x \in I_P$ *and any* y *with* $|y| \leq q_P(|x|)$*, one can decide in polynomial time whether* $y \in S_P(x)$*;*
- *there is a (monotone nondecreasing) polynomial (say* r_P*) such that the objective function* m_P *is computable in* $r_P(|x|, |y|)$ *time for any* $x \in I_P$ *and* $y \in S_P(x)$*.*

All problems considered below will be in NPO and all considered classes will be subclasses of NPO.

Note that for any problem $P \in$ NPO and any $n \in \mathbb{N}$ the maximum objective value of instances of size n, i.e. $\max\{m_P(x, y) \mid x \in I_P, |x| = n, y \in S_P(x)\}$, is bounded by $2^{r_P(n, q_P(n))}$, as the objective function value of any $x \in I_P$ and $y \in S_P(x)$ can be represented by at most $r_P(|x|, |y|) \leq r_P(|x|, q_P(|x|))$ bits. Let $M_P(n) = 2^{r_P(n, q_P(n))}$.

If one compares NPO to the class NP, then PO is the 'equivalent' of P. *PO* is the class of problems in NPO for which an optimal solution $y^* \in S^*_P(x)$ can be computed in time polynomial in $|x|$ for any $x \in I_P$. Paz and Moran [221] proved that P=NP implies PO=NPO and vice versa. Because it is not expected that all problems in NPO also fall in PO, several classes have been defined that contain NPO-problems for which an *approximate* solution can be found in polynomial time. Approximation algorithms are classified by two properties: their running time and their approximation ratio.

Definition 2.1.3 ([19, 115]) *For an optimization problem* $P \in$ *NPO, any* $x \in I_P$*, and any* $y \in S_P(x)$*, the* approximation ratio *achieved by* y *for* x *is*

$$R(x, y) = \max\left\{\frac{m_P(x, y)}{m^*_P(x)}, \frac{m^*_P(x)}{m_P(x, y)}\right\}.$$

Problem class	Running time	Approx. ratio
APX	Polynomial in $\lvert x\rvert$	c
PTAS	Polynomial in $\lvert x\rvert$ (for every fixed ϵ)	$(1+\epsilon)$
FPTAS	Polynomial in $\lvert x\rvert$ and $1/\epsilon$	$(1+\epsilon)$
FIPTAS	Polynomial in $\lvert x\rvert$	$(1+\epsilon)$
PO	Polynomial in $\lvert x\rvert$	1

Table 2.1: Problem classes and the distinguishing properties of the approximation algorithms admitted by problems in a particular class.

We say y is within (a factor) r *of* $m^*_P(x)$ *if* $R(x,y) \leq r$. *The* approximation ratio *of an algorithm $\mathcal{A}$ is defined as*

$$R_{\mathcal{A}} = \max\{R(x, \mathcal{A}(x)) \mid x \in I_P\}.$$

Observe that irrespective of whether $\text{goal}_P = \min$ or $\text{goal}_P = \max$, the approximation ratio is a number that is at least 1. Sometimes however, in the case where $\text{goal}_P = \max$, we will instead use

$$R'(x,y) = \frac{1}{R(x,y)} = \frac{m_P(x,y)}{m^*_P(x)},$$

which is at most 1. To keep the exposition simple, we will only use $R(x,y)$ in this chapter.

Any textbook on approximation algorithms covers at least the classes of Table 2.1. The table should be interpreted as follows: *PTAS* for instance is the class of optimization problems P in NPO having a *ptas*, i.e. having an algorithm $\mathcal{A}$ such that for any instance $x \in I_P$ and any $\epsilon > 0$, $\mathcal{A}(x,\epsilon)$ runs in time polynomial in $|x|$ for every fixed ϵ and the solution output by $\mathcal{A}(x,\epsilon)$ has approximation ratio $(1+\epsilon)$. We use lowercase for a scheme name and uppercase for the name of the corresponding class (i.e. ptas and PTAS).

APX is the class of problems having a *constant-factor approximation algorithm*, meaning a polynomial-time algorithm that returns a solution of approximation ratio c, for some fixed constant c.

The class FIPTAS (Fully Input-Polynomial-Time Approximation Scheme) in Table 2.1 is a new class. Clearly, FIPTAS=PO (use $\epsilon = 1/M_P(|x|)$), but the reason for defining this class will become apparent later.

A relatively new class that is of increasing interest is EPTAS [26, 53].

Definition 2.1.4 *Algorithm $\mathcal{A}$ is an* efficient polynomial-time approximation scheme (eptas) *for problem P if there is a computable function $f : \mathbb{Q}^+ \to \mathbb{N}$ such that for any $x \in I_P$ and any $\epsilon > 0$, $\mathcal{A}(x,\epsilon)$ runs in time $f(1/\epsilon)$ times a polynomial in $|x|$ and the solution output by $\mathcal{A}(x,\epsilon)$ has approximation ratio $(1+\epsilon)$. An NPO-problem is in the class* EPTAS *if and only if it has an eptas.*

The popularity of eptas is not only due to the separate dependence on $1/\epsilon$ and instance size in the running time, but also to the beautiful relation to the widely researched class FPT: any problem admitting an eptas is also in FPT w.r.t. its standard parameterization [26, 53]. An intriguing exploration of the type of problems that admit an eptas may be found in Cai et al. [48].

It is well-known that PO $\subseteq$ FPTAS $\subseteq$ EPTAS $\subseteq$ PTAS $\subseteq$ APX $\subseteq$ NPO. In most cases, the inclusion is strict (unless P=NP), except that EPTAS $\subset$ PTAS unless FPT=W[1] [26, 53]. The question whether FPT=W[1] is an open problem for fixed-parameter complexity theory akin to the question whether P=NP for classic complexity theory (see e.g. Downey and Fellows [92]).

2.2 Asymptotic Approximation Schemes

We introduce a new type of approximation scheme, the asymptotic approximation scheme.

Definition 2.2.1 *An approximation scheme $\mathcal{A}$ for an optimization problem $P \in NPO$ is* asymptotic *if there is a computable function $a : \mathbb{Q}^+ \to \mathbb{N}$ (the* threshold function*) such that for any $\epsilon > 0$ and any $x \in I_P$, it returns a $y \in S_P(x)$ and if $|x| \geq a(1/\epsilon)$, then y is within $(1+\epsilon)$ of $m^*_P(x)$.*

This definition leads to the following classes of asymptotic approximation schemes.

Class	Running time	Approx. ratio
PTAS$^\omega$	Polynomial in $\lvert x\rvert$ (for every fixed ϵ)	$(1+\epsilon)$ if $\lvert x\rvert \geq a(1/\epsilon)$
FPTAS$^\omega$	Polynomial in $\lvert x\rvert$ and $1/\epsilon$	$(1+\epsilon)$ if $\lvert x\rvert \geq a(1/\epsilon)$
FIPTAS$^\omega$	Polynomial in $\lvert x\rvert$	$(1+\epsilon)$ if $\lvert x\rvert \geq a(1/\epsilon)$

Example 2.2.2 *Maximum Independent Set has a fiptas$^\omega$ on bounded ply disk graphs (see the proof of Theorem 7.3.9). Disk graphs are intersection graphs of disks in the plane. A set of disks has ply γ if γ is the smallest integer such that any point of the plane is strictly contained in at most γ disks. One can find in $\mathrm{O}(n^{10}\log^4 n)$ time an independent set of a graph induced by n disks. If for some $\epsilon > 0$ an odd integer k can be chosen such that $\max\{5, 4(1+\epsilon)/\epsilon\} \leq k \leq c_1 \log n / \log(c_2\gamma)$ (where c_1, c_2 are fixed constants), then this independent set will be within $(1+\epsilon)$ of the optimum. If $\gamma = \gamma(n) = \mathrm{O}(n^{\mathrm{o}(1)})$, such an integer exists if $|x| \geq n \geq a(1/\epsilon)$ for some function a.*

We start with some easy observations about the asymptotic classes.

Proposition 2.2.3 *The following relations hold:*

- *FIPTAS$^\omega$ $\subseteq$ FPTAS$^\omega$ $\subseteq$ PTAS$^\omega$ and*
- *FIPTAS $\subseteq$ FIPTAS$^\omega$, FPTAS $\subseteq$ FPTAS$^\omega$, PTAS $\subseteq$ PTAS$^\omega$.*

The relations given by this proposition are straightforward and one might expect that the inclusions are strict under some hardness condition. However, this turns out not to be true for all of them. We can in fact prove very interesting equivalences and at the same time tie these new classes to existing approximation classes, in particular to EPTAS.

Theorem 2.2.4 *EPTAS = FPTAS$^\omega$ = FIPTAS$^\omega$.*

Proof: We first show that EPTAS $\subseteq$ FIPTAS$^\omega$. Let $P \in$ EPTAS and let $\mathcal{A}$ be an eptas for P with running time at most $p(|x|) \cdot f(1/\epsilon)$ for some computable function f and polynomial p. Construct a fiptas$^\omega$ for P as follows. Given an arbitrary instance $x \in I_P$ and an arbitrary $\epsilon > 0$, run $\mathcal{A}(x, \epsilon)$ for $p(|x|) \cdot |x|$ time steps. If $\mathcal{A}(x, \epsilon)$ finishes, return the solution given by $\mathcal{A}(x, \epsilon)$. Otherwise, return $\mathcal{A}(x, 1/2)$. This algorithm clearly runs in time polynomial in $|x|$ and always returns a feasible solution. Furthermore if $|x| \geq f(1/\epsilon)$, $\mathcal{A}(x, \epsilon)$ always finishes and returns a feasible solution with approximation ratio $(1+\epsilon)$. Hence we constructed a fiptas$^\omega$ for P with $a = f$.

We next prove that FPTAS$^\omega$ $\subseteq$ EPTAS. Let $P \in$ FPTAS$^\omega$ and let $\mathcal{A}$ be an fptas$^\omega$ for P with threshold function a. Construct an eptas as follows. Given an arbitrary instance $x \in I_P$ and an arbitrary $\epsilon > 0$, compute $a(1/\epsilon)$. By assumption, $a(1/\epsilon)$ is computable. The amount of time it takes to compute $a(1/\epsilon)$ is some computable function depending on $1/\epsilon$. If $|x| \geq a(1/\epsilon)$, simply compute and return $\mathcal{A}(x, \epsilon)$ in time polynomial in $|x|$ and $1/\epsilon$. If $|x| < a(1/\epsilon)$, proceed as follows. As FPTAS$^\omega$ $\subseteq$ NPO, any feasible solution for x has size at most $q(|x|)$ for some polynomial q. Furthermore, given any y with $|y| \leq q(|x|)$, one can determine in polynomial time whether $y \in S_P(x)$. The objective value of a feasible solution can also be computed in polynomial time. Hence by employing exhaustive search, one can find a $y^* \in S_P^*(x)$ in time

$$2^{q(|x|)} \cdot r_P(|x|, q(|x|)) = 2^{q(a(1/\epsilon))} \cdot \mathrm{poly}(a(1/\epsilon)).$$

The result is an eptas for P with appropriately defined function f.

Since FIPTAS$^\omega$ $\subseteq$ FPTAS$^\omega$, we have EPTAS $\subseteq$ FIPTAS$^\omega$ $\subseteq$ FPTAS$^\omega$ $\subseteq$ EPTAS, and hence all classes must be equal. □

The exponential increase in running time in the reduction from an fptas$^\omega$ to an eptas might be reduced by using an exact or fixed-parameter algorithm specific to the problem.

The equivalence of F(I)PTAS$^\omega$ and EPTAS allows an indirect proof of the existence of an eptas for a problem, where a direct proof seems more difficult. For instance, Maximum Independent Set on disk graphs of bounded ply has a fiptas$^\omega$ (Example 2.2.2) and thus, as an immediate consequence of Theorem 2.2.4, it also has an eptas.

We now show that PTAS$^\omega$ and PTAS are equivalent.

Theorem 2.2.5 *PTAS = PTAS$^\omega$.*

Proof: By Proposition 2.2.3 it suffices to prove that $\text{PTAS}^{\omega} \subseteq \text{PTAS}$. Let $P \in \text{PTAS}^{\omega}$ and let $\mathcal{A}$ be a ptas$^{\omega}$ for P. For an arbitrary instance $x \in I_P$ and an arbitrary $\epsilon > 0$, compute $a(1/\epsilon)$. If $|x| \geq a(1/\epsilon)$, compute and return $\mathcal{A}(x, \epsilon)$. Otherwise, apply the same exhaustive search trick as in the proof of Theorem 2.2.4. The result is a ptas for P. $\square$

This implies that $\text{FPTAS}^{\omega} \subset \text{PTAS}^{w}$, unless FPT=W[1].