Polynomial-time approximation scheme

In computer science, a **polynomial-time approximation scheme** (**PTAS**) is a type of approximation algorithm for optimization problems (most often, NP-hard optimization problems).

A PTAS is an algorithm which takes an instance of an optimization problem and a parameter ε > 0 and, in polynomial time, produces a solution that is within a factor 1 + ε of being optimal (or 1 - ε for maximization problems). For example, for the Euclidean traveling salesman problem, a PTAS would produce a tour with length at most (1 + ε)*L*, with *L* being the length of the shortest tour.[1] There exists also PTAS for the class of all dense CSP problems.[2]

The running time of a PTAS is required to be polynomial in *n* for every fixed ε but can be different for different ε. Thus an algorithm running in time $O(n^{1/\varepsilon})$ or even $O(n^{\exp(1/\varepsilon)})$ counts as a PTAS.

1 Variants

1.1 Deterministic

A practical problem with PTAS algorithms is that the exponent of the polynomial could increase dramatically as ε shrinks, for example if the runtime is $O(n^{(1/\varepsilon)!})$. One way of addressing this is to define the **efficient polynomial-time approximation scheme** or **EPTAS**, in which the running time is required to be $O(n^c)$ for a constant *c* independent of ε. This ensures that an increase in problem size has the same relative effect on runtime regardless of what ε is being used; however, the constant under the big-O can still depend on ε arbitrarily. Even more restrictive, and useful in practice, is the **fully polynomial-time approximation scheme** or **FPTAS**, which requires the algorithm to be polynomial in both the problem size *n* and 1/ε. All problems in FPTAS are fixed-parameter tractable. An example of a problem that has an FPTAS is the knapsack problem.

Any strongly NP-hard optimization problem with a polynomially bounded objective function cannot have an FPTAS unless P=NP.[3] However, the converse fails: e.g. if P does not equal NP, knapsack with two constraints is not strongly NP-hard, but has no FPTAS even when the optimal objective is polynomially bounded.[4]

Unless P = NP, it holds that FPTAS ⊊ PTAS ⊊ APX.[5] Consequently, under this assumption, APX-hard problems do not have PTASs.

Another deterministic variant of the PTAS is the **quasi-polynomial-time approximation scheme** or **QPTAS**. A QPTAS has time complexity $n^{\text{polylog}(n)}$ for each fixed $\epsilon > 0$.

1.2 Randomized

Some problems which do not have a PTAS may admit a randomized algorithm with similar properties, a **polynomial-time randomized approximation scheme** or **PRAS**. A PRAS is an algorithm which takes an instance of an optimization or counting problem and a parameter ε > 0 and, in polynomial time, produces a solution that has a *high probability* of being within a factor ε of optimal. Conventionally, “high probability” means probability greater than 3/4, though as with most probabilistic complexity classes the definition is robust to variations in this exact value (the bare minimum requirement is generally greater than 1/2). Like a PTAS, a PRAS must have running time polynomial in *n*, but not necessarily in ε; with further restrictions on the running time in ε, one can define an **efficient polynomial-time randomized approximation scheme** or **EPRAS** similar to the EPTAS, and a **fully polynomial-time randomized approximation scheme** or **FPRAS** similar to the FPTAS.[3]

2 As a complexity class

The term PTAS may also be used to refer to the class of optimization problems that have a PTAS. PTAS is a subset of APX, and unless P = NP, it is a strict subset. [5]

Membership in PTAS can be shown using a PTAS reduction, L-reduction, or P-reduction, all of which preserve PTAS membership, and these may also be used to demonstrate PTAS-completeness. On the other hand, showing non-membership in PTAS (namely, the nonexistence of a PTAS), may be done by showing that the problem is APX-hard, after which the existence of a PTAS would show P = NP. APX-hardness is commonly shown via PTAS reduction or AP-reduction.

3 References

[1] Sanjeev Arora, Polynomial-time Approximation Schemes for Euclidean TSP and other Geometric Problems, Journal of the ACM 45(5) 753–782, 1998.

[2] Arora, S.; Karger, D.; Karpinski, M. (1999), "Polynomial Time Approximation Schemes for Dense Instances of NP-Hard Problems", *Jourlal of Computer and System Sciences*, **58** (1): 193–210, doi:10.1006/jcss.1998.1605

[3] Vazirani, Vijay V. (2003). *Approximation Algorithms*. Berlin: Springer. pp. 294–295. ISBN 3-540-65367-8.

[4] H. Kellerer and U. Pferschy and D. Pisinger (2004). *Knapsack Problems*. Springer.

[5] Jansen, Thomas (1998), "Introduction to the Theory of Complexity and Approximation Algorithms", in Mayr, Ernst W.; Prömel, Hans Jürgen; Steger, Angelika, *Lectures on Proof Verification and Approximation Algorithms*, Springer, pp. 5–28, doi:10.1007/BFb0053011, ISBN 9783540642015. See discussion following Definition 1.30 on p. 20.

4 External links

- Complexity Zoo: PTAS, EPTAS, FPTAS
- Pierluigi Crescenzi, Viggo Kann, Magnús Halldórsson, Marek Karpinski, and Gerhard Woeginger, *A compendium of NP optimization problems* – list which NP optimization problems have PTAS.

5 Text and image sources, contributors, and licenses

5.1 Text

- **Polynomial-time approximation scheme** *Source:* https://en.wikipedia.org/wiki/Polynomial-time_approximation_scheme?oldid=746708498 *Contributors:* Dcoetzee, Altenmann, Mellum, Andris, Mboverload, Schizoid, Creidieki, Rich Farmbrough, Pak21, DcoetzeeBot~enwiki, Oleg Alexandrov, LOL, Decrease789, Chobot, YurikBot, Tribaal, Cedar101, SmackBot, DKalkin, AndreasKaiser, T.Friedrich, David Eppstein, Singleheart, Daveagp, Bender2k14, Vanished user uih38riiw4hjlsd, Addbot, Luckas-bot, Yobot, AnomieBOT, JackieBot, Citation bot, Miym, M.E.Kramer, RobinK, Helpful Pixie Bot, Tmigler, AustinBuchanan, Infinitestory, Bender the Bot and Anonymous: 19

5.2 Images

5.3 Content license