Approximation-preserving reduction

In computability theory and computational complexity theory, especially the study of approximation algorithms, an **approximation-preserving reduction** is an algorithm for transforming one optimization problem into another problem, such that the distance of solutions from optimal is preserved to some degree. Approximation-preserving reductions are a subset of more general reductions in complexity theory; the difference is that approximation-preserving reductions usually make statements on approximation problems or optimization problems, as opposed to decision problems.

Intuitively, problem A is reducible to problem B via an approximation-preserving reduction if, given an instance of problem A and a (possibly approximate) solver for problem B, one can convert the instance of problem A into an instance of problem B, apply the solver for problem B, and recover a solution for problem A that also has some guarantee of approximation.

1 Definition

Unlike reductions on decision problems, an approximation-preserving reduction must preserve more than the truth of the problem instances when reducing from one problem to another. It must also maintain some guarantee on the relationship between the cost of the solution to the cost of the optimum in both problems. To formalize:

Let A and B be optimization problems.

Let x be an instance of problem A , with optimal solution $OPT(x)$. Let $c_A(x, y)$ denote the cost of a solution y to an instance x of problem A . This is also the metric used to determine which solution is considered optimal.

An **approximation-preserving reduction** is a pair of functions (f, g) (which often must be computable in polynomial time), such that:

- f maps an *instance* x of A to an *instance* x' of B .
- g maps a *solution* y' of B to a *solution* y of A .
- g preserves some guarantee of the solution's *performance*, or *approximation ratio*, defined as $R_A(x, y) = \max\left(\frac{c_A(x,OPT(x))}{c_A(x,y)}, \frac{c_A(x,y)}{c_A(x,OPT(x))}\right)$.

2 Types of approximation-preserving reductions

There are many different types of approximation-preserving reductions, all of which have a different guarantee (the third point in the definition above). However, unlike with other reductions, approximation-preserving reductions often overlap in what properties they demonstrate on optimization problems (e.g. complexity class membership or completeness, or inapproximability). The different types of reductions are used instead as varying reduction techniques, in that the applicable reduction which is most easily adapted to the problem is used.

Not all types of approximation-preserving reductions can be used to show membership in all approximability complexity classes, the most notable of which are PTAS and APX. A reduction below **preserves membership** in a complexity class C if, given a problem A that reduces to problem B via the reduction scheme, and B is in C, then A is in C as well. Some reductions shown below only preserve membership in APX or PTAS, but not the other. Because of this, careful choice must be made when selecting an approximation-preserving reductions, especially for the purpose of proving completeness of a problem within a complexity class.

Crescenzi suggests that the three most ideal styles of reduction, for both ease of use and proving power, are PTAS reduction, AP reduction, and L-reduction. [1] The reduction descriptions that follow are from Crescenzi's survey of approximation-preserving reductions.

2.1 Strict reduction

Strict reduction is the simplest type of approximation-preserving reduction. In a strict reduction, the approximation ratio of a solution y' to an instance x' of a problem B must be at most as good as the approximation ratio of the corresponding solution y to instance x of problem A. In other words:

$R_A(x, y) \leq R_B(x', y')$ for $x' = f(x), y = g(y')$.

Strict reduction is the most straightforward: if a strict reduction from problem A to problem B exists, then problem A can always be approximated to at least as good a ratio as problem B. Strict reduction preserves membership in both PTAS and APX.

There exists a similar concept of an **S-reduction,** for which $c_A(x, y) = c_B(x', y')$, and the optima of the two corresponding instances must have the same cost as well. S-reduction is a very special case of strict reduction, and is even more constraining. In effect, the two problems A and B must be in near-perfect correspondence with each other. The existence of an S-reduction implies not only the existence of a strict reduction but every other reduction listed here.

2.2 L-reduction

Main article: L-reduction

L-reductions preserve membership in PTAS as well as APX (but *only for minimization problems in the case of the latter*). As a result, they cannot be used in general to prove completeness results about APX, Log-APX, or Poly-APX, but nevertheless they are valued for their natural formulation and ease of use in proofs.[1]

2.3 PTAS-reduction

Main article: PTAS reduction

PTAS-reduction is another commonly used reduction scheme. Though it preserves membership in PTAS, it does not do so for APX. Nevertheless, APX-completeness is defined in terms of PTAS reductions.

PTAS-reductions are a generalization of P-reductions, shown below, with the only difference being that the function g is allowed to depend on the approximation ratio r .

2.4 A-reduction and P-reduction

A-reduction and P-reduction are similar reduction schemes that can be used to show membership in APX and PTAS respectively. Both introduce a new function c , defined on numbers greater than 1, which must be computable.

In an A-reduction, we have that

$$R_B(x', y') \leq r \rightarrow R_A(x, y) \leq c(r)$$

In a P-reduction, we have that

$$R_B(x', y') \leq c(r) \rightarrow R_A(x, y) \leq r$$

The existence of a P-reduction implies the existence of a PTAS-reduction.

2.5 E-reduction

E-reduction, which is a generalization of strict reduction but implies both A-reduction and P-reduction, is an example of a less restrictive reduction style that preserves membership not only in PTAS and APX, but also the larger classes Log-APX and Poly-APX. E-reduction introduces two new parameters, a polynomial p and a constant β . Its definition is as follows.

In an E-reduction, we have that for some polynomial p and constant β ,

- $c_B(OPT_B(x')) \leq p(|x|)c_A(OPT_A(x))$, where $|x|$ denotes the size of the problem instance's description.
- For any solution y' to B , we have $R_A(x, y) \leq 1 + \beta \cdot (R_B(x', y') - 1)$.

To obtain an A-reduction from an E-reduction, let $c(r) = 1 + \beta \cdot (r - 1)$, and to obtain a P-reduction from an E-reduction, let $c(r) = 1 + (r - 1)/\beta$.

2.6 AP-reduction

AP-reductions are used to define completeness in the classes Log-APX and Poly-APX. They are a special case of PTAS reduction, meeting the following restrictions.

In an AP-reduction, we have that for some constant α ,

$$R_B(x', y') \leq r \rightarrow R_A(x, y) \leq 1 + \alpha \cdot (r - 1)$$

with the additional generalization that the function g is allowed to depend on the approximation ratio r , as in PTAS-reduction.

AP-reduction is also a generalization of E-reduction. An additional restriction actually needs to be imposed for AP-reduction to preserve Log-APX and Poly-APX membership, as E-reduction does: for fixed problem size, the computation time of f, g must be non-increasing as the approximation ratio increases.

2.7 Gap reduction

Main article: Gap reduction

A gap reduction is a type of reduction that, while useful in proving some inapproximability results, does not resemble the other reductions shown here. Gap reductions deal with optimization problems within a decision problem container, generated by changing the problem goal to distinguishing between the optimal solution and solutions some multiplicative factor worse than the optimum.

3 See also

- Reduction (complexity)
- PTAS reduction
- L-reduction
- Approximation algorithm

4 References

[1] Crescenzi, Pierluigi (1997). "A Short Guide To Approximation Preserving Reductions". *Proceedings of the 12th Annual IEEE Conference on Computational Complexity*. Washington, D.C.: IEEE Computer Society: 262–.

5 Text and image sources, contributors, and licenses

5.1 Text

- **Approximation-preserving reduction** *Source:* https://en.wikipedia.org/wiki/Approximation-preserving_reduction?oldid=752879871 *Contributors:* Edemaine, Ulric1313, SaschaWolff, Npinsker, Infinitestory and Anonymous: 1

5.2 Images

5.3 Content license