Algorithmic Modeling and Complexity **Fall 2003**

Lecture 14: 16 October

Lecturer: J. van Leeuwen *Scribe: E. Maes*

14.1 Overview

This lecture considers the matter of 'approximability and in-approximability' by means of polynomial-time algorithms. We have seen many computationally hard problems, which can be 'solved' efficiently using an approximation algorithm with a provable bound on the performance ratio. Can this always be done? We will see that there are problems for which no efficient heuristics of this kind appear to exist. Our leading example will be the *Maximum Independent Set* problem.

In practice the conclusions like derived in this lecture are often taken for granted. A large number of algorithmic techniques exist e.g. using 'local search', randomization or heuristics inspired by natural processes that perform either within good expected bounds or without any specific guarantees at all and yet often solve problems satisfactorily,.

14.2 Optimization problems and approximability

The context of this lecture will be *optimization problems*, i.e. models in which some goal function must be minimized or maximized subject to certain constraints. For example, consider a network $G =< V, E >$. An *independent set* is any subset $A \subseteq V$ such that no nodes of A are connected by an edge. Determining large independent sets arises e.g. in scheduling, when determining a large set of tasks that do not interfere with each other. In an earlier lecture we saw that independent sets are complementary to vertex covers. The (optimization version of the) Maximum Independent Set problem is the following problem:

> Given a network G, what is the size of the largest possible independent set in G.

The decision version of the Maximum Independent Set problem was proved to be *NP-complete* in Lecture 12. The question is whether there exists an efficient, polynomial-time algorithm that can approximate the answer to any given instance of the Maximum Independent Set problem within a good and guaranteed performance ratio. We first have to clarify what we mean by 'good approximability'.

14.2.1 Complexity classes based on approximability

Consider optimization problem where instances I have an optimum solution $OPT(I) = OPT$. Let a *polynomial-time* approximation algorithm deliver a *feasible* solution of value $RES(I) =$

RES. Let PO be the class of polynomial-time solvable optimization problems. Let NPO be the class of all 'NP-optimization problems', which have e.g. only polynomial size feasible solutions and admit a polynomial-time algorithm for checking feasibility of any purported solution.

One can distinguish between the following qualities of approximation for NPO problems by polynomial-time algorithms:

- ABS: the class of problems solvable to within an absolute constant, $|OPT - RES| \leq c$ for fixed c.
- APX: the class of problems solvable to within a performance ratio $\leq 1 + \epsilon$, for some fixed $\epsilon > 0$.
- $f(n)$-APX: the class of problems solvable to within a performance ratio $\leq f(n)$, for some fixed function f depending on the problem size $n = |I|$.
- PTAS: the class of problems solvable to within a performance ratio $\leq 1 + \epsilon$ for any $\epsilon > 0$, by a polynomial-time approximation scheme $A(I, \epsilon)$ that runs in time polynomial in $|I|$ for any fixed ϵ.
- FPTAS: the class of problems solvable to within a performance ratio $\leq 1 + \epsilon$ for any $\epsilon > 0$, by a certain polynomial-time approximation scheme $A(I, \epsilon)$ that runs in time polynomial in $|I|$ and $\frac{1}{\epsilon}$.

Lemma 14.1 *$PO \subseteq ABS$ and $PO \subseteq FPTAS \subseteq PTAS \subseteq APX \subseteq f(n) - APX \subseteq NPO$.*

It can be shown that $P = NP \Leftrightarrow PO = NPO$ and that all inclusions are strict unless $P = NP$.

Exercise. Verify that $PO \subseteq FPTAS$.

14.2.2 Approximability to within an absolute constant

One of the simplest notions of polynomial-time approximability concerns the approximability to within a small absolute 'distance' from the optimum. For example, Minimum Graph Coloring for planar graphs is in ABS.

Exercise. Design a polynomial-time algorithm for planar graph coloring such that $|OPT - RES| \leq 1$ for all instances. (Hint: use that every planar graph is 4-colorable in polynomial-time, and that 2-colorability can be decided in polynomial-time.)

On the other hand, many problems appear not to be in ABS (unless $P = NP$). We give two examples: 1-dimensional packing, and the Maximum Independent Set problem.

Proposition 14.2 *1-Dimensional packing $\notin$ ABS, unless $P = NP$.*

Proof: Suppose 1-dimensional packing is approximable in polynomial time by an algorithm that achieves $OPT - RES \leq c$ for some constant c. Without loss of generality we may assume that c is an integer ≥ 1. Consider an arbitrary instance of 1-dimensional packing:

$$I\,:\,S = \{a_1, \ldots, a_n\} \text{ with } s(a_i) = s_i,\;\; p(a_i) = p_i,\;\; \text{packing bound } B.$$

Construct a new instance by multiplying all profits by a factor $c+1$:

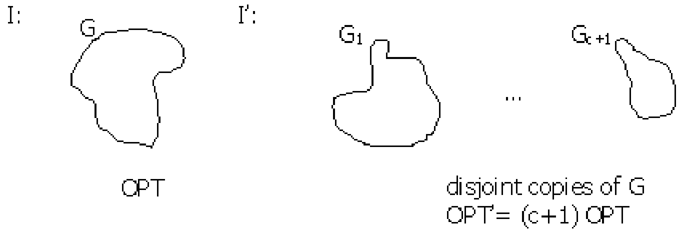
$$I'\,:\,S = \{a_1, \ldots, a_n\} \text{ with } s(a_i) = s_i,\;\; p(a_i) = (c+1)\cdot p_i,\;\; \text{packing bound } B.$$

If I has optimum profit OPT, then I' has optimum $(c+1)\cdot$ OPT and vice versa. Apply the approximation algorithm to I' rather than I. Because the result must be feasible, RES is a multiple of $c+1$ and $(c+1)OPT - RES \leq c$. Hence $RES = (c+1)OPT$ and OPT follows, in polynomial time. As 1-dimensional packing is NP-complete, this is not possible unless $P = NP$. ∎

Exercise. Show that the Traveling Salesman problem is $\notin$ ABS, unless $P = NP$. Show this even for the symmetric metric case. (Hint: use a same technique as above.)

Proposition 14.3 *Maximum Independent Set $\notin$ ABS, unless $P = NP$.*

Proof: Suppose there is a polynomial-time algorithm for the Maximum Independent Set problem that achieves $OPT - RES \leq c$, for some constant c. Again we may assume w.l.o.g. that c is an integer ≥ 1. Let G be a network for which we want to determine the size of a maximum independent set. Construct a network G' consisting of $c+1$ disjoint copies of G, see the figure.



G' can be constructed in polynomial time, and its maximum independent set has a size of $(c+1)\cdot OPT$. Run the approximation algorithm on G': it gives an independent set J in G' with $(c+1)OPT - |J| \leq c$ and thus

$$OPT - \frac{|J|}{c+1} \leq \frac{c}{c+1} < 1.$$

In G' at least one of the $c+1$ copies of G must have an independent set J' with $|J'| \geq \frac{|J|}{c+1}$. By the above inequality we have

$$OPT - |J'| < 1 \quad \Rightarrow \quad |J'| = OPT,$$

and J' is a maximum independent set in G! This contradicts the NP-completeness of Maximum Independent Set, unless $P = NP$. ■

14.2.3 Approximability by polynomial-time approximation schemes

FPTAS and PTAS are more interesting classes. They contain the problems that can be approximated to within any performance ratio $1+\epsilon$, albeit at a cost in terms of computation time the smaller ϵ is chosen. For example, in Lecture 13 we proved that 1-dimensional packing can be solved by a scheme that can achieve a performance ratio of $1+\epsilon$ for any $\epsilon > 0$, at the costs of a running time of $O(\frac{n^3}{\epsilon})$. Thus:

Proposition 14.4 *1-dimensional packing $\in$ FPTAS.*

Fact 14.5 (Lipton and Tarjan, 1980) *Maximum Independent Set restricted to planar networks is $\in$ PTAS but $\notin$ FPTAS, unless $P = NP$.*

As a further example we consider the *Minimum Bin Packing* problem. It concerns packing objects in bins without exceeding the capacity of each bin:

> Given a set of objects $\{a_1, \ldots, a_n\}$ with positive integer sizes $s(a_i) = s_i$ and an unlimited number of bins of integer size B, determine the smallest number of bins in which all objects can be packed. (It is assumed that $s_i \leq B$ for all i.)

The (decision version of the) Minimum Bin Packing problem is *NP-complete*, but the following result holds.

Theorem 14.6 *Minimum Bin Packing $\in$ APX, but $\notin$ PTAS unless $P = NP$.*

Proof: Consider the so-called *Next-Fit* strategy: it takes the objects in the given order, and fills the bins maximally one after the other. Say m bins are used for packing all n objects, and let b_i be the size filled in the i-th bin. Then $b_i + b_{i+1} \geq B$ by the way we fill successive bins, thus $2 \cdot \sum_1^m b_i \geq m \cdot B$. On the other hand, OPT bins suffice and thus $\sum_1^m b_i \leq OPT \cdot B$. We conclude that $m \leq 2 \cdot OPT$. Next-Fit clearly works in polynomial time. Thus Minimum Bin Packing $\in$ APX.

There are bin packing strategies that achieve slightly better performance ratios, but can arbitrary ratios $1+\epsilon$ be reached? Suppose Minimum Bin Packing $\in$ PTAS. Choose an ϵ such that $1+\epsilon < \frac{3}{2}$ and consider the polynomial-time algorithm that allegedly approximates Minimum Bin Packing instances with a performance ratio $< \frac{3}{2}$. We will use the following problem called PARTITION and known to be NP-complete:

> Given a set of objects $S = \{a_1, \ldots, a_n\}$ with positive integer sizes $s(a_i) = s_i$, can S be partitioned in *two* parts of equal size. (It is assumed that $s_i \leq \frac{1}{2}\sum_{i=1}^n s_i$ for every i, otherwise the problem is trivial.)

Given an instance of PARTITION, consider the corresponding instance of Minimum Bin Packing with the same objects and $B = \frac{1}{2}\sum_{i=1}^{n} s_i$. Obviously the instance of PARTITION has a positive answer if and only if the objects fit in ≤ 2, thus in exactly 2 bins. (Note that for the instance 3 bins always suffice: together the objects span a size of 2B, cut through the middle, put the objects to the left and to the right of the cut each in a bin and the object that is cut, if any, in a third bin.)

Run the polynomial-time approximation algorithm on the Bin Packing instance: it will give an answer that is feasible and $< \frac{3}{2} \cdot OPT$. Now observe: if $OPT = 2$ then necessarily $RES < \frac{3}{2} \cdot 2 = 3$, thus $RES = 2$. If $OPT \neq 2$, then $OPT \geq 3$ and necessarily $RES \geq 3$ as well. Thus PARTITION has a positiver answer if and only if $RES = 2$, and we can decide PARTITION in polynomial time. Contradiction unless $P = NP$. ■

Corollary 14.7 *Minimum Bin Packing does not have a polynomial-time approximation algorithm with a performance ratio $< \frac{3}{2}$, unless $P = NP$.*

Fact 14.8 (Simchi-Levi, 1994) *There exists a polynomial-time approximation algorithm for the Minimum Bin Packing problem that achieves a performance ratio of $\frac{3}{2}$.*

14.2.4 Max2SAT and APX-completeness

The results in this section were mentioned but not proved in class. Consider the optimization version of 2SAT, denoted by Max2SAT:

> Given a set of clauses $C = \{C_1, \ldots, C_m\}$ with ≤ 2 literals per clause, determine the largest number of clauses in the set that can be simultaneously satisfied.

Problems like Max3SAT and MaxSAT are defined similarly. In a previous Lecture we showed that 2SAT $\in P$. The following result may therefore surprise.

Lemma 14.9 *The decision version of Max2SAT is NP-complete.*

Proof: The decision version is clearly $\in NP$. To prove the NP-completeness, we show that 3SAT $\preccurlyeq_p$ Max2SAT.

Consider an instance of 3SAT. Let $(x \vee y \vee z)$ be an arbitrary clause in it with literals x, y and z. Allocate a new variable w and consider the following set $C_{x,y,z}$ of 10 clauses with ≤ 2 literals per clause:

$$x, \ y, \ z, \ w$$

$$\overline{x} \vee \overline{y}, \ \overline{y} \vee \overline{z}, \ \overline{z} \vee \overline{x}$$

$$x \vee \overline{w}, \ y \vee \overline{w}, \ z \vee \overline{w}.$$

One verifies:

- $(x \vee y \vee z)$ is true $\Leftrightarrow$ 7 clauses of $C_{x,y,z}$ are simultaneously satisfiable.
- $(x \vee y \vee z)$ is false $\Leftrightarrow$ 6 clauses of $C_{x,y,z}$ are simultaneously satisfiable.

Transform the 3SAT instance into a Max2SAT instance by replacing every clause $(x \vee y \vee z)$ by the corresponding set $C_{x,y,z}$. This is a polynomial-time computable transformation. If the 3SAT instance has m clauses, then the 3SAT instance is satisfiable if and only if $\geq 7m$ clauses in the Max2SAT can be simultaneously satisfied. This proves the p-reduction. ■

Theorem 14.10 *MaxSAT, thus also Max2SAT, is $\in APX$.*

Proof: Let $C = \{C_1, \ldots, C_m\}$ be a set of clauses. Consider the following strategy, which follows a greedy approach:

Algorithm GRS

count $:= 0$

while there are non-empty clauses left bf do

determine the variable x that occurs most often (as x or $\overline{x}$) in the current clauses
let $c_1 =$ the number of occurrences of x
let $c_0 =$ the number of occurrences of $\overline{x}$
if $c_1 \geq c_0$ **then** $x := true$ **else** $x := false$
substitute the truth value of x throughout
do the administration
if $c_1 \geq c_0$ **then** $count := count + c_1$ **else** $count := count + c_0$
delete the clauses that have become satisfied or 'false'

end

GRS is obviously a polynomial-time algorithm and it always delivers a feasible solution. We demonstrate that its performance ratio is ≤ 2.

We prove by induction on n, the number of variables in C, that GRS always satisfies $\geq \frac{1}{2}m$ clauses. For $n = 1$ this is trivial, observing that GRS is done in one iteration and precisely chooses the majority of the clauses to be satisfied. Let the induction hypothesis hold for set of clauses over n variables. Consider a set of clauses C over $n + 1$ variables, and see what GRS does. Assume w.l.o.g. that $c_1 \geq c_0$. By substituting for x, GRS will satisfy c_1 clauses, and a set of $\geq m - c_1 - c_0$ clauses will be left for consideration in the next iteration of the algorithm. By induction we will satisfy at least

$$c_1 + \frac{1}{2}(m - c_1 - c_0) = \frac{1}{2}m + \frac{1}{2}c_1 - \frac{1}{2}c_0 \geq \frac{1}{2}m$$

clauses. This completes the induction step. ■

The following result from modern complexity theory will be crucial in the next section.

Fact 14.11 *Max2SAT $\in$ APX, but $\notin$ PTAS unless $P = NP$.*

Within the class NPO a suitable notion of 'approximation-preserving reduction' between optimization problems can be defined, similar to 'p-reducibility'. The classes like PTAS and APX are closed under this notion of reducibility. In a way very similar to NP-completeness for decision problems one can define *APX-completeness* for NPO-problems.

Fact 14.12 *Max2SAT, Max3SAT and MaxSAT are APX-complete.*

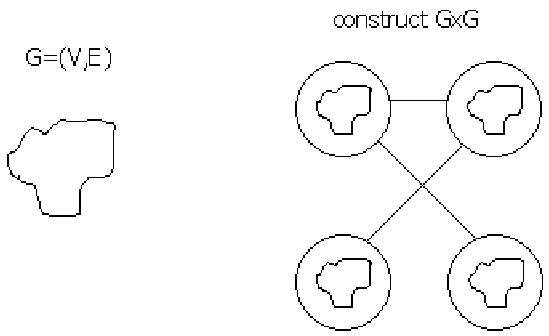
14.3 (In-)approximability of Maximum Independent Set

How well can the Maximum Independent Set problem on arbitrary networks be approximated? Our aim is to show the remarkable fact that Maximum Independent Set $\notin$ APX, implying that the problem is very hard to approximate unless $P = NP$.

14.3.1 Self-reduction of the problem

Lemma 14.13 *If Maximum Independent Set has a polynomial-time approximation algorithm with performance ratio $\leq c$ for some c, then it has one with performance ratio $\leq \sqrt{c}$.*

Proof: Suppose there is a polynomial-time algorithm for the problem with performance ratio $\leq c$, for some $c > 0$. Suppose we want to approximate the size of the maximum independent set in $G =< V, E >$.



Construct $G \times G$. It is the graph in which the nodes u of G act as 'supernodes' and all contain one copy of the original graph G. Inside a supernode, nodes are connected as in G. If u and u' are supernodes, v sits inside u, and v' sits inside u', then v and v' are connected if and only of u and u' are connected as supernodes in G. More formally, $G \times G$ has nodes $< u, v >$ and there is and edge between $< u, v >$ and $< u', v' >$ iff $u = u'$ and $(v, v') \in E$, or $(u, u') \in E$.

Claim 14.14 *G has an independent set I of size $\geq k$ iff $G \times G$ has an independent set of size $\geq k^2$.*

Proof: ($\Rightarrow$) Take the independent sets I in the copies of G that sit in the 'supernodes' of $G \times G$ corresponding to the nodes of I. This gives an independent set of size $|I|^2 \geq k^2$ in $G \times G$.

($\Leftarrow$) Let J be an independent set in $G \times G$. This induces many independent sets in G: one set consisting of the supernodes in which the J-nodes sit and one within each copy of G inside a supernode. At least one of them must have size $\geq \sqrt{|J|}$. ■

Run the approximation algorithm on $G \times G$. By the claim the maximum independent set in $G \times G$ has size OPT^2. The result of the algorithm is an independent set J in $G \times G$ with $OPT^2 \leq c \cdot |J|$, thus $OPT \leq \sqrt{c} \cdot \sqrt{|J|}$ and by the claim one can actually retrieve an independent set of size $\sqrt{|J|}$ in G. This algorithm has a performance ratio $\leq \sqrt{c}$. ■

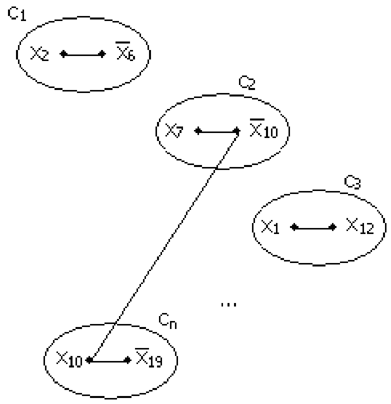
Corollary 14.15 *If Maximum Independent Set has a polynomial-time approximation algorithm with a performance ratio $\leq c$ for some c, then Maximum Independent Set $\in$ PTAS.*

Proof: Suppose we want performance ratio $\leq 1 + \epsilon$ for some given ϵ. Consider the algorithm scheme that iterates the construction in Lemma 14.13 k times, with k such that after square-rooting k times we get $c^{\frac{1}{2^k}} \leq 1 + \epsilon$. Thus $k \geq \log \frac{\log c}{\log(1+\epsilon)}$ suffices, and this k is 'constant' in terms of given ϵ. The algorithm has the desired performance ratio and is polynomial in $|I|$. ■

14.3.2 In-approximability

Theorem 14.16 *Maximum Independent Set $\notin$ APX, unless $P = NP$.*

Proof: By Corollary 14.15 all we have to show is that Maximum Independent Set is $\notin$ PTAS! Suppose it is. Consider the following reduction of Max2SAT to Maximum Independent Set.



Let $C = \{C_1, \ldots, C_n\}$ be an instance of Max2SAT. Assume w.l.o.g. that all clauses have precisely 2 literals and that trivially satisfied clauses of the form $x \vee \overline{x}$ are omitted. Design a network G as shown. There is a supernode corresponding to every clause: it contains two nodes, one for each literal in the clause. Draw an edge between the two literal-nodes within every supernode. Furthermore, draw an edge between every two nodes in different supernodes that are each others opposite as literals. The interpretation of an edge is to prevent conflicts in truth value.

Claim 14.17 *At least k clauses of C can be satisfied simultaneously iff G has an independent set of size $\geq k$.*

Proof: If k clauses in C can be satisfied simultaneously, then the truth value assignment makes at least one literal in each of the clauses true. The corresponding nodes in G form an independent set of size $\geq k$. The converse follows immediately as well. ■

By this direct correspondence, the assumption leads to a polynomial-time approximation scheme for Max2SAT! This contradicts Fact 14.11, unless $P = NP$. ■

14.4 Further remarks

The Lecture showed that for many problems the prospects for polynomial-time approximability are bleak. The $P - versus - NP$ problem appears to be the crucial bottleneck even here. As a surprising example we showed that Maximum Independent Set $\notin$ APX, i.e. there does not exist a polynomial-time algorithm for the problem with a performance ratio bounded by a constant, no matter how large this constant is chosen, unless $P = NP$.

Fact 14.18 (Boppana and Halldórsson, 1992) *Maximum Independent Set* $\in \frac{n}{\log^2 n} - APX$.

But things can be worse yet. We mention the following result for the Traveling Salesman problem, which translates to many more involved vehicle routing problems.

Fact 14.19 (Sahni and Gonzalez, 1976) *The Traveling Salesman problem cannot be approximated by any polynomial-time approximation algorithm within a performance ratio $\leq p(n)$ for any fixed polynomial p, i.e. TSP $\notin$ poly-APX unless $P = NP$.*

Many facts about the polynomial-time (in-)approximability of algorithmic models are listed in the compendium of Crescenzi and Kann [2].

References

[1] G. Ausiello *et al. Complexity and approximation - Combinatorial optimization problems and their approximability properties*, Springer-Verlag, Berlin, 1999.

[2] P. Crescenzi, V. Kann (Eds). *A compendium of NP optimization problems*, website at: *http://www.nada.kth.se/~viggo/problemlist/compendium.html*.