Approximation beyond P

Vangelis Th. Paschos
LAMSADE, Université Paris-Dauphine

Ecole Jeunes Chercheurs du GDR-RO, 2 septembre 2015

Quick recall: the problems discussed, the two established solution paradigms and a basic question

Generate a “small” number of candidate solutions

Split the instance (divide & approximate)

Approximately prune the search tree

Optimally solve a “small” part of the instance

Combine polynomial approximation with the techniques above

Quick recall: the problems discussed, the two established solution paradigms and a basic question

Generate a “small” number of candidate solutions

Split the instance (divide & approximate)

Approximately prune the search tree

Optimally solve a “small” part of the instance

Combine polynomial approximation with the techniques above

The problems

MAX INDEPENDENT SET: Given a graph $G(V, E)$, MAX INDEPENDENT SET consists of finding a set $S \subseteq V$ of maximum size such that for any $(u, v) \in S \times S$, $(u, v) \notin E$

MAX CLIQUE: Given a graph $G(V, E)$, MAX CLIQUE consists of finding a set $K \subseteq V$ of maximum size such that for any $(u, v) \in K \times K$, $(u, v) \in E$

MIN SET COVER: Given a set C of cardinality n and a system $S = \{S_1, \ldots, S_m\} \subseteq 2^C$, MIN SET COVER consists of determining a minimum size subsystem S' such that $\bigcup_{S \in S'} S = C$

More problems

MIN COLORING: Given a graph $G(V, E)$, MIN COLORING consists of partitioning the vertex-set V of G into a minimum number of independent sets.

MIN INDEPENDENT DOMINATING SET: Given a graph $G(V, E)$, MIN INDEPENDENT DOMINATING SET consists of finding the smallest independent set of G that is maximal for the inclusion

Polynomial approximation (1)

$\mathrm{opt}(I)$: the value of an optimal solution of an instance I of a problem Π

$m(\mathtt{A}, I, S)$: the value of the solution S computed by an approximation algorithm $\mathtt{A}$ on I

Approximation ratio of $\mathtt{A}$

$$\rho(\mathtt{A}, I) = \max\left\{\frac{m(\mathtt{A}, I, S)}{\mathrm{opt}(I)}, \frac{\mathrm{opt}(I)}{m(\mathtt{A}, I, S)}\right\}$$

The closer the ratio to 1, the better the performance of $\mathtt{A}$

Polynomial approximation (2)

Inapproximability result

A statement that a problem is not approximable within ratios better than some approximability level unless something very unlikely (e.g., $\boldsymbol{P} = \boldsymbol{NP}$) happens in complexity theory

Polynomial approximation (2)

Inapproximability result

A statement that a problem is not approximable within ratios better than some approximability level unless something very unlikely (e.g., $\boldsymbol{P} = \boldsymbol{NP}$) happens in complexity theory

For the problems discussed in this talk:

- MAX INDEPENDENT SET, MAX CLIQUE, MIN INDEPENDENT DOMINATING SET, MIN COLORING inapproximable within ratios $\Omega\left(n^{1-\epsilon}\right)$, $\forall \epsilon > 0$
- MIN SET COVER within ratios $o(\log n)$

Exact computation with worst-case bounds (1)

Exact computation with worst-case bounds (1)

*Determine an optimal solution for an **NP**-hard problem with provably non trivial worst-case time-complexity*

Exact computation with worst-case bounds (1)

*Determine an optimal solution for an **NP**-hard problem with provably non trivial worst-case time-complexity*

Example: MAX INDEPENDENT SET

Exact computation with worst-case bounds (1)

*Determine an optimal solution for an **NP**-hard problem with provably non trivial worst-case time-complexity*

Example: MAX INDEPENDENT SET

- Exhaustively generate any subset of V and get a maximum one among those that are independent sets: $O(2^n)$ (trivial exact complexity)

Exact computation with worst-case bounds (1)

*Determine an optimal solution for an **NP**-hard problem with provably non trivial worst-case time-complexity*

Example: MAX INDEPENDENT SET

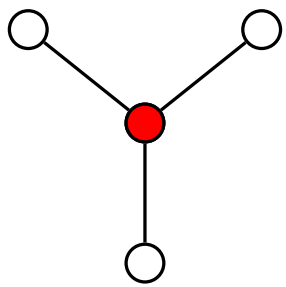
- Exhaustively generate any subset of V and get a maximum one among those that are independent sets: $O(2^n)$ (trivial exact complexity)
- Find all the maximal independent sets of the input graph: $O(1.4422^n)$

Exact computation with worst-case bounds (2)

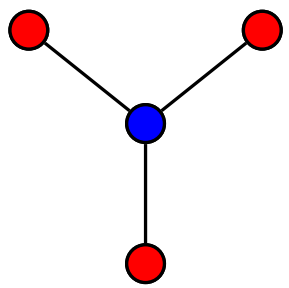
Pruning the search tree

Fix a vertex of maximum degree (> 2, else the problem is polynomial) and:

- either don't take it and delete it from the instance
- or take it and delete it from the graph together with its neighbors



(a) 1 vertex fixed

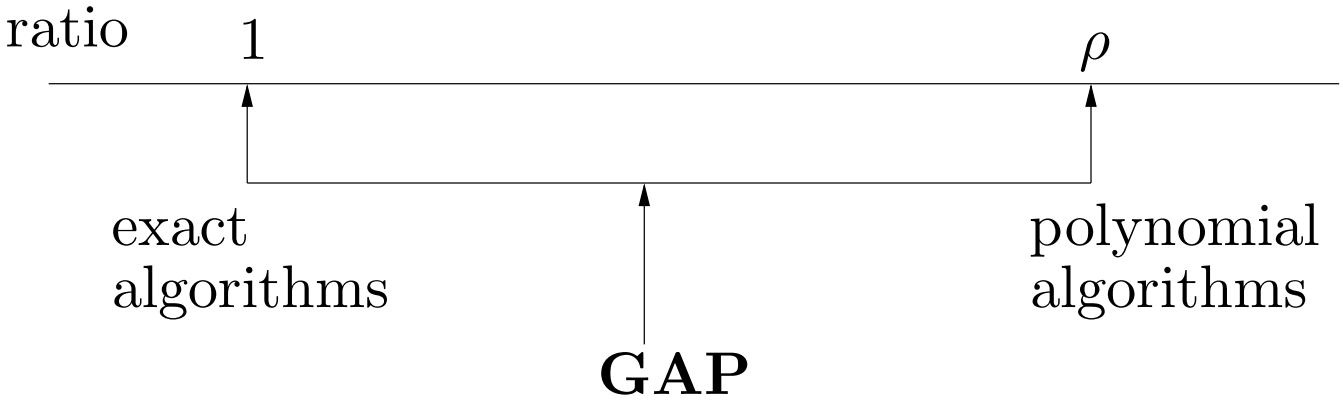


(b) $\geqslant$ 4 vertices fixed

$$T(n) \leqslant T(n-1) + T(n-4) + p(n) \;\; ' \;\; O(1.385^n)$$

A basic question

A basic question



- How to bridge **GAP**?

Quick recall: the problems discussed, the two established solution paradigms and a basic question

Generate a “small” number of candidate solutions

Split the instance (divide & approximate)

Approximately prune the search tree

Optimally solve a “small” part of the instance

Combine polynomial approximation with the techniques above

The key-idea

The key-idea

- Exhaustively generate a number of potential solutions

The key-idea

- Exhaustively generate a number of potential solutions
- If one of them is feasible, just return it

The key-idea

- Exhaustively generate a number of potential solutions
- If one of them is feasible, just return it
- Else, make some trivial or polynomial thing

MAX INDEPENDENT SET

MAX INDEPENDENT SET

- Generate all the $\overline{p}{n}$-subsets of V

MAX INDEPENDENT SET

- Generate all the $\overline{p}{}_{\overline{n}}$-subsets of V
- If one of them is independent, then return it

MAX INDEPENDENT SET

- Generate all the $\overline{p}_{\overline{n}}$-subsets of V
- If one of them is independent, then return it
- Else return a vertex at random

MAX INDEPENDENT SET

- Generate all the $p \atop n$-subsets of V
- If one of them is independent, then return it
- Else return a vertex at random

Approximation ratio: $n^{1/2}$ (impossible in polynomial time)

MAX INDEPENDENT SET

- Generate all the $\overline{p}$-$\overline{n}$-subsets of V
- If one of them is independent, then return it
- Else return a vertex at random

Approximation ratio: $n^{1/2}$ (impossible in polynomial time)

Worst-case complexity: $O\left(\binom{n}{\sqrt{n}}\right) \leqslant O\left(2^{\sqrt{n}\log n}\right)$

Subexponential

Quick recall: the problems discussed, the two established solution paradigms and a basic question

Generate a “small” number of candidate solutions

Split the instance (divide & approximate)

Approximately prune the search tree

Optimally solve a “small” part of the instance

Combine polynomial approximation with the techniques above

What is it?

What is it?

Optimally solve a problem in a series of (small) sub-instances of the initial instance

What is it?

Optimally solve a problem in a series of (small) sub-instances of the initial instance

- Appropriately split the instance in a set of sub-instances (whose sizes are functions of the ratio that is to be achieved)

What is it?

Optimally solve a problem in a series of (small) sub-instances of the initial instance

- Appropriately split the instance in a set of sub-instances (whose sizes are functions of the ratio that is to be achieved)
- Solve the problem in this set

What is it?

Optimally solve a problem in a series of (small) sub-instances of the initial instance

- Appropriately split the instance in a set of sub-instances (whose sizes are functions of the ratio that is to be achieved)
- Solve the problem in this set
- Compose a solution for the initial instance using the solutions of the sub-instances

MAX INDEPENDENT SET

MAX INDEPENDENT SET

Assume that an optimal solution for MAX INDEPENDENT SET *can be found in* $O(\gamma^n)$
Then, for any fixed r*, an* r*-approximation can be computed in* $O(\gamma^{n/r})$

MAX INDEPENDENT SET

Assume that an optimal solution for MAX INDEPENDENT SET *can be found in* $O(\gamma^n)$
Then, for any fixed r*, an* r*-approximation can be computed in* $O(\gamma^{n/r})$

It works for any problem defined upon a hereditary property

MAX INDEPENDENT SET

Assume that an optimal solution for MAX INDEPENDENT SET *can be found in* $O(\gamma^n)$
Then, for any fixed r*, an* r*-approximation can be computed in* $O(\gamma^{n/r})$

It works for any problem defined upon a hereditary property

- Split G into r induced subgraphs G_i of order n/r

MAX INDEPENDENT SET

Assume that an optimal solution for MAX INDEPENDENT SET *can be found in* $O(\gamma^n)$
Then, for any fixed r*, an* r*-approximation can be computed in* $O(\gamma^{n/r})$

It works for any problem defined upon a hereditary property

- Split G into r induced subgraphs G_i of order n/r
- Optimally solve MAX INDEPENDENT SET in every G_i

MAX INDEPENDENT SET

Assume that an optimal solution for MAX INDEPENDENT SET *can be found in* $O(\gamma^n)$
Then, for any fixed r*, an* r*-approximation can be computed in* $O(\gamma^{n/r})$

It works for any problem defined upon a hereditary property

- Split G into r induced subgraphs G_i of order n/r
- Optimally solve MAX INDEPENDENT SET in every G_i
- Output the best among the solutions computed

Example for $r = 2$

Example for $r = 2$

S^*: a maximum independent set in G

S_i^*: a maximum independent set in G_i, $i = 1, 2$

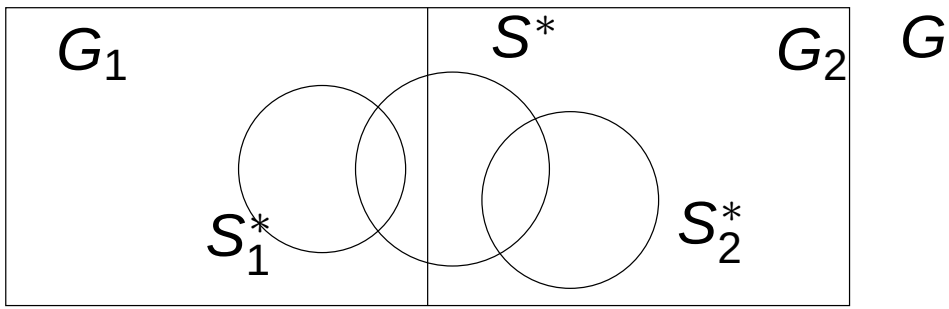
Here, G_i, $i = 1, 2$, the half of G

Example for $r = 2$

S^*: a maximum independent set in G

S_i^*: a maximum independent set in G_i, $i = 1, 2$

Here, G_i, $i = 1, 2$, the half of G

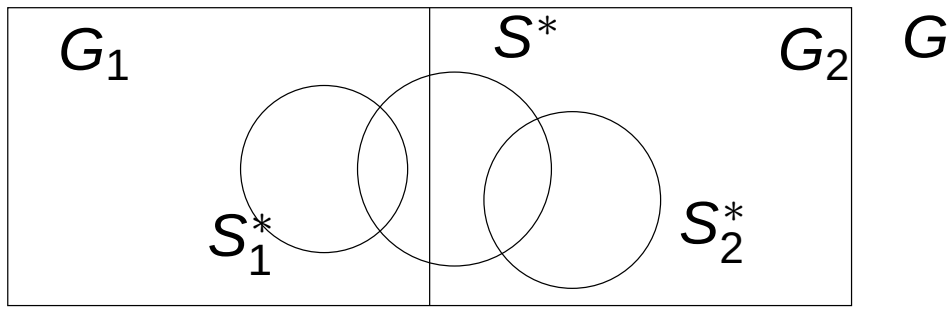


Example for $r = 2$

S^*: a maximum independent set in G

S_i^*: a maximum independent set in G_i, $i = 1, 2$

Here, G_i, $i = 1, 2$, the half of G



$$|S^* \setminus V(G_i)| \leqslant |S_i^*| \implies |S^*| \leqslant |S_1^*| + |S_2^*| \leqslant 2\max\{|S_1^*|, |S_2^*|\}$$

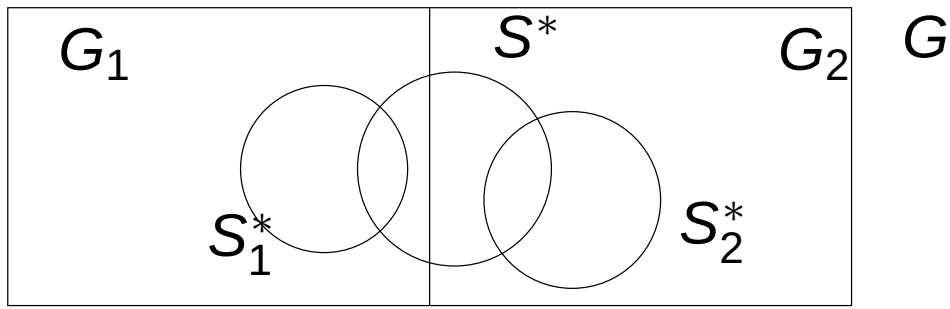
$$\implies \frac{|S^*|}{\max\{|S_1^*|, |S_2^*|\}} \leqslant 2$$

Example for $r = 2$

S^*: a maximum independent set in G

S_i^*: a maximum independent set in G_i, $i = 1, 2$

Here, G_i, $i = 1, 2$, the half of G



$$|S^* \setminus V(G_i)| \leqslant |S_i^*| \implies |S^*| \leqslant |S_1^*| + |S_2^*| \leqslant 2\max\{|S_1^*|, |S_2^*|\}$$

$$\implies \frac{|S^*|}{\max\{|S_1^*|, |S_2^*|\}} \leqslant 2$$

Complexity: $O\left(\gamma^{n/2}\right)$

MIN COLORING

MIN COLORING

- Pouvons-nous appliquez le *instance splitting* à la coloration ?

MIN COLORING

- Pouvons-nous appliquez le *instance splitting* à la coloration ?
- En garantissant quel rapport ?

MIN COLORING

- Pouvons-nous appliquez le *instance splitting* à la coloration ?
- En garantissant quel rapport ?
- Le nombre chromatique de chaque G_i est au plus $\chi(G)$

MIN COLORING

- Pouvons-nous appliquez le *instance splitting* à la coloration ?
- En garantissant quel rapport ?
- Le nombre chromatique de chaque G_i est au plus $\chi(G)$
- Donc, en prenant l'union de toutes les couleurs calculées sur toutes les sous-instances on aura pas plus que $r \ \ \chi(G)$ couleurs

MIN COLORING

- Pouvons-nous appliquez le *instance splitting* à la coloration ?
- En garantissant quel rapport ?
- Le nombre chromatique de chaque G_i est au plus $\chi(G)$
- Donc, en prenant l'union de toutes les couleurs calculées sur toutes les sous-instances on aura pas plus que $r \quad \chi(G)$ couleurs
- Combien de temps nous faut-il?

MIN COLORING

- Pouvons-nous appliquez le *instance splitting* à la coloration ?
- En garantissant quel rapport ?
- Le nombre chromatique de chaque G_i est au plus $\chi(G)$
- Donc, en prenant l'union de toutes les couleurs calculées sur toutes les sous-instances on aura pas plus que $r \ \ \chi(G)$ couleurs
- Combien de temps nous faut-il?
- $O\left(\gamma^{n/r}\right)$

MAX CLIQUE

MAX CLIQUE

Assume that an optimal solution for MAX INDEPENDENT SET *can be found in* $O(\gamma^n)$
Then, for any fixed $r \geq 2$ $\mathbb{Q}$, *an* r*-approximation for* MAX CLIQUE *can be computed with complexity* $O(\gamma^{r\Delta})$
where Δ *is the maximum degree of the input-graph*

MAX CLIQUE

Assume that an optimal solution for MAX INDEPENDENT SET *can be found in* $O(\gamma^n)$
Then, for any fixed $r \geq 1$, *an* r*-approximation for* MAX CLIQUE *can be computed with complexity* $O(\gamma^{r\Delta})$
where Δ *is the maximum degree of the input-graph*

- Split G into n induced subgraphs $G_i = G[\{v_i\} \cup \Gamma(v_i)]$, solve MAX INDEPENDENT SET in $\bar{G}_i$ (recover cliques in G_i's) and output the best

Quick recall: the problems discussed, the two established solution paradigms and a basic question

Generate a “small” number of candidate solutions

Split the instance (divide & approximate)

Approximately prune the search tree

Optimally solve a “small” part of the instance

Combine polynomial approximation with the techniques above

The key idea

Perform a branch-and-cut by allowing a “bounded error” in order to accelerate the algorithm (i.e., make the instance-size decreasing quicker than in exact computation by keeping the produced error “small”)

MIN SET COVER (1)

MIN SET COVER (1)

- Pick r sets and

MIN SET COVER (1)

- Pick r sets and
- Either remove them from the instance

MIN SET COVER (1)

- Pick r sets and
- Either remove them from the instance
- Or take them in the solution and remove them from the instance together with the elements they cover

MIN SET COVER (1)

- Pick r sets and
- Either remove them from the instance
- Or take them in the solution and remove them from the instance together with the elements they cover
- Solve recursively the two resulting subinstances and output the best solution computed

MIN SET COVER (2)

Fix an optimal solution S^*, built by an optimal branching algorithm, and see the above branching algorithm as follows:

MIN SET COVER (2)

Fix an optimal solution S^*, built by an optimal branching algorithm, and see the above branching algorithm as follows:

- *Any time S^* takes at least one of the r sets the algorithm above takes all of them*

MIN SET COVER (2)

Fix an optimal solution S^*, built by an optimal branching algorithm, and see the above branching algorithm as follows:

- *Any time S^* takes at least one of the r sets the algorithm above takes all of them*
- *The fact that S^* doesn't take any of them corresponds to a branch where the sets are not taken*

MIN SET COVER (2)

Fix an optimal solution S^*, built by an optimal branching algorithm, and see the above branching algorithm as follows:

- *Any time S^* takes at least one of the r sets the algorithm above takes all of them*
- *The fact that S^* doesn't take any of them corresponds to a branch where the sets are not taken*

There is a leaf of this branching tree corresponding to a solution containing S^* and having size at most $r \cdot |S^*|$

MIN SET COVER (2)

Fix an optimal solution S^*, built by an optimal branching algorithm, and see the above branching algorithm as follows:

- *Any time S^* takes at least one of the r sets the algorithm above takes all of them*
- *The fact that S^* doesn't take any of them corresponds to a branch where the sets are not taken*

There is a leaf of this branching tree corresponding to a solution containing S^* and having size at most $r \cdot |S^*|$

Any branching reduces the number of sets by r: $O\left(2^{m/r}\right)$

MAX INDEPENDENT SET

MAX INDEPENDENT SET

Can we obtain a, say, 2-approximation for MAX INDEPENDENT SET following a similar approach?

MAX INDEPENDENT SET

Can we obtain a, say, 2-approximation for MAX INDEPENDENT SET following a similar approach?

Approximately prune the MAX INDEPENDENT SET search tree

Fix a vertex of maximum degree and:

- either don't take it and delete it from the instance
- or take it and delete it from the graph together with its neighbors *plus a random vertex*

MAX INDEPENDENT SET

Can we obtain a, say, 2-approximation for MAX INDEPENDENT SET following a similar approach?

Approximately prune the MAX INDEPENDENT SET search tree

Fix a vertex of maximum degree and:

- either don't take it and delete it from the instance
- or take it and delete it from the graph together with its neighbors *plus a random vertex*

If the optimum takes $\alpha(G)$ vertices the algorithm above takes at least $\alpha(G)/2$ vertices (all the randomly removed vertices belonged to the optimum)

Quick recall: the problems discussed, the two established solution paradigms and a basic question

Generate a “small” number of candidate solutions

Split the instance (divide & approximate)

Approximately prune the search tree

Optimally solve a “small” part of the instance

Combine polynomial approximation with the techniques above

The key idea

Optimally solve the problem in a “small” part of the input and (eventually) approximately extend the solution to fit the whole instance

MIN INDEPENDENT DOMINATING SET

1. Compute all independent dominating sets of at most size n/r
2. If such a set exist, return it
3. Otherwise return some maximal independent set

MIN INDEPENDENT DOMINATING SET

1. Compute all independent dominating sets of at most size n/r
2. If such a set exist, return it
3. Otherwise return some maximal independent set

For any $k \geqslant 3$, it is possible to enumerate all independent dominating sets (i.e., maximal independent sets) of size at most n/k with running time $O(k^{n/k})$

- In Step 2 ratio 1, in Step 3 ratio r
- Complexity of Step 1: $O\left(r^{n/r}\right) = O\left(2^{n \log r/r}\right)$

Quick recall: the problems discussed, the two established solution paradigms and a basic question

Generate a “small” number of candidate solutions

Split the instance (divide & approximate)

Approximately prune the search tree

Optimally solve a “small” part of the instance

Combine polynomial approximation with the techniques above

MAX INDEPENDENT SET (again)

MAX INDEPENDENT SET (again)

1. If $\Delta(G) \leqslant 7$, then approximate MAX INDEPENDENT SET polynomially;
2. Else, run the branching algorithm seen before

MAX INDEPENDENT SET (again)

1. If $\Delta(G) \leqslant 7$, then approximate MAX INDEPENDENT SET polynomially;
2. Else, run the branching algorithm seen before

- If $\Delta(G) \leqslant 7$, approximation ratio $1/2$ (ratio $5/(\Delta(G)+3)$ in polynomial time)
- If $\Delta(G) \geqslant 8$, ratio 1

MAX INDEPENDENT SET (again)

1. If $\Delta(G) \leqslant 7$, then approximate MAX INDEPENDENT SET polynomially;
2. Else, run the branching algorithm seen before

- If $\Delta(G) \leqslant 7$, approximation ratio $1/2$ (ratio $5/(\Delta(G)+3)$ in polynomial time)
- If $\Delta(G) \geqslant 8$, ratio 1

Complexity: $T(n) = T(n-1) + T(n-8) \leqslant \ldots$

BREAK