Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision
UMR 7243

CAHIER DU LAMSADE

331

Décembre 2012

Moderate exponential time approximation and branching algorithms

B. Escoffier, V.Th. Paschos, E. Tourniaire

Moderate exponential time approximation and branching algorithms*

Bruno Escoffier Vangelis Th. Paschos[(a)] Émeric Tourniaire
PSL Research University, Université Paris-Dauphine
LAMSADE, CNRS UMR 7243, France
{escoffier,paschos,emeric.tourniaire}@lamsade.dauphine.fr

December 13, 2012

Abstract

We study links between approximation, exponential time computation and fixed parameter tractability. In particular, rather than focusing on one particular optimization problem, we tackle the question of finding sufficient conditions for a problem to admit "good" approximation algorithms in exponential time. In particular, we focus on the existence of "approximation schemata" (ratios $1 \pm \epsilon$ for arbitrarily small ϵ) and we exhibit conditions under which a technique of devising approximate branching algorithms reaches interesting results.

1 Introduction

Polytime approximation, exponential time computation and fixed parameter tractability (FPT) are three main fields aiming at coping with **NP**-hard problems. Considering an optimisation problem, a ρ-approximation algorithm is an algorithm that, on any instance, outputs a solution whose value is at least (for a maximisation problem, at most for a minimization problem) ρ times the optimal value. Polytime approximation is the field of studying whether one can derive approximation algorithms working in polynomial time for an **NP**-hard problem (see for instance [1]). The goal of exponential time computation (see [19]) is to devise exact algorithms whose worst case complexity, though not polynomially bounded, is as low as possible. For many **NP**-hard problems, the best known algorithm works in exponential time $O^*(\gamma^n)$ (meaning $O(\gamma^n p(n))$ for some polynomial p). A natural objective is then to minimize the value of γ. FPT is in some sense a complementary field where an instance is given together with a parameter $k \in \mathbb{N}$; the goal is to know whether the problem is solvable by an algorithm working in time $O(f(k)p(n))$ for some function f and some polynomial p [12]. Such an algorithm is called an FPT algorithm, and if such an algorithm exists the problem is in the **FPT** class (for this particular parameterization). A usual parameter (called standard parameterization) is the value of the sought solution. Formally, for the standard parameterization[1], a problem is in **FPT** if there exists an algorithm such that given an instance I of size n and an integer k, (i) it outputs YES if there is a solution of value at least k (for a maximization problem, at most k for a minimization problem) and NO otherwise; (ii) it works in time $O(f(k)p(n))$ for some function f and polynomial p.

Interestingly, negative results have been obtained in these three fields. For polytime approximation, tight lower bounds are known for many optimization problem, under the hypothesis $\mathbf{P} \neq \mathbf{NP}$. For instance, Max 3SAT[2] is well known to be approximable in polytime within ratio 7/8, and not

*Research supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010
(a)Institut Universitaire de France

[1]We will use the standard parameterization in all this article, so we will omit to precise it.

[2]Given a set of binary variables and a set of clauses, Max SAT is to find a truth assignment of variables that maximizes the number of satisfied clauses. Max kSAT is a restricted version where each clause contains exactly k literals.

approximable in polytime within ratio $7/8+\epsilon$ unless $\mathbf{P}=\mathbf{NP}$ for any $\epsilon>0$ [24]. Dealing with exponential time computation, [25] have introduced the Exponential Time Hypothesis (*ETH*) which supposes that there is no algorithm for Max 3SAT that works in time $2^{o(n)}$ where n is the number of variables. Under *ETH*, many classical optimization problems do not have subexponential time algorithms[3]. Finally, the parameterized complexity theory considers the hypothesis $\mathbf{W[1]}\neq\mathbf{FPT}$ (see [12] for the definition of the class $\mathbf{W[1]}$) under which it is shown that many optimization problems do not admit FPT algorithms (under the standard parameterization for instance).

In recent years, a growing interest appeared in trying to combine these approaches. The first way that has been considered is to devise (sub)exponential approximation algorithms. In this topic, we take a problem that is inapproximable with some ratio ρ (in polytime), and for which the best known complexity is $O^*(\gamma^n)$. A natural question is whether there exists a ρ-approximation algorithm that works in time much smaller than $O^*(\gamma^n)$. This issue has been considered for several optimization problems such as Max SAT [8, 15], Max Independent Set and Min Vertex Cover [4], Min Bandwidth [7, 21], ... In these articles, approximation algorithms working in exponential time are provided. Interestingly, the existence of subexponential time approximation algorithms (achieving constant ratios forbidden in polytime) seems to be unlikely for several classical optimization problems (see [29, 14]).

Another research direction introduced in [6, 13] aims at combining approximation algorithms and fixed parameter tractability. In the parameterized framework, given an instance I of an optimization problem and an integer k, a ρ-approximation algorithm either outputs a solution of value at least ρk (for a maximization problem, at most ρk for a minimization problem), or answers NO, asserting in this latter case that there is no solution of value at least (resp., at most) k. Then, if a problem is both $\mathbf{W[1]}$-hard and hard to approximate within ratio ρ in polytime, the existence of an FPT ρ-approximation algorithm is worth being considered. Both positive and negative results have been obtained (see the survey [28]). If a problem is solvable in time $O^*(f(k))$ but hard to approximate within ratio ρ in polytime, it is also interesting to seek a parameterized ρ-approximation algorithm working in time $O^*(f'(k))$ for some function f' significantly lower than f. In this direction, positive answers have been provided for instance for Min Vertex Cover [3, 17].

In this work, we further study the link between approximation algorithms, exponential time computation and fixed parameter tractability. In particular, rather than focusing on one particular optimization problem, we tackle the question of finding sufficient conditions for a problem to admit "good" approximation algorithms in exponential time. Especially, we focus on the existence of "approximation schemata" (ratios $1\pm\epsilon$ for arbitrarily small ϵ). The article is organized as follows. We define approximation classes and give some basic properties in Section 2. In Section 3, we give sufficient conditions to derive approximation schemata in exponential time using branching algorithms, which is the main result of this article. We conclude the article in Section 4 where we mention in particular FPT approximation algorithms.

2 Approximation classes and first results

2.1 Approximation classes

We focus on the well known class **NPO** of optimization problems whose decision versions are in **NP**. An **NPO** problem P is defined on a set $\mathscr{I}$ of instances. To each instance $I\in\mathscr{I}$ corresponds a set $\mathscr{S}(I)$ of feasible solutions. A function f associates for each instance I and each feasible solution $S\in\mathscr{S}(I)$ a value $f(I,S)\in\mathbb{N}$, to be either maximized or minimized. Furthermore, (i) instances of P are recognized in polynomial time; (ii) the size of feasible solutions are polynomially bounded in the size of the instance and, moreover, it can be checked in polynomial time whether a given S is in $\mathscr{S}(I)$ or not; (iii) f is polynomially computable.

An optimal solution of an instance I will be denoted $\mathrm{OPT}(I)$, or simply OPT if there is no ambiguity. We use the notation $f(I)$ for $f(I,\mathrm{OPT}(I))$.

Following the discussion in introduction, considering an optimization problem that is solvable in time $O^*(\gamma^n)$, we may wonder whether the problem is ρ-approximable in time $O^*(\gamma_\rho^n)$ with $\gamma_\rho<\gamma$

[3]Note that speaking of (sub)exponential time algorithm requires to have defined a parameter representing the size of the instance, such as the number of vertices in a graph, with respect to which the complexity is measured.

for some ratio ρ, or even for any ratio $\rho \neq 1$, or not. This motivates the definition of the following two classes. Note that we give the definition for maximization problems, but everything said in this section can be easily translated to minimization problems.

> *Class* 1 (**EAS**)*:* For $\delta > 1$, a maximization **NPO** problem is in **EAS**$[\delta]$ if, for any $\rho < 1$, there exists a ρ-approximation algorithm with computation time $O^*(\delta_\rho^n)$, with $\delta_\rho < \delta$.

This defines a notion of approximation schema, the existence of which of course depends on the parameter δ which is the "target" basis of the exponential function expressing the running time.

Dealing with approximation for a specific ratio, we define the class **EAX**$[\gamma, \rho]$.

> *Class* 2 (**EAX**)*:* For $\gamma > 1$, $\rho \neq 1$, an **NPO** problem is in **EAX**$[\gamma, \rho]$ if there exists a ρ-approximation algorithm with computation time $O^*(\gamma^n)$.

Let us begin with some properties of these classes. First, note that of course any problem that can be solved exactly in time $O^*(\delta^n)$ is in **EAS**$[\delta']$ for any $\delta' > \delta$. On the other hand, one may expect at first sight that if a problem is in **EAS**$[\delta]$, then the problem is solvable in $O^*(\delta^n)$. For several exponential approximation schemata obtained so far in the literature problems shown to be in **EAS**$[\delta]$ are indeed solvable in $O^*(\delta^n)$, but the following result is worth being mentioned: Max Unused Colors[4] can be approximated with ratio $(1-\epsilon)$ in time $O^*(2^n)$ and polynomial space for any $\epsilon > 0$, whereas the best known exact algorithm in polynomial space is in $O^*(2.25^n)$ [5].

However, we say that a problem has a *full γ-exponential approximation schema*, if it is possible to perform a $(1-\epsilon)$ approximation in time $O(p(1/\epsilon) \times \gamma_\epsilon^n)$ for some polynomial p and $\gamma_\epsilon < \gamma$. Then, we have the following easy result.

Proposition 3. *If an **NPO** maximization problem has a full γ-exponential approximation schema and is polynomially bounded*[5]*, then it can be solved in time $O^*(\gamma^n)$.*

Proof. If the value is bounded by $P_m(n)$, with P_m a polynomial, then using the schema, we can get a $(1+1/P_m(n))$-approximation in time $O(P(P_m(n))\gamma^n)$, which is a computation time $O^*(\gamma^n)$. With this ratio, the difference between the optimal solution and the obtained solution would be less than one, which means that we have an exact solution. □

Now, let us make some observations on the classes **EAS** and **EAX**.

Proposition 4. *The following holds: (i) $\forall \delta < \delta'$ **EAS**$[\delta] \subseteq$ **EAS**$[\delta']$; (ii) $\forall \gamma < \gamma', \forall \rho,$ **EAX**$[\gamma, \rho] \subseteq$ **EAX**$[\gamma', \rho]$; (iii) $\forall \gamma, \forall \rho' < \rho,$ **EAX**$[\gamma, \rho] \subseteq$ **EAX**$[\gamma, \rho']$. Furthermore, assuming the SETH*[6]*, the above inclusions are strict.*

Proof. Inclusions are trivial. We prove the strict inclusions under *SETH*. We first deal with the first inclusion Let δ and δ' be two real numbers such that $1 < \delta < \delta'$, and δ'' be such that $\delta < \delta'' < \delta'$. We create a problem where the value of solutions is either 0 or 1, and such that it is solvable in $O^*(\delta''^n)$ (and hence is in **EAS**$[\delta']$) but not in $O^*(\delta^n)$ under *SETH* (and hence not in **EAS**$[\delta]$ since obviously any approximate solution allows to solve optimally the problem).

Let $P_{\delta,\delta''}$ be the following problem. An instance of it, is a set of n variables $(x_1, \cdots, x_n)$, n nonnegative integers $(a_1, \cdots, a_n)$ such that the average value $\overline{a_i} = \sum_{i=1}^n a_i/n$ lies in the interval $(\ln(\delta)/\ln(2), \ln(\delta'')/\ln(2)]$, and a polynomially computable function $f : \mathbb{N}^n \to \{0,1\}$. Every variable x_i can take values from 0 to $2^{a_i} - 1$ (this defines the set of feasible solutions), and the value of a solution is $f(x_1, \cdots, x_n)$, to be maximized. This problem is obviously in **NPO**.

Note that $\prod_{i=1}^n 2^{a_i} = 2^{\sum_{i=1}^n a_i} \leq \delta''^n$; so, an exhaustive search allows us to solve the problem in time $O^*(\delta''^n)$.

Now, take an instance of SAT with N variables. Let $n = \lceil \ln(2)N/\ln(\delta'') \rceil$. Note that $\ln(2)N/\ln(\delta'') \leq n < \ln(2)N/\ln(\delta'')+1$, meaning that $\ln(2)/\ln(\delta'') \leq n/N < \ln(2)/\ln(\delta'')+1/N \leq$

[4] Given a graph G on n vertices, the goal is to find a proper coloring of G maximizing $n-k$ where k is the number of colors of the coloring.

[5] This means that the value of a solution is polynomialy bounded in the size of the instance.

[6] The strong exponential time hypothesis (*SETH* [25]) claims that there is no constant $\gamma < 2$ such that SAT is solvable in time $O^*(\gamma^n)$.

$\ln(2)/\ln(\delta)$ for N large enough. Then choose n nonnegative integers a_i such that $\sum_{i=1}^{n} a_i = N$. Intuitively, if for instance $a_i = 2$ this means that our variable x_i will have value in $\{0, 1, 2, 3\}$ which allows to simulate 2 variables of the SAT instance. This way, since $\sum_{i=1}^{n} a_i = N$, each variable of the SAT instance is simulated in our set of n variables with value in $\{0, \cdots, 2^{a_i} - 1\}$. The function f is now trivial: its value is 1 if $(x_1, \cdots, x_n)$ corresponds to a truth value that satisfies the SAT instance, and 0 otherwise. Now, note that $\delta^n < 2^{\sum_{i=1}^{n} a_i} = 2^N$, so an algorithm solving $P_{\delta,\delta'}$ in $O^*(\delta^n)$ would contradict *SETH*.

The same argument shows the second inclusion claimed. Just take a problem that is not solvable in $O^*(\gamma^n)$ under *SETH* like in the above construction, but define the values to be either 1 or ρ (instead of either 1 or 0). □

2.2 FPT and exponential time approximation schemata

As explained in introduction, our main concern in this article is to provide sufficient conditions for a problem solvable in time $O^*(\gamma^n)$ to be in $\mathbf{EAS}[\gamma]$.

In this line of work, a first general result has been obtained in [4] for hereditary graph problems[7]: using a method consisting of partitioning the instance, the authors show that if an hereditary graph problem is solvable in $O^*(\gamma^n)$ then it is ρ-approximable in time $O^*(\gamma^{\rho n})$ for any ratio ρ, and hence it is in $\mathbf{EAS}[\gamma]$.

As a first simple result here, we show that exponential time approximation schemata can be easily derived from FPT problems having some basic properties.

Proposition 5. *Let P be a maximization (resp., minimization) problem in **FPT** solvable in $O^*(\delta^k)$ such that: (i) the value of any feasible solution is at most n (n is the size of the instance); (ii) if there exists a solution of value k, then there exists a solution of size l, for any $l \leqslant k$ (resp., for any $k \leqslant l \leqslant n$). Then P is also in **EAS**[δ], and it is possible to reach any approximation ratio ρ in computation time $O^*(\delta^{\rho n})$ (resp., $O^*(\delta^{n/\rho})$).*

Proof. Let us first consider that P is a maximization problem. Given a $\rho \in (0, 1)$, we apply the parameterized algorithm with every value for k between 0 and ρn. If the best solution has value at most ρn, we find it. Otherwise, the optimum value is between ρn and n, hence there is a solution of value ρn. We will find such a solution, that trivially achieves an approximation ratio ρ. The complexity is clearly in $O^*(\delta^{\rho n})$.

If P is a minimization problem, for $\rho \geq 1$, we apply the parameterized algorithm with every value for k between 0 and n/ρ. If no solution is found, we output any feasible solution (by hypothesis we know how to compute a feasible solution in polynomial time). If the optimal value is at most n/ρ we find it. Otherwise, any feasible solution achieves an approximation ratio ρ. The complexity is $O^*(\delta^{n/\rho})$. □

The conditions in Proposition 5 are not very restrictive so that this proposition applies to a large range of problems, including for instance Min Vertex Cover, Max SAT, ...

3 Approximation schemata by branching algorithms

Hereditary problems and FPT problems having additional properties admit approximation schemata in exponential time. However, many other problems are known to be (exactly) solvable in exponential time, and a classical way to achieve these exponential time algorithms is to use a search tree (see [19] for instance). Then, a natural idea is to extend this technique to derive exponential time approximation algorithms. This has been done for Max SAT in [8, 15]. In this section, we formalize and generalize this technique, and show a sufficient condition on the problem dealt that makes the technique efficient.

[7] A graph property π is hereditary if the following holds: if π holds for a graph G, then π holds for any induced subgraph of G. Given an hereditary property π, a usual goal is to find the largest induced subgraph of a given graph that satisfies the property π.

Example 6*:* We give a basic overview of the schema on Max Independent Set. On this problem, using classical reduction rules (including vertex folding, see for instance [18]), vertices of degree at most two can be removed from the instance in polynomial time. When there is no such vertex, the naive exact algorithm is to pick a vertex, and to branch with two instances: either we add the vertex in the final solution and remove its neighbors from the "child" instance, either we remove this vertex from the instance. In the first case, the size of the instance decreases by at least 4 (one vertex in the solution, at least three vertices removed) ; in the second, the size decreases by 1, and this gives a complexity $O^*(1.38^n)$ (solution of $C(n) \leqslant C(n-4)+C(n-1)$). In this situation we can achieve a 1/2-ratio by removing an additional vertex in the first branching case. The complexity of this new algorithm will satisfy $C(n) \leqslant C(n-5) + C(n-1)$ giving a complexity $O^*(1.32^n)$. Although in this case, this is not an improvement over some existing approximation algorithms, this is a good illustration of the schema.

We will now introduce some notation to formalize what is a (exact) branching algorithm. We then present what is a diminution rule, and the results on approximate branching algorithms in Section 3.1, and finally illustrate this technique on a very detailed example and some others in Section 3.2.

Given an optimization problem, we have a size function $I \mapsto |I|$. Note that this can be any measure of the size of the instance (number of vertices in a graph, or number of edges, ...). We require that when $|I| = 0$ the instance is solvable in polynomial time. For the sake of clarity we will suppose that $|I| \in \mathbb{N}$ but similar results can be obtained with $|I| \in \mathbb{R}_+$ (in particular $|I|$ can be the weight associated to $|I|$ in the measure and conquer paradigm).

A branching algorithm solves a given optimization problem by building a search tree. To each node of the tree is associated an instance; a branching rule (such as "take a particular vertex in the solution or do not take it") gives birth to several subinstances that are the children of the node in the search tree[8]. The leaves of the tree correspond to trivial instances. This might be formalized as follows.

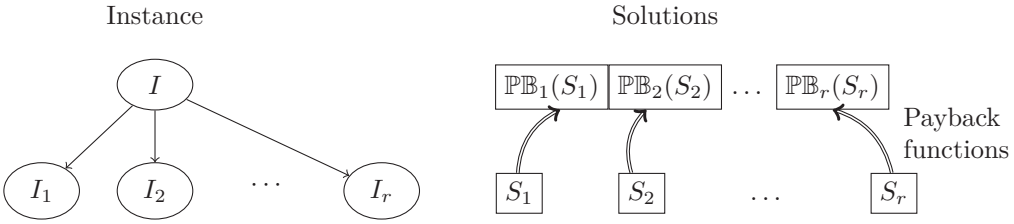


Figure 1: Branching rule

Definition 7 (Branching rule; Figure 1)*:* A branching rule of fan-out r is a mapping from an instance I to a set of r subinstances $(I_1, I_2, \cdots, I_r)$ and a set of algorithms $(\mathbb{PB}_1, \mathbb{PB}_2, \cdots, \mathbb{PB}_r)$, that we call payback functions, with the following properties:
a) If S_i is a solution for the subinstance I_i, then $\mathbb{PB}_i(I, I_i, S_i)$ is a solution for I (for simplicity, we will denote $\mathbb{PB}_i(I, I_i, S_i)$ by $\mathbb{PB}_i(S_i)$).
b) The branching and every payback function are computable in polynomial time with respect to the size of the instance.
c) (size reduction) We have a family of positive numbers $(a_i)_{i=1\cdots,r}$ giving a guarantee on the instance size decreasing. This gives **Rule (B1)**: $\forall i$, $|I_i| \leqslant |I| - a_i$.
d) (exhaustiveness) For (at least) one of the subinstances I_i, the payback function increases the value of a feasible solution by at least the difference in the optimal between I and I_i. This, gives **Rule (B2)**: $\exists i, \forall S \in \mathscr{S}(I_i)$, $f(\mathbb{PB}_i(S)) - f(S) \geqslant f(I) - f(I_i)$.

[8]Note that branching algorithms usually use reduction rules that are polynomial time rules applied on each node to simplify the instance. Reduction rules can be formally incorporated in the branching rule, so we do not need to state them explicitly.

Of course, the greater the a_i (in Rule (B1)), the faster the algorithm. Exhaustiveness ensures that an optimal solution for an instance can be found with the branching rule: indeed, exhaustiveness is satisfied in branching algorithms in the branch corresponding to the choice of an optimum solution.

Given an optimization problem P with such a branching rule, we define the following branching algorithm.

> *Algorithm* 8*:* Input: an instance I_0. Output: an optimal solution for I_0.
> We build a tree using the following rules:
> a) each node is labeled with an instance, the root's label being I_0.
> b) each node with a non-empty instance I has r children, given by the branching rule.
> c) each node with an empty instance is a leaf.
> Then, for each leaf in the tree, we compute an optimal solution. We use the payback functions to get a solution for the root of the tree in a bottom up fashion: the solution for a node with instance I is the best solution among the r solutions computed by the payback functions.

The following result is very common in the domain's literature.

Theorem 9. *Algorithm 8 gives an optimal solution in time $O^*(\gamma^{|I_0|})$, with γ the only positive solution of the equation $1 = \sum_{i=1}^{r} \gamma^{-a_i}$.*

Proof. Let us call I_0 the instance in the root of the tree. We know that at least one child of I_0 satisfies Rule (B2). We call this child I_1. Then I_1 has a child I_2 that satisfies the same rule. And so on till we reach a leaf, I_k. The instance on the leaf is solvable exactly in polynomial time, giving a solution S_k such that $f(S_k) = f(I_k)$. Then, we use the payback functions to compute solutions for every instance from I_k to I_0, and we call these solutions $S_k, S_{k-1}, \cdots, S_0$. With these solution, we have, using (B2), $f(S_i) - f(S_{i+1}) \geqslant f(I_i) - f(I_{i+1})$, for $i = 0, \cdots, k-1$. Summing these inequalities, we get $f(S_0) - f(S_k) \geqslant f(I_0) - f(I_k)$. So, $f(S_0) = f(I_0)$, which proves the correctness of the algorithm.

The complexity is given by the number of nodes in the tree. For an instance of size n, this number satisfies the equation $C(n) \leqslant C(n-a_1) + C(n-a_2) + \cdots + C(n-a_r)$, which gives the complexity claimed. □

3.1 Approximate branching algorithms

Suppose that we have for a given maximization problem a branching algorithm as given above. We are interested in modifying this algorithm to get an approximation algorithm with a better running time. We show that two properties are sufficient to reach this for any ratio $\rho \neq 1$, i.e., to get an exponential time approximation schema. The first condition, called *strict monotonicity*, deals with the branching rule. Note that it is generally satisfied by most of the branching rules used in the literature for classical optimization problems. The other condition, called *diminution rule*, is independent of the branching rule (it it directly linked to the problem, not to a particular branching algorithm).

> *Definition* 10 (Strict monotonicity)*:* A branching rule is said to be strictly monotonic if for at least one subinstance I_i the payback function increases (strictly) the value of feasible solutions (while the other ones do not decrease the value of feasible solutions). Formally, there exists a family of nonnegative integers $(s_i)_{i=1\cdots,r}$ with at least one of them greater than 0 such that the following rule is satisfied: **Rule (B3)**: $\forall i, \forall S \in \mathscr{S}(I_i)$, $f(\mathbb{PB}_i(S)) \geqslant f(S) + s_i$.

Of course, the greater the integers s_i, the faster our approximation algorithm. In Example 6, the branching rule is strictly monotonic: the payback function will increase by one the value of solutions in the branch where we have taken v (and by 0 in the other branch).

> *Definition* 11 (Diminution rule; Figure 2)*:* A diminution rule is a mapping from an instance I to another instance I' together with an polynomial time algorithm (which we call payback function) $\mathbb{PB}'$, with the following properties:
> 1) The new instance's size has decreased by at least one: **Rule (D1)**: $|I'| \leqslant |I| - 1$.

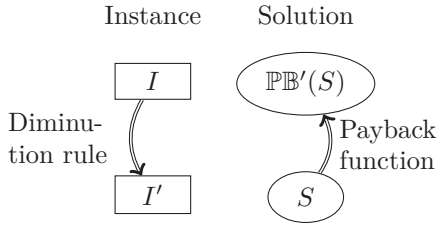


Figure 2: Diminution rule

2) The value of optimal solutions has not decreased by more than one. This gives **Rule (D2)**: $f(I') \geqslant f(I) - 1$.
3) The payback function $\mathbb{PB}'$ preserves the value of the solutions. This gives **Rule (D3)**: $\forall S' \in \mathscr{S}(I')$, $f(\mathbb{PB}'(S')) \geqslant f(S')$.

Example 12*:* For Max Independent Set (see also Example 6), the diminution rule we use is to remove a vertex from the graph: from a graph G we obtain a graph G' with one vertex less (Rule (D1)). Of course the optimum value in G is at most the optimum value in G' plus one (Rule (D2)). The payback function is the identity function (an independent set in G' is an independent set in G) and satisfies Rule (D3).

Given a problem that has both a diminution rule and a strictly monotonic branching rule, we can define the following ρ-approximation algorithm.

Algorithm 13*:* Input: an instance I_0 and a ratio ρ. Output: a ρ-approximate solution for I_0.
We define α_0 such that $\alpha_0 = \frac{\rho}{1-\rho}$. We build a tree using the following rules:
a) each node is labeled with an instance I and a number α. We write a node $[I, \alpha]$. The root's label is $[I_0, 0]$.
b) for each node $[I, \alpha]$, with I non-empty and $\alpha < \alpha_0$, we create up to r children $([I_1, \alpha + s_1], \cdots, [I_r, \alpha + s_r])$, corresponding to the branching rule.
c) for each node $[I, \alpha]$, with I non-empty and $\alpha \geqslant \alpha_0$, we create one child. Its instance is obtained by applying the diminution rule, and its integer is $\alpha - \alpha_0$.
d) the leaves are the empty instances (with any number).
Then, for each leaf in the tree, we compute an optimal solution. We use the payback functions to get a solution for the root of the tree in a bottom up fashion: in case b), the solution for a node with instance I is the best solution among the r solutions computed by the payback functions. In case c), the solution for a node with instance I is given by the payback function of the diminution rule (see Figure 3 for an illustration of the algorithm).

Theorem 14. *Algorithm 13 outputs a ρ-approximate solution in time $O^*(\gamma_\rho^{|I|})$, where γ_ρ is the only positive root of the equation $1 = \sum_{i=1}^{r} \gamma_\rho^{-a_i - \alpha_0 s_i}$. In particular, the problem is in $\boldsymbol{EAS}[\gamma]$ where γ is the only positive root of the equation $1 = \sum_{i=1}^{r} \gamma^{-a_i}$.*

Proof. We first analyze the approximation ratio of the algorithm. Let us show that each non-leaf node $[I, \alpha]$ in the tree has at least one child $[I', \alpha']$ with a payback function $\mathbb{PB}$, and satisfying the following:

$$\forall S' \in \mathscr{S}(I'), f(\mathbb{PB}(S')) - f(S') + \frac{1}{1+\alpha_0}(\alpha - \alpha') \geqslant \frac{\alpha_0}{1+\alpha_0}(f(I) - f(I')) \tag{1}$$

We examine the following two cases concerning α and α_0.

i) $\alpha \geqslant \alpha_0$. Then I' is derived from I using the diminution rule. For any solution $S' \in \mathscr{S}(I')$, we call $S = \mathbb{PB}(S')$. Then using Rule (D3), we have $f(S) - f(S') \geqslant 0$. Also, as we used a diminution, we know that $\alpha' = \alpha - \alpha_0$, and so, $\frac{1}{1+\alpha_0}(\alpha - \alpha') = \frac{1}{1+\alpha_0} \times \alpha_0 = \frac{\alpha_0}{1+\alpha_0}$.

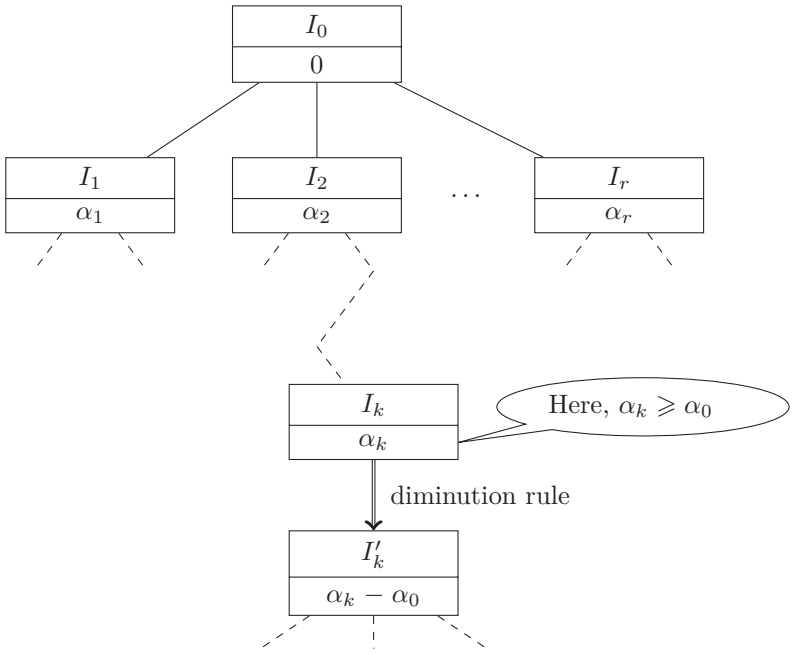


Figure 3: Illustration of Algorithm 13

Using Rule (D2), since $f(I) - f(I') \leqslant 1$, $\frac{\alpha_0}{1+\alpha_0}(f(I) - f(I')) \leqslant \frac{1}{1+\alpha_0}(\alpha - \alpha') \leqslant f(S) - f(S') + \frac{1}{1+\alpha_0}(\alpha - \alpha')$.

ii) $\alpha < \alpha_0$. Then, I has up to r children, and using Rule (B2), we know that for one child $[I', \alpha']$, and for any solution $S' \in \mathscr{S}(I')$ $f(S) - f(S') \geqslant f(I) - f(I')$ where $S = \mathbb{PB}(S')$. In particular:

$$\frac{\alpha_0}{1+\alpha_0}\left(f(S) - f(S')\right) \geqslant \frac{\alpha_0}{1+\alpha_0}\left(f(I) - f(I')\right) \tag{2}$$

Also, we use the definition of α and α', and Rule (B3), and we have $f(S) - f(S') \geqslant \alpha' - \alpha$. So:

$$\frac{1}{1+\alpha_0}(f(S) - f(S')) + \frac{1}{1+\alpha_0}(\alpha - \alpha') \geqslant 0 \tag{3}$$

Summing (2) and (3), we get $\frac{1+\alpha_0}{1+\alpha_0}\left(f(S) - f(S')\right) + \frac{1}{1+\alpha_0}(\alpha - \alpha') \geqslant \frac{\alpha_0}{1+\alpha_0}(f(I) - f(I'))$, which derives (1).

From the above cases, one can see that it is possible to find a branch in the tree, going from the root $[I_0, 0]$ to a leaf $[I_f, \alpha_f]$, satisfying the Proposition (1) at each step. We find in polynomial time a solution S_f for the instance I_f, and then apply the paybacks functions to find a solution S_0 on I_0. If we sum the equations derived from (1), we get $f(S_0) - f(S_f) + \frac{1}{1+\alpha_0}(-\alpha_f) \geqslant \frac{\alpha_0}{1+\alpha_0}(f(I_0) - f(I_f))$.

As $\alpha_f \geqslant 0$, and $f(S_f) = f(I_f) \geqslant 0$, we have: $f(S) - f(S_f) \geqslant \frac{\alpha_0}{1+\alpha_0} f(I) - \frac{\alpha_0}{1+\alpha_0} f(I_f)$ i.e., $f(S) \geqslant \frac{\alpha_0}{1+\alpha_0} f(I) \underbrace{f(S_f) - \frac{\alpha_0}{1+\alpha_0} f(I_f)}_{\geqslant 0} \geqslant \frac{\alpha_0}{1+\alpha_0} f(I) = \rho f(I)$, that concludes the proof of the ratio.

We now study the running-time of the algorithm. Set, for each node $[I, \alpha]$, $T([I, \alpha]) = |I| - \alpha/\alpha_0$.

When doing a branching rule, a node is replaced by at most r nodes of sizes $|I| - a_1, |I| - a_2, \cdots, |I| - a_r$, and with α values $\alpha + s_1, \alpha + s_2, \cdots, \alpha + s_r$. Note that, for each child, there are at most $|I|$ (i.e., a polynomial number) diminutions (the size decrease by one after a diminution).

In any diminution that transforms $[I, \alpha]$ into $[I', \alpha']$, the quantity T becomes T', and $T' \leqslant T$, so these will not affect the overall computation time of our algorithm. Indeed, we know that $|I'| \leqslant |I| - 1$ because of Rule (D1). Also we have $\alpha' = \alpha - \alpha_0$, due to the definition of the algorithm. Then, $T' = |I'| - \frac{\alpha'}{\alpha_0} \leqslant |I| - 1 - \frac{\alpha - \alpha_0}{\alpha_0} = |I| - \frac{\alpha}{\alpha_0} = T$.

With the branching rule, if the k-th child of $[I, \alpha]$ is $[I_k, \alpha_k]$, and the quantity T becomes T_k, then we have $T_k \leqslant T - a_k - s_k/\alpha_0$. Indeed, we know using Rule (B1) that $|I_k| \leqslant |I| - a_k$. Also, by definition, we have $\alpha_k = \alpha + s_k$. Then, $T_k = |I_k| - \frac{\alpha_k}{\alpha_0} \leqslant |I| - a_k - \frac{\alpha + s_k}{\alpha_0} = T - a_k - \frac{s_k}{\alpha_0}$.

So, the complexity according to the quantity T satisfies $C(T) \leqslant \sum_{k=1}^{r} C(T - a_k - \frac{s_k}{\alpha_0})$. For the first node, the quantity T equals the size of the initial instance. Finally, we get a complexity $O^*(\gamma_\rho^{|I|})$, with γ_ρ root of $1 = \sum_{k=1}^{r} \gamma_\rho^{-a_k - s_k(1-\rho)/\rho}$, as claimed. □

Note that γ corresponds to the complexity of the exact branching algorithm (its complexity is $O^*(\gamma^n)$). The fact that $\gamma_\rho < \gamma$ follows from the fact that at least one s_i is strictly positive by hypothesis.

We can adapt this technique to make it work for minimization problems. The necessary condition to use this algorithm is as following: we need a branching rule with nearly the same properties as before: Rule (B2) becomes the following **Rule (B2')**: $\exists i, \forall S_i \in \mathscr{S}(I_i)$, $f(\mathbb{PB}_i(S_i)) - f(S_i) \leqslant f(I) - f(I_i)$ (note that Rule (B3) remains the same).

Example 15*:* For Min Vertex Cover, the branching rule "corresponding" to the one in Example 6 is either to add a vertex v to the vertex cover (and to remove it from the graph), or to take all its neighbors in the vertex cover (and to remove them and v from the graph). This branching rule clearly satisfies Rules (B2') and (B3).

For the diminution rule, we need the same properties with the following modifications:
– Rule (D2) becomes: **Rule (D2')**: $f(I') \leqslant f(I)$;
– Rule (D3) becomes: **Rule (D3')**: $\forall S' \in \mathscr{S}(I')$, $f(\mathbb{PB}'(S')) \leqslant 1 + f(S)$, i.e., the $\mathbb{PB}'$ function will not increase the size of a solution by more than one.

Take for Min Vertex Cover the rule consisting in removing a vertex v in the graph G (thus getting a graph G') and putting it into the vertex cover under construction. Given a vertex cover on G', the payback function simply adds v to G', thus producing a vertex cover of G. Then Rules (D1), (D2') and (D3') are obviously satisfied.

Theorem 16. *For a minimization problem, with the new set of properties, Algorithm 13 has an approximation ratio of* $(\alpha_0 + 1)/\alpha_0$ *and the same complexity as before.*

Proof. To prove the correctness, we have nearly the same property as in the demonstration above: each non-leaf node $[I, \alpha]$ in the tree has at least one child $[I', \alpha']$ with a payback function $\mathbb{PB}$, and satisfying:

$$\forall S' \in \mathscr{S}(I'), f(\mathbb{PB}(S')) - f(S') + \frac{\alpha' - \alpha}{\alpha_0} \leqslant \frac{\alpha_0 + 1}{\alpha_0}(f(I) - f(I')) \tag{1'}$$

Indeed, we have two cases with respect to α.

i) $\alpha \geqslant \alpha_0$. In this case, I' is derived from I using the diminution rule. For any solution $S' \in \mathscr{S}(I')$, we call $S = \mathbb{PB}'(S')$. Then with Rule (D3'), we have $f(S) - f(S') \leqslant 1$. Also, as we used a diminution, we know that $\alpha' = \alpha - \alpha_0$, and so $f(S) - f(S') + \frac{\alpha' - \alpha}{\alpha_0} \leqslant 1 - 1 \leqslant 0$. Using Rule (D2'), we know that $f(I) - f(I') \geqslant 0$, which gives (1').

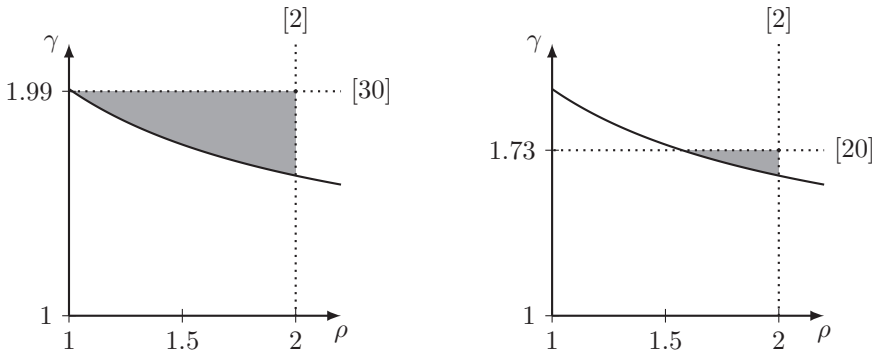
ii) $\alpha < \alpha_0$. Then, I has up to r children, and using Rule (B2'), we know that for one child $[I', \alpha']$, and for any solution $S' \in \mathscr{S}(I')$, $f(S) - f(S') \leqslant f(I) - f(I')$ where $S = \mathbb{PB}(S')$. Also, $f(S) - f(S') \geqslant \alpha' - \alpha$ (cf. Rule (B3)). So, it holds that $f(S) - f(S') + \frac{\alpha' - \alpha}{\alpha_0} \leqslant \frac{1+\alpha_0}{\alpha_0}(f(S) - f(S')) \leqslant \frac{1+\alpha_0}{\alpha_0}(f(I) - f(I'))$.

The end of the proof is therefore the same as previously. □

Let us conclude this section by the following remark. For clarity, we have fixed constants to 1 in Rules (D1) and (D2). It is worth noticing that similar results would immediately follow from using other constants. In particular (this will be useful for the examples below), suppose that we simply replace Rule (D2) by a new one: *(D2"): There exists a* k *such that* $f(I') \geqslant f(I) - k$. In this case, all we have to do is dividing the evaluation function by k, which preserves the approximation ratio. Therefore, the parameters s_i will become s_i/k (see Rule (B3)), and the complexity will be $O^*\left(\gamma^{|I|}\right)$, with γ root of $1 = \sum_{i=1}^{r} \gamma^{-a_i} - (s_i/k)(1-\rho)/\rho$.

3.2 Getting results with approximate branching

The techniques devised in Section 3.1 may be applied to a large variety of optimization problems. We first present in this section a detailed example on the Feedback Vertex Set problem. Then



(a) On undirected graphs (b) On directed graphs

Figure 4: Approximating Feedback Vertex Set. The grey area represents the improvement over the existing.

we show that it is in some sense more general than the technique of partitioning the instance [4] since it also works for hereditary problems. Finally, we present three other examples, handling the Feedback Vertex Set problem, the Max Cut problem and a non-hereditary problem, the Max k-Coverage problem. Note that we mention here simple applications of this technique: as already mentioned, a more involved implementation of this technique has been used in [15] to derive the best known bounds (until now) for approximating Max Sat for some ratios.

3.2.1 Feedback Vertex Set

Given a graph of order n, Feedback Vertex Set consists of finding a minimum-size set of vertices whose deletion makes the graph acyclic. This problem is **NP**-hard in both directed or undirected graphs ([27]) and cn be optimally solved in time $O^*(1.9977^n)$ in the former class of graphs [30], and in time $O^*(1.7347^n)$ in the latter one [20]. Both cases are approximable in polynomial thime within ratio 2 [2]. We show how our method can be used in order to obtain ratios between 1 and 2 with improved computation times.

To obtain a branching rule on this problem, we formalize an instance of Feedback Vertex Set as a graph $G(V, E)$ with a set D of vertices excluded from the solution (we set $D = \emptyset$ at the beginning of the algorithm). The size of an instance is $|V| - |D|$. If we have an instance $I = (G, D)$, we can pick a vertex v from $V - D$ and define $I_1 = (G[V - \{v\}], D)$ and $I_2 = (G, D \cup \{v\})$. Our payback functions are $\mathbb{PB}_1(S) = S \cup \{v\}$ and $\mathbb{PB}_2(S) = S$. Note that if $D \cup \{v\}$ contains a cycle, then we can skip the I_2 branch as this instance will not have a solution. This branching has the desired properties: the size of the instance decreases by one on each branch, the payback functions give a feasible solution, they can be computed in polynomial time, and the optimal solution can be reached (take the first branch only when the vertex v is in the optimal solution), which ensures exhaustiveness.

We also have a diminution rule: if we have an instance $I = (G, D)$, then we pick any vertex v and we have $I' = (G[V - \{v\}], D)$ and $\mathbb{PB}'(S) = S \cup \{v\}$. Also here we see that the size of the instance decreases by one, that the computations are polynomial, and the payback function does not increase the size of a solution by more than one.

Theorem 17. *Feedback Vertex Set can be approximately solved within any ratio ρ in time $O^*(\gamma^n)$, with γ solution of the equation $\gamma \leqslant \gamma^{-1} + \gamma^{-\rho}$ (this result is summarized Figure 4).*

3.2.2 Hereditary problems

Our method can also handle hereditary problems. Let P be a maximization problem where, given an instance I, feasible solutions are subsets of a given set S that satisfy a given hereditary property π (if $S' \subseteq S$ satisfies π, then any $S'' \subseteq S'$ satisfies π), the goal being to maximize the size of the subset.

To make hereditary problems fitting the previous framework, we formulate P in a different (equivalent) manner, defining a problem P' where an instance is given by: (i) an instance I for the problem P; (ii) a set of nodes in the solution $T \subseteq S$; (iii) a set of nodes not in the solution $D \subseteq S$ with $D \cap T = \emptyset$. Feasible solutions of an instance (I, T, D) of P' are subsets $S' \subseteq S \setminus (T \cup D)$ such that $T \cup S'$ satisfies π (i.e., these are the feasible solutions of I containing T). The value of the solutions S' is its size $|S'|$.

As already mentioned, for any instance (I, T, D) of this new problem P', we have a natural branching rule: we take an element from S that is neither in T nor in D, and we add it either in T or in D. The branching continue as long as T is a feasible solution. This gives a trivial branching algorithm in time $O^*(2^n)$, where $n = |S|$. This branching rule is strictly monotonic. As a diminution rule, we only add one element to D. In the worst case, the element was in the optimal solution, then its size decreases by one.

So our algorithm applies, and gives an answer with approximation ratio ρ and computation time in $O^*(\gamma_\rho^n)$, γ_ρ being the greatest real solution of the equation $1 = \gamma_\rho^{-1} + \gamma_\rho^{-1-(1-\rho)/\rho}$.

Note that if this indicates that approximate branching algorithms seems to apply to more problems than the technique of partitioning the instance, it is worth noticing that the latter technique derives better running times for hereditary problems (compare $\gamma^{\rho n}$ with the running time of Theorem 14).

3.2.3 Minimization Problems

Analogous results can be obtained for minimization problems where feasible solutions are subsets of a given set S that satisfy some property π such that if $S' \subseteq S$ satisfies π, then any $S'' \supseteq S'$ satisfies π, the goal being to minimize the size of the subset.

Suppose that P is such a problem. Then, we rephrase P as in the maximization case by considering the sets T and D of already taken and discarded vertices. A feasible solution is a set $S' \subseteq S \setminus (T \cup D)$ such that $S' \cup T$ satisfies π, and its value is $|S'|$. We can use as branching rule the same as above. The diminution rule is to take an element that and put it in the solution, which satisfies Rule (D2'). This allow us to find a ρ-approximation in $O^*(\gamma^n)$, γ being the solution of the equation $1 = \gamma^{-1} + \gamma^{-\rho}$.

For example, Min Set Cover consists, giving a set $E = \{E_1, E_2, \cdots, E_n\}$ of sets, of finding a subset E' of E with as few elements as possible such that $\bigcup_{E_i \in E'} E_i = \bigcup_{E_i \in E} E_i$. This problem is **NP**-hard. We can can use the method of approximate branching because this problem is to find a subset of a solution satisfying the property above, and any set containing a solution is a solution.

3.2.4 Max Cut

Max Cut consists, given a graph $G(V, E)$, of partitioning V into two subsets V_1 and V_2 such that the number of edges having one endpoint in V_1 and the other one in V_2 is maximal. The set of these crossing edges is called the cut (associated to V_1).

This problem could also be seen as an hereditary problem: if a set S of edges is a cut, then any subset of S is also a cut (or at least a part of a cut). This would lead, using our algorithm, to an approximation algorithm with a complexity of $O^*(\gamma^{|E|}) = O^*(\gamma^{(n^2)})$ (with $\gamma > 1$ a constant depending on the approximation ratio). On the other hand, the basic branching schema consisting of fixing, let say, set V_1 and then of considering, for every vertex, that either it belongs, or it does not belongs to V_1, solves the problem in time $O^*(2^n)$ and this is the best worst-case running-time known for this problem in general graphs. However, in sparse graphs, i.e., in graphs with maximum degree bounded by d, the problem is solvable in time $O^*(2^{(1-(2/d))n})$ [10]. On the other hand, Max Cut is approximable in polynomial time within ratio 0.8785 [23], but it is **APX**-hard [31].

Consider a graph with maximum degree d. We will show how we can use the techniques developed above in order to get non-trivial moderately exponential ratios.

Branching rule. We need a branching rule that guarantee us that, for each instance of the problem, at last one child will increase the size of any solution by one (cf. Rule (B3)). To achieve this, we first eliminate one pathologic case. If the graph is not connected, then we apply our algorithm on each connected component (which gives, of course, the optimal solution). Therefore, we will suppose in the following that our graph is connected. In the initial instance, we pick one

vertex randomly, and we assign it in V_1. Then, the branching rule is to chose a vertex that is adjacent to another one in V_1 or V_2, and to make two children, placing this vertex in V_1 or V_2. If the first node was in V_1, then placing the node in V_2 will guarantee an increasing of the solution of at least one.

Diminution rule. If we now consider a graph G of degree at most d, then we can use the following easy result.

Proposition 18. *In a graph of degree at most d, removing a vertex and all its edges decreases the size of the maximum cut by at most d.*

Now we have a diminution rule: take a vertex, throw it away and remove every edge adjacent to this vertex in the same time, and the following theorem holds.

Theorem 19. *If $G(V,E)$ is a graph of maximum degree d, then we can compute a ρ-approximation for Max Cut in time $O^*(\gamma^{|V|})$, with γ solution of $1=\gamma^{-1}+\gamma^{-1-(1-\rho)/d\rho}$.*

3.2.5 Max k-Coverage

Let us finally consider a non-hereditary problem, the Max k-Coverage problem in graphs with maximum degree d. In the Max k-Coverage problem, given a graph $G(V,E)$ and an integer k, one asks for determining a subset S of V of size k that maximizes the number of edges incident to a vertex in S.

Here again, we will consider graphs of bounded degree d. Note that this problem is solvable in $O^*(2^{(d-1)n/(d+1)})$ by the basic branching schema (take a vertex or do not take it), in graphs with maximum degree d [9], and it is approximable in polynomial time within ratio 3/4 [16, 26], but it is **APX**-hard as generalization of Min Vertex Cover.

Max k-Coverage problem is polynomial if k is bounded by a fixed constant. The branching rule is trivial: take a vertex, add it in the solution or not. Stop branching when k vertex have been added. The diminution rule is simply to take a vertex and eliminate it. As the degree of the graph is bounded, then this elimination can't decrease the solution's size by more than d.

Theorem 20. *If $G(V,E)$ is an instance of Max k-Coverage of degree at most d, then we can compute a ρ-approximation of the solution with complexity in $O^*(\gamma^{|V|})$, with γ solution of $1=\gamma^{-1}+\gamma^{-1-(1-\rho)/d\rho}$.*

4 Around FPT approximation

We have proposed in this article a general framework to devise approximate branching algorithms that achieve polynomial time approximation schemata. We conclude this article by some considerations on similar issues for parameterized complexity. As mentioned in introduction, several recent articles aim at combining parameterized complexity and approximation algorithms (see for instance [28]). Following these works, we shall define approximation classes in the parameterized framework[9].

Class 21 (**AFPT**)*:* A problem is in **AFPT**$[\rho]$ if there is a function f such that, for any instance of size n, it is possible in time $O^*(f(k,\rho))$ either to find a solution of size at least ρk or to show that there is no solution of size k.

A similar definition can be given for minimization problems.

Class 22 (**ASFPT**)*:* A problem is said to have an **ASFPT** if it is in **AFPT**$[\rho]$ for any $\rho \neq 1$.

Clearly, we know that **FPT** $\subseteq$ **AFPT**$[\rho]$ for any ρ. Though there does not exist to our knowledge a "natural" problem that distinguishes between these classes, the inclusion is indeed strict. Take a $\rho > 1$ and the following problem: given a graph G, feasible solutions are proper colorings of G, and a coloring has value 3 if it uses at most 3 colors and 3ρ otherwise. Clearly, this problem is ρ approximable in polynomial time (hence in FPT time) and is not in **XP** (so not in **FPT**) unless **P** = **NP**.

[9] Note that we only consider here the standard parameterization by the value of the solution, results dealing with other parameters have also been considered [28].

Proposition 23. *$\forall\rho$, $\boldsymbol{FPT} \subsetneq \boldsymbol{AFPT}[\rho]$, unless $\boldsymbol{P} = \boldsymbol{NP}$.*

We also know that $\mathbf{APX}[\rho] \subseteq \mathbf{AFPT}[\rho]$ (where $\mathbf{APX}[\rho]$ denotes the set of problems ρ-approximable in polynomial time). Note that Min Vertex Cover is in **FPT** but is **NP**-hard to approximate with ratio $\rho < 1.36$ [11]. This shows strict inclusion for ratios $\rho < 1.36$. An amplification of the value of solutions allows to extend the strict inclusion for any ratio ρ: take t such that $1.36^t > \rho$, and define a modified vertex cover problem where the value of a vertex cover V' is $|V'|^t$. Then the problem is still in **FPT** but a polytime approximation ratio smaller than 1.36^t would solve vertex cover within ratio 1.36.

Proposition 24. *$\forall\rho$, $\boldsymbol{APX}[\rho] \subsetneq \boldsymbol{AFPT}[\rho]$ unless $\boldsymbol{P} = \boldsymbol{NP}$.*

Finally, in the field of approximation parameterized algorithms, strong negative results have been obtained for Min Independent Dominating Set[10] by [13], leaving as open questions the existence of approximation parameterized algorithms for Max Independent Set and Min Dominating Set. As a last result, we deduce from the self-improvement property of Max Independent Set [22] that either Max Independent Set has a parameterized approximation schema or it does not admit constant approximation parameterized algorithms for any ratio. Note that this result has been used in [14] to obtain conditional negative results for the existence of parameterized approximation algorithms for Max Independent Set.

Theorem 25. *If there is a ratio ρ such that Max Independent Set is in $\boldsymbol{AFPT}[\rho]$, then it is also in $\boldsymbol{AFPT}[1-\epsilon]$ for any $\epsilon > 0$.*

Proof. We suppose that Max Independent Set is $\mathbf{AFPT}[\rho]$, which means that we have an algorithm with complexity $O(f(k)p(n))$ (with p a polynomial function) that either outputs an independent set of size at least ρk or outputs no (and in this latter case there is no independent set of size k).

Let $G(V, E)$ be a graph of size n. We build a new graph G' [22] with n^2 vertices $V_{i,j}$, with $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n$. There is an edge between (i_1, j_1) and (i_2, j_2) if:

- $i_1 = i_2$ and $(j_1, j_2) \in E$.
- $i_1 \neq i_2$ and $(i_1, i_2) \in E$.

First, we remark that if we have an independent set of size k in G, then we have an independent set of size k^2 in G'. Reciprocally, if we have an independent set of size k in G', then we can find an independent set of size $\sqrt{k}$ in G:

- either this independent set has vertices in more than $\sqrt{k}$ copy of G, and then the numbers of these copy give an independent set on G.
- either is has vertices in less than $\sqrt{k}$ copy of G, and then there is at least $\sqrt{k}$ vertices in one of them, which gives an independent set on G too.

When given a graph G and a parameter k, we apply our parameteric algorithm to G' with the parameter k^2. If the algorithm outputs no, then we know that there is no independent set of size k in G and we output no too. Else, the algorithm outputs an independent set of size ρk^2 on G'. We know that this leads to an independent set of size $\sqrt{\rho}k$ on G. The complexity of this new algorithm $O(f(k^2)p(n^2))$. In other words, we know that Max Independent Set is now $\mathbf{AFPT}[\sqrt{\rho}]$. Iterating this process proves the claimed result. □

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation. Combinatorial optimization problems and their approximability properties.* Springer-Verlag, Berlin, 1999.

[2] A. Becker and D. Geiger. Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83 (1): 167–188, 1996.

[10] Given a graph G, find the minimum size vertex subset which is both an independent set and a dominating set.

[3] L. Brankovic and H. Fernau. Combining two worlds: parameterized approximation for vertex cover. In O. Cheong and K.-Y. Chwa ans K. Park, editors, *Proc. International Symposium on Algorithms and Computation, ISAAC'10*, volume 6506 of *Lecture Notes in Computer Science*, pages 390–402. Spinger-Verlag, 2010.

[4] N. Bourgeois, B. Escoffier, and V. Th. Paschos. Approximation of max independent set, min vertex cover and related problems by moderately exponential algorithms. *Discrete Appl. Math*, 159 (17): 1954–1970, 2011.

[5] N. Bourgeois, B. Escoffier, and V. Th. Paschos. Efficient approximation of MIN COLORING by moderately exponential algorithms. *Inform. Process. Lett.*, 109(16):950–954, 2009.

[6] L. Cai and X. Huang. Fixed-parameter approximation: conceptual framework and approximability results. In H. L. Bodlaender and M. A. Langston, editors, *Proc. International Workshop on Parameterized and Exact Computation, IWPEC'06*, volume 4169 of *Lecture Notes in Computer Science*, pages 96–108. Springer-Verlag, 2006.

[7] M. Cygan and M. Pilipczuk. Exact and approximate bandwidth. *Theoret. Comput. Sci.*, 411(40–42):3701–3713, 2010.

[8] E. Dantsin, M. Gavrilovich, E. A. Hirsch, and B. Konev. MAX SAT approximation beyond the limits of polynomial-time approximation. *Ann. Pure and Appl. Logic*, 113:81–94, 2001.

[9] F. Della Croce and V. Th. Paschos. Efficient algorithms for the MAX k-VERTEX COVER problem. *J. Comb. Optim.* To appear

[10] F. Della Croce, M. J. Kaminski, and V. Th. Paschos. An exact algorithm for MAX CUT in sparse graphs. *Oper. Res. Lett.*, 35:403–408, 2007.

[11] I. Dinur and M. Safra. The importance of being biased. *Proc. STOC 2002*: 33-42.

[12] R. G. Downey and M. R. Fellows *Parameterized Complexity.* Monographs in Computer Science. Springer, 1999.

[13] R. G. Downey, M. R. Fellows, and C. McCartin. Parameterized approximation problems. In H. L. Bodlaender and M. A. Langston, editors, *Proc. International Workshop on Parameterized and Exact Computation, IWPEC'06*, volume 4169 of *Lecture Notes in Computer Science*, pages 121–129. Springer-Verlag, 2006.

[14] B. Escoffier, E. Kim and V. Th. Paschos: Subexponential and FPT-time Inapproximability of Independent Set and Related Problems. `http://www.lamsade.dauphine.fr/sites/default/IMG/pdf/cahier_321.pdf` *Cahier du Lamsade 321*, 2012.

[15] B. Escoffier, V.Th. Paschos and E. Tourniaire. Approximating Max Sat by moderately exponential and parameterized algorithms. In *Proc. TAMC'12*, to appear in *Lecture Notes in Computer Science*. Springer-Verlag, 2012.

[16] U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *J. Algorithms*, 41(2), 2001.

[17] M. R. Fellows, A. Kulik, F. A. Rosamond and H. Shachnai. Parameterized Approximation via Fidelity Preserving Transformations. *Proc. of ICALP (1) 2012*, pages 351-362, 2012.

[18] F. V. Fomin, F. Grandoni and D. Kratsch. A measure & conquer approach for the analysis of exact algorithms. *J. ACM* 56(5), 2012.

[19] F. V. Fomin and D. Kratsch *Exact Exponential Algorithms.* Springer, 2011.

[20] Fomin, Fedor, Villanger and Yngve Finding induced subgraphs via minimal triangulations *Proc. of STACS'10*, 2010.

[21] M. Fürer, S. Gaspers, and S. P. Kasiviswanathan. An exponential time 2-approximation algorithm for bandwidth. In *Proc. International Workshop on Parameterized and Exact Computation, IWPEC'09*, volume 5917 of *Lecture Notes in Computer Science*, pages 173–184. Springer, 2009.

[22] M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. W. H. Freeman, San Francisco, 1979.

[23] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6):1115–1145, 1995.

[24] J. Håstad. Some optimal inapproximability results. *Proc. 29th Ann. ACM Symp. on Theory of Comp., ACM*, pages 1–10, 1997.

[25] R. Impagliazzo and R. Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.* 62(2): 367-375, 2001.

[26] G. Jäger and A. Srivastav. Improved approximation algorithms for maximum graph partitioning problems. *J. Comb. Optim.*, 10(2):133–167, 2005.

[27] R. Karp Reducibility Among Combinatorial problems. *Complexity of Computer Computations*, 85–103, 1972.

[28] D. Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

[29] D. Moshkovitz and R. Raz. Two-query PCP with subconstant error. *J. ACM* 57(5), 2010.

[30] I. Razgon. Computing minimum directed feedback vertex set in $O^*(1.9977^n)$. *Proceedings of the 10th Italian Conference on Theoretical Computer Science* 70–81, 2007.

[31] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *J. Comput. System Sci.*, 43:425–440, 1991.