Super-polynomial approximation branching algorithms

Bruno Escoffier, Vangelis Th. Paschos, Emeric Tourniaire

RAIRO-Oper. Res. 50 (2016) 979–994
DOI: 10.1051/ro/2015060

RAIRO Operations Research
www.rairo-ro.org

SUPER-POLYNOMIAL APPROXIMATION BRANCHING ALGORITHMS*

Bruno Escoffier[1], Vangelis Th. Paschos[2] and Emeric Tourniaire[2]

Abstract. We give sufficient conditions for deriving moderately exponential and/or parameterized time approximation schemata (*i.e.*, algorithms achieving ratios $1 \pm \epsilon$, for arbitrarily small ϵ) for broad classes of combinatorial optimization problems via a well-known technique widely used for deriving exact algorithms, namely the branching tree pruning.

Mathematics Subject Classification. 68W25, 05C85, 68Q25.

Received June 11, 2015. Accepted November 23, 2015.

1. Introduction

Polynomial time approximation, exponential time computation and fixed parameter tractability (FPT) are three main fields aiming at coping with **NP**-hard problems.

In polynomial time approximation (see for instance [1]), one tries to devise polynomial time algorithms that compute feasible solution that are as close as possible to the optimal one, under an *a priori* defined criterion, called approximation ratio.

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 $O(\gamma^n \text{poly}(n))$ for some polynomial poly), for some $\gamma > 1$. A natural objective is then to minimize the value of γ.

Fixed parameter tractability 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 $O(f(k)\text{poly}(n))$ for some function f and some polynomial p [14]. 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 parameter) is the value of the sought solution. Formally, for the standard parameterization (we will use the standard parameterization in all this article), 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 $O(f(k)\text{poly}(n))$ for some function f and polynomial p.

Keywords. Branching algorithm, moderately exponential approximation, approximation schema.

* *Research supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010*

[1] Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu, 75005 Paris, France.
bruno.escoffier@lip6.fr

[2] PSL Research University, Université Paris-Dauphine, LAMSADE CNRS UMR 7243, Paris, France.
{paschos,tourniaire}@lamsade.dauphine.fr

Article published by EDP Sciences

Interestingly, negative results have been obtained in these three fields. For polynomial time approximation, tight lower bounds are known for many optimization problem, under the hypothesis $\mathbf{P} \neq \mathbf{NP}$. For instance, Max 3SAT[3] is well known to be approximable in polynomial time within ratio 7/8, and not approximable in polynomial time within ratio $7/8 + \epsilon$ unless $\mathbf{P} = \mathbf{NP}$ for any $\epsilon > 0$ [25]. Dealing with exponential time computation, the Exponential Time Hypothesis (*ETH*), introduced in [26, 27], conjectures 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[4]. Finally, the parameterized complexity theory considers the hypothesis $\mathbf{W[1]} \neq \mathbf{FPT}$ (see [14] 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 several very interesting studies try to put together either polynomial approximation and exact computation, or parameterized computation and polynomial approximation, in order to build new tools coping with intractability (see, for example [5, 7, 8, 10, 11, 15, 16, 22, 30]. One of the basic goals of all these works is to show how one can achieve ratios unachievable in polynomial time spending time that is much lower than the best-known exact or parameterized time for the problems handled. Let us note that we say that a minimization (maximization, respecitvely) problem Π, together with a parameter p, is *parameterized ρ-approximable* if there exists an FPT-time algorithm which computes a solution of size at most (at least, respectively) ρp whenever the input instance has a solution of size at most (at least, respectively) p; otherwise, the algorithm returns some feasible solution.

Our aim in this paper is to study links between approximation and exact computation (whatever the parameter measuring complexity is) by devising a unified framework that handles exact and parameterized computations and approximation. Furthermore, rather than focusing on one particular optimization problem, we tackle the question of giving sufficient conditions for any problem optimally solved by branching algorithms to admit good approximation algorithms in exponential time, with respect to any parameter used to measure its complexity.

Our basic guideline here is the following. Consider a problem Π parameterized by some parameter p (that can be the size n of its instance, the value k of its optimal solution, the treewidth of the input-graph, if it is a graph-problem,...). Suppose that Π is inapproximable within some ratio ρ (in polynomial time) and, moreover, that the best-known algorithm for Π runs in $O^*(F(p))$[5], for some function F that increases with p. Does there exist an approximation algorithm for Π, achieving ratio better than ρ and running in time much smaller than $O^*(F(p))$? Especially, we focus on the existence of "approximation schemata", *i.e.*, algorithms with ratios $1 \pm \epsilon$, for arbitrarily small ϵ, with running time $O^*(F_\epsilon(p))$, where $F_\epsilon < F$ for a problem solvable (exactly) in $O^*(F(p))$. Typically, if $F(p) = \gamma^p$ for some γ, we seek $F_\epsilon(p) = \gamma_\epsilon^p$ for $\gamma_\epsilon < \gamma$.

The paper is organized as follows. In Section 2, we give sufficient conditions to derive approximation schemata using branching algorithms. Then, in Section 3, we derive such schemata parameterized by n (the size of the instance) and p (the standard parameter), for some well-known problems. Finally, in Section 4, we sketch a formal framework of a structure for moderately exponential and parameterized approximation.

2. Approximation schemata by branching algorithms

In both exact and parameterized computation, a very popular technique for achieving exact algorithms with non-trivial running-time results is the so-called "search-tree pruning". So, a natural question that arises ant is handled in this section, is whether its applicability can be extended for deriving good approximation results or not.

[3] 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.

[4] 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.

[5] Note that in this article, with a slight abuse of notation, $O^*(F(p))$ means $F(p)q(|I|)$ for some polynomial q.

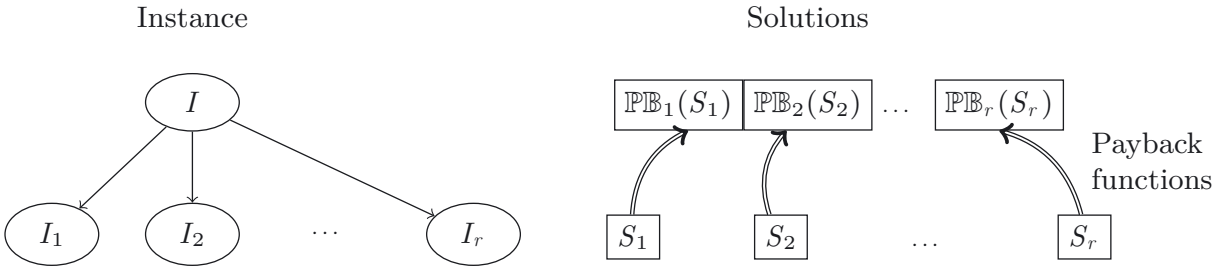

FIGURE 1. Branching rule.

Let us first note that a class of problems is already known to admit such exponential time approximation schemata (with respect to n), the class of hereditary maximization problems[6] (see, for instance, [5]): by partitioning the instance, if a hereditary graph problem is solvable in $O^*(\gamma^n)$ then it is ρ-approximable in time $O^*(\gamma^{\rho n})$ for any ratio ρ. The same holds also for Max SAT in [11, 16]. In the second of these papers such schemata are obtained with respect to both n (the number of variables), m (the number of clauses), and k the standard parameter, *i.e.*, the number of satisfied clauses.

Example 1. We give a basic overview of the schema for Max Independent Set (with respect to n). Using classical reduction rules (including vertex folding, see for instance [21]), 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 on it in the following way: either we add the vertex in the final solution and remove its neighbors, or we remove this vertex. In the first case, the size of the instance decreases by at least 4 (one vertex in the solution and 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 satisfies $C(n) \leqslant C(n-5) + C(n-1) \leqslant O^*(1.32^n)$. Note that much better exact algorithms are known for Max Independent Set, and that it is easy to improve this technique of approximation (remove a clique for instance). The aim of this simple example is only to illustrate the basic idea that will be developed in the sequel.

Given an instance I of an optimization problem, we denote by $\mathscr{S}(I)$ the set of feasible solutions of I, by $f(I)$ the optimal value of I, and by $f(I,S) \in \mathbb{N}$, or $f(S)$ when no confusion is possible, the value of $S \in \mathscr{S}(I)$. We adopt a measure function $I \mapsto p$. Note that p can be any parameter of the instance (including $f(I)$, $|I|$,...). We just require that when $f(I) = 0$ the instance is solvable in polynomial time[7]. For clarity we will suppose that $p \in \mathbb{N}$ but similar results can be obtained with $p \in \mathbb{R}_+$ (in particular p can be the weight associated to $|I|$ in the measure and conquer paradigm [21]).

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 rise to several instances that are the children of this node[8]. The leaves of the tree correspond to trivial instances. This might be formalized as follows in the case of a maximization problem.

Definition 2.1 (Branching rule; Fig. 1)**.** A branching rule of fan-out r is a mapping from an instance$n$ I to a set of r instances $(I_1, I_2, \ldots, I_r)$ and a set of algorithms $(\mathbb{PB}_1, \ldots, \mathbb{PB}_r)$, called payback functions, with the following properties:

(a) If S_i is a solution for the instance 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 $\mathbb{PB}_i(S_i)$).

[6] Maximization problems 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.

[7] This is obviously the case for any parameterized problem in **XP**.

[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.

(b) The branching and every payback function are computable in polynomial time with respect to the size of the instance.
(c) (parameter reduction) There exists a sequence of positive numbers $(a_i)_{i=1,\dots,r}$ giving a guarantee on the instance size decreasing. This gives **Rule (B1)**: $\forall i$, $p_i \leqslant p - a_i$.
(d) (exhaustiveness) For (at least) one of the instances I_i, the payback function increases the value of a feasible solution by at least the difference in the optimum between I and I_i. This, gives **Rule (B2)**: $\exists i, \forall S \in \mathscr{S}(I_i)$, $f(\mathbb{PB}_i(S)) - f(S) \geqslant f(I) - f(I_i)$.

Obviously, the larger 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.

Example 2 (Example 1 continued)**.** For Max Independent Set, if we branch on a vertex v and I_1 (resp., I_2) is the branch where v is taken (resp., discarded), then $\mathbb{PB}_1(I, I_1, S) = S \cup \{v\}$ and $\mathbb{PB}_2(I, I_2, S) = S$.

Given an optimization problem P with such a branching rule, we define the branching Algorithm 1. The following result is folklore. We give it for reasons of self-readability of the paper.

Algorithm 1: Naive branching algorithm.

Input : An instance I_0
Output: An optimal solution for I_0
1 Build a tree using the following rule:
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 reduction rule.
c) Each node with an empty instance is a leaf.
2 **for** *each leaf in the tree* **do**
3 Compute an optimal solution. 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.
4 **end**
5 Output the solution of the tree's root.

Proposition 2.2. *Algorithm 1 gives an optimal solution in time* $O^*(\gamma^{p_0})$*, with* γ *the only positive solution of the equation* $1 = \sum_{i=1}^{r} \gamma^{-a_i}$.

Proof. Let us denote by I_0 the instance in the root of the tree. 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 until we reach a leafk 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 $S_k, S_{k-1}, \dots, S_0$. With these solution, we have, using (B2), $f(S_i) - f(S_{i+1}) \geqslant f(I_i) - f(I_{i+1})$, for $i = 0, \dots, k-1$. 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 a parameter p, this number satisfies the equation $C(p) \leqslant C(p - a_1) + C(p - a_2) + \dots + C(p - a_r)$, which gives the complexity claimed. □

In the case of a parameterized problem with the standard parameterization, the previous algorithm provides (with the same complexity) an optimal solution if there is a solution of value at least p in the instance (with a slight obvious modification in the leaves when there is no solution of the seek value).

Given a maximization problem, assume that a branching algorithm as Algorithm 1 has been devised. We are interested in modifying it, in order to get an approximation algorithm with a better running time. We show that

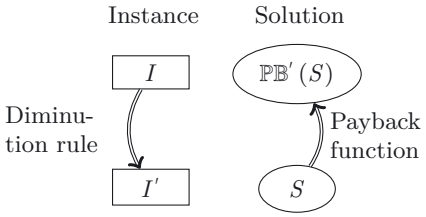

FIGURE 2. Diminution rule.

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 on the branching rule (it it directly linked to the problem and not to a particular branching algorithm).

Definition 2.3 (Strict monotonicity)**.** A branching rule is said to be strictly monotonic if for at least one instance I_i the payback function increases (strictly) the value of such solutions (while the other ones do not decrease the value of feasible solutions). Formally, there exists a sequence of nonnegative integers $(s_i)_{i=1,\dots,r}$, with at least one of them greater than 0 such that the following rule is satisfied: **Rule (B3)**: $\forall i, \forall S \in \mathscr{S}(I_i)$, $f(\mathbb{PB}_i(S)) \geqslant f(S) + s_i$.

Obviously, the greater the integers s_i, the faster the approximation algorithm. In Example 1, the branching rule is strictly monotonic: the payback function will increase by one the value of the solutions in the branch where we have taken v (and by 0 in the other branch).

Definition 2.4 (Diminution rule; Fig. 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 parameter decreases by at least one. This gives **Rule (D1)**: $p' \leqslant p - 1$.
(2) The value of the optimal solutions does not decrease by more than one. This results in **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 3 (Example 1 continued)**.** For Max Independent Set, using as parameter the number of vertices, the diminution rule used consists of removing a vertex from the graph: from a graph G we obtain a graph G' with one vertex less (Rule (D1)). Obviously, 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) that satisfies Rule (D3).

Given a problem that has both a diminution rule and a strictly monotonic branching rule, we can define the ρ-approximation Algorithm 2. As for the exact branching algorithm, the case of a parameterized problem with standard parameter, a slight and obvious modification of Algorithm 2 allows to output a solution of value at least ρp (and even at least $\rho f(I)$) if there is a solution of value at least p in the instance.

Theorem 2.5. *Algorithm 2 outputs a ρ-approximate solution in time $O^*(\gamma_\rho^p)$, where γ_ρ is the only positive root of the equation $1 = \sum_{i=1}^{r} \gamma_\rho^{-a_i - s_i(1-\rho)/\rho}$.*

Algorithm 2: A ρ-approximation pruning algorithm.

Input : An instance I_0 and a ratio ρ.
Output: A ρ-approximate solution for I_0.

1 Define α_0 such as $\alpha_0 = \frac{\rho}{1-\rho}$.
2 Build a tree using the following rule:
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 $([I_1, \alpha + s_1], \ldots, [I_r, \alpha + s_r])$, corresponding to the branching rule.
c) For each node $[I, \alpha]$, with I non-empty and $\alpha \geqslant \alpha_0$, build 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).
3 **for** *every leaf in the tree* **do**
4 Compute backward a solution 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.
5 **end**
6 Output the best solutions among those calculated.

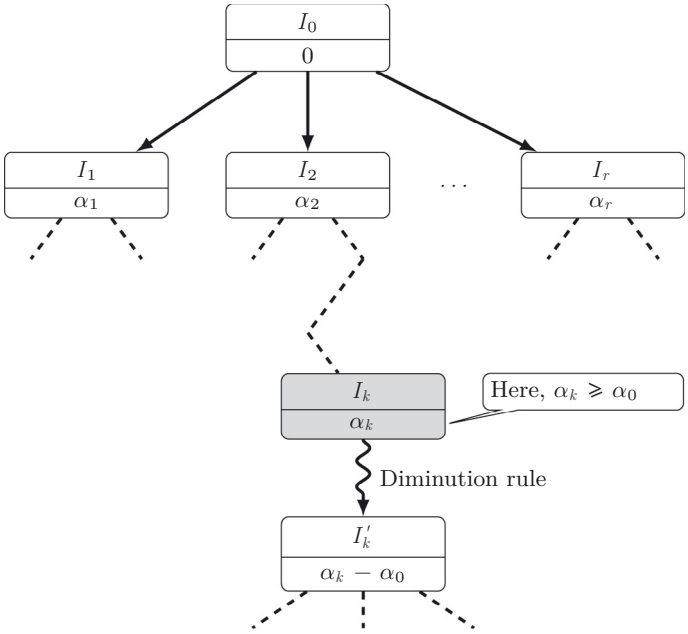

FIGURE 3. Illustration of Algorithm 2.

Note that γ_1 (for $\rho = 1$) corresponds to the complexity of the exact branching algorithm (its complexity is $O^*(\gamma_1^n)$). The fact that $\gamma_\rho < \gamma_1$ for any $\rho \neq 1$ follows from the fact that at least one s_i is strictly positive by hypothesis.

Proof. (Thm. 2.5). We first analyze the approximation ratio of the algorithm. Let us show that each internal node $[I, \alpha]$ in the tree has at least one child $[I', \alpha']$ with a payback function $\mathbb{PB}$, and satisfying the following:

$$\forall S' \in \mathscr{S}(I'), f(\mathbb{PB}(S')) - f(S') + \frac{1}{1+\alpha_0}(\alpha - \alpha') \geqslant \frac{\alpha_0}{1+\alpha_0}(f(I) - f(I')). \tag{2.1}$$

Concerning α and α_0, we examine the following two cases.

(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:

$$\frac{1}{1+\alpha_0}(\alpha - \alpha') = \frac{1}{1+\alpha_0} \times \alpha_0 = \frac{\alpha_0}{1+\alpha_0}.$$

Using Rule (D2), since $f(I) - f(I') \leqslant 1$, it holds that:

$$\frac{\alpha_0}{1+\alpha_0}(f(I) - f(I')) \leqslant \frac{1}{1+\alpha_0}(\alpha - \alpha') \leqslant f(S) - f(S') + \frac{1}{1+\alpha_0}(\alpha - \alpha').$$

(ii) $\alpha < \alpha_0$. Then, I has up to r children. Using Rule (B2), we immediately derive 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:

$$\frac{\alpha_0}{1+\alpha_0}(f(S) - f(S')) \geqslant \frac{\alpha_0}{1+\alpha_0}(f(I) - f(I')). \tag{2.2}$$

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

$$\frac{1}{1+\alpha_0}(f(S) - f(S')) + \frac{1}{1+\alpha_0}(\alpha - \alpha') \geqslant 0. \tag{2.3}$$

Summing (2.2) and (2.3), we get:

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

which derives (2.1).

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 (2.1) at each step. We build in polynomial time a solution S_f for the instance I_f, and then apply the payback functions to build a solution S_0 on I_0. If we sum the equations derived from (2.1), we get:

$$f(S_0) - f(S_f) + \frac{1}{1+\alpha_0}(-\alpha_f) \geqslant \frac{\alpha_0}{1+\alpha_0}(f(I_0) - f(I_f)).$$

Since $\alpha_f \geqslant 0$, and $f(S_f) = f(I_f) \geqslant 0$, we have:

$$f(S) - f(S_f) \geqslant \frac{\alpha_0}{1+\alpha_0} f(I) - \frac{\alpha_0}{1+\alpha_0} f(I_f)$$
$$\Longrightarrow f(S) \geqslant \frac{\alpha_0}{1+\alpha_0} f(I) + \underbrace{f(S_f) - \frac{\alpha_0}{1+\alpha_0} f(I_f)}_{\geqslant 0} \geqslant \frac{\alpha_0}{1+\alpha_0} f(I) = \rho f(I)$$

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 applying a branching rule, a node is replaced by at most r nodes of sizes $|I| - a_1, |I| - a_2, \ldots, |I| - a_r$, and with α values $\alpha + s_1, \alpha + s_2, \ldots, \alpha + s_r$. 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 $T' \leqslant T$; so, this will not affect the overall computation time of the algorithm. Indeed, $|I'| \leqslant |I| - 1$, 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 kth 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, using Rule (B1), $|I_k| \leqslant |I| - a_k$. Also, by definition, $\alpha_k = \alpha + s_k$. 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:

$$C(T) \leqslant \sum_{k=1}^{r} C\left(T - a_k - \frac{s_k}{\alpha_0}\right).$$

For the first node, the quantity T equals the parameter of the initial instance. Finally, we get a complexity $O^*(\gamma_\rho^{|I|})$, where γ_ρ is the largest real root of the equation $1 = \sum_{k=1}^{r} \gamma_\rho^{-a_k - s_k(1-\rho)/\rho}$, as claimed. □

This technique can be adapted to work for minimization problems also. It suffices that Rule (B2) becomes **Rule (B2')**: $\exists i, \forall S_i \in \mathscr{S}(I_i)$, $f(\mathbb{PB}_i(S_i)) - f(S_i) \leqslant f(I) - f(I_i)$.

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 parameter of a solution by more than one.

Example 4. For Min Vertex Cover, the branching rule "corresponding" to that of Example 1 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 together with v from the graph). This branching rule clearly satisfies Rules (B2') and (B3). Now take the rule consisting of 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. This example is also a simple example (not leading to some interesting result) for illustrating the idea in minimization problems.

When setting $\alpha_0 = \frac{1}{\rho - 1}$ in Algorithm 2 (now $\rho > 1$) to deal with minimization problems, then the following holds.

Theorem 2.6. *For a minimization problem, with the new set of properties, Algorithm 2 outputs a ρ-approximate solution in time $O^*(\gamma_\rho^p)$, where γ_ρ is the only positive root of the equation $1 = \sum_{i=1}^{r} \gamma_\rho^{-a_i - s_i(\rho-1)}$.*

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

$$\forall S' \in \mathscr{S}(I'), f(\mathbb{PB}(S')) - f(S') + \frac{\alpha' - \alpha}{\alpha_0} \leqslant \frac{\alpha_0 + 1}{\alpha_0}(f(I) - f(I')). \tag{2.1'}$$

Indeed, we have two cases with respect to α.

$\alpha \geqslant \alpha_0$. 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 (2.1').

$\alpha < \alpha_0$. 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 simplicity, we have fixed differences in Rules (D1) and (D2) to 1. It is worth noticing that similar results would immediately follow using other (constant) values. In particular (this will be useful for the results given later), suppose that we simply replace Rule (D2) by a new **Rule (D2")**: there exists 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 now be $O^*(\gamma^p)$, with γ root of $1 = \sum_{i=1}^{r} \gamma^{-a_i-(s_i/k)(1-\rho)/\rho}$.

3. Using approximate branching to get approximation schemata

3.1. Parameterization by the instance size

3.1.1. 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 and undirected graphs [29] and can be optimally solved in $O^*(1.9977^n)$ [32], and in $O^*(1.7347^n)$ [20], respectively. Both cases are approximable in polynomial time 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 for 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 have the following diminution rule: in instance $I = (G, D)$, pick any vertex v and we have $I' = (G[V - \{v\}], D)$ and $\mathbb{PB}'(S) = S \cup \{v\}$. We can see that the size of the instance decreases by one, the computations are polynomial and the payback function does not increase the size of a solution by more than one.

Discussion above, immediately derives the following theorem.

Theorem 3.1. *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 Fig. 4).*

More generally, suppose that P is a minimization problem where given an instance I, feasible solutions are subsets of a given set S that satisfy a given 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.

Then, we rephrase the problem 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 s claimed, a ρ-approximation in $O^*(\gamma^n)$, γ being the solution of the equation $1 = \gamma^{-1} + \gamma^{-\rho}$.

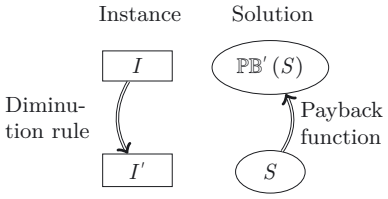

FIGURE 4. Approximating Feedback Vertex Set. The grey area represents the improvement over the existing.

3.1.2. Max cut

Given a graph $G(V, E)$, Max Cut consists 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).

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 $O^*(2^n)$. In sparse graphs, *i.e.*, in graphs with maximum degree bounded by d, the problem is solvable in time $O^*(2^{(1-(2/d))n})$ [12]. On the other hand, Max Cut is approximable in polynomial time within ratio 0.8785 [24], 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.

To obtain a branching rule for this problem guaranteeing that, at least one child will increase the size of any solution by one (*cf.* Rule (B3)), we first eliminate one pathological 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 arbitrarily, and we assign it to V_1. Then, the branching rule is to choose 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.

For the diminution rule, if we consider a graph G of degree at most d, then we can use the following easy result: *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*. So, the diminution rule is as follows: take a vertex, throw it away and remove every edge adjacent to this vertex in the same time, and the following holds.

Proposition 3.2. *Max Cut can be approximately solved within ratio ρ in time $O^*(\gamma^{|V|})$, with γ the largest real solution of $1 = \gamma^{-1} + \gamma^{-1-(1-\rho)/d\rho}$.*

3.1.3. Maximization 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, 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 elements in the solution $T \subseteq S$; (iii) a set of elements 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 Thm. 2.5).

3.1.4. Max k-coverage

In the Max k-coverage problem, given a graph $G(V, E)$ and an integer k, we ask for finding 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 [17, 28], 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. We get a similar result as in the case of Max cut.

3.2. Standard parameterization and minimization problems

Let us consider the Min Vertex Cover problem with the standard parameterization. Here, there exists a naive well known branching with complexity $O^*(2^p)$ that allows to determine if there is a vertex cover of size p (and even to build one if so): consider an uncovered edge, and branch adding either one or the other of its extremities. This branching fits our definition, and we use the following diminution rule: we consider any edge, and we add both its endpoints in the solution, following the spirit of the modified version of Algorithm 2 for minimization problems. When applying this diminution rule on an instance I we get an instance I' where $f(I') \leqslant f(I) - 1$, $p' \leqslant p - 1$ and $f(\mathbb{PB}(S)) \leqslant 2 + f(PB(S))$, so the rule fulfills the requirements.

Note that this algorithm, leading to an approximation schema, is very similar in the case of vertex cover to the works of [18] and [6]. Note also that the diminution rule can be combined with other parameterized algorithms (better than $O^*(2^p)$) to derive better running times exactly in the same way.

The discussion made above can be generalized to capture more FPT problems (parameterized by the standard parameter). Consider, for instance, problems consisting of finding a minimum-size set of items satisfying some property π such that every set containing a subset satisfying π, also satisfies π.

If there exists a branching rule that any time adds an item among r in the solution (that is the parameter diminishes by at least 1 on any branch), this gives rise to an FPT exact algorithm running in time $O^*(r^p)$. Then, a simple reduction rule consists of adding the r items of a given step in the solution. Using the same analysis as in Theorem 2.6, we derive the following result.

Theorem 3.3. *Problems satisfying properties as π are ρ-approximable, for any $\rho > 1$, in FPT-time $O^*(r^{k(r-\rho)/(r-1)})$.*

The above discussion can be immediately generalized to enhance also maximization problems.

4. Towards a formal definition of structure for moderately exponential and parameterized approximation

4.1. Approximation classes for moderately exponential approximation

We focus on the well known class **NPO** of optimization problems: an **NPO** problem Π 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. In the class **NPO**, the following assumptions are required:

- instances can be recognized in polynomial time;

- the size of feasible solutions are polynomially bounded in the size of the instance; moreover, it can be checked in polynomial time whether a given S is in $\mathscr{S}(I)$ or not;
- 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 $\gamma_\rho < \gamma$ for some ratio ρ, or even for any ratio ρ, 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 $\mathbf{EAS}[\delta]$ 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 $\mathbf{EAX}[\gamma, \rho]$.

Class 2 (EAX). For $\delta > 1$, an **NPO** problem is in $\mathbf{EAX}[\gamma, \rho]$ 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, obviously, any problem that can be solved exactly in time $O^*(\delta^n)$ is in $\mathbf{EAS}[\delta']$ for any $\delta' > \delta$. On the other hand, one may expect at first sight that if a problem is in $\mathbf{EAS}[\delta]$, then the problem will be solvable in $O^*(\delta^n)$. This is the case for several exponential approximation schemata obtained so far in the literature, but the following result is worth being mentioned: Max Unused Colors[9] 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 $O^*(2.25^n)$ [4].

To get this "reverse" property, we need to enforce the notion of approximation schema. As in the polytime approximation framework, we may say that a problem has a *full γ-exponential approximation schema* if it is possible to perform a $(1-\epsilon)$ approximation in time $p(1/\epsilon, n) \times \gamma_\epsilon^n$ for some polynomial p and $\gamma_\epsilon < \gamma$. Then, we have the following easy property.

Property 4.1. If an **NPO** maximization problem has a full γ-exponential approximation schema and is polynomially bounded[10], then it can be solved in time $O^*(\gamma^n)$.

Proof. If the value is bounded by $q(n)$, with q a polynomial, then using the schema, we can get a $(1+1/(q(n)+1))$-approximation in time $O(p(q(n)+1, n)\gamma^n)$, 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 observation on the classes **EAS** and **EAX**. The following inclusions are trivial.

Property 4.2. The following holds:

(i) $\forall \delta < \delta' \mathbf{EAS}[\delta] \subseteq \mathbf{EAS}[\delta']$;
(ii) $\forall \gamma < \gamma', \forall \rho, \mathbf{EAX}[\gamma, \rho] \subseteq \mathbf{EAX}[\gamma', \rho]$;
(iii) $\forall \gamma, \forall \rho' < \rho, \mathbf{EAX}[\gamma, \rho] \subseteq \mathbf{EAX}[\gamma, \rho']$.

An interesting question is whether these inclusions are strict or not. Let us consider the strong exponential time hypothesis (SETH) [26], which claims that there is no constant $\gamma < 2$ such that SAT is solvable in time $O^*(\gamma^n)$.

[9] 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.

[10] This means that the value of a solution is bounded by a polynomial in the size of the instance.

Property 4.3. Assuming SETH:

(i) $\forall \delta < \delta' \mathbf{EAS}[\delta] \subsetneq \mathbf{EAS}[\delta']$;
(ii) $\forall \gamma < \gamma', \forall \rho, \mathbf{EAX}[\gamma, \rho] \subsetneq \mathbf{EAX}[\gamma', \rho]$;
(iii) $\forall \gamma, \forall \rho' < \rho, \mathbf{EAX}[\gamma, \rho] \subsetneq \mathbf{EAX}[\gamma, \rho']$.

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

Let $P_{\delta,\delta''}$ be the following problem. An instance of the problem is a set of n variables $(x_1, \dots x_n)$, n non-negative integers $(a_1, \dots, a_n)$ 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 $f(x_1, \dots, x_n)$, 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 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:

$$\frac{\ln(2)N}{\ln(\delta'')} \leq n < \frac{\ln(2)N}{\ln(\delta'')} + 1 \Longrightarrow \frac{\ln(2)}{\ln(\delta'')} \leq \frac{n}{N} < \frac{\ln(2)}{\ln(\delta'')} + \frac{1}{N} \leq \frac{\ln(2)}{\ln(\delta)}$$

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. In 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 $\{0, \dots, 2^{a_i} - 1\}$. The function f is now trivial: its value is 1 if $(x_1, \dots, x_n)$ 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 claimed inclusion. 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).

For the third one, simulate with the previous trick SAT by a problem not solvable in $O^*(\delta^n)$ under SETH, and state $f(x_1, \dots, x_n) = 1$ if this corresponds to a satisfying assignment, and $f(x_1, \dots, x_n) = \rho'$ otherwise. Any solution is a ρ' approximation, while getting a ρ approximation would solve SAT. □

4.2. FPT and exponential time approximation schemata

As pointed out in Section 1, our main concern in this article is to provide sufficient conditions for a problem solvable in time $O^*(\gamma^n)$ to be in $\mathbf{EAS}[\gamma]$. To this aim, we provided in Section 2 and 3 a general method to derive exponential time approximation schemata. We now show that schemata can be also derived in an easy way for FPT problems having some basic properties.

Property 4.4. Let Π be a maximization (resp., minimization) problem in **FPT** solvable in $O^*(\delta^k)$ such that:

1. the value of any feasible solution is at most n (n is the size of the instance);
2. if there exists a solution of value k, then there is a solution of value l, for any $l \leqslant k$ (resp., for any $k \leqslant l \leqslant n$).

Then Π is also in $\mathbf{EAS}[\delta]$, 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 Π 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 Π 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 Property 4.4 are not very restrictive and applies to a large range of problems, including for instance Min Vertex Cover, Max SAT, ...

On the basis of what has been discussed in this section, we conclude the paper by sketching a class-landscape for parameterized approximation. As mentioned in introduction, several recent articles aim at combining parameterized complexity and approximation algorithms (see, for instance, [30]). Note that we only consider here the standard parameterization.

Class 3 (AFPT). A problem is in **AFPT**$[\rho]$ if there exists 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 value at least ρk or to show that there is no solution of value k.

A similar definition can be given for minimization problems.

Class 4 (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**.

Property 4.5. $\forall\rho$, **FPT** $\subsetneq$ **AFPT**$[\rho]$, unless **P** = **NP**.

We also know that **APX**$[\rho] \subseteq$ **AFPT**$[\rho]$ (where **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 $\rho < 1.36$ [13]. 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 FPT but a polytime approximation ratio smaller than 1.36^t would solve vertex cover within ratio 1.36.

Property 4.6. $\forall\rho$, **APX**$[\rho] \subsetneq$ **AFPT**$[\rho]$ unless **P** = **NP**.

Finally, in the field of approximation parameterized algorithms, strong negative results have been obtained for Min Independent Dominating Set[11] by [15], 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 [23] 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 [3] to obtain conditional negative results for the existence of parameterized approximation algorithms for Max Independent Set.

Theorem 4.7. *If there is a ratio ρ such as Max Independent Set is in **AFPT**$[\rho]$, then it is also in **AFPT**$[1-\epsilon]$, for any ϵ.*

Proof. We suppose that Max Independent Set is **AFPT**$[\rho]$, which means that we have an algorithm with complexity $O(f(k)\text{poly}(n))$ (with poly 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

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

of size n. We build a new graph G' [23] 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;
- or 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 parameterized 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 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. *Artif. Intell.* **83** (1996) 167–188.

[3] E. Bonnet, B. Escoffier, E. Kim and V.Th. Paschos, On subexponential and fpt-time inapproximability. In *Proc. of International Workshop on Parameterized and Exact Computation, IPEC'13*, edited by G. Gutin and S. Szeider. Vol. 8246 of *Lect. Notes Comput. Sci.* Springer-Verlag (2013) 54–65.

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

[5] 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** (2011) 1954–1970.

[6] L. Brankovic and H. Fernau, Combining two worlds: parameterized approximation for vertex cover. In *Proc. of International Symposium on Algorithms and Computation, ISAAC'10*, edited by O. Cheong and K.-Y. Chwa ans K. Park. Vol. 6506 of *Lect. Notes Comput. Sci.* Spinger-Verlag (2010) 390–402.

[7] L. Cai and X. Huang, Fixed-parameter approximation: conceptual framework and approximability results. In *Proc. of International Workshop on Parameterized and Exact Computation, IWPEC'06*, edited by H.L. Bodlaender and M.A. Langston. Vol. 4169 of *Lect. Notes Comput. Sci.* Springer-Verlag (2006) 96–108.

[8] Y. Chen, M. Grohe and M. Grüber, On parameterized approximability. In *Proc. of International Workshop on Parameterized and Exact Computation, IWPEC'06*, edited by H.L. Bodlaender and M.A. Langston. Vol. 4169 of *Lect. Notes Comput. Sci.* Springer-Verlag (2006) 109–120.

[9] F. Della Croce and V.Th. Paschos, Efficient algorithms for the MAX k-VERTEX COVER problem. *J. Comb. Optim.* **28** (2014) 674–691.

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

[11] E. Dantsin, M. Gavrilovich, E.A. Hirsch and B. Konev, MAX SAT approximation beyond the limits of polynomial-time approximation. *Ann. Pure Appl. Logic* **113** (2002) 81–94.

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

[13] I. Dinur and M. Safra, The importance of being biased. In *Proc. of STOC'02* (2002) 33–42.

[14] R.G. Downey and M.R. Fellows, Parameterized complexity. *Monogr. Comput. Sci.* Springer, New York (1999).

[15] R.G. Downey, M.R. Fellows and C. McCartin, Parameterized approximation problems. In *Proc. of International Workshop on Parameterized and Exact Computation, IWPEC'06*, edited by H.L. Bodlaender and M.A. Langston. Vol. 4169 of *Lect. Notes Comput. Sci.* Springer-Verlag (2006) 121–129.

[16] B. Escoffier, V. Th. Paschos and E. Tourniaire, Approximating max sat by moderately exponential and parameterized algorithms. In *Proc. of Theory and Applications of Models of Computation, TAMC'12*, edited by M. Agrawal, S. Barry Cooper and A. Li. Vol. 7287 of *Lect. Notes Comput. Sci.* Springer-Verlag (2012) 202–213.

[17] U. Feige and M. Langberg, Approximation algorithms for maximization problems arising in graph partitioning. *J. Algorithms* **41** (2001) 174–211.

[18] M.R. Fellows, A. Kulik, F.A. Rosamond and H. Shachnai, Parameterized approximation via fidelity preserving transformations. In *Proc. of ICALP'12*, edited by A. Czumaj, K. Mehlhorn, A. Pitts and R. Wattenhofer. Vol. 7391 of *Lect. Notes Comput. Sci.* Springer-Verlag (2012) 351–362.

[19] F.V. Fomin and D. Kratsch, Exact exponential algorithms. *EATCS*. Springer-Verlag (2010).

[20] F.V. Fomin and Y. Villanger, Finding induced subgraphs via minimal triangulations. In *Proc. of International Symposium on Theoretical Aspects of Computer Science, STACS'10*, edited by J.-Y. Marion and T. Schwentick. Nancy, France (2010) 383–394.

[21] F.V. Fomin, F. Grandoni and D. Kratsch, A measure & conquer approach for the analysis of exact algorithms. *J. Assoc. Comput. Mach.* **56** (2009) 1–32.

[22] M. Fürer, S. Gaspers and S.P. Kasiviswanathan, An exponential time 2-approximation algorithm for bandwidth. In *Proc. of International Workshop on Parameterized and Exact Computation, IWPEC'09*. Vol. 5917 of *Lect. Notes Comput. Sci.* Springer (2009).

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

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

[25] J. Håstad, Some optimal inapproximability results. *J. Assoc. Comput. Mach.* **48** (2001) 798–859.

[26] R. Impagliazzo and R. Paturi, On the complexity of k-sat. *J. Comput. System Sci.* **62** (2001) 367–375.

[27] R. Impagliazzo, R. Paturi and F. Zane, Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.* **63** (2001) 512–530.

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

[29] R.M. Karp, Reducibility among combinatorial problems. *Complexity of computer computations*, edited by R.E. Miller and J.W. Thatcher. Plenum Press, New York (1972) 85–103.

[30] D. Marx, Parameterized complexity and approximation algorithms. *Comput. J.* **51** (2008) 60–78.

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

[32] I. Razgon, Computing minimum directed feedback vertex set in $o(1.9977^n)$. In *Proc. of Italian Conference in Theoretical Computer Science, ICTCS'07*, edited by G.F. Italiano, E. Moggi and L. Laura. World Scientific (2007) 70–81.