Average Case Analysis of NP-complete Problems: Maximum Independent Set and Exhaustive Search Algorithms

BANDERIER Cyril[§] HWANG Hsien-Kuei[⋆]
RAVELOMANANA Vlady[§] ZACHAROVAS Vytas[⋆]

⋆ Institute of Statistical Science
Academia Sinica
Taipei 115, Taiwan
{hkhwang,vytas}@stat.sinica.edu.tw

§ Laboratoire d'informatique de Paris Nord
UMR CNRS 7030 – Institut Galilée - Université Paris XIII
99, Av. J.B. Clément, 93430 Villetaneuse, France
{cb,vlad}@lipn.univ-paris13.fr

Abstract

In this article, we deal with rigorous average case analysis of NP complete problems, relying on some mathematical tools from complex analysis and probability theory. We consider here one of the historical prototype of such problems: Maximum Independent Set (MIS).

For a large class of exhaustive algorithms which always find exactly a MIS, their complexity is directly related to their number of iterations. Under the $\Gamma(n, m)$ and $G(n, p)$ distribution for graphs, we give some fascinating phase transitions between exponential (A^n), superpolynomial ($n^{\ln n}$), and polynomial (n^d) average complexities.

Our approach gives a precise picture of the "complexity landscape" of these algorithms, depending on the average degree (or the ratio vertices/edges) of the graph, and gives access to the location of the "hard regions" where people could then sample their inputs if they want to make some benchmarks (may it be for the worst or average case).

The challenging associated mathematical aspects force us to introduce new analyses (for a large class of recurrences), which will clearly be of interest for many other NP hard problems (typically, optimization problems on graphs).

Direct applications cover graph 3-coloring (which is also a deep motivation for considering our class of exhaustive Tarjan-Chvàtal like algorithms), and numerous constraint satisfaction problems.

1 Introduction

1.1 Some considerations on worst and average case analysis of algorithms

Recently, there have been some studies on relationships between worst case and average case complexity (e.g. the introduction of the notion of "smoothed analysis" allowed to prove that perturbation of worst cases of some simplex-like algorithms are easy to solve [44] ; for some discrete algorithms, it was also be proven [2] that some natural perturbations of the worst cases don't coincide with the hardest regions of the algorithm ; these ideas have also been applied to Knapsack like problems [5]). This allows to have a better understanding of complexity landscapes of algorithms: where are the "easy regions", the hard ones?

This can be useful e.g. in cryptography [33], so that people don't sample inputs (in an *easy on average* region) for an NP-complete problem. This can also be useful in order to tune some algorithms so that they behave faster (this kind of reverse engineering questions are in one sense the starting idea of some sampling algorithms [3] or the Boltzman method [18]).

One of the exiting aspects of Spielman and Teng's notion of smoothed analysis is that they succeeded in applying this idea to combinatorial optimization, where there have been thousands of results related to worst-case studies, but much less related to average-case complexity (let us just mention some interesting cases: the polynomial complexity of the simplex algorithms [9], works on 2-SAT [8], and different other success stories mainly related to random graphs with some tools from probability theory [43] or statistical physics [37] or, in a sense, a mixture of them [32]).

The lack of average-case results in combinatorial optimization can partly be explained by the fact that optimization problems often involved many variables, and that rigorous analysis of their complexity will therefore often rely on multivariate asymptotics, a subject which is too difficult for the actual mathematical technologies. This explains partly why so many articles are dedicated to heuristics/experimentations. Another reason is that for a lot of optimization problems, this is not clear to say that such given distribution on the inputs is more pertinent that such other one. As average case analysis relies first on a chosen distribution (e.g. the uniform distribution), we could then have as many complexity results as assumptions on the distribution. From a mathematical point of view, it is also often the case that the iterations of the algorithm destroy the initial probabilistic model of the inputs, making any rigorous analysis very hard. Last but not least, even if people agree on a probabilistic model, it could sometimes be very difficult to generate inputs according to this distribution, and it can be almost impossible (too time-consuming, exponential complexity of NP hard problems...) to make benchmarks.

All of this makes average-case analysis of optimization problems, a field still largely open, and deeply challenging.

Note that we deal in this article with average-case complexity of some algorithms for an NP problem, and not of *all* algorithms for this problem. We therefore don't have the pretencion to bring new results on $\mathrm{P}\stackrel{?}{=}\mathrm{NP}$, however we believe that our analysis gives more insight on this question.

In this article, we have what we believe to be an original approach to optimization NP-hard problems, for two reasons:

- first, the tools: it is one of the first applications of analytic combinatorics (in the sense of Flajolet-Knuth-Sedgewick [25, 35]) to optimization problems. We therefore hope to give the desire to people from analytic combinatorics to have a look on this large field of applications, and to people from combinatorial optimization to have a look on the kind of tools we are using, in order to recycle them! This also leads to some new interesting mathematical challenges (e.g. asymptotics of new families of recurrences).

- second, the results: we make a precise average case analysis of some NP-hard problems, giving more insights on their thresholds phenomena (why and where a large class of algorithms behaves polynomially/sub-exponientially/exponentially). This also leads to some new interesting computer science challenges (for designing more efficient algorithms).

In this article, we consider one of prototype of NP-complete problems: the maximum independent set problem (but our results can be applied, for free, to many more NP-complete problems!).

1.2 Maximum independent set

Definitions (MIS): In a graph G, an "independent set" (also called "stable set") is a set of vertices without edges between them. Amongst all independent sets of G, consider those of maximum size (=those containing

the largest number of vertices). These are called "maximum independent set" (MIS)[1]. Finding such a set is a NP-hard problem. Deciding if a graph have a MIS of size k, deciding if a subset is a MIS, are NP-complete problems (this is typically proven via a reduction to SAT). This problem was one of Richard Karp's original 21 problems shown NP-complete in his 1972 seminal article [34].

There are many fascinating aspects of MIS with respect to approximation: if we draw with probability 1/2 the edges of a graph, almost all such graphs have a MIS of size $2\log_2 n(1+o(1))$, but it is still an open problem to find (in polynomial time) an independent set of size $(1+\epsilon)\log_2 n$, while a greedy algorithm can reach $\log_2 n$ (a similar challenge exist for different distributions on the graphs) [26]. If we care for worst-case approximation, the situation is even... worst: even when the size of the MIS is almost n, there is no algorithm which can always say "OK, this graph has a MIS of size larger than n^{ϵ}" (more rigorously, under the technical assumption $NP \neq ZPP$, there is no way to find a $n^{1-\epsilon}$ approximation algorithm for the maximum independent set problem [27]). On the other hand, it is possible to detect in polynomial time the existence of larger MIS (of size at least $n(\ln\ln n)^2/(\ln n)^3$, see [23]). All these results may sound anecdotal, but it is noteworthy that most of them were derived thanks the introduction of tools which have deep repercussions on some other computer science questions.

For people not familiar with the zoo of complexity classes, we refer to [1], and for those who wants a quick (but detailed!) overview of approximations results, we refer to [14].

For finding thousands of articles related to our work, it could be useful to mention that the size of a MIS of a graph G is often denoted $\alpha(G)$ and called the "stability number" of G. Note that a MIS will always be a maximal clique in the complement graph[2], so these two problems (MIS and max clique) are "equivalent". Other "equivalent" questions are minimum independent dominating problem or minimal vertex cover (= the complement of a maximal independent set, which have been studied in statistical mechanics in connection with the hard-sphere lattice gas model, a mathematical abstraction of fluid-solid state transitions). These are all sets that can be found by an algorithm that lists all MIS. As cliques and questions of colorability of graphs have a strong relationships, it is not surprising that the exhaustive algorithms we study can be used to tackle graph coloring. E.g., Lawler [36] observed that listing maximal independent sets can also be used to find 3-colorings of graphs: a graph can be 3-colored if and only if the complement of one of its maximal independent sets is bipartite (and then, for sure, you can test bipartiteness in linear time (and constant memory) by a parity argument, or in constant time (and linear memory) with slight changes on the basic algorithms).

For a lot of NPO problems, there will be the similar questions: cost of finding a set, cost of giving (or bounding) its size, in the worst or average case, with memory concerns or not. Some results lead to a speed-up of the algorithms/bounds by an ad-hoc structural study of the inputs, here we will consider some exhaustive algorithms, working on graphs. The huge advantage of this is that is clearly an upper bound to the average-complexity of algorithms for all these previous questions. If we can give upper bound for our "stupid" scheme, then they can be applied directly to the worst and average-complexity of hundreds of related problems !

All of this gives strong motivations for studying the exhaustive algorithms that we present hereafter.

[1] In the sequel, we will always abbreviate "maximum independent set" into MIS, in order to spare trees. Note that "maximal independent set" is another notion: it is a set such that if you try to add any vertex to it, then there is an edge linking this vertex to another vertex from the set. Any MIS is maximal IS, but a maximal IS is often not a MIS! Note that the exhaustive algorithm we will study is immediately suitable to list all maximal IS.

[2] There is an edge between vertice a and b in the "complement" of a graph G if and only if there is no edge between a and b in G. It is then clear that there is a clique of size at least k if and only if there is an independent set of size at least k in the complement graph, since if a subgraph is complete, its complement subgraph has no edges at all.

2 Algorithms

`ExhaustiveMIS:=procedure(`G, IS`) global List;`
`if` G`=empty graph then`
`List:=List` $\cup$ `IS //`IS is a maximal IS, and the algorithm needs to end its exhaustive search to decide if it is actually a MIS.
`else //`we now consider the "pivot" vertex v of smallest label:
`ExhaustiveMIS(`$G - v, IS$`) //`assuming $v \notin$ IS.
`ExhaustiveMIS(`$G -$ `(`v `and all its neighbours),` $IS \cup \{v\}$`) //`assuming $v \in$ IS
`end if`
`end procedure;`
`BasicAlgorithm:=procedure(`G`);`
`List:={}; //`will contain the list of all the maximal IS.
`ExhaustiveMIS(`$G, \{\}$`);`
`RETURN(the largest sets in List)` // this will give the list of the MIS
`end;`

The Basic Algorithm can be optimized by removing, at the beginning of each iteration all the "tree components" from the graph. It is indeed possible to find in polynomial time the MIS of these components. It is even also possible to remove all the prunable[3] nodes; we call the "*optimized algorithm*" the algorithm which does these prunings. Note that this optimization won't affect the "nature" of the complexity of this algorithm, may it be polynomial $O(n^d)$, subexponential $O(n^{\ln(n)})$, or exponential $O(A^n)$.

We call the class of algorithm having such a scheme "algorithms of the Chvátal-Tarjan type", as they are directly inspired of [12] and [45]. The principle of this algorithm is at the core of many algorithms, which all behave like $(1+\alpha)^n$, and there is a quest for finding the smallest α, even for some restricted classes of graphs (e.g., graphs of bounded degree).

It is for sure also possible to optimize the choice of the "pivot" vertex v (mostly if we care to find just a MIS). This leads to what Chvàtal called f-driven algorithms (f being the function which chooses which v we take after each iteration, it could e.g. be the vertex of smallest degree). Chvàtal also then considers at once numerous possible pruning or branching/bounding[4] strategies by what he calls "length of recursive proofs": this will correspond to the smallest part of the binary tree associated to the iterations of the algorithm that you need in order to be sure you got a MIS.

Indeed, pruning just the "prunable" nodes remains a naive approach, and we can go to much more complicated kind of prunings (as this is the case in the article by Tarjan and Trojanowski), it is also easy to design several bounding techniques (e.g. relying on the current depth in the tree, and the size of previously obtained IS). We don't say more on all these possible improvements and we will in this article only consider our "stupid" schemes (the "basic algorithm" and the "optimized algorithm") as they give bounds for any more cleaver algorithm, and as we believe these bounds to be more general and tight (in terms of the classification we address in this article: polynomial, superpolynomial, subexponential, exponential complexities).

Our believing relies on the analysis of the following basic recurrence for μ_n, the average number of iterations of our algorithms:

$$\mu_n := \mu_{n-1} + \sum_{k=0}^{n-1} p_{n,k}\, \mu_{n-k-1} \tag{1}$$

which mimics the fact that we always first consider the vertex v as not being in the IS (and therefore we have

[3] We call "prunable" any node of degree 0,1,2 (after iterations of deletion of such nodes).

[4] From Wikipedia: "Branch and bound is a general algorithm for finding optimal solutions of various optimization problems, especially in discrete and combinatorial optimization. It consists of a systematic enumeration of all candidate solutions, where large subsets of fruitless candidates are discarded *en masse*, by using upper and lower estimated bounds of the quantity being optimized."

to pay the cost of a T_{n-1} exhaustive search) and then we consider v as being in the IS and so we have to pay the price of an exhaustive search on the graph remaining after removing v and its neighbours ($p_{n,k}$ being the probability to have k neighbours). Now let us imagine that we get one day a much faster algorithm (such a scheme is also possible for other problems than MIS) such that:

$$\mu_n := \mu_{n-A} + \mu_{qn}$$

which mimics the fact that we always first remove a fixed number A (and therefore it remains to pay the cost of a μ_{n-A} exhaustive search) or a given proportion p ($0 < p < 1$) of vertices (and therefore it remains to pay the cost of a μ_{qn} exhaustive search, where $q := 1-p$). We can prove that such a (yet unknown) tremendously cleaver algorithm has still an $n^{\ln n}$ complexity (like our stupid algorithm). This explains why we don't want to go into a precise analysis of deeply optimized algorithms, as it will with high probability lead to complexities of the same order as our optimized algorithm! What is more, some of these optimizations are however a pain in the neck from a practical point of view, as they can lead to a drastic increased need of memory (e.g. pruning computations "already done" would lead from a polynomial ($n^3/3$) to an exponential n^2A^n space complexity !). Another drawback of optimization will be that we lost some applications of exhaustive search (like 3-colorability).

As pointed out by Scott and Sorkin [43], it is (much) more difficult to get results *on expectation* than *almost always true* results: exponentially rare events could have an exponentially large contribution, therefore impacting the average cost of the algorithm ; and thus it may be possible that these rare events lead to an exponential average-complexity for the algorithm, whereas almost all inputs give a linear complexity !

We therefore, on these aspects, improve some results of Chvàtal and Pittel. We also give (exponentially better) bounds for several areas, and give precise locations of the phase transitions.

It is also noteworthy that our scheme of algorithms is finally not "necessarily" linked to the MIS problem but in fact to quite a large bunch of branch and bound algorithms, a similar scheme can indeed be applied to numerous MAX 2-CSP (binary 2-variable constraint satisfaction problems) like Max Cut, Max Dicut, Max 2 Lin, Max 2-SAT Max Ones-2-SAT and other examples from [43].

2.1 The $G(n,p)$ graph model

This model corresponds to a probabilistic model over the graphs with n vertices for which we put (with probability p) an edge between each pair of vertices. I.e., one has $2^{\binom{n}{2}}$ graphs, and each graph with n vertices and m edges appears with the probability $\binom{\binom{n}{2}}{m} p^n (1-p)^{\binom{n}{2}-m}$. We refer to [7, 31] for the nice probabilistic methods typically used for this model. Note that under the $G(n,p)$ model, the fundamental recurrence (3) is rephrased as

$$X_n = X_{n-1} + X_{n-1-\mathrm{Bin}(n-1,p)} \tag{2}$$

$$\mu_n = \mu_{n-1} + \sum_{k=0}^{n-1} \binom{n-1}{k} p^k q^{n-k-1} \mu_{n-k-1} \text{ where } \mu_n = E[X_n]\,. \tag{3}$$

2.2 The $\Gamma(n,m)$ graph model

This model corresponds to the uniform distribution amongst all graphs with n vertices having the same number of edges, i.e. each graph with n vertices and m edges appear with the probability $1/\binom{\binom{n}{2}}{m}$.

We refer to [30, 25] for the nice analytic methods typically used for this model.

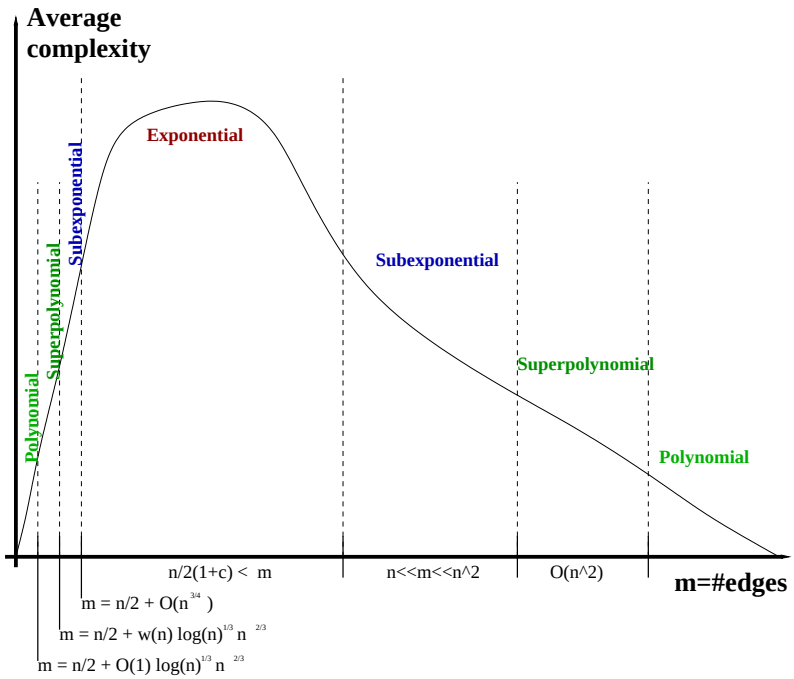

Figure 1: The complexity landscape of the optimized algorithm for exhaustive search of all the maximum independent sets.

Proposition 2.1 (The decreasing property)**.** *The average-complexity of the basic algorithm (=its number of iterations) is decreasing, going from* 2^n *(when* $p = 0$*) to* $2n + 1$ *(when* $p = 1$*).*

Proof. Looking at the basic recurrence (3), it is no so easy to prove analytically. It becomes however trivial to prove if you remember the combinatorial meaning of this recurrence ; indeed, the more we increase p, the more we have chances to have neighbours (as the graphs are still obeying the $G(n, p)$ model, even after iterations), the more we will delete vertices, and so the algorithm is then faster. When $p = 0$ or $p = 1$, the analysis is deterministic, and looking the trace of the algorithm (=the associated binary tree) for an input which a set of isolated vertices ($p = 0$) or the complete graph ($p = 1$) directly gives the values of the proposition. □

For the optimized algorithm, we don't have at all this decreasing property: there is a counterbalance between the fact the algorithm is fast if you are able to kill a lot of neighbours (which is the case if p is large) and the fact the algorithm is fast if you are able to kill a lot of prunable nodes (which is the case if p is small), what is more, after few iterations of the algorithm (in fact, just one iteration), we don't have anymore a $G(n, p)$ distribution, so the situation is much more messy to analyse than for the basic algorithm and it sounds difficult to give asymptotics. However the following claim will be useful:

Proposition 2.2 (The upper-bound property)**.** *The average-complexity of the optimized algorithm is always smaller (modulo a polynomial factor) than the average-complexity of the basic algorithm.*

Proof. Here again, considering the binary tree associated to the recursive calls of the algorithm is the key point: the optimized algorithm is just pruning some part of the binary tree associated to the recursive calls of the basic algorithm, so it will always have less iterations. But there is a price for pruning, and as this price is always bounded by a polynomial cost, we have our claim. □

3 Sparse graphs (almost forests): a polynomial complexity

Theorem 3.1. *For graphs which are almost a forest (i.e. for* $m < n/2$*), it is possible to find a MIS in polynomial time.*

Proof. Direct consequence of a dynamic programming approach. □

It is in fact possible to extend this result to graphs for which all nodes are prunable, but such graphs become exponentially rare for large values of m. Note that the optimized algorithm is in fact of polynomial complexity until $m = n/2 + O((\ln n)^{1/3})n^{2/3}$ as can be proven more via more complicated arguments, see the theorem in the next section.

4 Sparse graphs (few cycles): a superpolynomial complexity

This Section and the 2 next ones are covering the case of sparse graphs with number of edges m ranging from $\frac{n}{2}$ to $\frac{n}{2}(1+\varepsilon)$ for some $\varepsilon > 0$. We will show some "smooth" phase transitions occurring during a relatively short window :

- from polynomial to superpolynomial as m ranges from $\frac{n}{2} + O(\log n^{1/3})n^{2/3}$ to $\frac{n}{2} + O(\omega(n))n^{2/3}$,
- from superpolynomial to subexponential as m ranges from $\frac{n}{2} + O(\omega(n))n^{2/3}$ to $\frac{n}{2} + o(n)$,
- finally, from subexponential to exponential as m ranges from $\frac{n}{2} + o(n)$ to $\frac{n}{2} + O(n)$,

where $\omega(n)$ is such that $\log n^{1/3} \ll \omega(n) \ll n^{1/3}$.

Theorem 4.1. *If $m = \frac{n}{2} + \mu n^{2/3}$ with $\mu = o(n^{1/3})$ then the average cost of the optimized algorithm is at most*

$$\exp\left(\frac{2\log 2}{3}\,(5\log 2 + 4)\,\mu^3\right)\Big(1 + o(1)\Big)\,. \tag{4}$$

To obtain our results, we shall use exponential generating functions (EGFs) to study random graphs. We recall briefly the main EGFs needed for our purposes and for which we refer the reader to the celebrated "giant paper" of Janson, Knuth, Łuczak and Pittel for more details (namely [30, Section 6]).

Let $T(z)$ be the EGF of rooted trees, that is

$$T(z) = ze^{T(z)} = \sum_{n=1}^{\infty} n^{n-1}\frac{z^n}{n!}\,. \tag{5}$$

The EGF of unrooted trees is given by $U(z) = T(z) - \frac{T(z)^2}{2}$. The EGF of unicyclic components is given by[5]

$$W_0(z) = \frac{1}{2}\log\frac{1}{1-T(z)} - \frac{T(z)}{2} - \frac{T(z)^2}{2}\,. \tag{6}$$

More generally, E. M. Wright [46] was able to compute the EGFs W_r, $r \geq 1$. He has shown that for each $r \geq 1$, there exists rational coefficients $(w_{r,d})_{d\in\{0,\cdots,3r+2\}}$ such that

$$W_r(z) = \sum_{d=0}^{3r+2}\frac{w_{r,d}}{(1-T(z))^{3r-d}}\,. \tag{7}$$

Let $E_r(z)$ be the EGF all complex graphs having exactly r edges more than vertices. The sequence $(E_r)_{r\geq 0}$ can be computed recursively as shown by [30, equations (6.7), (6.8), (6.9)], satisfies $E_0(z) = 1$ and

$$E_r(z) = \sum_{d\geq 0} e_{rd}\frac{T(z)^{5r-d}}{(1-T(z))^{3r-d}}\,. \tag{8}$$

[5] We remove the widehat notations associated with graphs from [30] for sake of brevity and clarity.

Let $\mathbb{P}_r(n,\, m)$ be the probability that a random graph with n vertices and m edges has excess r, then we have

$$\mathbb{P}_r(n,\, m) = \frac{n!}{\binom{\binom{n}{2}}{m}}\, [z^n]\, \frac{U(z)^{n-m+r}}{(n-m+r)!} e^{W_0(z)} E_r(z)\,, \tag{9}$$

where $[z^n]\, F(z)$ denotes the n-th coefficient of the EGF $F(z)$. To get our results, we use a saddle point approach which gives estimates or upper-bounds of $\mathbb{P}_r(n,\, m)$ for values of m such that $m \leq \frac{n}{2} + O(n)$ (full details omitted in this extended abstract, see however the next section and the Appendix).

5 Sparse graphs (more cycles): a subexponential complexity

First, we focus on values of m such that $m = \frac{n}{2} + o(n^{3/4})$ which we shall call the *polynomial-superpolynomial-subexponential ranges*. In fact, our results show that the optimized algorithm changes its complexity there from polynomial to superpolynomial and then from superpolynomial to subexponential. Next, we will study the *subexponential-exponential ranges*, that is whenever the number of edges m ranges from $\frac{n}{2} + O(n^{3/4})$ to $\frac{n}{2}(1+\mu)$ (with fixed $\mu > 0$).

Pittel and Wormald [40] were able to obtain the limit joint distribution of the number of vertices in the 2-core[6], the excess of the 2-core, and the number of vertices not in the 2-core. More precisely, let B_n be the event that "there is a unique component of size between $0.5bn$ and $2bn$, and none larger", where b is defined as $b := b(c) = 1 - \frac{t}{c}$, and $t \in (0,\, 1)$ is the unique root of $te^{-t} = ce^{-c}$ (see [40, equations (2.14), (2.15)]). Then, Pittel and Wormald [40, Theorem 6] gave a local limit theorem for the joint distribution of these three parameters in $\Gamma(n, m)$ *conditioned* upon B_n. In what follows, we offer an alternative approach to prove that the limit distribution of excess of random graphs is Gaussian when the number of edges m satisfies $m = \frac{n}{2} + o(n^{3/4})$. Contrary to [40, Theorem 6], our result does not rely on the event B_n defined above.

Theorem 5.1. *The probability that a graph with n vertices and $m = \frac{n}{2} + \mu n^{2/3}$ edges has excess $r = \frac{16}{3}\mu^3 + O(\mu^{3/2})$ is given by*

$$\begin{aligned}\mathbb{P}_r(n,\, m) \;=\; & \left(\frac{3}{160\pi\mu^3}\right)^{1/2} \exp\left(-\frac{3}{160}\frac{\left(r-\frac{16}{3}\mu^3\right)^2}{\mu^3}\right) \\ & \times\left(1 + O\left(\frac{\left(r-\frac{16}{3}\mu^3\right)}{\mu^3}\right) + O\left(\frac{1}{r^{3/4}}\right) + O\left(\frac{\mu^4}{n^{1/3}}\right) + O\left(\frac{r^{4/3}}{n^{1/3}}\right) + O\left(\frac{\mu r}{n^{1/3}}\right)\right)\end{aligned} \tag{10}$$

uniformly as $r,\, \mu,\, n \to \infty$ and $|r - \frac{16}{3}\mu^3| \leq O(\mu^{3/2})$, $r \leq O(n^{1/4})$ and $\mu \leq O(n^{1/12})$.

6 Sparse graphs (large linear number of edges): an exponential complexity

Theorem 6.1. *For a large linear number of edges, the optimized algorithm has an exponential complexity, more precisely if $m = \frac{n}{2}(1+\varepsilon)$ with $n^{-1/4} \leq \varepsilon < \delta$ and fixed δ then the number of iterations of the algorithm is at most*

$$\exp\left(\frac{5(\log 2)^2}{12}\varepsilon^3\, n + \frac{\log 2(\varepsilon^2 - \sigma^2)}{4(1+\varepsilon)}\, n\right)\Big(1 + o(1)\Big)\,, \tag{11}$$

where $\sigma = \sigma(\varepsilon)$ is defined by the formula

$$(1+\varepsilon)e^{-\varepsilon} = (1-\sigma)e^{\sigma}\,. \tag{12}$$

[6] The 2-core is the largest subgraph of minimum degree 2 or more.

7 Sparse-dense graphs: a subexponential complexity

For this Section and the next ones, the asymptotics are given for the basic algorithm, under the $G(n,p)$ probabilistic model. Let us call "sparse-dense graphs", the graphs which have more than a linear number of edges, but less than a quadratic number of edges (for sure, this makes sense only by considering some asymptotic families of graphs). An extrapolation of the Theorem 8.1 (in the next Section) then gives that it costs a subexponential number of iterations to list all the MIS in any family of sparse-dense graphs. With respect to $G(n,p)$, this covers the area $p = c(n)/n$ (for any function $1 \ll c(n) \ll n$), and a full variety of subexponential behaviour.

Theorem 7.1. *As $p \to 0$, we have*

$$\mu_n = \frac{1 - \frac{W(np-p)}{np-p}}{1 + W(np-p)} \left(1 - \frac{W(np-p)}{np-p}\right)^{-n} (1 + O(p)) \ , \tag{13}$$

where $W(z) = ze^{-W(z)}$ is the Lambert W function.

Example 1 : $p = \frac{\log n}{n}$**.** We obtain an average cost growing as $\exp\left(n\frac{\log\log n}{\log n}\right)$.
Example 2 : $p = \frac{1}{n^{1/2}}$**.** The average cost grows as $\exp\left(\frac{1}{2}n^{1/2}\log n\right)$

8 Dense graphs: a superpolynomial complexity

Theorem 8.1. *The average number μ_n of iterations of the basic algorithm is of the type $n^{\ln n}$. (Much) more precisely, with $\rho = 1/\log(1/q)$ and $r = \frac{W(n/\rho)}{n/\rho}$, where W is Lambert-W function (principal solution of the equation $W(z)e^{W(z)} = z$) then μ_n satisfies*

$$\mu_n \sim \frac{e^{(\rho/2)(\log(1/r))^2} G(\rho\log(1/r))}{r^{\rho+1/2}\sqrt{2\pi\rho\log(1/r)}}, \tag{14}$$

as $x \to \infty$, where G is continuous periodic function with period 1:

$$G(u) = q^{(\{u\}^2+\{u\})/2} \sum_{-\infty<j<\infty} \frac{q^{j(j+1)/2}}{1+q^j q^{-\{u\}}} q^{-(j+1)\{u\}}.$$

In Equation (2), the random variables X_j are dependent. This makes it very difficult to obtain any information about asymptotics of the higher moments of X_j as $j \to \infty$. What will happen if we assume that the variables X_j are independent (which seems to be asymptotically true in numerous examples)? The answer to this question is provided by the following theorem.

Theorem 8.2. *Suppose a sequence of independent random variables S_n satisfies a recurrence*

$$S_n \stackrel{d}{=} S_{n-1} + S^*_{n-1-\mathrm{Binom}(n-1;p)}, \tag{15}$$

with $S_0 = 0$ and $S_1 = 1$, where $\stackrel{d}{=}$ stands for equality in distribution, S^*_n has the same distribution as S_n and $\mathrm{Binom}(n-1;p)$ is an independent binomial distribution with parameters $n-1$ and p. Then S_n is asymptotically normal, that is

$$\frac{S_n - \mu_n}{\sigma_n} \stackrel{d}{\to} \mathscr{N}(0,1),$$

where $\stackrel{d}{\to}$ denotes convergence in distribution, while $\mu_n = \mathbb{E}S_n$ and $\sigma_n = \sqrt{\mathbb{V}S_n}$.

9 Dense (almost complete) graphs: a polynomial complexity

When p is very near from 1, even the basic algorithm is very fast: most of the nodes have a linear number of neighbours, and we get all the (very small) independent sets in a polynomial number of iterations (e.g. the algorithm is linear if $p = 1 - c/n$). It is in fact possible to argue a little more, and bootstrapping a polynomial behaviour in the recurrence for the basic algorithm (3) allows to prove that if $p = 1 - c/n^{1/d}$ (for $c = O(1)$) then we have an $\mu_n = O(n^d)$ complexity. If we let d going to $\ln n$, we then have a behaviour coherent with the superpolynomial phase (that we got in Section 9 for any fixed p independent of n). All of this is consistent with our simulations and concludes our description of the complexity landscape of this family of algorithms.

10 Conclusion

Why to give so "precise" analysis (with so much math involved) for our "stupid" algorithms whereas we claim that one of their main use is to bound the complexity of many other algorithms ? Well, because we believe that these bounds are "robust", they should (with high probability) give the *exact* order of the complexity of many other algorithms, because most of the optimizations that one could imagine will (with high probability) only affects marginally the recurrence (i.e., a polynomial speed up, no more) : our classification into the four complexity types (polynomial/superpolynomial/subexponential/exponential) should remain the same. For sure, it is more a matter of believing because it is very hard to *prove* lower bounds for *any* (yet unknown) algorithm (cf $P \neq NP$). But, at least, our analyses suggest that it is a dead-end to look for more and more intricate reductions/prunings, IF you hope to get a tremendous speedup (i.e., a change of complexity type). However, from a practical point of view, there are not a dead-end as it is always very useful to kill e.g. a n^3 factor in front of a superpolynomial average-complexity (as for e.g. $n = 500$, we can then reach in one second what would have taken 4 years before, and therefore doing experiments on larger values of n).

We can apply our approach to a natural extension of the recurrence we studied :

$$X_n \stackrel{d}{=} X_{n-b} + X^*_{n-b-\mathrm{Binom}(n-b;p)},$$

with $X_n = 0$ for $n < b$ and $X_b = 1$, where $b \geqslant 1$. It makes sense to link these quantities to some kind of internal right path length in digital search trees, or to problems like maximum clique partition. We could also go to recurrences with e.g. a disjunction into 3 cases instead of 2 (the trace of the algorithm would then be ternary trees). It could be interesting to see what can be obtained with the contraction method [38].

Further similar approaches can be done for studying the average size of the MIS, the number of MIS, and approximation schemes (e.g. : what happens if one cuts the recursion tree at a given depth). Here also, the approaches should hold for a much more general class of problems than MIS.

References

[1] Scott Aaronson and Christopher Kuperberg, Gregand Granade. *Complexity Zoo*. http://qwiki.stanford.edu/wiki/Complexity_Zoo, 2008.

[2] Cyril Banderier, René Beier, and Kurt Mehlhorn. Smoothed analysis of three combinatorial problems. In *Mathematical foundations of computer science 2003*, volume 2747 of *Lecture Notes in Comput. Sci.*, pages 198–207. Springer, Berlin, 2003.

[3] Cyril Banderier, Philippe Flajolet, Gilles Schaeffer, and Michèle Soria. Random maps, coalescing saddles, singularity analysis, and Airy phenomena. *Random Structures Algorithms*, 19(3-4):194–246, 2001. Analysis of algorithms (Krynica Morska, 2000).

[4] Paul Beame, Russell Impagliazzo, and Ashish Sabharwal. The resolution complexity of independent sets and vertex covers in random graphs. *Comput. Complexity*, 16(3):245–297, 2007.

[5] Rene Beier and Berthold Vöcking. An experimental study of random knapsack problems. *Algorithmica*, 45(1):121–136, 2006.

[6] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM J. Comput.*, 36(4):1119–1159 (electronic), 2006.

[7] Béla Bollobás. *Random graphs*, volume 73 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, second (first ed. in 1985) edition, 2001.

[8] Béla Bollobás, Christian Borgs, Jennifer T. Chayes, Jeong Han Kim, and David B. Wilson. The scaling window of the 2-SAT transition. *Random Structures Algorithms*, 18(3):201–256, 2001.

[9] K.-H. Borgwardt. The average number of pivot steps required by the simplex-method is polynomial. *Z. Oper. Res. Ser. A-B*, 26(5):A157–A177, 1982.

[10] Neil Calkin and Alan Frieze. Probabilistic analysis of a parallel algorithm for finding maximal independent sets. *Random Structures Algorithms*, 1(1):39–50, 1990.

[11] A. Cayley. A theorem on trees. *Quart. J. XXIII.*, pages 376–378, 1888.

[12] V. Chvátal. Determining the stability number of a graph. *SIAM J. Comput.*, 6(4):643–662, 1977.

[13] S.A. Cook. The complexity of theorem proving procedures. In *Proceedings, Third Annual ACM Symposium on the Theory of Computing*, pages 151–158. ACM, New York, 1971.

[14] Pierluigi Crescenzi and Viggo Kann. *A compendium of NP optimization problems*. http://www.csc.kth.se/ṽiggo/wwwcompendium/, 2008.

[15] H. Daudé and V. Ravelomanana. Random 2-xorsat at the satisfiability threshold. In E. S. Laber, C. F. Bornstein, L. T. Nogueira, and Faria L., editors, *Proceedings of LATIN 2008: Theoretical Informatics, 8th Latin American Symposium (Lecture Notes in Computer Science 4957)*, pages 12–23. Springer, 2008.

[16] N. G. de Bruijn. On Mahler's partition problem. *Nederl. Akad. Wetensch., Proc.*, 51:659–669 = Indagationes Math. 10, 210–220 (1948), 1948.

[17] N. G. de Bruijn. *Asymptotic methods in analysis*. Dover Publications Inc., New York, third (1st ed.:1958) edition, 1981.

[18] Philippe Duchon, Philippe Flajolet, Guy Louchard, and Gilles Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combin. Probab. Comput.*, 13(4-5):577–625, 2004.

[19] Philippe Dumas and Philippe Flajolet. Asymptotique des récurrences mahlériennes: le cas cyclotomique. *J. Théor. Nombres Bordeaux*, 8(1):1–30, 1996.

[20] Martin Dyer, Alan Frieze, and Mark Jerrum. On counting independent sets in sparse graphs. *SIAM J. Comput.*, 31(5):1527–1541 (electronic), 2002.

[21] David Eppstein. Small maximal independent sets and faster exact graph coloring. *J. Graph Algorithms Appl.*, 7(2):131–140 (electronic), 2003.

[22] P. Erdős and A. Rényi. On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, 5:17–61, 1960.

[23] Uriel Feige. Approximating maximum clique by removing subgraphs. *SIAM J. Discrete Math.*, 18(2):219–225 (electronic), 2004.

[24] Philippe Flajolet, Donald E. Knuth, and Boris Pittel. The first cycles in an evolving graph. *Discrete Math.*, 75(1-3):167–215, 1989. Graph theory and combinatorics (Cambridge, 1988).

[25] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. Cambridge University Press, Cambridge, 2008. Available at $http://algo.inria.fr/flajolet/Publications/books.html$.

[26] G. R. Grimmett and C. J. H. McDiarmid. On colouring random graphs. *Math. Proc. Cambridge Philos. Soc.*, 77:313–324, 1975.

[27] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182(1):105–142, 1999.

[28] Russell Impagliazzo and Leonid A. Levin. No better way to generate hard np instances than picking uniformly at random. *Proc. of the 31st IEEE Symp. on Foundations of Computer Science*, pages 812–821, 1990.

[29] Svante Janson. Brownian excursion area, Wright's constants in graph enumeration, and other Brownian areas. *Probab. Surv.*, 4:80–145 (electronic), 2007.

[30] Svante Janson, Donald E. Knuth, Tomasz Łuczak, and Boris Pittel. The birth of the giant component. *Random Structures Algorithms*, 4(3):231–358, 1993. With an introduction by the editors.

[31] Svante Janson, Tomasz Łuczak, and Andrzej Rucinski. *Random graphs*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience, New York, 2000.

[32] Mark Jerrum. Large cliques elude the Metropolis process. *Random Structures Algorithms*, 3(4):347–359, 1992.

[33] Ari Juels and Marcus Peinado. Hiding cliques for cryptographic security. *Designs, Codes, and Cryptography*, 20(3):269 – 280, 2000.

[34] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972.

[35] Donald E. Knuth. *The art of computer programming. Volume 1: Fundamental algorithms*. Addison-Wesley, third (1st ed.: 1968 edition, 1997.

[36] E. L. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Lett.*, 5(3):66–67, 1976.

[37] Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. Determining computational complexity from characteristic "phase transitions". *Nature*, 400(6740):133–137, 1999.

[38] Ralph Neininger and Ludger Rüschendorf. A survey of multivariate aspects of the contraction method. *Discrete Math. Theor. Comput. Sci.*, 8(1):31–56 (electronic), 2006.

[39] B. Pittel. On the probable behaviour of some algorithms for finding the stability number of a graph. *Math. Proc. Cambridge Philos. Soc.*, 92(3):511–526, 1982.

[40] B. Pittel and N. Wormald. Counting connected graphs inside-out. *Journal of Combinatorial Th., Series B.*, 93:127 – 172, 2005.

[41] Y. V. Prohorov. Asymptotic behavior of the binomial distribution. *Select. Transl. Math. Statist. Prob.*, 1:87 – 95, 1961.

[42] Vlady Ravelomanana. Another proof of Wright's inequalities. *Inform. Process. Lett.*, 104(1):36–39, 2007.

[43] Alexander D. Scott and Gregory B. Sorkin. Solving sparse random instances of Max Cut and Max 2-CSP in linear expected time. *Combin. Probab. Comput.*, 15(1-2):281–315, 2006.

[44] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463 (electronic), 2004.

[45] Robert Endre Tarjan and Anthony E. Trojanowski. Finding a maximum independent set. *SIAM J. Comput.*, 6(3):537–546, 1977.

[46] E. M. Wright. The number of connected sparsely edged graphs. *J. Graph Theory*, 1(4):317–330, 1977.

11 Appendix

11.1 Proof of Theorem 5.1 [sketch]

Proof. The idea is to analyse the average number of iteration of the optimized algorithm with random graphs from $\Gamma(n, m)$ as input.

For theoretical purposes, it proves important to partition a random graph into its *acyclic part* (trees) and its *cyclic part*. Then in its turn, the cyclic part can be partitioned into the *unicyclic part* (consisting of entirely of unicyclic components) and the *complex part* with connected components that have more edges than vertices. These decompositions allow penetrating and stringent analyses by means of generating functions as shown in [24, 30]

The process of pruning recursively all vertices of degree 1 and folding all vertices of degree 2 is called *reduction*. The obtained graph (possibly a multigraph) after a reduction is said *reduced*. Apart the isolated vertices, the minimum degree of the reduced graph is 3. The algorithm transforms the original graph into a reduced one in polynomial time as explained in [43] (see also Section 1 and 2).

As in [30, Section 13], define the *excess* of a graph as its number of edges plus the number of acyclic components, minus the number of vertices. The proof of our results rely on the most probable excesses of the reduced graph. More precisely, we will show that the probability that the excess of the underlying random graphs is not of order magnitude $O(\mu^3)$ is exponentially small whenever the number of edges satisfies $m = \frac{n}{2}(1+\mu)$ for all values of μ ranging from $0 \leq \mu \leq O(n)$. By [43, Theorem 5], in time $O(\sqrt{2}^{\text{excess}})$ the algorithm finds the optimal solution. Therefore, the average cost of the algorithm is at most roughly $\sqrt{2}^{O(\mu^3)}$. More precisely, by [43, Theorem 5] the average cost is bounded by

$$\sum_{r=0}^{m} \sqrt{2}^{r} \mathbb{P}_r(n,\, m)\,, \tag{16}$$

where $\mathbb{P}_r(n,\, m)$ is the probability that a random graph with n vertices and m edges has excess r. Then, by computing the asymptotic values of $\mathbb{P}_r(n,\, m)$ according to the stated ranges, we get theorem (4.1). □

11.2 Proof of Theorem 6.1 [sketch]

Proof. We just have to give an upper-bound of the probability of excess for m from $\frac{n}{2}+O(n^{3/4})$ to $\frac{n}{2}(1+\varepsilon)$. Since there are more multigraphs (allowing self-loops and multiple edges) than graphs, by means of [30, Theorem 13] we get an upper-bound of the probability of interest. More precisely for any $\varepsilon > 0$, the upper-bound of the probability that a random graph with n vertices and $m = \frac{n}{2}(1+\varepsilon)$ has excess r is roughly

$$\left(\frac{3}{20\pi\varepsilon^3 n}\right)\exp\left(-\frac{3(r-r_0)^2}{20\varepsilon^3 n}\right)(1+o\,(1))\ , \tag{17}$$

when $n^{-1/4} \leq \varepsilon < \delta$ and δ is fixed, and where $r_0 = \frac{\varepsilon^2-\sigma^2}{2(1+\varepsilon)}n$ and $\sigma = \sigma(\varepsilon)$ is the solution of $(1+\varepsilon)e^{-\varepsilon} = (1-\sigma)e^{\sigma}$. □

11.3 Proof of Theorem 7.1 [sketch]

Proof. Using the difference between the binomial and the Poisson distributions (see [41]), we get from Equation (3),

$$\mu_n = \mu_{n-1} + \sum_{j=0}^{n-1}\frac{(n-1)^j p^j}{j!}e^{-(n-1)p}\mu_{n-1-j}\,(1+O(p))\,. \tag{18}$$

Thus, up to a factor $(1+O(p))$ μ_n behaves as

$$\hat{\mu}_n = \hat{\mu}_{n-1} + \sum_{j=0}^{n-1}\frac{((n-1)p)^j}{j!}e^{-(n-1)p}\hat{\mu}_{n-1-j} \tag{19}$$

whose generating function $\hat{M}(z) = 1 + \sum_{n=1}^{\infty}\hat{\mu}_n z^n$ satisfies

$$\hat{M}(z) = \frac{1}{1-z-e^{-(n-1)p+(n-1)pz}}\,. \tag{20}$$

Using singularity analysis of generating functions [25], we find

$$\hat{\mu}_{n+1} \sim \frac{1}{1+W(np)}\left(1-\frac{W(np)}{np}\right)^{-n}\,. \tag{21}$$

□

Remarks. Recall the asymptotic behaviour of $W(x)$ (see [17],

$$W(x) = \log x - \log\log x + O\left(\frac{\log\log x}{\log x}\right)\,, \qquad x\to\infty\,. \tag{22}$$

Then as $np\to\infty$, $\frac{W(np-p)}{np-p}$ is of order of magnitude $\log\,(np-p)/np$ The "exponential" quantity of (13) is then roughly

$$\exp\left(-n\log\left(1-\frac{\log\,(np)}{np}\right)\right) = \exp\left(\frac{\log\,(np)}{p}+O\left(\frac{\log np^2}{np^2}\right)\right) \tag{23}$$

11.4 Proof of theorem 8.1 [Sketch]

Proof. Let $f(z) := \sum_{n\geqslant 0} \mu_n z^n/n!$ denote the exponential generating function (EGF) of μ_n. Then f satisfies the equation $f'(z) = 1 + f(z) + e^{pz} f(qz)$, or, equivalently, denoting by $\tilde{f}(z) := e^{-z} f(z)$ the Poisson generating function (PGF) of μ_n, $\tilde{f}'(z) = \tilde{f}(qz) + e^{-z}$. By taking Laplace transform

$$\tilde{f}^{\star}(s) := \frac{1}{s} \int_0^{\infty} e^{-x/s} \tilde{f}(x)\, \mathrm{d}x. \tag{24}$$

of the above equation and iterating the obtained identity indefinitely, we obtain an explicit expression

$$\tilde{f}^{\star}(s) = \sum_{j\geqslant 0} \frac{q^{j(j+1)/2}}{1+q^j s}\, s^{j+1}. \tag{25}$$

By Laplace inversion formula, we have

$$\tilde{f}(x) = \frac{1}{2\pi i} \int_{r-i\infty}^{r+i\infty} \frac{e^{xs}}{s} \tilde{f}^{\star}\left(\frac{1}{s}\right)\, \mathrm{d}s, \tag{26}$$

where $r > 0$. Using (25) we can obtain an asymptotic estimate of the integral in the above expression. This way we obtain an estimate of $\tilde{f}(x)$ for real values of x.

Further we link the asymptotics of μ_n and $\tilde{f}(n)$ by means of de-Poissonization procedure and prove that

$$\mu_n = \tilde{f}(n) - \frac{n}{2}\, \tilde{f}''(n) + O\left(n^{-2}(\log n)^4 \tilde{f}(n)\right) = \tilde{f}(n)\left(1 + O\left(n^{-1}(\log n)^2\right)\right) \tag{27}$$

□

11.5 Proof of Theorem 8.2 [Sketch]

Proof. The proof of this theorem is done by the method of moments. Independence of variables S_n together with recurrence (15) allows us obtain some recurrent relations for the moments $\mathbb{E}(S_n - \mu_n)^k$. For example, computing variance of the both parts of the recurrence relation (15) we obtain

$$\sigma_n^2 = \sigma_{n-1}^2 + \sum_{0\leqslant j<n} \binom{n-1}{j} q^j p^{n-1-j} \sigma_j^2 + T_{n,2}, \tag{28}$$

where

$$T_{n,2} = \sum_{0\leqslant j<n} \binom{n-1}{j} q^j p^{n-1-j} (\mu_j + \mu_{n-1} - \mu_n)^2.$$

Applying now Theorem 8.1 together with (27) we prove that

$$T_{n,2} \sim q^{-1} p \rho^4 n^{-3} (\log n)^4 \tilde{f}(n)^2.$$

This estimate together with the recurrence relation for variance (28) allows us to obtain an asymptotic estimate of σ_n^2. In a similar way we obtain asymptotics of higher moments, which happen to converge to the moments of normal distribution. Which implies that S_n is asymptotically normal. □