Smoothed Analysis of Three Combinatorial Problems

Cyril Banderier Kurt Mehlhorn Rene Beier*

April 3, 2003

Abstract

Smoothed analysis combines elements over worst-case and average case analysis. For an instance x the smoothed complexity is the average complexity of an instance obtained from x by a perturbation. The smoothed complexity of a problem is the worst smoothed complexity of any instance. Spielman and Teng introduced this notion for continuous problems. We apply the concept to combinatorial problems and study the smoothed complexity of three classical discrete problems: quicksort, left-to-right maxima counting, and shortest paths. This opens a vast field of nice analyses (using for example generating functions in the discrete case) which should lead to a better understanding of complexity landscapes of algorithms.

1 Introduction

For most algorithms, there is a discrepancy between the worst case and the average case behavior. Both quantities convey very useful informations and lead to different type of analysis. For combinatorial algorithms, in the Art of Computer Programming [14] Knuth exhaustively illustrated how discrete mathematics and analysis nicely meets computer science to give incredibly accurate informations, for example leading to full asymptotic expansions for the complexity of some algorithms. In this article, we concentrate on a new notion, called "smoothed analysis" (recently introduced by Spielman and Teng [20]) which is intermediate between average case analysis and worst case analysis and which (we will see) allows to follow the nice wedding initiated by Knuth. The smoothed complexity of an algorithm is

$$\max_x \mathrm{E}_{y \in U_\epsilon(x)} \; C(y) \, ,$$

where x ranges over all inputs, y is a random instance in a neighborhood of x (whose size depends on the smoothing parameter ϵ), E denotes expectation, and $C(y)$ is the cost of the algorithm on input y. In other words, worst-case complexity is *smoothed* by considering the expected running time in a neighborhood of an instance instead of the running time at the instance. If $U_\epsilon(x)$ is the entire input space, smoothed analysis becomes average case analysis (whereas it becomes worst case analysis if $U_\epsilon(x)$ is reduced to x). Smoothed analysis gives information whether instance space contains dense regions of hard instances, see Figure 1. The smoothed complexity of an algorithm is low if worst-case instances are "isolated events" in the instance space.

*Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken (Germany) `banderie, rbeier,mehlhorn@mpi-sb.mpg.de`, `http://www.mpi-sb.mpg.de/units/ag1/people.html`; the three authors are supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT)

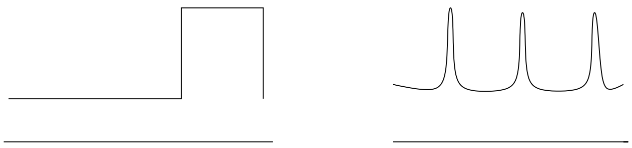

Figure 1: Instance space is indicated by the lines at the bottom of the figures and the neighborhood of an instance is simply an interval around the instance. In the situation on the left, the smoothed complexity will be equal to the worst case complexity (for all small enough ϵ), and in the situation on the right, the smoothed complexity decreases sharply as a function of ϵ.

Spielman and Teng [20] showed that the smoothed complexity of the simplex algorithm (with the shadow-vertex pivot rule) for linear programming is polynomial. Linear programming is a continuous problem. The input is a sequence of real numbers[1] (a cost vector, a constraint matrix, a right-hand side). The smoothing operation adds Gaussian noise with parameter σ to each number in the input. The expected running time of the simplex algorithm for such a perturbed instance is polynomial in $1/\sigma$ and the number of input variables. The other papers on smoothed analysis [2, 7] also discuss continuous problems.

We apply the concept of smoothed analysis to problems defined on sequences and natural numbers. In both cases we first define a natural model of perturbation and then analyze some algorithms.

Partial Permutations: Our first model applies to problems defined on sequences. It is parameterized by a real parameter p with $0 \leq p \leq 1$ and is defined as follows. Consider a sequence $s_1, s_2, \ldots, s_n$. Select each element (independently) with probability p and let m be the number of selected elements (in average $m = pn$). Take one of the $m!$ permutations of m elements (uniformly at random) and let it act on the selected elements. E.g., for $p = 1/2$ and $n = 7$, one might select $m = 3$ elements (namely, s_2, s_4, and s_7) out of an input sequence $(s_1, \overline{s_2}, s_3, \overline{s_4}, s_5, s_6, \overline{s_7})$. Applying the permutation (312) to the selected elements yields $(s_1, \overline{s_7}, s_3, \overline{s_2}, s_5, s_6, \overline{s_4})$. The probability to obtain this sequence in this way is $p^3(1-p)^4/3!$. We will analyze quicksort (Section 2) and maxima finding (Section 3) under the partial permutations model.

Partial Bit Randomization: Our second model applies to problems involving natural numbers. It is parameterized by an integer k ($k \geq 0$). For each integer, the last k bits are randomly modified. This model is a discrete analogue of the model considered by Spielman and Teng. However, in our model the expectation of the resulting distribution is not necessarily equal to the unperturbed value. We analyze the running time of a shortest path algorithm under partial bit randomization (Section 4).

2 Quicksort

We analyze quicksort under partial permutations. We assume that quicksort takes the first element of the list as the pivot and splits the input list with respect to the pivot into two

[1] By suitable scaling we may assume that all numbers are in $[-1, +1]$.

parts: the elements smaller than the pivot and the elements larger than the pivot. We assume that the order of elements in the resulting two sublist is unchanged.

Theorem 1 (Quicksort under Limited Randomness) *The expected running time (i.e., number of comparisons) of quicksort on a partial permutation of n elements is $O((n/p)\ln n)$.*

Proof: We utilize a proof, based on randomized incremental constructions [5], for the fully randomized version of quicksort. We will only count the number of comparisons C. Assume that we have a permutation of the numbers 1 to n. Let X_{ij} be the indicator variable which is 1 iff i and j are compared in a run of quicksort with i being the pivot. Clearly $C = \sum_{i,j} X_{ij}$.

Fact 1 $X_{ij} = 1$ *iff i occurs first among the elements with value between i and j.*

Thus for a random permutation $\text{prob}(X_{ij} = 1) = 1/(j - i + 1)$ and hence the expected number of comparisons is

$$\sum_{i \neq j} \frac{1}{j - i + 1} \leq 2n \sum_{2 \leq k \leq n} \frac{1}{k} \leq 2n \ln n \ .$$

Next we estimate $\text{prob}(X_{ij} = 1)$ for partial permutations. Let $s_1, \ldots, s_n$ be our initial permutation and let $L = (8/p)\ln n$. If i is among $s_1, \ldots, s_L$, or $|j - i| \leq L$, we estimate $\text{prob}(X_{ij} = 1)$ for a total contribution of $O(n/p \ln n)$.

Next assume that there are at least L elements preceding i in the initial permutation and that $|i - j| > L$. We split our estimate for $\text{prob}(X_{ij} = 1)$ into two parts. For the first part, we assume that i is selected and for the second part, we assume that i is not selected.

So assume first that i is selected and let $l = |i - j|$. The probability that at most $lp/2$ elements between i (exclusive) and j (inclusive) are selected is less than $\exp(-lp/8)$. If more than $lp/2$ elements are selected, $X_{ij} = 1$ implies that i is first in the permutation of the selected elements and hence $\text{prob}(X_{ij} = 1) \leq 8/(lp)$. Together we obtain

$$\text{prob}(X_{ij} = 1) \leq \exp(-lp/8) + 8/(lp)$$

and hence

$$\sum_{1 \leq i \leq n} \sum_{l \geq L} \exp(-lp/8) + 8/(lp) = O(n/p \ln n) \ .$$

Assume next that i is not selected and let i be the k_i-th element in the initial sequence. The probability that less than $pk_i/2$ elements before i are chosen or less than $p|j - i|/2$ elements between i and j or more than $2pn$ elements are chosen altogether is less than

$$\exp(-pk_i/8) + \exp(-p|j - i|/8) + \exp(-pn/2).$$

We need to sum over i and j and obtain:

$$\begin{aligned}
\sum_{k_i \geq L} \sum_{j} \exp(-pk_i/8) &= \sum_{l \geq L} \sum_{j} \exp(-pl/8) \\
&= n \sum_{l \geq 0} \exp(-pL/8)\exp(-p/8)^l \\
&= \sum_{l \geq 0} \exp(-p/8)^l \\
&\leq \frac{1}{1 - \exp(-p/8)} \\
&= O(1/p) \ .
\end{aligned}$$

and, by the same argument:

$$\sum_{i\geq L}\sum_{l\geq L}\exp(-lp/8) = O(1/p)$$

and, since $n \geq k_i$ for all i:

$$\sum_{i}\sum_{j}\exp(-pn/2) = O(1/p).$$

So assume that the required number of elements are chosen. If i is before $i+1$ to j in the partial permutation, it must be the case that none of the $pl/2$ selected ($l = |j-i|$) elements between i and j is inserted before i. The probability for this is less than

$$\left(\frac{2pn - ip/2}{2pn}\right)^{lp/2} \leq \exp(-ilp/(4n))$$

Next observe that

$$\begin{aligned}\sum_{1\leq i\leq n}\sum_{1\leq l\leq n}\exp(-ilp/(4n)) &\leq \sum_{1\leq i\leq n}\frac{1}{1-\exp(-ip/(4n))}\\ &\leq \sum_{1\leq i\leq n}\frac{8n}{ip}\\ &\leq \frac{8n\ln n}{p}\end{aligned}$$

since $1-e^{-x} \geq x/2$ for $0 \leq x \leq 1$ and hence $1/(1-e^{-x}) \leq 2/x$. ∎

Remark: When we consider the perturbation of the classical worst-case; we are able to get closed form formulae for the X_{ij}'s (one has to distinguish 10 subcases, most of them involving 7 nested sums). From these sums (involving binomials), it is possible to get the differential equation satisfied by their generating functions, and then the Frobenius method allows to get the full asymptotic scale, which gives a $\frac{2}{p}n\ln n$ complexity. A generating function approach can also be used for the next section. More details are given in the Appendix.

Pitfalls: The expected running time of quicksort on random permutations can be analyzed in many different ways. Many of them rely on the fact that the subproblems generated by recursive calls are again random permutations. This is *not true* for partial permutations[2] as the following example demonstrates.

Consider an input $1, 2, 3, 4$ and define $q := 1-p$. Assume that 2 is the pivot element and hence the second subproblem consists of the numbers $3, 4$. If 2 is the pivot (first element after permutation), at least the numbers 1 and 2 are selected. Conditioned on the fact that 1 and 2 are selected and 2 is made the first element we obtain subproblem $(3,4)$ with probability $\text{prob}((3,4)) = q^2+3/2pq+p^2/2$ and subproblem $(4,3)$ with probability $\text{prob}((4,3)) = 1/2pq+p^2/2$. Applying partial permutations on input sequence $3,4$ gives $\text{prob}((3,4)) = q^2+2pq+p^2/2$ and $\text{prob}((4,3)) = p^2/2$.

[2]In the first version of this paper, we fell into this pitfall.

We also point out that the content of the first position, even if it is selected, is *not* a random element of the sequence. It is more likely to be the original element than any other element. The other elements are equally likely. This unbalance results from the fact that if only one element is selected, the permutation of the selected elements has very little freedom.

The expected maximum recursion depth of quicksort on random permutations is $O(\ln n)$. For partial permutations the expected maximum recursion depth is $\Omega(\sqrt{n/p})$. We will show in the next section that the number of left-to-right-maxima in a partial permutation might be as large as $\Omega(\sqrt{n/p})$. The number of left-to-right-maxima is the number of times the element n is compared to a pivot element. Thus some elements may take part in as many as $\Omega(\sqrt{n/p})$ recursive calls. Thus it is not true that every element takes part in $O((1/p)\ln n)$ calls with high probability.

The asymptotics expansion that we got for quicksort shows that this algorithm is already quite efficient for $p \gg \frac{1}{n \ln n}$; this gives a threshold after which the divide and conquer strategy of quicksort "wins" (ceases to have a quadratic complexity), even if the inputs is (in one sense) already almost sorted. We also showed that the perturbation of the worst case for quicksort is eventually the worst case among all the perturbations: quicksort has a dominant pic, with a rather sharp transition (cf Figure 1). We will see in the next section that is not always the case: another simple combinatorial algorithms, like finding a maximum in a list, can reveal some surprises!

3 Left-to-Right Maxima

The simplest strategy to determine the largest element in a sequence is to scan the sequence from left to right and to keep track of the largest element seen. The number of changes to the current maximum is called the number of *left-to-right maxima* in the sequence. The sequence $1, \ldots, n$ has n left to right maxima and the expected number of left to right maxima in a random permutation of n elements is $H_n = 1 + 1/2 + \cdots + 1/n$.

It is somehow surprising that the perturbation of the above mentioned worst case is not the worst case among all perturbations (when we switch from the classical uniform distribution model to the partial permutation model):

Theorem 2 (Left-to-Right Maxima under Limited Randomness) *Under the partial permutation model, the smoothed number of left-to-right maxima is*

$$\Omega(\sqrt{n}) \text{ and } O(\sqrt{(n/p)\log n})$$

whereas the number of left-to-right maxima of the list $(1, \ldots, n)$ *is then*

$$\ln(pn) + \gamma + 2\,\frac{1-p}{p} + \left(\frac{1}{2} + \frac{2\,(1-p)}{p^2}\right)\frac{1}{n} + O(\frac{1}{n^2})\,,$$

where $\gamma \approx .5772$ *is Euler's constant.*

Proof: We first give the two first asymptotic terms for the perturbation of the classical worst-case (see the appendix for a generating function proof which gives the full asymptotics).

The sequence $1, \ldots, n$ has n left-to-right maxima. Smoothing decreases the number to about $\ln(pn) + 2/p$ as we show next. Let X_i be the probability that the i-th position is not selected and is a maximum and let Y_i be the probability that the i-th position is selected and is a maximum.

Consider first a selected position i. A selected position contains a maximum iff it is a maximum among the selected elements. Assume that it is a maximum among the selected elements. Then its value is at least i and hence it is also a maximum when the elements not selected are considered. Thus $\sum_i Y_i$ is simply the number of maxima among the selected elements. The number of selected elements concentrates around pn and hence

$$\mathrm{E}[\sum_i Y_i] \approx \log(pn) \ .$$

Assume next that i is not selected. We start with the observation that X_i and X_{n+1-i} have the same distribution. Consider $i < n/2$. Position i stays a maximum if non of the preceding $i-1$ elements move to a position larger than i. Analogously, position $n+1-i$ stays a maximum if non of the succeeding $i-1$ elements move to a position smaller than $i+1-i$. We therefore concentrate on $i \le n/2$.

If k_1 elements among the first $i-1$ and k_2 elements among the last $n-i$ are selected, the probability that i stays a maximum is

$$f(k_1, k_2) = \frac{k_1! \cdot k_2!}{(k_1+k_2)!} \ .$$

The expression for $f(k_1, k_2)$ is decreasing in both arguments. Namely,

$$\frac{f(k_1, k_2+1)}{f(k_1, k_2)} = \frac{k_1! \cdot (k_2+1)! \cdot (k_1+k_2)!}{(k_1+k_2+1)! \cdot k_1! \cdot k_2!} = \frac{k_2+1}{k_1+k_2+1} \le 1 \ .$$

We want to compute $\mathrm{E}[\sum_{i \le n/2} X_i]$. We split the sum into two parts: $i \le (16/p)\log n$ and $i \ge (16/p)\log n$.

For the second part, $i \ge (16/p)\log n$, we expect to select about $pi \ge 16 \log n$ elements less than i and about $p(n-i) \ge pn/2$ elements larger than i. The probability that we select less than half the stated number in either part is less than $\exp(-(16/8)\log n) = O(n^{-2})$ by Chernoff bounds. If at least $8 \log n$ elements smaller i are selected and at least $pn/4$ elements larger i are selected the probability that i is a maximum is less than

$$f(8 \log n, pn/4) = O(n^{-2}).$$

Thus $\mathrm{prob}(X_i = 1) = O(n^{-2})$.

We turn to the i's with $i \le (16/p)\log n$. If none of the first $i-1$ elements is selected i stays a maximum. If at least one for the first $i-1$ elements is chosen, the probability that i stays a maximum is at most $e^{-pn/16} + 4/pn$. The first term accounts for the fact that less $pn/4$ elements larger i are selected and the second term accounts for the fact that at least $pn/4$ elements larger i are selected and none of them is moved to a position before i. Thus

$$\mathrm{prob}(X_i = 1) \le (1-p)\big((1-p)^{i-1} + e^{-pn/16} + 4/pn\big)$$

and hence

$$\mathrm{E}[\sum_{i \le (16/p)\log n} X_i] \le \frac{1-p}{p} + (1-p)\frac{16 \log n}{p}(e^{-pn/16} + 4/pn) = \frac{1-p}{p}(1+o(1)) \ .$$

We conclude

$$\mathrm{E}[\sum_i (X_i + Y_i)] \le \log(pn) + \frac{2(1-p)}{p} + o(1)$$

for constant p. In fact, constant p is not required. The argument works as long as $\log n/(p^2 n) = o(1)$, i.e., for $p \gg \sqrt{\log n/n}$.

We now come to the first affirmation of the theorem: the complexity of the worst case among all perturbations. We show that, for $p < 1/2$, the smoothed number of left-to-right maxima in a permutation of n elements may be $\Omega(\sqrt{n/p})$. Consider the sequence

$$n-k, n-k+1, \ldots, n, 1, 2, \ldots, n-k-1 \qquad \text{(where } k = \sqrt{n/p}\text{)}\,.$$

Let $a \approx pk$ and $b \approx p(n-k)$ be the number of selected elements in the first and second part of the sequence respectively; the first part consists of the first k elements. For large n, the probability that $a > 2pk$ or $b < pn/2$ is exponentially small by Chernoff bounds. So assume $a \le 2pk$ and $b \ge pn/2$. The probability that all elements selected in the first part are put into the second part by the random permutation of the selected elements is at least

$$q := \frac{b \cdot (b-1) \cdots (b-a+1)}{(a+b) \cdot (a+b-1) \cdots (b+1)}$$

since the number of choices for the first element is only b out of $a+b$, the number of choices for the second elements is only $b-1$ out of $a+b-1$, and so on. We have

$$q \ge \left(\frac{b-a}{a+b}\right)^a = \left(1 - \frac{2a}{a+b}\right)^a = \exp\left(a \ln\left(1 - \frac{2a}{a+b}\right)\right) \ge \exp\left(-\frac{4a^2}{a+b}\right)\,.$$

since $\ln(1-x) \ge -2x$ for $0 \le x \le 3/4$. Using the bounds $a \le 2pk$ and $b \ge pn/2$ we get

$$q \ge \exp\left(-\frac{4a^2}{a+b}\right) \ge \exp\left(-\frac{4(2p)^2 n/p}{pn/2}\right) \ge e^{-32}\,.$$

We conclude that with constant probability the number of left-to-right maxima in the perturbed sequence is at least $k - a \ge k(1-2p) = \Omega(\sqrt{n/p})$ for $p < 1/2$.

We next show an almost matching upper bound. Let $s_1, \ldots, s_n$ be an arbitrary permutation of the numbers 1 to n, let $k = \sqrt{8(n/p)\log n}$, and let I be the set of indices such that $i \ge k$ and $s_i \le n-k$. Basically, I ignores the first k and the largest k elements of the permutation. We estimate how many s_i with $i \in I$ are left-to-right maxima in the perturbed sequence. Then the total number of maxima is at most $2k$ larger.

Consider a fixed s_i with $i \in I$. If s_i is selected and is a maximum in the partial permutation, it must be a maximum among the selected elements. The expected number of left right maxima among the selected elements is $\ln pn$.

So assume that s_i is not selected. With high probability there are at least $kp/2$ elements preceding s_i among the selected elements, there are at least $kp/2$ elements larger than s_i among the selected elements, and there are at most $2np$ selected elements. Therefore the probability that s_i is a maximum in the perturbed sequence is bounded by

$$\left(\frac{2np - kp/2}{2np}\right)^{kp/2} \le \left(1 - \frac{k}{4n}\right)^{kp/2} \le \exp(-k^2 p/(8n)) = \frac{1}{n}$$

and hence the expected number of left-to-right maxima in the perturbed sequence is

$$O(\sqrt{(n/p)\log n})\,. \qquad \blacksquare$$

4 Single Source Shortest Path Problems

We consider the single source shortest path problem with nonnegative integer edge weights. As usual, let n and m denote the number of nodes and edges respectively. We assume our edge weights to be in $[0, 2^K - 1]$, i.e., edge weights are K bit integers. Meyer [16] has shown that the average complexity of the problem is linear $O(n+m)$. He assumes edge weights to be random K bit integers and that a certain set of primitive operations on such integers can be performed in constant time (addition, finding the first bit where two integers differ, ...). The algorithm can be used for arbitrary graphs. An alternative algorithm was later given by Goldberg [10] and his work is the starting point for this section. The worst case complexity of his algorithm is $O(m + nK)$. Algorithms with better worst case behavior are known [1, 3, 18, 12].

Theorem 3 (Shortest Paths under Limited Randomness) *Let G be an arbitrary graph, let $c : E \mapsto [0, \ldots, 2^K - 1]$ be an arbitrary cost function, and let k be such that $0 \le k \le K$. Let $\overline{c}$ be obtained from c by making the last k bits of each edge cost random. Then the single source shortest path problem can be solved in expected time $O(m + n(K - k))$.*

With full randomness the expected running time is $O(m + n)$, with no randomness the running time is $O(m + nK)$. Limited randomness interpolates linearly between the extremes.
Proof:

For a node v, let $min\_in\_cost(v)$ be the minimum cost of an incoming edge. Goldberg has shown that the running time of his algorithm is

$$O(n + m + \sum_v (K - \log min\_in\_cost(v) + 1),$$

where $min\_in\_cost(v)$ denotes the minimal cost of an (directed) edge with target node v. Next observe that $min\_in\_cost(v)$ is the minimum of $indeg(v)$ numbers of which the last k bits are random; here $indeg(v)$ is the indegree of v. For an edge e, let $r(e)$ be the number of leading zeroes in the random part of e. Then $\mathrm{E}[r(e)] = 2$ and

$$\begin{aligned} K - \log min\_in\_cost(v) &\le K - k + \max\{r(e)\,;\, e \in inedges(v)\} \\ &\le K - k + \sum\{r(e)\,;\, e \in inedges(v)\} \end{aligned}$$

Thus

$$\mathrm{E}[K - \log min\_in\_cost(v)] \le K - k + O(indeg(v))$$

and the time bound follows.

In our model of limited randomness, the last k bits of each weight are set randomly. Alternatively, one might select bits with probability p and set selected bits to random values. With this definition, the smoothed complexity becomes $O(m/p)$. For an edge e, let $r(e)$ be the number of leading zeroes in the weight of e. Then $\mathrm{E}[r(e)] \le 2/p$ and

$$\begin{aligned} K - \log min\_in\_cost(v) &\le \max\{r(e)\,;\, e \in inedges(v)\} \\ &\le \sum\{r(e)\,;\, e \in inedges(v)\} \end{aligned}$$

Thus

$$\mathrm{E}[K - \log min\_in\_cost(v)] \le O(indeg(v)/p)$$

and the time bound follows. ∎

5 Conclusion

We analyzed the smoothed complexity of three combinatorial problems. Smoothed complexity gives additional information about the distribution of hard instances in instance space. We believe, that the analysis of further discrete problems is a worthwhile task.

From a technical viewpoint, asymptotic expansions for higher moments (e.g., the variance) and the limit laws seems to be harder to get for yet, even with the help of computer algebra systems (some stochastic inequalities could be a way to tackle the distribution problem). It involves a kind of functional equations (see our Quicksort analysis), sparsely encountered in analysis of algorithms until now, which seems however to have some deep combinatorial properties. It can be expected that most of these functional equations will in fact behave like differential equations of the Cauchy–Euler type, which are better understood (see the rather complete article [4]) but for which to get explicit formulae (or numerical approximations) for the constants hidden in the big-oh term (and in further asymptotic terms) is sometimes a real challenge.

From a more theoretical viewpoint, it is natural to raise the question "Is there any relevant notion of smoothed complexity completeness?", quite similarly to the $DistNP$ complete class defined by Levin [15, 11]. While classical worst-case complete problems are rather well-known, this average-complete problems were introduced quite recently. A smoothed complexity completeness would give a valuable criterion for problems on which one could use cryptographic schemes on a rather wide region of instances. For example, the paper [21] shows that random knapsack has an almost linear time complexity. Its smoothed analysis approach confirms something which is sometimes forgotten: the fact that NP-completeness is not the most relevant notion for cryptography (even if this sometimes implies an exponential average case complexity, like for the permanent); many exponential worst-case algorithms have in fact a polynomial complexity, independently of $P \neq NP$! This also confirms the conjecture that worst-case NP problems are thought to have an average case polynomial complexity, as soon as the initial distribution of inputs is "reasonable" (computable in "polynomial time").

From a practical viewpoint, the thresholds obtained with this smoothed analysis approach (binding the perturbative parameter with n, and getting the ratio for which there is a jump of complexity, if any, in the algorithms) could also suggest to use meta-algorithms which test (if this test is cheap) if one is in a "hot" area of complexity (with respect to a first algorithm) and then to switch to another algorithm which is known to have a better behavior for this area.

In conclusion, we really believe that smoothed complexity is a key idea which suggests us to revisit all the classical algorithms, using some new tricks or some nice mathematics for their analyses, while allowing us to get a better understanding of the complexity landscapes of these algorithms.

Acknowledgements. This work was partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT). During the redaction of this article, Cyril Banderier was supported by the INRIA postdoctoral program and by the Max-Planck-Institut.

References

[1] Ravindra K. Ahuja, Kurt Mehlhorn, James B. Orlin, and Robert E. Tarjan. Faster algorithms for the shortest path problem. *J. Assoc. Comput. Mach.*, 37(2):213–223, 1990.

[2] A. Blum and J.D. Dunagan. Smoothed analysis of the perceptron algorithm for linear programming. In *SODA'02*, 2002.

[3] Boris V. Cherkassky, Andrew V. Goldberg, and Craig Silverstein. Buckets, heaps, lists, and monotone priority queues. *SIAM J. Comput.*, 28(4):1326–1346 (electronic), 1999.

[4] Hua-Huai Chern, Hsien-Kuei Hwang, and Tsung-Hsi Tsai. An asymptotic theory for Cauchy–Euler differential equations with applications to the analysis of algorithms. *Journal of Algorithms*, To appear, 2002. Analysis of algorithms (Krynica Morska, 2000).

[5] Kenneth L. Clarkson, Kurt Mehlhorn, and Raimund Seidel. Four results on randomized incremental constructions. *Comput. Geom.*, 3(4):185–212, 1993.

[6] Eric V. Denardo and Bennett L. Fox. Shortest-route methods. I. Reaching, pruning, and buckets. *Oper. Res.*, 27(1):161–186, 1979.

[7] J.D. Dunagan, D.A. Spielman, and S-H. Teng. Smoothed analysis of the renegar's condition number for linear programming. In *SIAM Conference on Optimization*, 2002.

[8] Philippe Flajolet, Gaston Gonnet, Claude Puech, and J. M. Robson. Analytic variations on quadtrees. *Algorithmica*, 10(6):473–500, 1993.

[9] Philippe Flajolet and Robert Sedgewick. *Symbolic and Analytic Combinatorics.* Book in preparation: http://algo.inria.fr/flajolet/Publications/books.html.

[10] Andrew V. Goldberg. A simple shortest path algorithm with linear average time. In *Proceedings of the 9th European Symposium on Algorithms (ESA '01)*, pages 230–241. Springer Lecture Notes in Computer Science LNCS 2161, 2001.

[11] Yuri Gurevich. Average case completeness. *Journal of Computer and System Sciences*, 42(3):346–398, 1991. Twenty-Eighth IEEE Symposium on Foundations of Computer Science (Los Angeles, CA, 1987).

[12] Torben Hagerup. Improved shortest paths on the word RAM. In *Automata, languages and programming (Geneva, 2000)*, pages 61–72. Springer, Berlin, 2000.

[13] Peter Henrici. *Applied and computational complex analysis. Vol. 2.* John Wiley & Sons Inc., New York, 1991. Reprint of the 1977 original.

[14] Donald E. Knuth. *The Art of Computer Programming.* Addison-Wesley Publishing Co., third edition of volume I & II, second edition of volume III edition, 1997–198.

[15] Leonid A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986.

[16] Ulrich Meyer. Shortest-Paths on arbitrary directed graphs in linear Average-Case time. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-01)*, pages 797–806. ACM Press, 2001.

[17] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, Cambridge, 1995.

[18] Raman. Recent results on the single-source shortest paths problem. In *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, volume 28, pages 61–72. Springer, Berlin, 1997.

[19] Bruno Salvy and Paul Zimmermann. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software*, 20(2):163–177, 1994.

[20] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. In *proceedings of the The Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 296–3051, 2001.

[21] Berthold Voecking and Rene Beier. Random knapsack in almost linear time. *In preparation*, 2003.

6 Appendix

Due to the limitation on the number of pages, we give in this appendix a proof of our assertion (in Theorem 2) about the perturbation of the classical worst case for left-to-right maxima.

In combinatorics, left-to-right maxima are often called "records"; a bijection due to Foata links them to the number of cycles in the permutation (for which a generating function approach is easy, leading to $(1-z)^{-u}$ and thus to the Stirling numbers). This suggests that generating functions could also be useful for our smoothed analysis. This is indeed the case (we refer to [9] for an extensive presentation of generating functions and use of complex analysis for the analysis of algorithms/combinatorial structures):

Theorem 4 *The average number of left-to-right maxima in a partial permutation of* $(1,\dots,n)$ *is*

$$\mathrm{LR}(n) = \ln(pn) + \gamma + 2\,\frac{1-p}{p} + \left(\frac{1}{2} + \frac{2\,(1-p)}{p^2}\right)\frac{1}{n} + O(\frac{1}{n^2})\,,$$

where $\gamma \approx .5772$ *is Euler's constant.*

What should we expect? At both ends of the partial permutation, pieces of length $1/p$ (in average) are not selected and in the remaining part, one should see about $\ln(pn)$ left-to-right maxima from the selected elements. Here is a rigorous proof of this:

Proof: [Proof via generating functions.] Note $\mathrm{LR}_{\mathrm{s}}(n)$ (resp. $\mathrm{LR}_{\mathrm{ns}}(n)$) the average number of left-to-right maxima arising from selected (resp. nonselected) elements. One now studies the asymptotic contribution of these two quantities to the sum $\mathrm{LR}(n) = \mathrm{LR}_{\mathrm{s}}(n) + \mathrm{LR}_{\mathrm{ns}}(n)$.

Contribution of selected left-to-right maxima:

$$\mathrm{LR}_{\mathrm{s}}(n) = E[\sum_{i=1}^{n} Y_i] = \sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k} H_k\,,$$

(with $H_0 = 0$). This formula is obtained by looking at all the configurations with k selected elements, each left-to-right maximum there is a left-to-right for the whole sequence, and each selected left-to-right maximum in the whole sequence is a left-to-right maximum in the subsequence made up of the selected elements, thus in average there is H_k such left-to-right maxima. Set $\alpha = \frac{p}{1-p}$ and $q := 1-p$, thus

$$\mathrm{LR}_{\mathrm{s}}(n) = q^n \sum_{k=0}^{n} \binom{n}{k} \alpha^k H_k\,. \tag{1}$$

In this sum, one recognizes an Euler transform: the transformation of a sequence f_n into a sequence $\sum_{k=0}^{n}\binom{n}{k} f_k$. It is easy to check that if $F(z)$ is the generating function associated to f_n, then $\frac{F(\frac{z}{1-z})}{1-z}$ is the generating function associated to its Euler transform. In our case, the generating function of $\alpha^k H_k$ is $\frac{\ln\frac{1}{1-\alpha z}}{1-\alpha z}$, and its Euler transform is

$$\sum_{n\geq 0}\left(\sum_{k=0}^{n}\binom{n}{k}\alpha^k\right) H_k\, z^n = \frac{\ln\frac{1}{1-\alpha\frac{z}{1-z}}}{1-\alpha\frac{z}{1-z}}\,\frac{1}{1-z} = \frac{\ln\frac{1-z}{1-(\alpha+1)z}}{1-(\alpha+1)z}\,. \tag{2}$$

Taking into account the multiplication by q^n in (1), one performs the substitution $z \mapsto qz$ in (2) and one gets

$$\sum_{n\geq 1} \mathrm{LR_s}(n) z^n = \frac{\ln(\frac{1-qz}{1-z})}{1-z}\,. \tag{3}$$

The radius of convergence is $1 < 1/q$ and analysis of singularity gives $\mathrm{LR_s}(n) = H_n + \ln(p) + o(C^n)$ (for $C < 1$), then, using $H_n = \ln(n) + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + O(\frac{1}{n^3})$, one has

$$\mathrm{LR_s}(n) = \ln(pn) + \gamma + \frac{1}{2n} + O(\frac{1}{n^2})\,.$$

Contribution of nonselected left-to-right maxima: The probability to have a non selected maximum at position i is given by

$$\mathrm{prob}(X_i = 1) = \sum_{k_1=0}^{i-1} \sum_{k_2=0}^{n-i} \binom{i-1}{k_1} \binom{n-i}{k_2} p^{k_1+k_2} q^{n-k_1-k_2} \frac{k_1!k_2!}{(k_1+k_2)!}\,,$$

where $q := 1-p$. This formula is obtained by considering the k_1 selected elements before i (as i is non selected, the value i is at position i and all the k_1 selected elements are $\leq i-1$ which is a left-to-right maximum) and k_2 selected elements after i ($\frac{1}{(k_1+k_2)!}$ is the "weight" of the chosen permutation). A nice (conjectural) observation is that $\mathrm{prob}(X_i \neq 1)$ is unimodal (for $i = 1, \ldots, n$) but we don't need this fact. One wants to find the asymptotics of

$$\begin{aligned} \mathrm{LR_{ns}}(n) &= \sum_{i=1}^{n} \mathrm{prob}(X_i = 1) \\ &= \sum_{i=1}^{n} \sum_{k_1=0}^{i-1} \sum_{k_2=0}^{n-i} \binom{i-1}{k_1} \binom{n-i}{k_2} p^{k_1+k_2} q^{n-k_1-k_2} \frac{k_1!k_2!}{(k_1+k_2)!}\,. \end{aligned}$$

In order to get the generating function associated to this triple sum, one can use some basic transformations, like the (inverse) Borel transform (multiplication or division of the sequence by $n!$), the Euler transform, the Hadamard product... All these transformations are closed in the space of D-finite functions (functions which satisfy a linear differential equations with polynomial coefficients), and are easily performed with a computer algebra package such as Gfun [19]. For example, the generating function (3) is given by $\mathcal{B}^{-1}\left(\exp(z)\ \mathcal{B}\left(\frac{\ln(\frac{1}{1-\alpha z})}{1-\alpha z}\right)\right)$ where $\mathcal{B}$ stands for the Borel transform and $\mathcal{B}^{-1}$ for the inverse Borel transform. Similar manipulations for the $LR_{ns}(n)$'s lead to the generating function

$$\sum_{n\geq 1} \mathrm{LR_{ns}}(n) z^n = \frac{-q\ln(1-z) + q\ln(1-qz)}{2(1-z(1-\frac{p}{2}))^2} + \frac{q}{(1-z)(1-z(1-\frac{p}{2}))}\,.$$

As $1 < 1/(1-p/2) < 1/q$, the radius of convergence is 1 for the first log and the last summand, and $\frac{1}{1-p/2}$ for the second log, so singularity analysis gives (with $r := \frac{1-p}{2}$):

$$\mathrm{LR_{ns}}(n) = r\big(\frac{4}{p^2 n} + O(\frac{1}{n^2})\big) + r\big(n\ln(\frac{2}{p} - 1) r^n + O(r^n)\big) + \frac{2q}{p} + \frac{q}{r}(\frac{p}{p-2})^n$$

Summing the contribution of $\mathrm{LR_{ns}}(n)$ and $\mathrm{LR_s}(n)$ gives the theorem. ∎

This "shows" that the contribution of the subsequent summands begins to be as important as the log part when $p \sim \frac{1}{\sqrt{n \ln n}}$.