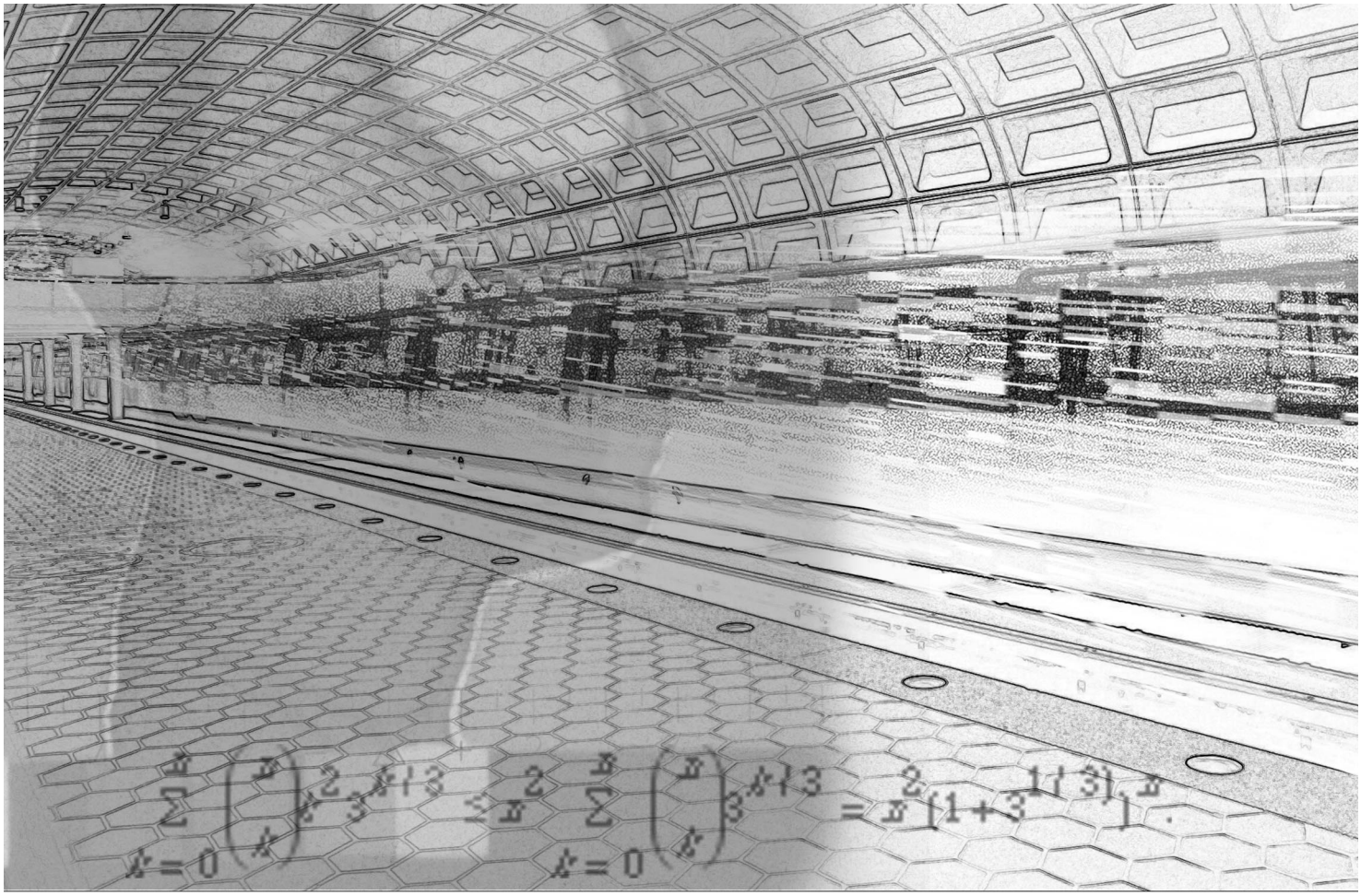
Nº68

OPTIMA

Mathematical Programming Society Newsletter

OCTOBER 2002



68

Exact algorithms for NP-hard problems

Gerhard J. Woeginger*

We discuss fast exponential time solutions for NP-complete problems. We survey known results and approaches, we provide pointers to the literature, and we discuss several open problems in this area.

Introduction

Every NP-complete problem can be solved by exhaustive search. Unfortunately, when the size of the instances grows the running time for exhaustive search soon becomes forbiddingly large, even for instances of fairly small size. For some problems it is possible to design algorithms that are significantly faster than exhaustive search, though still not polynomial time. This survey deals with such fast, super-polynomial time algorithms that solve NP-complete problems to optimality. In recent years there has been growing interest in the design and analysis of such super-polynomial time algorithms. This interest has many causes.

- It is now commonly believed that P↑NP, and that super-polynomial time algorithms are the best we can hope for when we are dealing with an NP-complete problem. There is a handful of isolated results scattered across the literature, but we are far from developing a general theory. In fact, we have not even started a systematic investigation of the worst case behavior of such super-polynomial time algorithms.

- Some NP-complete problems have better and faster exact algorithms than others. There is a wide variation in the worst case complexities of known exact (super-polynomial time) algorithms. Classical complexity theory can not explain these differences. Do there exist any relationships among the worst case behaviors of various problems? Is progress on the different problems connected? Can we somehow classify NP-complete problems to see how close we are to the best possible algorithms?

- With the increased speed of modern computers, large instances of NP-complete problems can be solved effectively. For example it is nowadays routine to solve traveling salesman (TSP) instances with up to 2000 cities. And if the data is nicely structured, then instances with up to 13000 cities can be handled in practice (Applegate, Bixby, Chvátal & Cook [1]). There is a huge gap between the empirical results from testing implementations and the known theoretical results on exact algorithms.

- Fast algorithms with exponential running times may actually lead to practical algorithms, at least for moderate instance sizes. For small instances, an algorithm with an exponential time complexity of $O(1.01^n)$ should usually run much faster than an algorithm with a polynomial time complexity of $O(n^4)$.

In this article we survey known results and approaches to the worst case analysis of exact algorithms for NP-hard problems, and we provide pointers to the literature.

How do we measure the quality of an exact algorithm for an NP-hard problem? Exact algorithms for NP-complete problems are sometimes hard to compare, since their analysis is done in terms of different parameters. For instance, for an optimization problem on graphs the analysis could be done in terms of the number n of vertices, or possibly in the number m of edges. Since the standard reductions between NP-complete problems may increase the instance sizes, many questions in computational complexity theory depend delicately on the choice of parameters. The right approach seems to be to include an explicit complexity parameter in the problem specification (Impagliazzo, Paturi & Zane [12]). Recall that the decision version of every problem in NP can be formulated in the following way:

> Given x, decide whether there exists y so that $|y| \leq m(x)$ and $R(x, y)$.

Here x is an instance of the problem; y is a short YES-certificate for this instance; $R(x, y)$ is a polynomial time decidable relation that verifies certificate y for instance x; and $m(x)$ is a polynomial time computable and polynomially

*g.j.woeginger@math.utwente.nl. Faculty of Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands.

bounded *complexity parameter* that bounds the length of the certificate y. A trivial exact algorithm for solving x would be to enumerate all possible strings with lengths up to $m(x)$, and to check whether any of them yields a YES-certificate. Up to polynomial factors that depend on the evaluation time of $R(x,y)$, this would yield a running time of $2^{m(x)}$. The first goal in exact algorithms always is to break the triviality barrier, and to improve on the time complexity of this trivial enumerative algorithm.

Throughout this survey, we will measure the running times of algorithms only with respect to the complexity parameter $m(x)$. We will use a modified big-Oh notation that suppresses all other (polynomially bounded) terms that depend on the instance x and the relation $R(x,y)$. We write $O^*(T(m(x)))$ for a time complexity of the form $O^*(T(m(x))\cdot\text{poly}(|x|))$. This modification may be justified by the exponential growth of $T(m(x))$. Note that for instance for simple graphs with $m(x)=n$ vertices and m edges, the running time $1.7344^n\cdot n^2m^5$ is sandwiched between the running times 1.7344^n and 1.7345^n. We stress, however, the fact that the complexity parameter $m(x)$ in general is not unique, and that it heavily depends on the representation of the input. For an input in the form of an undirected graph, for instance, the complexity parameter might be the number n of vertices or the number m of edges.

Time complexities and classes of optimization problems. Consider a problem in NP as defined above, with instances x and with complexity parameter $m(x)$. An algorithm for this problem has *sub-exponential* time complexity, if the running time depends polynomially on $|x|$ and if the logarithm of the running time depends sub-linearly on $m(x)$. For instance, a running time of $|x|^5\cdot 2^{\sqrt{m(x)}}$ would be sub-exponential. A problem in NP is contained in the complexity class SUBEXP (the class of SUB-EXPonentially solvable problems) if for every fixed $\varepsilon > 0$, it can be solved in poly $(|x|)\cdot 2^{\varepsilon\cdot m(x)}$ time.

Let us briefly discuss some basic classes of optimization problems that contain many classical problems: the class of subset problems, the class of permutation problems, and the class of partition problems. In a *subset problem*, every feasible solution can be specified as a subset of an underlying ground set. For instance, fixing a truth-assignment in the satisfiability problem corresponds to selecting a subset of TRUE variables. In the independent set problem, every subset of the vertex set is a solution candidate. In a *permutation problem*, every feasible solution can be specified as a total ordering of an underlying ground set. For instance, in the TSP every tour corresponds to a permutation of the cities. In single machine scheduling problems, feasible schedules are often specified as permutations of the jobs. In a *partition problem*, every feasible solution can be specified as a partition of an underlying ground set. For instance, a graph coloring is a partition of the vertex set into color classes. In parallel machine scheduling problems, feasible schedules are often specified by partitioning the job set and assigning every part to another machine.

As we observed above, all NP-complete problems possess trivial algorithms that simply enumerate and check all feasible solutions. For a ground set with cardinality n, subset problems can be trivially solved in $O^*(2^n)$ time, permutation problems can be trivially solved in $O^*(n!)$ time, and partition problems are trivial to solve in $O^*(c^{n\log n})$ time; here $c > 1$ denotes a constant that does not depend on the instance. These time complexities form the triviality barriers for the corresponding classes of optimization problems.

Organization of the survey. The four Sections 1-4 introduce and explain four central techniques for designing fast exact algorithms: Section 1 deals with dynamic programming across the subsets, Section 2 discusses pruning of search trees, Section 3 illustrates the power of preprocessing the data, and Section 4 considers approaches based on local search. Finally, Section 5 gives some concluding remarks.

Throughout this paper, *open problems* refer to unsolved research problems, while *exercises* pose smaller questions and puzzles that should be fairly easy to solve. A long version of this survey will appear in the Springer book *"Eureka, you shrink!"* edited by Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. The long version contains more examples, more exercises, and more problems, and it also discusses the interface to complexity theory.

1 Technique: Dynamic programming across the subsets

A standard approach for getting fast exact algorithms for NP-complete problems is to do dynamic programming across the subsets. For every 'interesting' subset of the ground set, there is a polynomial number of corresponding states in the state space of the dynamic program. In the cases where all these corresponding states can be computed in reasonable time, this approach usually yields a time complexity $O^*(2^n)$. We will illustrate these benefits of dynamic programming by discussing the traveling salesman problem. Sometimes, the number of 'interesting' subsets is fairly small, and then an even better time complexity might be possible. This will be illustrated by discussing the graph 3-colorability problem.

The traveling salesman problem (TSP). A traveling salesman has to visit the cities 1 to n. He starts in city 1, runs through the remaining $n-1$ cities in arbitrary order, and in the very end returns to his starting point in city 1. The distance from city i to city j is denoted by $d(i,j)$. The goal is to minimize the total travel length of the salesman. A trivial algorithm for the TSP checks all $O(n!)$ permutations.

We now sketch the exact TSP algorithm of Held & Karp [8] that is based on dynamic programming across the subsets. For every non-empty subset $S \subseteq \{2,\ldots,n\}$ and for every city $i \in S$ we denote by $\text{OPT}[S; i]$ the length of the shortest path that starts in city 1, then visits all cities in $S - \{i\}$ in arbitrary order, and finally stops in city i. Clearly, $\text{OPT}[\{i\};i] = d(1, i)$ and

$$\text{OPT}[S; i] = \min \{\text{OPT}[S - \{i\};j] + d(j, i) : j \in S - \{i\}\}.$$

By working through the subsets S in order of increasing cardinality, we can compute the value $\text{OPT}[S; i]$ in time proportional to $|S|$. The optimal travel length is given as the minimum value of $\text{OPT}[\{2,\ldots,n\};j]+d(j,1)$ over all j with $2 \neq j \neq n$. This yields an overall time complexity of $O(n^2 2^n)$ and hence $O^*(2^n)$.

This result was published in 1962, and from nowadays point of view almost looks trivial. Still, it yields the best time complexity that is known today.

Open problem 1.1 *Construct an exact algorithm for the traveling salesman problem with time complexity $O^*(c^n)$ for some $c < 2$. In fact, it even would be interesting to reach such a time complexity $O^*(c^n)$ with $c < 2$ for the closely related, but slightly simpler Hamiltonian cycle problem (given a graph G on n vertices, does it contain a spanning cycle).*

Hwang, Chang & Lee [11] describe a sub-exponential time $O(c^{\sqrt{n}\log n})$ exact algorithm with some constant $c > 1$ for the Euclidean TSP. The Euclidean TSP is a special case of the TSP where the cities are points in the Euclidean plane and where the distance between two cities is the Euclidean distance. The approach in [11] is heavily based on planar separator structures, and it cannot be carried over to the general TSP.

Graph coloring. Given a graph $G = (V, E)$ with n vertices, color the vertices with the smallest possible number of colors such that adjacent vertices never receive the same color. This smallest possible number is the *chromatic* number $\chi(G)$ of the graph. Every color class is a vertex set without induced edges; such a vertex set is called an *independent* set. An independent set is *maximal*, if none of its proper supersets is also independent. For any graph G, there exists a feasible coloring with $\chi(G)$ colors in which at least one color class is a maximal independent set. Moon & Moser [19] have shown that a graph with n vertices contains at most $3^{n/3} \oplus 1.4422^n$ maximal independent sets. By considering a collection of $n/3$ independent triangles, we see that this bound is best possible. Paull & Unger [24] designed a procedure that generates all maximal independent sets in a graph in $O(n^2)$ time per generated set.

Based on the ideas introduced by Lawler [16], we present a dynamic program across the subsets with a time complexity $O^*(2.4422^n)$. For a subset $S \subseteq V$ of the vertices, we denote by $G[S]$ the subgraph of G that is induced by the vertices in S, and we denote by $\text{OPT}[S]$ the chromatic number of $G[S]$. If S is empty, then clearly $\text{OPT}[S] = 0$. Moreover, for $S \uparrow \emptyset$ we have

$$\text{OPT}[S] = 1 + \min \{\text{OPT}[S - T] : T \text{ maximal indep. set in } G[S]\}.$$

We work through the sets S in order of increasing cardinality, such that when we are handling S, all its subsets have already been handled. Then the time needed to compute the value $\text{OPT}[S]$ is dominated by the time needed to generate all maximal independent subsets T of $G[S]$. By the above discussion, this can be done in $k^2 3^{k/3}$ time where k is the number of vertices in $G[S]$.This leads to an overall time complexity of

$$\sum_{k=0}^{n} \binom{n}{k} k^2 3^{k/3} \le n^2 \sum_{k=0}^{n} \binom{n}{k} 3^{k/3} = n^2 (1 + 3^{1/3})^n.$$

Since $1 + 3^{1/3} \oplus 2.4422$, this yields the claimed time complexity $O^*(2.4422^n)$. Very recently, Eppstein [6] managed to improve this time complexity to $O^*(2.4150^n)$ where $2.4150 \oplus 4/3 + 3^{4/3}/4$. His improvement is based on carefully counting the *small* maximal independent sets in a graph.

Finally, we turn to the (much easier) special case of deciding whether $\chi(G) = 3$. Lawler [16] gives a simple $O^*(1.4422^n)$ algorithm: Generate all maximal independent sets S, and check whether their complement graph $G[V - S]$ is bipartite. Schiermeyer [30] describes a rather complicated modification of this idea that improves the time complexity to $O^*(1.415^n)$. The first major progress is due to Beigel & Eppstein [3] who get a running time of $O^*(1.3446^n)$ by applying the technique of pruning the search tree; see Section 2 of this survey. The current champion algorithm has a time complexity of $O^*(1.3289^n)$ and is due to Eppstein [5]. This algorithm combines pruning of the search tree with several tricks based on network flows and matching.

Exercise 1.2 *(Nielsen [20])*
Find an $O^*(1.7851^n)$ exact algorithm that decides for a graph on n vertices whether $\chi(G)=4$. Hint: Generate all maximal independent sets of cardinality at least n/4 (why?), and use the algorithm from [5] to check their complement graphs.

Eppstein [5] also shows that for $n/4 \le k \le n/3$, a graph on n vertices contains at most $O(3^{4k-n}4^{n-3k})$ maximal independent sets. Apply this result to improve the time complexity for 4-coloring further to $O^*(1.7504^n)$.

2 Technique: Pruning the search tree

Every NP-complete problem can be solved by enumerating and checking all feasible solutions. An organized way for doing this is to (1) concentrate on some piece of the feasible solution, to (2) determine all the possible values this piece can take, and to (3) branch into several subcases according to these possible values. This naturally defines a search tree: Every branching in (3) corresponds to a branching of the search tree into subtrees. Sometimes, it can be argued that certain values for a certain piece can never lead to an optimal solution. In these cases we may simply ignore all these values, kill the corresponding subtrees, and speed-up the search procedure. Every Branch-and-Bound algorithm is based on this idea, and we can also get exact algorithms with good worst case behavior out of this idea. However, to get the worst case analysis through, we need a good mathematical understanding of the evolution of the search tree, and we need good estimates on the sizes of the killed subtrees and on the number and on the sizes of the surviving cases.

We will illustrate the technique of pruning the search tree by developing algorithms for the satisfiability problem, and for the independent set problem in graphs.

The satisfiability problem. Let $X = \{x_1, x_2, ..., x_n\}$ be a set of logical variables. A variable or a negated variable from X is called a *literal*. A *clause* over X is the disjunction of literals from X. A Boolean formula is in *conjunctive normal form (CNF)*, if it is the conjunction of clauses over X. A formula in CNF is in *k-CNF*, if all clauses contain at most k literals. A formula is *satisfiable*, if there is a truth assignment from X to $\{0,1\}$ which assigns to each variable a Boolean value (0=false, 1=true) such that the entire formula evaluates to true. The k-satisfiability problem is the problem of deciding whether a formula F in k-CNF is satisfiable. It is well-known that 2-satisfiability is polynomially solvable, whereas k-satisfiability with $k \ge 3$ is NP-complete. A trivial algorithm checks all possible truth assignments in $O^*(2^n)$ time.

We will now describe an exact $O^*(1.8393^n)$ algorithm for 3-satisfiability that is based on the technique of pruning the search tree. Let F be a Boolean formula in 3-CNF with m clauses ($m \le n^3$). The idea is to branch on one of the clauses c with three literals ℓ_1, ℓ_2, ℓ_3. Every satisfying truth assignment for F must fall into one of the following three classes:

(a) literal ℓ_1 is true;

(b) literal ℓ_1 is false, and literal ℓ_2 is true;

(c) literals ℓ_1 and ℓ_2 are false, and literal ℓ_3 is true.

We fix the values of the corresponding one, two, three variables appropriately, and we branch into three subtrees according to these cases (a), (b), and (c) with $n - 1$, $n - 2$, and $n - 3$ unfixed variables, respectively. By doing this, we cut away the subtree where the literals ℓ_1, ℓ_2, ℓ_3

all are false. The formulas in the three subtrees are handled recursively. The stopping criterion is when we reach a formula in 2-CNF, which can be resolved in polynomial time. Denote by $T(n)$ the worst case time that this algorithm needs on a 3-CNF formula with n variables. Then

$$T(n) \leq T(n\text{-}1) + T(n\text{-}2) + T(n\text{-}3) + O(n+m).$$

Here the terms $T(n-1)$, $T(n-2)$, and $T(n-3)$ measure the time for solving the subcase with $n-1$, $n-2$, and $n-3$ unfixed variables, respectively. Standard calculations yield that $T(n)$ is within a polynomial factor of α^n where α is the largest real root of $\alpha^3 = \alpha^2 + \alpha + 1$. Since $\alpha \approx 1.8393$, this gives a time complexity of $O^*(1.8393^n)$.

In a milestone paper in this area, Monien & Speckenmeyer [18] improve the branching step of the above approach. They either detect a clause that can be handled without any branching, or they detect a clause for which the branching only creates formulas that contain one clause with at most $k-1$ literals. A careful analysis yields a time complexity of $O^*(\beta^n)$ for k-satisfiability, where β is the largest real root of $\beta = 2 - 1/\beta^{k-1}$. For 3-satisfiability, this time complexity is $O^*(1.6181^n)$. Schiermeyer [29] refines these ideas for 3-satisfiability even further, and performs a quantitative analysis of the number of 2-clauses in the resulting subtrees. This yields a time complexity of $O^*(1.5783^n)$. Kullmann [14, 15] writes half a book on the analysis of this approach, and gets time complexities of $O^*(1.5045^n)$ and $O^*(1.4963^n)$ for 3-satisfiability. The current champion algorithms for satisfiability are, however, not based on pruning the search tree, but on local search ideas; see Section 4 of this survey.

Exercise 2.1 *For a formula F in CNF, consider the following bipartite graph G_F: For every logical variable in X, there is a corresponding variable-vertex in G_F, and for every clause in F, there is a corresponding clause-vertex in G_F. There is an edge from a variable-vertex to a clause-vertex if and only if the corresponding variable is contained (in negated or un-negated form) in the corresponding clause. The planar satisfiability problem is the special case of the satisfiability problem that contains all instances with formulas F in CNF for which the graph G_F is planar.*

Design a sub-exponential time exact algorithm for the planar 3-satisfiability problem! Hint: Use the planar separator theorem of Lipton & Tarjan [17] to break the formula F into two smaller, independent pieces. Running times of roughly $O^*(c^{\sqrt{n}})$ are possible.

The independent set problem. Given a graph $G=(V, E)$ with n vertices, the goal is to find an independent set of maximum cardinality. An *independent* set $S \subseteq V$ is a set of vertices that does not induce any edges. Moon& Moser [19] have shown that a graph contains at most $3^{n/3} \approx 1.4422^n$ maximal (with respect to inclusion) independent sets. Hence the first goal is to beat the time complexity $O^*(1.4422^n)$.

We describe an exact $O^*(1.3803^n)$ algorithm for independent set that is based on the technique of pruning the search tree. Let G be a graph with m edges. The idea is to branch on a high-degree vertex: If all vertices have degree at most two, then the graph is a collection of cycles and paths. It is straightforward to determine a maximum independent set in such a graph. Otherwise, G contains a vertex v of degree $d \geq 3$; let $v_1,\ldots, v_d$ be the neighbors of v in G. Every independent set I for G must fall into one of the following two classes:

(a) I does not contain v.
(b) I does contain v; then I cannot contain any neighbor of v.

We dive into two subtrees. The first subtree deals with the graph that results from removing vertex v from G. The second subtree deals with the graph that results from removing v together with $v_1, \ldots, v_d$ from G. We recursively compute the maximum independent set in both subtrees, and update it to a solution for the original graph G. Denote by $T(n)$ the worst case time that this algorithm needs on a graph with n vertices. Then

$$T(n) \leq T(n-1) + T(n-4) + O(n+m).$$

Standard calculations yield that $T(n)$ is within a polynomial factor of γ^n where $\gamma \approx 1.3803$ is the largest real root of $\gamma^4 = \gamma^3 + 1$. This yields the time complexity $O^*(1.3803^n)$.

The first published paper that deals with exact algorithms for maximum independent set is Tarjan & Trojanowski [34]. They give an algorithm with running time $O^*(1.2599^n)$. This algorithm follows essentially the above approach, but performs a smarter (and pretty tedious) structural case analysis of the neighborhood around the high-degree vertex v. The algorithm of Jian [13] has a time complexity of $O^*(1.2346^n)$. Robson [26] further refines the approach. A combinatorial argument about connected regular graphs helps to get the running time down to $O^*(1.2108^n)$. Robson's algorithm uses exponential space. Beigel [2] presents another algorithm with a weaker time complexity of $O^*(1.2227^n)$, but polynomial space complexity. Robson [27] is currently working on a new algorithm which is supposed to run in time $O^*(1.1844^n)$. This new algorithm is based on a detailed computer generated subcase analysis where the number of subcases is in the tens of thousands.

Open problem 2.2 *(a) Construct an exact algorithm for the maximum independent set problem with time complexity $O^*(c^n)$ for some $c < 1.1$. If this really is doable, it will be very tedious to do.*

(b) Prove a lower bound on the time complexity of any exact algorithm for maximum independent set that is based on the technique of pruning the search tree and that makes its branching decision by solely considering the subgraphs around a fixed chosen vertex.

Exercise 2.3 *Design a sub-exponential time exact algorithm for the restriction of the maximum independent set problem to planar graphs! Hint: Use the planar separator theorem of Lipton & Tarjan [17].*

3 Technique: Preprocessing the data

Preprocessing is an initial phase of computation, where one analyzes and restructures the given data, such that later on certain queries to the data can be answered quickly. By preprocessing an exponentially large data set or part of this data in an appropriate way, we may sometimes gain an exponentially large factor in the running time. In this section we will use the technique of preprocessing the data to get fast algorithms for the subset sum problem and for the binary knapsack problem. We start this section by discussing two very simple, polynomially solvable toy problems where preprocessing helps a lot.

In the first toy problem, we are given two integer sequences $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$ and an integer S. We want to decide whether there exist an x_i and a y_i that sum up to S. A trivial approach would be to check all possible pairs in

$O(k^2)$ overall time. A better approach is to first preprocess the data and to sort the x_i in $O(k \log k)$ time. After that, we may repeatedly use bisection search in this sorted array, and search for the k values $S - y_j$ in $O(\log k)$ time per value. The overall time complexity becomes $O(k \log k)$, and we save a factor of $k / \log k$. By applying the same preprocessing, we can also decide in $O(k \log k)$ time, whether the sequences $\langle x_i \rangle$and $\langle y_j \rangle$ are disjoint, or whether every value x_i also occurs in the sequence $\langle y_j \rangle$.

In the second toy problem, we are given k points (x_i, y_i) in two-dimensional space, together with the n numbers $z_1, \ldots, z_k$, and a number W. The goal is to determine for every z_j the largest value y_i, subject to the condition that $x_i + z_j \leq W$. The trivial solution needs $O(k^2)$ time, and by applying preprocessing this can be brought down to $O(k \log k)$: If there are two points (x_i, y_i) and (x_j, y_j) with $x_i \leq x_j$ and $y_i \geq y_j$, then the point (x_j, y_j) may be disregarded since it is always dominated by (x_i, y_i). The subset of non-dominated points can be computed in $O(k \log k)$ time by standard methods from computational geometry. We sort the non-dominated points by increasing x-coordinates and store this sequence in an array. This completes the preprocessing. To handle a value z_j, we simply search in $O(\log k)$ time through the sorted array for the largest value x_i less or equal to $W - z_j$.

In both toy problems preprocessing improved the time complexity from $O(k^2)$ to $O(k \log k)$. Of course, when dealing with exponential time algorithms an improvement by a factor of $k / \log k$ is not impressive at all. The right intuition is to think of k as roughly $2^{n/2}$. Then preprocessing the data yields a speedup from $k^2 = 2^n$ to $k \log k = n2^{n/2}$, and such a speedup of $2^{n/2}$ indeed *is* impressive!

The subset sum problem. In this problem, the input consists of positive integers $a_1, \ldots, a_n$ and S. The question is whether there exists a subset of the a_i that sums up to S. The subset sum problem belongs to the class of subset problems, and can be solved (trivially) in $O^*(2^n)$ time. By splitting the problem into two halves and by preprocessing the first half, the time complexity can be brought down to $O^*(\sqrt{2^n}) \oplus O^*(1.4145^n)$.

Let X denote the set of all integers of the form $\Sigma_{i \in I} a_i$ with $I \subseteq \{1, \ldots, \lfloor n/2 \rfloor\}$, and let Y denote the set of all integers of the form $\Sigma_{i \in I} a_i$ with $I \subseteq \{\lfloor n/2 \rfloor + 1, \ldots, n\}$. Note that $0 \in X$ and $0 \in Y$. It is straightforward to compute X and Y in $O^*(2^{n/2})$ time by complete enumeration. The subset sum instance has a solution if and only if there exists an $x_i \in X$ and a $y_j \in Y$ with $x_i + y_j = S$. But now we are back at our first toy problem that we discussed at the beginning of this section! By preprocessing X and by searching for all $S - y_j$ in the sorted structure, we can solve this problem in $O(n2^{n/2})$ time. This yields an overall time of $O^*(2^{n/2})$.

Exercise 3.1 (Van Vliet [35])
In the Three-Partition problem, the input consists of 3n positive integers $a_1, \ldots a_n$, $b_1, \ldots, b_n$, and $c_1, \ldots c_n$, together with an integer D. The question is to determine whether there exist three permutations $\neq$, ψ, ϕ, of $\{1, \ldots, n\}$ such that $a_{\pi(i)} + b_{\psi(i)} + c_{\phi(i)} = D$ holds for all $i = 1, \ldots, n$. By checking all possible triples $(\neq, \psi, \phi)$ of permutations, this problem can be solved trivially in $O^*(n!^3)$ time.

Use the technique of preprocessing the data to improve the time complexity to $O^*(n!)$.

The binary knapsack problem. Here the input consists of n items that are specified by a positive integer value a_i and a positive integer weight w_i ($i = 1, \ldots, n$), together with a bound W. The goal is to find a subset of the items with the maximum total value subject to the condition that the total weight does not exceed W. The binary knapsack problem is closely related to the subset sum problem, and it can be solved (trivially) in $O^*(2^n)$ time. In 1974, Horowitz & Sahni [10] used a preprocessing trick to improve the time complexity to $O^*(2^{n/2})$.

For every $I \subseteq \{1,\ldots, \lfloor n/2 \rfloor\}$ we create a compound item x_I with value $a_I = \Sigma_{i \in I} a_i$ and weight $w_I = \Sigma_{i \in I} w_i$, and we put this item into the set X. For every $J \subseteq \{\lfloor n/2 \rfloor + 1,\ldots, n\}$ we put a corresponding compound item y_J into the set Y. The sets X and Y can be determined in $O^*(2^{n/2})$ time. The solution of the knapsack instance now reduces to the following: Find a compound item x_I in X and a compound item y_J in Y, such that $w_I + w_J \leq W$ and such that $a_I + a_J$ becomes maximum. But this can be handled by preprocessing as in our second toy problem, and we end up with an overall time complexity and an overall space complexity of $O^*(2^{n/2})$.

In 1981, Schroeppel & Shamir [33] improved the *space* complexity of this approach to $O^*(2^{n/4})$, while leaving its time complexity unchanged. The main trick is to split the instance into four pieces with $n/4$ items each, instead of two pieces with $n/2$ items. Apart from this, there has been no progress on exact algorithms for the knapsack problem since 1974.

Open problem 3.2 *Construct an exact algorithm for the subset sum problem or the knapsack problem with time complexity $O^*(c^n)$ for some $c < \sqrt{2}$, or prove that no such algorithm can exist under some reasonable complexity assumptions.*

4 Technique: Local search

The idea of using local search methods in designing exact exponential time algorithms is relatively new. A *local search* algorithm is a search algorithm that wanders through the space of feasible solutions. At each step, this search algorithm moves from one feasible solution to another one nearby. In order to express the word 'nearby' mathematically, we need some notion of distance or neighborhood on the space of feasible solutions. For instance in the satisfiability problem, the feasible solutions are the truth assignments from the set X of logical variables to {0, 1}. A natural distance between truth assignments is the *Hamming* distance, that is, the number of bits where two truth assignments differ.

In this section we will concentrate on the 3-satisfiability problem where the input is a Boolean formula F in 3-CNF over the n logical variables in $X = \{x_1, x_2,\ldots,x_n\}$; see Section 2 for definitions and notations for this problem. We will describe three exact algorithms for 3-satisfiability that all are based on local search ideas. All three algorithms are centered around the Hamming neighborhood of truth assignments: For a truth assignment t and a non-negative integer d, we denote by $\mathcal{H}(t, d)$ the set of all truth assignments that have Hamming distance at most d from assignment t. It is easy to see that $\mathcal{H}(t, d)$ contains exactly $\Sigma_{k=0}^{d} \binom{n}{k}$ elements.

Exercise 4.1 *For a given truth assignment t and a given non-negative integer d, use the technique of pruning the search tree to check in $O^*(3^d)$ time whether the Hamming neighborhood $\mathcal{H}(t, d)$ contains a satisfying truth assignment for the 3-CNF formula F.*

In other words, the Hamming neighborhood $\mathcal{H}(t, d)$ can be searched quickly for the 3-satisfiability problem. For the k-satisfiability problem, the corresponding time complexity would be $O^*(k^d)$.

First local search approach to 3-satisfiability. We denote by 0^n (respectively 1^n) the truth assignment that sets all variables to 0 (respectively, to 1). Any truth assignment is in $\mathcal{H}(0^n, n/2)$ or in $\mathcal{H}(1^n, n/2)$. Therefore by applying the search algorithm from Exercise 4.1 twice, we get an exact algorithm with running time $O^*(\sqrt{3}^n) \oplus O^*(1.7321^n)$ for 3-satisfiability. It is debatable whether this algorithm should be classified under pruning the search tree or under local search. In any case, it is due to Schöning [32].

Second local search approach to 3-satisfiability. In the first approach, we essentially covered the whole solution space by two balls of radius $d = n/2$ centered at 0^n and 1^n. The second approach works with balls of radius $d = n/4$. The crucial idea is to *randomly* choose the center of a ball, and to search this ball with the algorithm from Exercise 4.1. If we only do this once, then we ignore most of the solution space, and the probability for answering correctly is pretty small. But by repeating this procedure a huge number α of times, we can boost the probability arbitrarily close to 1. A good choice for α is $\alpha = 100 \cdot 2^n / \sum_{k=0}^{n/4} \binom{n}{k}$. The algorithm now works as follows: Choose α times a truth assignment t uniformly at random, and search for a satisfying truth assignment in $\mathcal{H}(t, n/4)$. If in the end no satisfying truth assignment has been found, then answer that the formula F is not satisfiable.

We will now discuss the running time and the error probability of this algorithm. By Exercise 4.1, the running time can be bounded by roughly $\alpha \cdot 3^{n/4}$. By applying Stirling's approximation, one can show that up to a polynomial factor the expression $\sum_{k=0}^{n/4} \binom{n}{k}$ behaves asymptotically like $(256/27)^{n/4}$. Therefore, the upper bound $\alpha \cdot 3^{n/4}$ on the running time is in $O^*((3/2)^n) = O^*(1.5^n)$.

Now let us analyze the error probability of the algorithm. The only possible error occurs, if the formula F is satisfiable, whereas the algorithm does not manage to find a good ball $\mathcal{H}(t, n/4)$ that contains some satisfying truth assignment for F. For a single ball, the probability of containing a satisfying truth assignment equals $\sum_{k=0}^{n/4} \binom{n}{k} / 2^n$, that is the number of elements in $\mathcal{H}(t, n/4)$ divided by the overall number of possible truth assignments. This probability equals $100/\alpha$. Therefore the probability of selecting a ball that does *not* contain any satisfying truth assignment is $1 - 100/\alpha$. The probability of α times *not* selecting such a ball equals $(1 - 100/\alpha)^\alpha$, which is bounded by the negligible value e^{-100}.

In fact, the whole algorithm can be derandomized without substantially increasing the running time. Dantsin, Goerdt, Hirsch, Kannan, Kleinberg, Papadimitriou, Raghavan & Schöning [4] do not choose the centers of the balls at random, but they take all centers from a so-called *covering code* so that the resulting balls cover the whole solution space. They show that such covering codes can be computed within reasonable amounts of time. The approach in [4] yields deterministic exact algorithms for k-satisfiability with running time $O^*((2 - \frac{2}{k+1})^n)$. For 3-satisfiability, [4] improve the time complexity further down to $O^*(1.4802^n)$ by using a smart idea for an underlying branching step. This is currently the fastest known deterministic algorithm for 3-satisfiability.

Third local search approach to 3-satisfiability. The first approach was based on selecting the center of a ball deterministically, and then searching through the whole ball. The second approach was based on selecting the center of a ball randomly, and then searching through the whole ball. The third approach now is based on selecting the center of a ball randomly, and then doing a short random walk within the ball. More precisely, the algorithm repeats the following procedure roughly $200 \cdot (4/3)^n$ times: Choose a truth assignment t uniformly at random, and perform $2n$ steps of a random walk starting in t. In each step, first select a violated clause at random, then select a literal in the selected clause at random, and finally flip the truth value of the corresponding variable. If in the very end no satisfying truth assignment has been found, then answer that the formula F is not satisfiable.

The intuition behind this algorithm is as follows. If we start far away from a satisfying truth assignment, then the random walk has little chance of stumbling towards a satisfying truth assignment. Hence, it is a good idea to terminate it quite early after $2n$ steps, without wasting time. But if the starting point is very close to a satisfying truth assignment, then the probability is high that the random walk will be dragged closer and closer towards this satisfying truth assignment. And if the random walk indeed is dragged into a satisfying truth assignment, then with high probability this happens within the first $2n$ steps of the random walk. The underlying mathematical structure is a Markov chain that can be analyzed by standard methods. Clearly, the error probability can be made negligibly small by sufficiently often restarting the random walk. And up to a polynomial factor, the running time of the algorithm is proportional to the number of performed random walks. This implies that the time complexity is $O^*((4/3)^n) \oplus O^*(1.3334^n)$.

This algorithm and its analysis are due to Schöning [31]. Some of the underlying ideas go back to Papadimitriou [21]} who showed that 2-SAT can be solved in polynomial time by a randomized local search procedure. The algorithm easily generalizes to the k-satisfiability problem, and yields a randomized exact algorithm with time complexity $(O^*((2(k-1)/k)^n)$. The fastest known randomized exact algorithm for 3-satisfiability is due to Hofmeister, Schöning, Schuler & Watanabe [9], and has a running time of $O^*(1.3302^n)$. It is based on a refinement of the above random walk algorithm.

Open problem 4.2 *Design better deterministic and/or randomized algorithms for the k-satisfiability problem.*

More results on exact algorithms for k-satisfiability and related problems can be found in the work of Paturi, Pudlak & Zane [22], Paturi, Pudlak, Saks & Zane [23], Pudlak [25], and Rodősek) [28].

5 Concluding remarks

Currently, when we are dealing with an optimization problem, we are used to look at its computational complexity, its approximability behavior, its online behavior (with respect to competitive analysis), its polyhedral structure. Exact algorithms with good worst case (time) behavior should probably become another standard item on this list, and we feel that the known techniques and results as described in Sections 1-4 deserve to be taught in our introductory algorithms courses.

There remain many open problems and challenging questions around the worst case analysis of exact algorithms for NP-hard problems. This seems to be a rich and promising area. We only have a handful of techniques available, and there is ample space for improvements and for new results.

References

[1] D. APPLEGATE, R. BIXBY, V. CHVÁTAL, AND W. COOK [1998]. On the solution of traveling salesman problems. *Documenta Mathematica* 3, 645-656.

[2] R. BEIGEL [1999]. Finding maximum independent sets in sparse and general graphs. *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms* (SODA'1999), 856-857.

[3] R. BEIGEL AND D. EPPSTEIN [1995]. 3-Coloring in time O(1.3446^n): A no-MIS algorithm. *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'1995)*, 444-453.

[4] E. DANTSIN, A. GOERDT, E.A. HIRSCH, R. KANNAN, J. KLEINBERG, C.H. PAPADIMITRIOU, P. RAGHAVAN, AND U. SCHÖNING) [2001]. A deterministic $(2-\frac{2}{k+1})^n$ algorithm for *k*-SAT based on local search. To appear in *Theoretical Computer Science*.

[5] D. EPPSTEIN [2001]. Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction. *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA'2001)*, 329-337.

[6] D. EPPSTEIN [2001]. Small maximal independent sets and faster exact graph coloring. *Proceedings on the 7th Workshop on Algorithms and Data Structures (WADS'2001)*, Springer, LNCS 2125, 462-470.

[7] M.R. GAREY AND D.S. JOHNSON [1979]. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco.

[8] M. HELD AND R.M. KARP [1962]. A dynamic programming approach to sequencing problems. *Journal of SIAM* 10, 196-210.

[9] T. HOFMEISTER, U. SCHÖNING, R. SCHULER, AND O. WATANEBE [2001]. A probabilistic 3-SAT algorithm further improved. Manuscript.

[10] E. HOROWITZ AND S. SAHNI [1974]. Computing partitions with applications to the knapsack problem. *Journal of the ACM 21*, 277-292.

[11] R.Z. HWANG, R.C. CHANG, AND R.C.T. LEE [1993]. The searching over separators strategy to solve some NP-hard problems in subexponential time. *Algorithmica 9*, 398-423.

[12] R. IMPAGLIAZZO, R. PATURI, AND F. ZANE [1998]. Which problems have strongly exponential complexity? *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS'1998)*, 653-663.

[13] T. JIAN [1986]. An O ($2^{0.304n}$) algorithm for solving maximum independent set problem. *IEEE Transactions on Computers 35*, 847-851

[14] O. KULLMANN [1997]. Worst-case analysis, 3-SAT decisions, and lower bounds: Approaches for improved SAT algorithms. In: *The Satisfiability Problem: Theory and Applications*, D. Du, J. Gu, P.M. Pardalos (eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science 35, 261-313.

[15] O. KULLMANN [1999]. New methods for 3-SAT decision and worst case analysis. *Theoretical Computer Science 223*, 1-72.

[16] E.L. LAWLER [1976]. A note on the complexity of the chromatic number problem. *Information Processing Letters 5*, 66-67.

[17] R.J. LIPTON AND R.E. TARJAN [1979]. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics 36*, 177-189.

[18] B. MONIEN AND E. SPECKENMEYER [1985]. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics 10*, 287-295.

[19] J.W. MOON AND L. MOSER [1965]. On cliques in graphs. *Israel Journal of Mathematics 3*, 23-28.

[20] J.M. NIELSEN [2001]. Personal communication.

[21] C.H. PAPADIMITRIOU [1991]. On selecting a satisfying truth assignment. *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS'1991)*, 163-169.

[22] R. PATURI, P. PUDLAK, AND F. ZANE [1997]. Satisfiability coding lemma. *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS'1997)*, 566-574.

[23] R. PATURI, P. PUDLAK, M.E. SAKS, AND F. ZANE [1998]. An improved exponential time algorithm for *k*-SAT. *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS'S1998)*, 628-637.

[24] M. PAULL AND S. UNGER [1959]. Minimizing the number of states in incompletely specified sequential switching functions. *IRE Transactions on Electronic Computers 8*, 356-367.

[25] P. PUDLAK [1998]. Satisfiability – algorithms and logic. *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science (MFCS'1998)*, Springer, LNCS 1450, 129-141.

[26] J.M. ROBSON [1986]. Algorithms for maximum independent sets. *Journal of Algorithms 7*, 425-440.

[27] J.M. ROBSON [2002]. Finding a maximum independent set in time $O(2^{n/4})$? Manuscript.

[28] R. RODOŠEK) [1996]. A new approach on solving 3-satisfiability. *Proceedings of the 3rd International Conference on Artificial Intelligence and Symbolic Mathematical Computation, Springer, LNCS 1138*, 197-212.

[29] I. SCHIERMEYER [1992]. Solving 3-satisfiability in less than $O(1.579^n)$ steps. *Selected papers from Computer Science Logic (CSL'1992)*, Springer, LNCS 702, 379-394.

[30] I. SCHIERMEYER [1993]. Deciding 3-colorability in less than $O(1.415^n)$ steps. *Proceedings of the 19th Workshop on Graph Theoretic Concepts in Computer Science (WG'1993)*, Springer, LNCS 790, 177-182.

[31] U. Schöning [1999]. A probabilistic algorithm for *k*-SAT and constraint satisfaction problems. *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS'1999)*, 410-414.

[32] U. Schöning [2001]. New algorithms for *k*-SAT based on the local search principle. *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS'2001)*, Springer, LNCS 2136, 87-95.

[33] R. SCHROEPPEL AND A. SHAMIR [1981]. A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems. *SIAM Journal on Computing 10*, 456-464.

[34] R.E. TARJAN AND A.E. TROJANOWSKI [1977]. Finding a maximum independent set. *SIAM Journal on Computing 6*, 537-546.

[35] A. VAN VLIET [1995]. Personal communication.

Report from the Integer Programming Conference in Honor of Egon Balas

Alberto Caprara

As somebody noted at the conference, Egon Balas is so efficient that he was born in early June, a perfect period to celebrate his birthday with a conference. His precise age is perhaps not particularly significant, considering that his aspect is essentially unchanged with respect to the picture from the 50s that appears in his biography (have you ever heard about "highlanders"?).

In any case, since Egon's age is an integer multiple of 10 starting from June 8, Gerard Cornuejols and Bill Pulleyblank organized at CMU a conference in his honor, which consisted of three days of invited plenary talks by people who have done outstanding work on Integer Programming, many of them being Egon's co-authors or former PhD students, and one afternoon of contributed talks selected from a wide set of submissions.

The conference organization was supported by Mary Bober and Barbara Carlson, whose kindness and efficiency reminds everybody that outstanding universities have outstanding staff. Also, a few PhD students at CMU were involved in the not-so-easy task of distributing the room keys to the many attendants staying in the CMU buildings and arriving at every possible hour during the night - this was probably a good training to stay up for watching the soccer World Cup.

Not only was there no conference fee (while there was plenty of good conference food and drinks), but the participation of a large number of PhD students was supported by the various sponsors.

The only bug in the organization was that the good weather was reserved only for the conference duration, whereas many people had to stay a few more days before or after, noting what is the meaning of bad weather in the summer in Pittsburgh (once back, I stopped complaining about the bad weather in Bologna).

Back to Egon, I am probably not the right person to say something on him from a scientific viewpoint, since my Balas number is only 2 (a "heavy" 2, however). I would simply like to note that his impact on the scientific community goes certainly beyond his many outstanding papers, co-authors, and PhD students. For instance, thinking about my personal case, Egon is one of those mythical scientists that attract in their work place many young people interested in doing research: even

if eventually most of these people do not actively work with the mythical scientist (simply because they are too many), they find a very interesting environment that changes their view of research. Moreover, and I would say most important, looking at Egon all researchers from my generation tend to have very optimistic expectations for their research activity in the next (at least) 50 years...

For those who want to hear something personal about Egon, there would be the talks given by Edith, Anna and Vera Balas, Sebastian Ceria, Manfred Padberg and Bill Pulleyblank at the conference dinner. We cannot give here all of them for several reasons. For instance, all the pictures in Sebastian's talk would require the space of a few issues of OPTIMA. On the other hand, we are very proud to report below the speech of somebody who certainly needs no introduction, having Balas number 1, a very heavy 1, and at the same time Padberg number 0!

Hi folks,

I am Manfred Padberg, Ph.D. 1971, GSIA, Carnegie-Mellon University. You must have heard that I am Egon's first student or, maybe, that I am his first retired student.

In any case, as Egon once told me and probably others, I am his FIRST AMERICAN STUDENT. Now that is THREE propositions in one: STUDENT, yes. That is in the GSIA

files. FIRST, no, not really anyway --Egon has helped others in their professional development before me: Peter Hammer is here tonight with us. Peter can attest to the fact that he benefitted from Egon's brain way before me. As did others like Cristian Bergenthaller, Mihai Dragomirescu back in Egon's native country. Now that leaves AMERICAN. And that's a good question: American, who? Egon, me? The answer seems easy at first: Egon carries an American passport since many years; so, *de jure et de facto* (by law and by his whereabouts), Egon is American. My own passport is still what it was when I came to this country in 1968; so, *de jure*, I am not an American to date; neither *de facto* --I live currently in France. Yet I am Egon's FIRST AMERICAN STUDENT; the man has said so himself! What better proof?

True, while neither of the two of us were Americans in this sense in 1968, undoubtedly, both of us have become AMERICANIZED over the many years that have passed since then. So here is the question that I will try to answer tonight: How AMERICAN has Egon gotten over these past thirty five something years that he and Edith have lived on this continent?

I will start with the time when I came to GSIA. This was at the beginning of the 1968/1969 academic year. Egon had just about one year earlier joined GSIA as the Ford Distinguished Research Professor and he was new, very much a *Homo Novus* in the GSIA spirit of that time. Egon was already very well-known internationally and certainly deserved the grand title. His 1965 paper "An additive algorithm for solving linear programs with zero-one variables", published in the flagship journal of our profession *Operations Research*, had rocked, yes rocked, the discrete optimization world. The "Balasian algorithm" --as it was called by several of our colleagues-- had opened a way to answer difficult problems in business. Problems that had been identified and formulated in the ten to fifteen years preceding Egon's paper and that could in fact not be solved before that time. This paper --like many other papers written by Egon later on-- is a true classic in the field of *Operations Research/Management Science*. In a study done in the mid eighties Egon's 1965 paper was identified as the most frequently cited article of the *Operations Research* journal during the twenty year period preceding the survey.

I had come from the University of Mannheim with a Ford Foundation Doctoral Fellowship to get my Ph.D. from GSIA. So both I and Egon have to thank Mr. Ford for giving us the chance of getting and working together. For some reason I was classified a "Special Student", "special" meaning "problematic" or likely not to succeed in the Ph.D. program. Don't ask me why. I managed to get into Egon's advanced classes right from the start nevertheless. It was not easy. For me this experience was an important first step towards my own Americanization. I'll tell you the details some other time.

That first semester at GSIA I took *Economics of the Firm* with Richard Roll, *Introduction to Management Analysis and Policy*, co-taught by Kalman Cohen and Alfred Kuehn, and Egon Balas' *Mathematical Programming~I*. Boy, what a difference in teaching styles and personalities. Cohen, Kuehn and Roll expected to --if not insisted on-- being called Kal, Al and Dick, respectively. I had seen that happen in American movies and here it was reality. But not so in Balas' case. With Egon it was either Professor Balas or Dr. Balas. There were other striking differences in the teaching style. But I can't dwell on them now.

What about Egon's teaching? One word and that one in capital letters suffices to describe it: SUPERB. The man would stroll into the class room on time, not one piece of paper in his hands or pockets, no "absent minded professor" stuff. Egon typically put one or two of his twenty-five or so students up on the blackboard to check on the previous assignments. Then he proceeded to deliver a meticulously prepared lecture --no repetitions, no story telling in Egon's classes. You want proof? Come and visit Suzy and me in Marseille. I still have my old note books from my student days at GSIA. I flipped through them the other day, just to remind me. For most classes, there are some four or five pages filled with my scriblings of that time and then many BLANK pages. Not so for Egon's classes. Indeed, these old notes from his classes are still helpful today, to grasp this fine point or that in the development of an argument.

Egon's teaching was inspiring and demanding at the same time. Persistence in the problem solving process is one of the prime qualities that I learned from Egon and that comes to my mind when I think back to those years. Let me tell you a little anedocte about that. In the Spring term of 1969 Egon finally discussed his additive algorithm for zero-one linear programs in class. He then gave an assignment, one of the number examples from his classic paper. Except Egon did not tell us so! Of course not! I believe it was the Eastern or Pentecoste weekend of 1969. The Padberg's had wanted to spend it at Lake Erie. To do the assignment took the entire weekend. No Lake Erie that weekend, my wife and kids were disenchanted to say the least, but I got the job done. Oh no, I was not the only one in Egon's class, mind you. Marcel Boyer, a French Canadian fellow student, had done it too. When Marcel and I compared notes on Tuesday before class, it had taken Marcel 48 handwritten pages compared to my 49 pages to do the assignment. Most importantly, we had found the same optimum solution. What had happened? Egon -- in one of his rare underperformances-- had inadvertently messed up one of the coefficients of the number example when he had put it on the black board. The example --when changed-- took at best two pages to solve. Morale? First, Egon had inspired at least two guys in his class-- Marcel Boyer and me-- to spoil a long Spring weekend for our respective families. For the betterment of science naturally! Second, this taught us that persistence is necessary to solve difficult problems.

Of course, inspiration and persistence hardly ever come from a single source. GSIA's eminent faculty in those days, like William (Bill) Cooper, the late Richard Cyert or Dick Cyert to us students, Gerald (Gerry) Thompson, Robert (Bob) Lucas, Herbert (Herb) Simon and many others had created an intellectual climate that was unique and conducive, really conducive to the use of mathematics in the social sciences. I hope that this spirit persists at GSIA to date.

At about the time that I took Egon's second class I was working with Kal Cohen. I had read the book "*Analytical Methods in Banking*" which Kal had coedited in 1966. This book is full of mixed-integer programming formulations for difficult, challenging and important problems from finance and the banking industry. Many of these problems remain important to date. Mathematically speaking, these were exactly the kind of problems that Egon was interested in. They were big and difficult to solve. Now that is exactly what Egon was really trying to do. Solve those difficult, big problems from industry. No wonder then that by my second year at GSIA, Egon had become my official advisor in GSIA's Ph.D. program.

I won't bore you with a day-by-day or a year-by-year account of all of these years. That would

be impossible here. Suffice it to say, that by about 1973/1974 Egon and I had written four or five joint articles; we had several papers in the "pipeline", some of which never saw the light of day. I had returned to Germany upon graduation in 1971. My job in Berlin permitted me to travel freely. Egon and I continued our cooperation by my frequent visits to Pittsburgh over several years. And yet: Egon was still "Professor Balas" to me. But then one day it happened. At the end of a long technical letter --Egon was dissecting a referee report on a joint paper and he still knows how to do that-- it said:

"By the way, Manfred, I think you should call me Egon."

I must have read this line twice or thrice, probably aloud to myself. What had happened? Egon had joined the world of the Kal's, the Al's, the Bill's, the Dick's, the Gerry's, of the American culture. No more European stiffness, that European professorial stuff. The inevitable Americanization of Professor Egon Balas had set in.

As it goes with such processes of assimilation to a different culture, there are relapses. It never comes easy. I remember that evening in 1981 in my apartment in Brussels, Belgium, when I had the impression that Egon might have preferred to be called "Professor Balas" again. But it was a fleeting impression. When several years later Egon told me that he was just a husband like the most of us, "the husband of a famous art historian by the name of Edith Balas", I knew that all was well, that things were going in the right direction. The final proof that Egon had become a true American came a few of years ago. He told me that the Balas' had left their lovely home overlooking the Shadyside section of Pittsburgh which I knew from the earlier years. He told me that they now owned a bigger property in the Squirrel Hill part of town and that they had their own swimming pool. A private swimming pool! The ultimate dream and achievement of the American couple! Now, there I had proof that Egon can proudly say to himself and to all of us:
Mathematicus Americanus sum!

Happy birthday, Egon!

Report from the 9th IPCO Conference

Alberto Caprara

The 9th Conference on Integer Programming and Combinatorial Optimization was held at the Massachusetts Institute of Technology (MIT) in the last week of May. This is the only selective conference in the "Optimization" field, as opposed to "Theoretical Computer Science", where such meetings are the norm. As a result, most of the participants have either a paper accepted or did not submit anything, both because many are not financially supported if they cannot give a talk, and because it is psychologically difficult to attend if the paper you submitted was rejected. This means that the number of participants is generally between 50 and 100 (actually it was more than a hundred this year), which makes it possible to talk at least a few minutes with all the people you know, something certainly impossible in very large conferences such as the Mathematical Programming Symposium. Moreover, selectivity and session uniqueness force the speakers to be more responsible (since acceptance of their paper implied rejection of others, and they are not competing for the audience with a few dozen other speakers in parallel), guaranteeing a very high level of the talks. In my opinion, this should stimulate the MPS authorities to consider the possibility in the future to have an IPCO *every year* instead of twice every three years, of course keeping the big Symposium every third year.

On the scientific side, the program committee was chaired by Bill Cook. The subjects covered by the 33 accepted papers (about one third of the 110 submitted, most of which I guess of very good quality) were essentially the same as for the previous IPCO conferences. In particular, the main topics, which refer to about 3/4 of the papers presented, are (in order of appearance at the conference): submodular function minimization and generalizations, multicommodity flows, integrality gaps, separation for the TSP polytope, properties and practical use of general cutting planes, approximation algorithms (and schemes) for scheduling, packing, facility location and network design problems, nonstandard approaches to integer programming (based on semidefinite and subadditive relaxations and on basis reduction).There was certainly more emphasis on the theoretical aspects of the problems rather than on their practical solution, although most problems discussed arise from important real-world applications. Personally, I noted an increasing interest in knapsack-like packing problems with respect to the past, contrary to my previous impression to be one of the very few researchers still interested in these problems.

As was (fortunately) always the case in all occasions in which I could attend an IPCO meeting, the organization, whose committee was chaired by Andreas Schulz, was very efficient. For the social part, this meant providing plenty of good food and drinks for the participants for three consecutive days - this forced me to make a trip to Boston without visiting any restaurant. In particular, on Sunday night, before the conference started, there was a reception at the MIT museum, where many participants could try an interesting combination of a 6-hours jet lag with special effects from the museum (such as holograms). This is at least as much fun as being drunk, with the notable difference that every effect disappears as soon as you go to bed, which is something that everybody above 30 appreciates a lot. The conference dinner was held at the Fogg Art Museum at Harvard University, a very nice location with very good food and drinks, which is a very rare and appreciated combination. Perhaps, the only drawback was the very high quality and easy accessibility of the wine, whose effect was not completely vanished the day after...

D. R. FULKERSON PRIZE

Call for nominations

The Fulkerson Prize Committee invites nominations for the Delbert Ray Fulkerson Prize, sponsored jointly by the Mathematical Programming Society and the American Mathematical Society. The Fulkerson Prize is for outstanding papers in the area of discrete mathematics. The Prize will be awarded at the XVIIIth International Symposium on Mathematical Programming to be held in Copenhagen, Denmark, August 18-22, 2003.

Eligible papers should represent the final publication of the main result(s) and should have been published in a recognized journal, or in a comparable, well-refereed volume intended to publish final publications only, during the six calendar years preceding the year of the Symposium (thus, from January 1997 through December 2002). The prizes will be given for single papers, not series of papers or books, and in the event of joint authorship the prize will be divided.

The term 'discrete mathematics' is interpreted broadly and is intended to include graph theory, networks, mathematical programming, applied combinatorics, applications of discrete mathematics to computer science, and related subjects. While research work in these areas is usually not far removed from practical applications, the judging of papers will only be based on their mathematical quality and significance.

Previous winners of the Fulkerson Prize are listed below. Further information about the Fulkerson Prize can be found at www.mathprog.org/prz/fulkerson.htm.

The Fulkerson Prize Committee consists of Gerard Cornuejols (Carnegie-Mellon University), Andrew M. Odlyzko (University of Minnesota), and David P. Williamson (IBM Almaden Research Center), chair.

Please send your nominations (including reference to the nominated article and an evaluation of the work) by January 31, 2003 to the chair of the committee. Electronic submissions are preferred.

David P. Williamson
Re: Fulkerson Prize
IBM Almaden Research Center, K53/B1
650 Harry Rd.
San Jose, CA 95120
USA
e-mail: dpw@almaden.ibm.com

Previous winners of the Fulkerson Prize:

1979: Kenneth Appel and Wolfgang Haken; Richard M. Karp; Paul D. Seymour
1982: L.G. Khachiyan/D.B. Iudin and A.S. Nemirovskii; G.P. Egorychev/D.I. Falikman; Martin Grotschel, Laszlo Lovasz, and Alexander Schrijver
1985: Jozsef Beck; H.W. Lenstra, Jr.; Eugene M. Luks
1988: Eva Tardos; Narendra Karmarkar
1991: Alfred Lehman; Nikolai E. Mnev; Martin Dyer, Alan Frieze, and Ravi Kannan
1994: Lou Billera; Neil Robertson, Paul D. Seymour, and Robin Thomas; Gil Kalai
1997: Jeong Han Kim
2000: Michel X. Goemans and David P. Williamson; Michele Conforti, Gerard Cornuejols, and M. R. Rao

New Editorial Board for *Operations Research Letters*

George Nemhauser, the founding editor of *Operations Research Letters*, has resigned after being at the helm for twenty years. Starting with Volume 30, the editorial board will be led by Jan Karel Lenstra (jkl@win.tue.nl). There are thirteen area editors, to whom papers should be submitted:

Sigrun Andradottir
(sa@isye.gatech.edu), simulation

Sem Borst (sem@cwi.nl), communication networks

Onno Boxma
(boxma@win.tue.nl),
queueing theory and applications

Vadim Linetsky
(linetsky@iems.nwo.edu),
financial engineering

Alexander Martin
(martin@mathematik.tu-darmstadt.de),
discrete optimization

S. Thomas McCormick
(stmv@adk.commerce.ubc.ca),
graphs and networks

Nimrod Megiddo
(megiddo@almaden.ibm.com),
OR with multiple decision makers

Martin Savelsbergh
(Martin.Savelsbergh@isye.gatech.edu),
routing, location and supply chain management

Stefan Scholtes
(s.scholtes@jims.cam.ac.uk),
continuous optimization

Rüdiger Schultz
(schultz@math.uni-duisburg.de),
linear and stochastic optimization

Sridhar Seshadri
(sseshadr@stern.nyu.edu),
inventory, reliability and control

Steef van de Velde
(s.velde@fac.fbk.eur.nl),
sequencing and scheduling

Gerhard Woeginger
(woeginge@math.utwente.nl),
approximation and heuristics

The new board intends to reinforce the character of ORL as a high-quality journal for the rapid publication of short papers and research announcements. It is committed to make a serious attempt to review each submission within three months. The electronic submission of papers is encouraged.

Next to concise articles, generally limited to ten journal pages, extended abstracts of two to four pages, announcing results without full proofs, are welcome too. The authors of such contributions may submit support material that enables the board to verify the results.

gallimaufry

Application for Membership

I wish to enroll as a member of the Society.

My subscription is for my personal use and not for the benefit of any library or institution.

☐ I will pay my membership dues on receipt of your invoice.

☐ I wish to pay by credit card (Master/Euro or Visa).

CREDIT CARD NO. EXPIRATION DATE

FAMILY NAME

MAILING ADDRESS

TELEPHONE NO. TELEFAX NO.

EMAIL

SIGNATURE ▶

Mail to:

Mathematical Programming Society
3600 University City Sciences Center
Philadelphia PA 19104-2688 USA

Cheques or money orders should be made payable to The Mathematical Programming Society, Inc. Dues for 2002, including subscription to the journal ***Mathematical Programming***, are US $80.
Student applications: Dues are one-half the above rate. Have a faculty member verify your student status and send application with dues to above address.

Faculty verifying status

Institution

Order Form

Springer

Simply fax this whole page to
212-533-5587

25% off all titles below for members of the **Mathematical Programming Society**

Please send me:

- ❑ **3-540-41744-3** Alevras, Padberg, Linear Optimization and Extensions. US ~~$49.95~~ $37.46
- ❑ **3-540-65431-3** Ausiello et al., Approximate Solutions of NP-hard Optimization Problems. US ~~$59.95~~ $44.96
- ❑ **0-387-98633-2** Aven, Jensen, Stochastic Models in Reliability. US ~~$71.95~~ $53.96
- ❑ **0-387-98859-9** Balakrishnan, Ranganathan, A Textbook of Graph Theory. US ~~$59.95~~ $44.96
- ❑ **1-85233-268-9** Bang-Jensen,Gutin, Digraphs. Theory, Algorithms and Applications Theory. US ~~$95.00~~ $71.25
- ❑ **0-387-98705-3** Bonnans, Shapiro, Perturbation Analysis of Optimization Problems. US ~~$79.95~~ $59.96
- ❑ **0-387-98535-2** Chicone, Ordinary Differential Equations with Applications. US ~~$59.95~~ $44.96
- ❑ **0-387-98727-4** Cohen, Advanced Topics in Computational Number Theory. US ~~$59.95~~ $44.96
- ❑ **0-387-98512-3** Crauel, Gundlach (Eds.), Stochastic Dynamics. US ~~$64.95~~ $48.71
- ❑ **0-387-98976-5** Diestel, Graph Theory. **Softcover.** US ~~$34.95~~ $26.21
- ❑ **0-387-95014-1** Diestel, Graph Theory. **Hardcover.** US ~~$69.95~~ $52.46
- ❑ **3-540-67787-9** Du et al. (Eds.), Computing and Combinatorics. US ~~$73.00~~ $54.75
- ❑ **0-387-98945-5** Dullerud, Paganini, A Course in Robust Control Theory. US ~~$59.95~~ $44.96
- ❑ **0-387-98836-X** Durrett, Essentials of Stochastic Processes. US ~~$69.95~~ $52.46
- ❑ **3-540-67191-9** Eiselt, Sandblom, Integer Programming and Network Models. US~~$94.00~~ $70.50
- ❑ **3-540-60931-8** Embrechts et al., Modelling Extremal Events. US ~~$79.95~~ $59.96
- ❑ **0-387-94527-X** Fishman, Monte Carlo.Concepts, Algorithms, and Applications. US ~~$74.95~~ $56.21
- ❑ **3-540-67633-3** Giancarlo, Sankoff (Eds.), Combinatorial Pattern Matching. US ~~$69.00~~ $51.75
- ❑ **3-540-64766-X** Goldreich, Modern Cryptography, Probalistic Proofs and Pseudorandomness. US ~~$79.95~~ $59.96
- ❑ **3-540-64902-6** Grimmett, Percolation. US ~~$99.00~~ $74.25
- ❑ **0-387-98736-3** Harris et al., Combinatorics and Graph Theory. US ~~$44.95~~ $33.71
- ❑ **3-540-66419-X** Jacod, Protter, Probability Essentials. US ~~$36.00~~ $27.00
- ❑ **3-540-67996-0** Jansen, Khuller (Eds.), Approximation Algorithms for Combinatorial Optimization. US ~~$52.00~~ $39.00
- ❑ **0-387-94839-2** Karatzas, Shreve, Methods of Mathematical Finance. US ~~$74.95~~ $56.21
- ❑ **0-387-98605-7** Kevorkian, Partial Differential Equations. US ~~$59.95~~ $44.96
- ❑ **3-540-67226-5** Korte, Vygen, Combinatorial Optimization. US ~~$45.00~~ $33.75
- ❑ **0-387-98725-8** Kulkarni, Modeling, Analysis, Design, and Control of Stochastic Systems. US ~~$74.95~~ $56.21
- ❑ **0-387-95139-3** Kushner, Dupuis, Numerical Methods for Stochastic Control Problems in Continuous Time. US ~~$69.95~~ $52.46
- ❑ **3-540-65274-4** Langtangen, Computational Partial Differential Equations. US ~~$74.95~~ $56.21
- ❑ **3-540-41554-8** Marks (Ed.), Graph Drawing. US ~~$59.00~~ $44.25
- ❑ **3-540-67296-6** Mei, Numerical Bifurcation Analysis for Reaction-Diffusion Equations. US ~~$84.00~~ $63.00
- ❑ **3-540-66061-5** Michalewicz, Fogel, How to Solve It: Modern Heuristics. US ~~$49.95~~ $37.46
- ❑ **3-540-67715-1** Montanari et al. (Eds.), Automata, Languages and Programming. US ~~$89.00~~ $66.75
- ❑ **3-540-66024-0** Murota, Matrices and Matroids for Systems Analysis. US ~~$112.00~~ $84.00
- ❑ **3-540-61477-X** Musiela, Rutkowski, Martingale Methods in Financial Modelling. US ~~$79.95~~ $59.96
- ❑ **3-540-41114-3** Nazareth, DLP and Extensions. US ~~$54.95~~ $41.21
- ❑ **3-540-66905-1** Nguyen et al. (Eds.), Optimization. US ~~$72.00~~ $54.00
- ❑ **0-387-98793-2** Nocedal, Wright, Numerical Optimization. US ~~$69.95~~ $52.46
- ❑ **3-540-41004-X** Paterson (Ed.), Algorithms - ESA 2000.US ~~$73.00~~ $54.75
- ❑ **1-85233-304-9** Pelsser, Efficient Methods for Valuing Interest Rate Derivatives. US ~~$64.95~~ 48.71
- ❑ **3-540-57045-4** Prokhorov, Statulevicius (Eds.), Limit Theorems of Probability Theory. US ~~$104.00~~ $78.00
- ❑ **3-540-67442-X** Rolim (Ed.), Parallel and Distributed Processing. US ~~$110.00~~ $82.50
- ❑ **0-387-98513-1** Sastry, Nonlinear Systems. US ~~$79.95~~ $59.96
- ❑ **0-387-98773-8** Serfozo, Introduction to Stochastic Networks. US ~~$74.95~~ $56.21
- ❑ **0-387-95016-8** Steele, Stochastic Calculus and Financial Applications. US ~~$69.95~~ $52.46
- ❑ **0-387-94654-3** Taylor, Partial Differential Equations. US ~~$49.95~~ $37.46
- ❑ **0-387-98346-5** Thomas, Numerical Partial Differential Equations. US ~~$59.95~~ $44.96
- ❑ **0-387-98779-7** Thorisson, Coupling, Stationarity, and Regeneration. US ~~$79.95~~ $59.96
- ❑ **3-540-66321-5** Varga, Matrix Iterative Analysis. US ~~$89.95~~ $67.46
- ❑ **3-540-65367-8** Vazirani, Approximation Algorithms. US ~~$34.95~~ $26.21
- ❑ **3-540-67853-0** Wesseling, Principles of Computational Fluid Dynamics. US ~~$89.00~~ $66.75

Please fill in your membership number:_____

BOOK SUBTOTAL $_____

SALES TAX $_____

SHIPPING $_____

TOTAL AMOUNT $_____

SALES TAX: RESIDENTS OF CA, IL, MA, MO, NJ, NY, PA, TX, VA, AND VT, PLEASE ADD SALES TAX.
CANADIAN RESIDENTS, PLEASE ADD 7% GST.

SHIPPING: PLEASE ADD $5.00 FOR SHIPPING ONE BOOK AND $1.00 FOR EACH ADDITIONAL BOOK.

Orders are processed upon receipt. If an order cannot be fulfilled within 90 days, payment will be refunded upon request. Prices quoted are payable in US currency or its equivalent and are subject to change without notice. Remember, your 30 day return privilege is always guaranteed.

METHOD OF PAYMENT:

❑ CHECK/MONEY ORDER ENCLOSED ❑ AMEX ❑ MC ❑ VISA ❑ DISCOVER

CARD NO _____ EXP. DATE _____

SIGNATURE _____ DATE _____

NAME _____

ADDRESS _____

CITY/STATE/ZIP _____

PHONE _____ E-MAIL _____

PLEASE MAIL ORDERS TO:
Mathematics Promotion
Promotion Code S350
Springer-Verlag New York, Inc.
175 Fifth Avenue
New York, NY 10010

CALL:
1-800-SPRINGER
FAX:
(212) 533-5587

Please make sure you mention promotion code S350 when calling or faxing. You will need this in order to receive the appropriate discount.

Promotion #S350

O P T I M A

MATHEMATICAL PROGRAMMING SOCIETY

UNIVERSITY OF
FLORIDA

Center for Applied Optimization
371 Weil
PO Box 116595
Gainesville FL 32611-6595 USA

FIRST CLASS MAIL

PUBLISHED BY THE
MATHEMATICAL PROGRAMMING SOCIETY &
GATOREngineering® PUBLICATION SERVICES
University of Florida