Computing in Combinatorial Optimization

William Cook(✉)

University of Waterloo, Waterloo, Canada
bico@uwaterloo.ca

Abstract. Research in combinatorial optimization successfully combines diverse ideas drawn from computer science, mathematics, and operations research. We give a tour of this work, focusing on the early development of the subject and the central role played by linear programming. The paper concludes with a short wish list of future research directions.

Keywords: Combinatorial optimization · Linear programming
Traveling salesman problem

1 In the Beginning was n Factorial

The design of efficient algorithms for combinatorial problems has long been a target of computer science research. Natural combinatorial models, such as shortest paths, graph coloring, network connectivity and others, come equipped with a wide array of applications as well as direct visual appeal. The discrete nature of the models allows them to be solved in finite time by listing candidate solutions one by one and selecting the best, but the number of such candidates typically grows extremely fast with the input size, putting optimization problems out of reach for simple enumeration schemes.

A central model is the *traveling salesman problem*, or TSP for short. It is simple to state. Given a set of cities, together with the distance between each pair, find the shortest way to visit all of them and return to the starting point, that is, find the shortest circuit containing every city in the specified set.

The TSP has its own array of applications, but its prominence is more due to its success as an engine of discovery than it is to miles saved by travelers. The cutting-plane method [20], branch-and-bound [22], local search [32], Lagrangian relaxation [53], and simulated annealing [65], to name just a few, were all developed with the salesman problem as the initial target. This success, coupled with the simplicity of the model, has made the TSP the poster child for the $\mathcal{NP}$-hard class of problems.

It was Karl Menger who first brought the algorithmic challenge of the TSP to the attention of the mathematics community, describing the model in a colloquium held in Vienna in 1930 [76].

B. Steffen and G. Woeginger (Eds.): Computing and Software Science, LNCS 10000, pp. 27–47, 2019.
https://doi.org/10.1007/978-3-319-91908-9_3

> We use the term *Botenproblem* (because this question is faced in practice by every postman, and, by the way, also by many travelers) for the task, given a finite number of points with known pairwise distances, to find the shortest path connecting the points. This problem is of course solvable by finitely many trials. Rules that give a number of trials below the number of permutations of the given points are not known.

Menger was quite informal about the notion of an algorithm, speaking of "rules". (He used the German word "Regeln".) Keep in mind that at the time Alan Turing was an undergraduate student at Cambridge and still several years away from his breakthrough concept of general-purpose computing machines [87].

Menger's call for a better-than-finite TSP algorithm arose at an important period of mathematical research, where fundamental issues in logic and computability were being explored. Indeed, the published statement of Menger's problem appears in the same volume as Gödel's paper [37] on incompleteness theorems. This is not entirely a coincidence, since Gödel and Menger were both Ph.D. students of Hans Hahn and active members of the Vienna Circle. Being a great fan of the salesman problem, I'd like to think Gödel and Menger spent time discussing TSP complexity in Viennese coffee houses, but there is no indication of that.

His challenge did however capture the imagination of the mathematics community. Denoting by n the number of cities to visit, beating $n!$ in a race to the optimal TSP tour became a well-established research topic by the late 1940s.

The first major paper was written by Robinson [82] in 1949, while she was a post-doctoral fellow at the RAND Corporation. Her Ph.D. thesis, *Definability and Decision Problems in Arithmetic* [81], placed her firmly on Gödel's side of the Vienna Circle activities, but the TSP proved hard to resist while she was at RAND as part of "what may have been the most remarkable group of mathematicians working on optimization every assembled" [50].

When we today read the statement of Menger's problem, we immediately think of it in terms of establishing an asymptotic complexity bound. But early researchers considered it not as an asymptotic challenge, but rather as a call for mathematical methods to handle modest-sized instances. The following passage is Robinson's description [82] of her research target.

> Since there are only a finite number of paths to consider, the problem consists in finding a method for picking out the optimal path when n is moderately large, say $n = 50$. In this case there are more than 10^{62} possible paths, so we can not simply try them all. Even for as few as 10 points, some short cuts are desirable.

Her approach towards a 50-point algorithm was to consider a relaxation of the TSP where each city is assigned to another (and no two cities are assigned to the same target city), so as to minimize the total distance between pairs of cities in the assignments. A solution to this relaxation gives a set of disjoint directed circuits (possibly including circuits having only two points) such that every city

is contained in exactly one of them. The total length of the solution provides a lower bound on the length of any TSP tour.

Robinson's relaxation is known as the *assignment problem*, where we typically think of assigning n workers to n tasks, rather than assigning cities to cities. Her solution method is an iterative approach, moving step by step towards the optimal assignment via a cycle-canceling operation (that later became a standard technique for minimum-cost flow problems).

> The method presented here of handling this problem will enable us to check whether a given system of circuits is optimal or, if not, to find a better one. I believe it would be feasible to apply it to as many as 50 points provided suitable calculating equipment is available.

She mentions again the specific target of 50-point problems, but the paper does not provide an analysis of the algorithm's running time.

Several years later, in 1956, Flood [32] wrote the following in a survey paper on TSP methods.

> It seems very likely that quite a different approach from any yet used may be required for successful treatment of the problem. In fact, there may well be no general method for treating the problem and impossibility results would also be valuable.

It is not clear what he meant by "impossible", but Flood may have had in mind the notion of polynomial time and was thus speculating that the TSP had no solution method with the number of steps bounded above by n^k for some constant k. Such an algorithm would certainly have been viewed as a major breakthrough at the time, possibly permitting the solution of the target-sized instances.

An argument for suggesting Flood was hoping to replace Menger's $n!$ by a more tidy n^k can be seen in Schrijver's beautifully detailed history of combinatorial optimization [84, Chap. 3]. Schrijver cites several examples from the early 1950s where authors point out polynomial running-time bounds. The earliest of these is the following passage from a lecture given by von Neumann [78] in 1951, concerning the assignment problem.

> We shall now construct a certain related 2-person game and we shall show that the extreme optimal strategies can be expressed in terms of the optimal permutation matrices in the assignment problem. (The game matrix for this game will be $2n \times n^2$. From this it is not difficult to infer how many steps are needed to get significant approximate solutions with the method of G. W. Brown and J. von Neumann. It turns out that this number is a moderate power of n, i.e. considerably smaller than the "obvious" estimate $n!$ mentioned earlier.)

Details are not provided in the transcript, but the use of the phrase "moderate power of n" fits with the overall theme of developing methods for use in practice.

An explicit running-time bound for solving the assignment problem was given by Munkres [77] in 1957, making use of a variant of the Hungarian algorithm.

The final maximum on the number of operations needed is

$$(11n^3 + 12n^2 + 31n)/6.$$

This maximum is of theoretical interest since it is so much smaller than the $n!$ operations necessary in the most straightforward attack on the problem.

Munkres's paper is entirely analysis; there is no mention of target applications or possible computational testing. Indeed, his use of the phrase "theoretical interest" is striking, making an early claim for the mathematics of algorithmic complexity.

From Menger to Robinson to von Neumann to Munkres, we see the modern treatment of combinatorial algorithms starting to take shape. But back in the TSP camp, the obvious $n!$ bound for worst-case complexity remained the state of the art through the 1930s, 40s, and 50s. This finally changed in 1962 with the publication of papers by Bellman [5] and Held and Karp [52], describing an elegant method for solving any n-city TSP in time proportional to $n^2 2^n$. Not the dreamed-for n^k, but still an answer to Menger's challenge.

The Bellman-Held-Karp algorithm adopts Bellman's general tool of dynamic programming [4], building an optimal tour from shortest paths through each of the 2^n subsets of points. Both teams were well aware of the asymptotic limitations of the algorithm. In a section titled "Computational Feasibility", Bellman gives estimates of the memory requirement (which also grows exponentially with n) for 11, 17, and 21 cities. He goes on to write the following passage [5].

> It follows that the case of 11 cities can be treated routinely, that 17 cities requires the largest of current fast memory computers, but that problems involving 21 cities are a few years at least beyond our reach. One can improve upon these numbers by taking advantage of the fact that the distances will be integers and that we need not use all the digits of one word to specify a distance, but this requires some fancy programming.

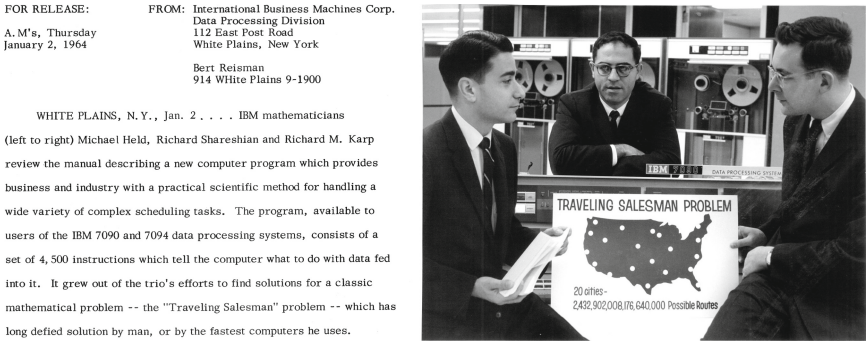
This is an interesting display of mathematical analysis with real-world computing in mind.

Held and Karp take this a step further, giving explicit computational results on IBM hardware: "An IBM 7090 program can solve any 13-city traveling-salesman problem in 17 seconds." And they describe how the exact algorithm can be used as a tool for high-quality solutions for larger instances [52].

> It is characteristic of the algorithms under discussion that their complexity, measured by numbers of arithmetic operations and storage requirements, grows quite rapidly. They are, however, a vast improvement over complete enumeration, and permit the rapid solution of problems of moderate size. In this section we show how the algorithms can be combined with a method of successive approximations to provide a systematic procedure for treating large problems. This procedure yields a sequence of permutations, each obtained from its predecessor by the solution of a derived subproblem of moderate size having the same structure as the given problem. The

associated costs form a monotone nonincreasing sequence which may not converge to the optimum solution; however, computer experimentation has yielded excellent results in a variety of cases.

Again, a fantastic combination of theory and practice. Their computer code, developed together with Richard Shareshian, was made available to users of IBM's hardware; an image of the 1964 press release is displayed in Fig. 1.



FOR RELEASE:
A. M's, Thursday
January 2, 1964

FROM: International Business Machines Corp.
Data Processing Division
112 East Post Road
White Plains, New York

Bert Reisman
914 WHite Plains 9-1900

WHITE PLAINS, N.Y., Jan. 2 . . . . IBM mathematicians (left to right) Michael Held, Richard Shareshian and Richard M. Karp review the manual describing a new computer program which provides business and industry with a practical scientific method for handling a wide variety of complex scheduling tasks. The program, available to users of the IBM 7090 and 7094 data processing systems, consists of a set of 4,500 instructions which tell the computer what to do with data fed into it. It grew out of the trio's efforts to find solutions for a classic mathematical problem -- the "Traveling Salesman" problem -- which has long defied solution by man, or by the fastest computers he uses.

Fig. 1. Michael Held, Richard Shareshian and Richard Karp, 1964. Courtesy of IBM Corporate Archives.

2 Dantzig, Linear Programming, and Cutting Planes

We mentioned that Julia Robinson was a post-doc at the RAND Corporation, a think tank for the United States government. Another member of the RAND group was the remarkable George Dantzig. His name is forever associated with his life's work: the creation of the linear programming model, the simplex method for its solution, and its application to problems far and wide. Grötschel [44] gave the following powerful summary.

> The development of linear programming is—in my opinion—the most important contribution of the mathematics of the 20th century to the solution of practical problems arising in industry and commerce.

This is from the operations research perspective, but Dantzig's LP model was also a bombshell for the general theory and practice of computing in combinatorial optimization.

Linear programming was introduced to the world in a lecture given by Dantzig on September 9, 1948, at an economics meeting at the University of Wisconsin in Madison [18].

> The basic assumptions of the model lead to a fundamental set of linear equations expressing the conditions which much be satisfied by the various levels of activity, X_i, in the dynamic system. These variables are subject

> to the further restriction $X_i \geq 0$. The determination of the "best" choice of X_i is made to depend on the maximization (or minimization) of linear form in X_i. ... It is proposed that computational techniques developed by J. von Neumann and by the author be used in connection with large scale digital computers to implement the solution of programming problems.

This last point, concerning digital computers, is important. Dantzig was working to get linear programming at the head of the queue for implementation on the first generation of electronic hardware. Computation pioneers Hoffman and Wolfe [57] write the following.

> Linear programming was an important force in sponsoring the early UNIVAC computers, as well as the SEAC at the National Bureau of Standards, because of U.S. Air Force funding in support of computations required by it and other planning tools developed by Dantzig and his associates in the Office of the Air Controller.

The computers were lined up and ready to go. And, indeed, extensive LP tests were made as early as 1952 [56,79].

The general LP model is to optimize a linear function $c^T x$ subject to constraints $Ax = b, x \geq 0$, where matrix A, vector b, and vector c are data and $x = (x_1, x_2, \ldots, x_n)$ is a vector of variables. Note that each x_i can be assigned possibly a fractional value. This makes the connection to combinatorial optimization subtle, since combinatorial objects, such as paths, correspond to integer-valued solutions. Indeed, most often in combinatorial models we have a logical choice for each variable, either we use it or we do not, $x_i = 1$ or $x_i = 0$.

In some basic cases, like the assignment problem, it can be shown there exists always an optimal LP solution that is integer valued. Geometrically, this means all vertices of the polyhedral set $\{x : Ax = b, x \geq 0\}$ have integer components. There are beautiful theorems in combinatorics that can be described in this way, but, for optimization models, such naturally-integer examples are the exception, rather than the rule.

The TSP is not one of the exceptional cases. That said, it is also true that any combinatorial problem is, in principle, an LP problem. Take the example of the salesman. A tour through n cities selects n direct point-to-point links, that is, n edges in the complete graph on n points. So a tour can be specified as a 0-1 vector indexed by the edges of the complete graph, where a 1 means we include the edge in the tour. The TSP is to minimize the total travel distance, which is a linear function, over these $(n-1)!$ vectors. That sounds unpleasant, but, for any finite set of vectors S, a classic result of Minkowski states that there exists a convex polytope P such that S is precisely the set of vertices of P, that is, P is the convex hull of S. For our set of tour vectors, by taking the linear constraints that describe the corresponding polytope P, we have formulated the TSP as an LP problem.

Cheers for Minkowski, but actually solving the LP problem could be difficult. Indeed, following up on Robinson's work, Heller [54] and Kuhn [68] presented results showing the number of linear constraints needed to describe the TSP

polytope grows exponentially fast with the number of cities. This is not a trivial statement. For example, there are $n!$ solutions to the n-dimensional assignment problem and yet its corresponding polytope is described by only $2n + 2n^2$ constraints. But not so for the TSP.

Dantzig brushed aside this concern. He was willing to gamble that only a small number of these exponentially many constraints would be needed to solve a target instance that was making the rounds of the US mathematics community, namely to visit one city chosen in each of the 48 states plus Washington, DC.[1] He and his RAND colleagues Ray Fulkerson and Selmer Johnson settled the challenge, finding a tour through the 49 cities together with an LP proof that it was the shortest possible.

The approach they invented in this TSP work is called the *cutting-plane method*. Rather than taking the full polytope as envisioned by Heller and Kuhn, they instead take a simple polytope that includes all of the tours, and refine it step by step, adding in each step constraints that are satisfied by all tours but violated by the optimal solution to the current LP relaxation. At the end of their computation, they have an LP relaxation of the TSP that has as an optimal solution a TSP tour. Game over.

In the research-report version of their paper, Dantzig et al. [19] wrote the following line to start a section titled "The method".

> The title of this section is perhaps pretentious, as we don't have a method in a precise sense.

This is typical of the modest style of writing at the time, but there were indeed certain ad hoc aspects to their work.

1. They did not know explicitly the full set of inequalities that define the TSP polytope for $n = 49$ points. Indeed, even today the full description is known only for $n \leq 9$. So it was possible they could reach an LP relaxation having a non-tour optimal solution and not have any means to refine their relaxation with an additional constraint.
2. Even for the classes of constraints they did know, they did not have available exact algorithms to test whether or not the current LP solution satisfies all member constraints of each class. In the computation, they relied on their own creativity to locate a violated constraint.
3. Despite Dantzig's willingness to place a wager, they had no upper bound on the number of iterations that might be needed to reach an LP optimal solution that was a tour. Indeed, without further restrictions on the constraints to be considered (such as defining facets of the TSP polytope) the process is not guaranteed to be finite.

That is three strikes. Nonetheless, Dantzig's team accomplished the task described in their abstract [20]: "It is shown that a certain tour of 49 cities, one

[1] This is literally true. Dantzig wagered Fulkerson one dollar that at most 25 inequalities would be needed on top of the assignment problem constraints [57].

in each of the 48 states and Washington, DC, has the shortest road distance." A feat no team would surpass until the 1970s.

Interestingly, rather than unleash the new electronic computers that Dantzig secured for LP testing, the TSP team carried out all computations by hand. This was due to the shear size of their model. The 1,176 variables in the 49-city example put it far beyond the capabilities of available hardware and software. In the long run this proved to be a good thing—the specialized techniques they developed to short-cut the computations provided a template for large-scale computing in the following decades.

The computation by Dantzig's team was an amazing accomplishment, but, due to the three strikes, the work made no direct contribution to the asymptotic complexity of the TSP. Here is the concluding paragraph of their paper [20].

> It is clear that we have left unanswered practically any question one might pose of a theoretical nature concerning the traveling-salesman problem; however, we hope that the feasibility of attacking problems involving a moderate number of points has been successfully demonstrated, and that perhaps some of the ideas can be used in problems of similar nature.

That is marvelous in its modesty. The team may not have given Karl Menger a satisfactory answer, but they showed us how to use LP to attack seemingly intractable problems.

The cutting-plane method is far and away the most successful technique for the exact solution of $\mathcal{NP}$-hard models in combinatorial optimization; a nice survey can be found in Jünger et al. [60].

For the TSP itself, it took twenty years for the community to catch up to the by-hand computations of Dantzig's team. But starting in the 1970s, led

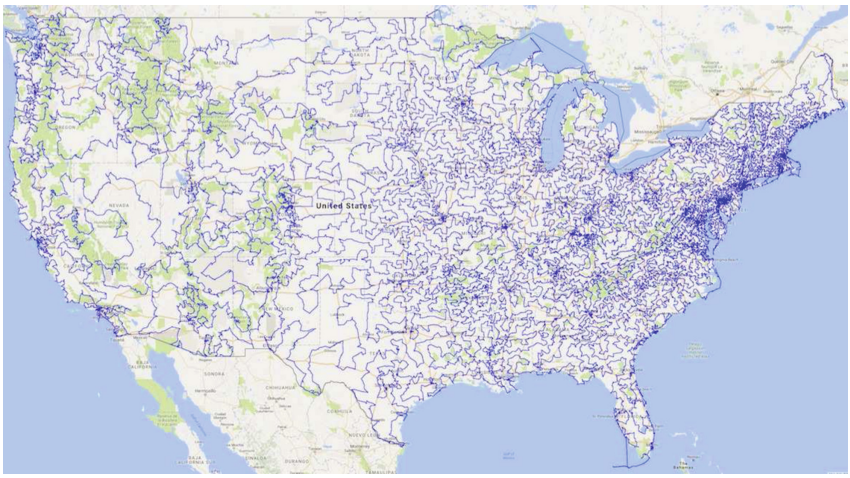


Fig. 2. Optimal walking tour to 49,603 sites from the National Register of Historic Places. Image Google Maps ©.

by Martin Grötschel and Manfred Padberg and aided by increasingly powerful computing platforms, great progress was made in exact methods [1,17,43,45,80]. It is now routine to exactly solve problems with many hundreds of cities.

The most recent study [15] reports the optimal solution of a 49,603-city USA instance with point-to-point distances measured by walking directions provided by Google Maps. I like to think Dantzig, Fulkerson, and Johnson would be proud (Fig. 2).

3 Edmonds, Matchings, and Polynomial Time

Following the success of Bellman, Held and Karp, the stage in the 1960s look set for the jump to an n^k algorithm for the TSP. And the best hope for making such a breakthrough was Jack Edmonds, a mathematician who brought the theory of combinatorial optimization into the modern era.

The 28-year-old Edmonds burst onto the scene in 1961, at a summer workshop held at the RAND Corporation. At the time, he was working for the National Bureau of Standards, having obtained a Master's degree at the University of Maryland. Balinsky, Edmonds, and several other young researchers had been invited to join a who-is-who list of stars from the field of combinatorics. He described his talk as follows [26].

> At this lecture, everybody I'd ever heard of was there. Gomory and Dantzig and Tutte and Fulkerson and Hoffman ... And I gave this grand philosophical speech ... Here is a good algorithm, here is a solved integer program. This was a sermon, this was a real sermon. Here is a solved integer program. It was my first glimpse of heaven.

This was a report on a polynomial-time algorithm for the matching problem in general graphs, a difficult generalization of the assignment problem. Along the way, he made a strong case for the mathematical importance of such *good algorithms*.

Edmonds' paper on these topics was written in 1963 and appeared in journal form [24] in 1965, the same year as Alan Cobham's paper on machine-independent computation [13]. Both authors present the notion of polynomial-time algorithms, but they emphasize different aspects of the theory. In his Turing Award lecture [14], Stephen Cook summarized the work as follows.

> Cobham pointed out that the class was well defined, independent of which computer model was chosen, and gave it a characterization in the spirit of recursive function theory. The idea that polynomial time computability roughly corresponds to tractability was first expressed in print by Edmonds.

It was certainly a great year for computational complexity—the classic paper by Hartmanis and Stearns [51] also appeared in journal form in 1965.

If you are a card-carrying complexity theorist, then you almost certainly turn to Cobham's elegant paper for an explanation of the class $\mathcal{P}$. For a combinatorial optimizer, however, Edmonds is the polynomial-time champion (Fig. 3). His

description is less formal, but he brings the topic to life, showing by example how to put combinatorial problems into $\mathcal{P}$. It is his glimpse of algorithmic heaven.

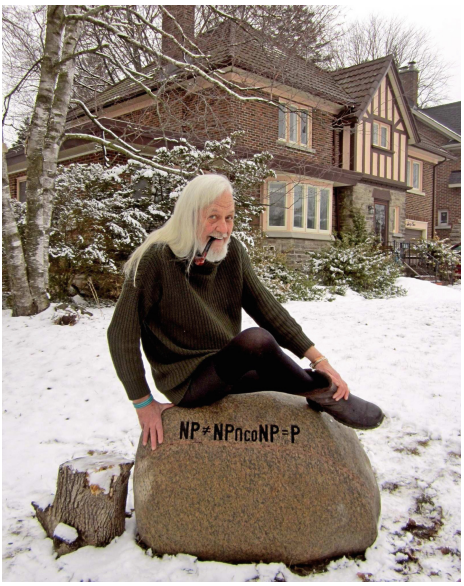


Fig. 3. Jack Edmonds, 2015. Photograph courtesy of Kathie Cameron.

The center piece of Edmonds' paper is his *blossom algorithm* for matching in a general graph. A *matching* is a set of edges that have no common end points. Unlike the assignment problem, the natural LP relaxation for matchings may have optimal solutions that are not integer valued. For this problem, however, Edmonds was able to describe explicitly the full set of linear inequalities that define the polytope promised by Minkowski. He thus obtained an LP model that returns always a matching as an optimal solution. The LP has exponentially many constraints, but he was able to devise a primal-dual algorithm to handle them all in polynomial time. And his algorithm was not just good in theory, but also in practice: careful implementations are able to handle graphs with a million or more points [2,16,66,75].

In his work, Edmonds was able to overcome all three strikes we had against the cutting-plane method. So the big question was whether or not his methodology would also give a means to devise a polynomial-time algorithm for the TSP. Edmonds writes that he did indeed follow up his success with matchings by taking a crack at the salesman problem [27].

> Inspired by Dantzig, Fulkerson, Johnson (1954), I became excited in 1961 to show that TSP is co-$\mathcal{NP}$ by finding an $\mathcal{NP}$ description of a set of

> inequalities whose solution set is the convex hull of all the 0,1 incidence vectors of tours in G. I failed, and so in 1966 I conjectured that TSP is not in $\mathcal{P}$.

Sadly, he became convinced we will never see an n^k algorithm for the TSP. He stated this explicitly in a paper [28] published in 1967: "I conjecture that there is no good algorithm for the traveling salesman problem." Bad luck for all fans of the TSP. And the news got even worse when both the directed and undirected versions of the TSP appeared on Karp's famous list [62] of 21 $\mathcal{NP}$-hard problems in 1972.

So the fate of the TSP is now tied to the great $\mathcal{P}$ versus $\mathcal{NP}$ question. But, along the way, Edmonds provided an important insight into possible solution methods. He made the following remark during a discussion that took place after a TSP lecture by Gomory [42] in 1964.

> For the traveling salesman problem, the vertices of the associated polyhedron have a simple characterization despite their number—so might the bounding inequalities have a simple characterization despite their number. At least we should hope they have, because finding a really good traveling salesman algorithm is undoubtedly equivalent to finding such a characterization.

Edmonds' thesis was that polynomial-time algorithms go hand-in-hand with polyhedral characterizations.

And he was right. Some twenty years later, Edmonds' thesis was proven in a deep result by Grötschel et al. [49]. The theorem is based on the ellipsoid algorithm for linear programming and it goes by the slogan *optimization* $\equiv$ *separation*. Roughly speaking, if we can solve an optimization problem in polynomial time, then we have an implicit description of the corresponding polytope, and, the other way around, if we understand the polytope then we have a polynomial-time algorithm for the optimization problem.

Beautiful! The new work brought together combinatorial-optimization theory, practice, algorithms, and complexity, all united via linear programming.

4 Sixty-Three Years of Progress

Let's take a step back to fill in other research highlights, before jumping ahead to current topics. In this quick survey, we start the combinatorial-optimization clock ticking with the 1954 publication of the Dantzig-Fulkerson-Johnson paper on the TSP.

Despite the ringing success of their cutting-plane method, the research carried out by Dantzig's team did not have an immediate impact on the computational side of the field. They were simply too far ahead of their time, both in the sophisticated use of LP theory and in their ability to perform by-hand calculations that were out of reach for existing computing platforms.

But the clock did not stand still in the 1950s. Led by Hoffman's min-max theorems [55], Ford and Fulkerson's network-flow algorithms [33], Gomory's cutting planes [41], and Eastman [22] and Land and Doig's [69] branch-and-bound method, the overall field advanced quickly. This work, particularly by Hoffman, showed further the central role that was to be played by linear programming.

The following decade was dominated by Edmonds on the algorithmic side, including polynomial-time results for matchings [23], matroid intersection [25], optimal branchings [28], Gaussian elimination [29] and, together with Karp, maximum flow and minimum-cost flow [30]. On the computational side, Lin and Kernighan's introduction of powerful heuristic methods for graph partitioning [64] and the TSP [72,73] established the study of heuristic search as an important and sophisticated component of combinatorial optimization.

The 1970s saw the introduction of the formal study of α-approximation algorithms by Johnson [58], where he considers polynomial-time methods that produce solutions guaranteed to have value no greater than α times that of an optimal solution. His paper was quickly followed by Christofides' 1.5-approximation for the TSP [11], and the sub-discipline was off and running. A highlight here is the spectacular result on approximating the maximum-cut problem by Goemans and Williamson [39] in 1995, where they use semi-definite programming to move beyond what can be obtained with the natural LP relaxation. Texts by Vazirani [88] and Williamson and Shmoys [89] provide great coverage of the area.

We have already discussed the iconic result of the 1980s, the decade of optimization $\equiv$ separation. But these years also saw, on the applied/computational side, the start of a great expansion of the use of the cutting-plane method. This work goes well beyond the confines of the TSP, led by successful projects by Grötschel et al. [46–48] and others. A huge boost to this computational area was the arrival of robust LP solvers that could be called as functions from within a cutting-plane code, in particular the CPLEX library created by Bixby [6].

Moving to the 1990s, a major development was the introduction of new hierarchical relaxations of combinatorial optimization models by Lovász and Schrijver [74], Sherali and Adams [85], and Lasserre [70]. The template for this line of work was the classic paper on cutting planes by Chvátal [12], where he shows that the convex hull associated with any combinatorial problem can be obtained by iteratively applying a simple rounding process. The new procedures expand on this idea, by considering the problem in higher-dimensional spaces (obtained by adding variables to the initial relaxation) where it can be easier to enforce integrality conditions.

I've been skipping over important work by a host of researchers, so with the field getting ever broader and more active, I prefer not to try for a two-line summary of the 2000s. But let me point out a clear highlight, the publication of Schrijver's monograph *Combinatorial Optimization: Polyhedra and Efficiency* [84]. His work, published in 2003, covers 1881 pages in three volumes and includes over 4,000 references. Schrijver's scholarly writing is amazing. And to bring you forward from 2003, there are excellent recent texts by Frank [35] and Korte and Vygen [67]. If you want to study the first sixty-three years of combinatorial optimization, these three books contain all the material you need.

5 Wish List of Research Directions

A good sign of the health of a field of study is the ease in which it is possible to list future research directions. Combinatorial optimization looks great under this metric. I could have started with any of the seventy-five open problems and conjectures listed in Schrijver's monograph [84]. But I decided to go with only a short wish list of five topics, all aimed at pushing forward the intersection of computation, algorithms, and theory.

5.1 Improving the Simplex Method

Perhaps the most important contribution the computer-science algorithms community could make to the field of operations research would be the delivery of ideas to improve the practical performance of the simplex method for solving linear programming problems. The simplex method was named one of the top ten algorithms of the century [21], but it needs help to continue to drive progress in OR applications. The steepest-edge pivot rules that are the state of the art in simplex implementations date back to the 1970s [40], and the best known methods for implementing these rules go back to the early 1990s [34]. Moreover, there currently are no implementations of the simplex method that make effective use of multi-core processors, GPUs, or other parallel computing platforms.

For solving any single large-scale LP model, the simplex method has a serious competitor in the class of polynomial-time algorithms called interior-point methods, that parallelize nicely. But for solving a sequence of closely related LP models, such as those that arise in the cutting-plane method or in a branch-and-bound search, the simplex method is the only game in town. This comes from the fact that the simplex method can be set up to start its search at the optimal solution to the previously solved model, dramatically decreasing the number of steps needed to reach an optimal solution to the new model. Interior-point methods are not able to do this effectively.

Here is the context. Over a broad class of models and considering the commercial software libraries CPLEX 1.0 to 11.0 and Gurobi 1.0 to 7.0, Bixby [7] reports a machine-independent speed-up in mixed-integer programming (MIP) codes of a factor over 1.3 million in the past 25 years, roughly a factor of 1.75 each year. That is amazing progress in practical computing and it has been a driving force in the growth of successful OR applications. But Bixby also reports that, with continued increases in model sizes, linear programming has become a roadblock towards solutions in a substantial fraction of MIP settings. Help is needed.

The big ticket item, from the CS theory side, is the fact that there is no known polynomial-time variant of the simplex method. Avis [3] sums things up nicely in a short note marking the 100th anniversary of the birth of George Dantzig.

> Surely the close collaboration of TCS and the optimization community would be able to settle this question: is there or is there not a polynomial

> time pivot selection rule for the simplex method? Of course I think all of us, including George, hope for a positive answer that is both strongly polynomial time and a winner in practice!

Settling this question may be difficult, but there has been progress in understanding the complexity of Dantzig's algorithm, including a long line of award winning papers from the OR, CS, optimization, and mathematics communities.

- Borgwardt [8], average-case analysis of the simplex method, Lanchester Prize 1982 (INFORMS).
- Kalai [61], quasi-polynomial bound on diameter of polyhedra, Fulkerson Prize 1994 (MOS and AMS).
- Spielman and Teng [86], smoothed analysis of the simplex method, Gödel Prize 2008 (ACM) and Fulkerson Prize 2009 (MOS and AMS).
- Friedmann [36], super-polynomial lower bounds for simplex pivot rules, Tucker Prize 2012 (MOS).
- Santos [83], counterexample to the Hirsh conjecture, Fulkerson Prize 2015 (MOS and AMS).

Each of these results concerns analysis of the simplex method or the paths it takes along the edges of polyhedra. What I'd like to emphasize is the need for turning analytical insights, such as these, into recommendations for improving the simplex method in practice.

For example, can machine-learning techniques be used to build pivot-selection rules that adapt to the properties of the input polyhedra? In this area, every saved pivot helps.

5.2 Language of Algorithms

I'd like to draw attention to a research direction that is more of a dream than a specific problem. Namely, the development and adoption of a more nuanced way of expressing accomplishments in the analysis of algorithms.

Avis [3] gives a dramatic example. Suppose history was reversed and Dantzig announced a polynomial-time interior-point algorithm in his 1948 talk in Madison. Would a paper today on a newly invented, exponential-time simplex method have a chance of being accepted into a major conference? Would it even be coded for testing? One might hope the operations research community would handle the task, but it takes an exceptional amount of care to implement the simplex method in a way that makes it competitive with interior-point codes. Without guidance from the algorithms community, it seems unlikely the necessary time and energy would be devoted to building the expertise needed to bring the simplex method into practical use.

I do not mean to criticize the focus on asymptotic analysis and polynomial-time results. Indeed, this focus drives CS theory, and CS theory has long supplied a lifeline of techniques for everyone in the business of attacking large-scale optimization models. But this focus, together with the use of big-oh notation and the hiding of logarithmic factors, can sometimes make it difficult to tap into ideas

that can have a major impact in computational studies. And, more importantly, it may sometimes hinder the creation of techniques that could have dramatic impact on computational practice, such as the simplex method in Avis's fable.

5.3 Understanding Heuristic Algorithms

The term *heuristic* is sometimes used to describe both non-exact techniques, such as simulated annealing, as well as exact, but exponential-time, techniques, such as the cutting-plane method. In my discussion I refer only to the first meaning. That is, a *heuristic algorithm* is one designed to find a hopefully good solution, but comes without a performance guarantee.

Heuristic algorithms are widely used in operations research and many other areas. They are used, but not understood. The success of these techniques far outstrips our ability to explain and evaluate analytically.

In his 2010 Knuth Prize lecture [59], David Johnson discussed the theme of how best to increase the impact of theoretical computer science research. His first rule was "Study problems people might actually want to solve" and his top open question in this regard was the following.

> When (and why) do metaheuristc approaches work well?

Just so. Computer science is the research community best equipped with analytic tools to address this issue.

And Johnson is not the only giant of computer science to bring up this topic. Richard Karp, in a lecture given at Harvard University in 2011 [63], made the following statement.

> Heuristics are often "unreasonably effective," for reasons not well understood.

This is certainly the case. Recall that I mentioned the report of an optimal TSP tour to visit 49,603 sites with distance measured by Google walking routes. To solve this instance, the total computing time for the cutting-plane method and branch-and-bound search was 178.9 years (on a network of processors). That is a great deal of computing power, but it turned out that the heuristic tour we had at the start of the search was in fact optimal. All of the computation was to verify there was no better solution. It is definitely unreasonable that a combination of local search and genetic algorithms was able to produce an optimal solution for such a complex optimization problem. Unreasonable and unexplained.

5.4 Analysis of Exact Algorithms for Hard Problems

Facing an $\mathcal{NP}$-hard optimization problem, the main targets for study are approximation methods and computational methods, combining heuristic search with lower-bound techniques like cutting planes. A third option is the study of exponential-time exact solution algorithms. Research in this direction can provide effective means to handle small problem instances (and these can in turn

be used to solve larger examples, using techniques such as the local-cuts procedure described in [1]), as well as providing insights that can be adopted in branch-and-bound and other computational methods.

A nice survey of this area, together with a list of open problems, is given in Woeginger [90]. Prominent among these is the challenge of improving the Bellman-Held-Karp $n^2 2^n$ bound for solving an n-city TSP, possibly replacing the exponential term by c^n for a constant $c < 2$. It has been 55 years since the publication of the BHK algorithm, but there has been no improvement for general instances. This is likely one of the longest-standing, non-trivial, complexity bounds for any combinatorial model.

5.5 Complexity of Cutting-Plane Methods

The cutting-plane method is a well-established computational technique, with successful application to a broad range of combinatorial models. As such, it is a good target for investigation from a computer science theory perspective. Possible topics include examining bounds on the complexity of the overall method, investigating algorithms for separation problems to deliver cutting planes for particular models, and obtaining insights into the selection of cutting planes to speed the convergence of the process.

A nice result of the first type was given recently by Chandrasekaran et al. [10], establishing a polynomial-time cutting-plane method for the matching problem. A direct challenge here would be to establish a similar bound for the subtour relaxation of the TSP.

For the second type of problem, there are interesting results for the separation of TSP inequalities by Carr [9], Fleischer and Tardos [31], Letchford [71] and others. But, even for this intensely studied model, there are far more open questions than results. For example, the comb-separation problem (the most basic question for TSP inequalities) is not known to be $\mathcal{NP}$-hard and also not known to be polynomial-time solvable. It would be interesting to see non-trivial approximation results in this area.

The third type of problem, the selection of cutting planes, is critical in practice. It would be great to see analysis for well-known models, such at the TSP, the maximum-cut problem, and the maximum stable-set problem. A nice paper here is an initial study of TSP inequalities by Goemans [38]. His results are worst-case comparisons, but they predict well the performance seen in computational studies (see [1, p. 524]).

References

1. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton (2006)
2. Applegate, D.L., Cook, W.: Solving large-scale matching problems. In: Johnson, D.S., McGeoch, C.C. (eds.) Algorithms for Network Flows and Matching. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 12, pp. 557–576. American Mathematical Society, Providence (1993)

3. Avis, D.: George Dantzig: father of the simplex method. Bull. EATCS **116** (2015)
4. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton (1957)
5. Bellman, R.: Dynamic programming treatment of the travelling salesman problem. J. ACM **9**, 61–63 (1962)
6. Bixby, R.: You have to figure out who your customer is going to be. Optima **101**, 1–6 (2016)
7. Bixby, R.: A saga of 25 years of progress in optimization. Lecture at the University of Tokyo, 1 December 2016
8. Borgwardt, K.-H.: Some distribution-independent results about the asymptotic order of the average number of pivot steps of the simplex-method. Math. Oper. Res. **7**, 441–462 (1983)
9. Carr, R.: Separating clique trees and bipartition inequalities having a fixed number of handles and teeth in polynomial time. Math. Oper. Res. **22**, 257–265 (1997)
10. Chandrasekaran, K., Végh, L.A., Vempala, S.S.: The cutting plane method is polynomial for perfect matchings. Math. Oper. Res. **41**, 23–48 (2015)
11. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Report no. 388, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh (1976)
12. Chvátal, V.: Edmonds polytopes and a hierarchy of combinatorial problems. Discret. Math. **4**, 305–337 (1973)
13. Cobham, A.: The intrinsic computational difficulty of functions. In: Proceedings of the 1964 International Congress for Logic, Methodology, and Philosophy of Sciences, pp. 24–30. North Holland, Amsterdam (1965)
14. Cook, S.A.: An overview of computational complexity. Commun. ACM **26**, 401–408 (1983)
15. Cook, W., Espinoza, D., Goycoolea, M., Helsgaun, K.: US50K (2016). http://www.math.uwaterloo.ca/tsp/us/index.html
16. Cook, W., Rohe, A.: Computing minimum-weight perfect matchings. INFORMS J. Comput. **11**, 138–148 (1999)
17. Crowder, H., Padberg, M.W.: Solving large-scale symmetric travelling salesman problems to optimality. Manag. Sci. **26**, 495–509 (1980)
18. Dantzig, G.: Programming in a linear structure. Econometrica **17**, 73–74 (1948)
19. Dantzig, G., Fulkerson, R., Johnson, S.: Solution of a large scale traveling salesman problem. Technical report P-510. RAND Corporation, Santa Monica (1954)
20. Dantzig, G., Fulkerson, R., Johnson, S.: Solution of a large-scale traveling-salesman problem. Oper. Res. **2**, 393–410 (1954)
21. Dongara, J., Sullivan, F.: The top 10 algorithms. IEEE Comput. Sci. Eng. **2**, 22–23 (2000)
22. Eastman, W.L.: Linear programming with pattern constraints. Ph.D. thesis, Department of Economics, Harvard University, Cambridge (1958)
23. Edmonds, J.: Maximum matching and a polyhedron with 0,1-vertices. J. Res. Nat. Bur. Stan. **69B**, 125–130 (1965)
24. Edmonds, J.: Paths, trees, and flowers. Can. J. Math. **17**, 449–467 (1965)
25. Edmonds, J.: Submodular functions, matroids, and certain polyhedra. In: Guy, R., Hanani, H., Sauer, N., Schönheim, J. (eds.) Combinatorial Structures and Their Applications, pp. 69–87. Gordon and Breach, New York (1970)
26. Edmonds, J.: A glimpse of heaven. In: Lenstra, J.K., et al. (eds.) History of Mathematical Programming-A Collection of Personal Reminiscences, pp. 32–54. North-Holland, Amsterdam (1991)

27. Edmonds, J.: EP and PPA: can it be hard to find if it's easy to recognize and you know it's there? Lecture at the 21st Combinatorial Optimization Workshop, Aussois, France, 13 January 2017
28. Edmonds, J.: Optimum branchings. J. Res. Nat. Bur. Stan. **71B**, 233–240 (1967)
29. Edmonds, J.: Systems of distinct representatives and linear algebra. J. Res. Nat. Bur. Stan. **71B**, 241–245 (1967)
30. Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. J. ACM **19**, 248–264 (1972)
31. Fleischer, L., Tardos, É.: Separating maximally violated comb inequalities in planar graphs. Math. Oper. Res. **24**, 130–148 (1999)
32. Flood, M.M.: The traveling-salesman problem. Oper. Res. **4**, 61–75 (1956)
33. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press, Princeton (1962)
34. Forrest, J.J., Goldfarb, D.: Steepest-edge simplex algorithms for linear programming. Math. Program. **57**, 341–274 (1992)
35. Frank, A.: Connections in Combinatorial Optimization. Oxford University Press, Oxford (2011)
36. Friedmann, O.: Exponential lower bounds for solving infinitary payoff games and linear programs. Ph.D. thesis, Ludwig-Maximilians-Universität München (2011)
37. Gödel, K.: Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I. Monats-hefte für Mathematik und Physik **38**, 173–198 (1931)
38. Goemans, M.: Worst-case comparison of valid inequalities for the TSP. Math. Program. **69**, 335–349 (1995)
39. Goemans, M., Williamson, D.: Improved approximation algorithms for maximum cut and satisfiability problems. J. ACM **42**, 1115–1145 (1995)
40. Goldfarb, D., Reid, J.K.: A practicable steepest-edge simplex algorithm. Math. Program. **1**, 361–371 (1977)
41. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. Bull. Am. Math. Soc. **64**, 275–278 (1958)
42. Gomory, R.E.: The traveling salesman problem. In: Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems, pp. 93–121. IBM, White Plains (1996)
43. Grötschel, M.: On the symmetric travelling salesman problem: solution of a 120-city problem. Math. Program. Study **12**, 61–77 (1980)
44. Grötschel, M.: Notes for a Berlin Mathematical School (2006)
45. Grötschel, M., Holland, O.: Solution of large-scale symmetric travelling salesman problems. Math. Program. **51**, 141–202 (1991)
46. Grötschel, M., Jünger, M., Reinelt, G.: A cutting plane algorithm for the linear ordering problem. Oper. Res. **32**, 1195–1220 (1984)
47. Grötschel, M., Jünger, M., Reinelt, G.: On the acyclic subgraph polytope. Math. Program. **33**, 28–42 (1985)
48. Grötschel, M., Jünger, M., Reinelt, G.: Facets of the linear ordering polytope. Math. Program. **33**, 43–60 (1985)
49. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Berlin (1988). https://doi.org/10.1007/978-3-642-97881-4
50. Grötschel, M., Nemhauser, G.L.: George Dantzig's contributions to integer programming. Discret. Optim. **5**, 168–173 (2008)
51. Hartmanis, J., Stearns, R.E.: On the computational complexity of algorithms. Trans. Am. Math. Soc. **117**, 285–306 (1965)

52. Held, M., Karp, R.M.: A dynamic programming approach to sequencing problems. J. Soc. Ind. Appl. Math. **10**, 196–210 (1962)
53. Held, M., Karp, R.M.: The traveling-salesman problem and minimum spanning trees: Part II. Math. Program. **1**, 6–25 (1971)
54. Heller, I.: On the problem of the shortest path between points. I. Abstract 664t. Bull. Am. Math. Soc. **59**, 551 (1953)
55. Hoffman, A.J.: Generalization of a theorem of Konig. J. Wash. Acad. Sci. **46**, 211–212 (1956)
56. Hoffman, A., Mannos, M., Sokolowsky, D., Wiegmann, N.: Computational experience in solving linear programs. J. Soc. Ind. Appl. Math. **1**, 17–33 (1953)
57. Hoffman, A.J., Wolfe, P.: History. In: Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (eds.) The Traveling Salesman Problem, pp. 1–15. Wiley, Chichester (1985)
58. Johnson, D.S.: Approximation algorithms for combinatorial problems. J. Comput. Syst. Sci. **9**, 256–278 (1974)
59. Johnson, D.S.: Knuth Prize Lecture (2010)
60. Jünger, M., Reinelt, G., Thienel, S.: Practical problem solving with cutting plane algorithms. In: Cook, W., Lovász, L., Seymour, P. (eds.) Combinatorial Optimization. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 20, pp. 111–152. American Mathematical Society, Providence (1995)
61. Kalai, G.: Upper bounds for the diameter and height of graphs of the convex polyhedra. Discret. Comput. Geom. **8**, 363–372 (1992)
62. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press, New York (1972)
63. Karp, R.M.: Implicit hitting set problems. Lecture at Harvard University, 29 August 2011
64. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. **49**, 291–307 (1970)
65. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
66. Kolmogorov, V.: Blossom V: a new implementation of a minimum cost perfect matching algorithm. Math. Program. Comput. **1**, 43–67 (2009)
67. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms. Springer, Berlin (2012). https://doi.org/10.1007/978-3-642-24488-9
68. Kuhn, H.W.: On certain convex polyhedra. Abstract 799t. Bull. Am. Math. Soc. **61**, 557–558 (1955)
69. Land, A.H., Doig, A.G.: An automatic method of solving discrete programming problems. Econometrica **28**, 497–520 (1960)
70. Lasserre, J.B.: Global optimization with polynomials and the problem of moments. SIAM J. Optim. **11**, 796–817 (2001)
71. Letchford, A.N.: Separating a superclass of comb inequalities in planar graphs. Math. Oper. Res. **25**, 443–454 (2000)
72. Lin, S.: Computer solutions of the traveling salesman problem. Bell Syst. Tech. J. **44**, 2245–2269 (1965)
73. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. **21**, 498–516 (1973)
74. Lovász, L., Schrijver, A.: Cones of matrices and set-functions, and 0–1 optimization. SIAM J. Optim. **1**, 166–190 (1991)

75. Mehlhorn, K., Schäfer, G.: Implementation of $O(nm \log n)$ weighted matchings in general graphs: the power of data structures. J. Exp. Algorithmics **7**, Article 4 (2002)
76. Menger, K.: Bericht über ein mathematisches Kolloquium. Monats-hefte für Mathematik und Physik **38**, 17–38 (1931)
77. Munkres, J.: Algorithms for the assignment and transportation problems. J. Soc. Ind. Appl. Math. **5**, 32–38 (1957)
78. von Neumann, J.: A certain zero-sum two-person game equivalent to the optimal assignment problem. In: Kuhn, H.W., Tucker, A.W. (eds.) Contributions to the Theory of Games, pp. 5–12. Princeton University Press, Princeton (1953). (Transcript of a seminar talk given by Professor von Neumann at Princeton University, 26 October 1951)
79. Orden, A.: Solution of systems of linear inequalities on a digital computer. In: Proceedings of the 1952 ACM National Meeting, pp. 91–95 (1952)
80. Padberg, M., Rinaldi, G.: A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Rev. **33**, 60–100 (1991)
81. Robinson, J.: Definability and decision problems in arithmetic. Ph.D. thesis, University of California, Berkeley (1948)
82. Robinson, J.: On the Hamiltonian game (a traveling salesman problem). RAND Research Memorandum RM-303. RAND Corporation, Santa Monica (1949)
83. Santos, F.: A counterexample to the Hirsh conjecture. Ann. Math. **176**, 383–412 (2011)
84. Schrijver, A.: Combinatorial Optimization. Springer, Berlin (2003)
85. Sherali, H.D., Adams, W.P.: A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Springer, Berlin (2013). https://doi.org/10.1007/978-1-4757-4388-3
86. Spielman, D.A., Teng, S.-H.: Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. J. ACM **51**, 385–463 (2004)
87. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proc. London Math. Soc. **s2-42**, 230–265 (1937)
88. Vazirani, V.V.: Approximation Algorithms. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-662-04565-7
89. Williamson, D.P., Shmoys, D.B.: The Design of Approximation Algorithms. Cambridge University Press, Cambridge (2011)
90. Woeginger, G.J.: Exact algorithms for NP-hard problems: a survey. In: Jünger, M., Reinelt, G., Rinaldi, G. (eds.) Combinatorial Optimization — Eureka, You Shrink!. LNCS, vol. 2570, pp. 185–207. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36478-1_17