Ulysses 2000: In Search of Optimal Solutions to Hard Combinatorial Problems

by

Martin Grötschel* and Manfred Padberg§

November 1993

---

* Konrad-Zuse Zentrum für Informationstechnik and Technische Universität, Berlin.

§ New York University, New York. Supported in part by grants from the Office of Naval Research (N00014-90-J-1324) and the Air Force Office of Scientific Research (F49620-90-C-0022).

Abstract

Combinatorial optimization problems pervade many areas of human problem solving especially in the fields of business, economics and engineering. Intensive mathematical research and vast increases in raw computing power have advanced the state of the art in exact and heuristic problem solving at a pace that is unprecedented in human history. We survey here in layman's terms some of the fundamental concepts and principles that have led this progress. (This article will appear – possibly in modified form – in a popular science magazine.)

Prologue Homer's Odyssey a travel guide? Indeed! Already Heinrich Schliemann(1820-1890) has interpreted the classic epic this way; and with Homer's detailed description of the Trojan war in his pocket, he discovered the remains of Troy and Mycenae. So why not view the whole story as a travel guide of Phoenician times to wonders of the Mediterranean such as the 16 places depicted in Figure 1a. Among them are contemporary tourist attractions like the rock of Gibraltar(11), the straits of Messina(13), the volcano of Stromboli(12), the Tunisian island of Djerba(5) and, of course, the ruins of Troy(2). Figure 1a not only shows sixteen Mediterranean tourist spots but also the order by which Ulysses presumably traveled some 3,000 years ago. His journey does evidently not correspond to the image that one has of a well-planned, organized tour – which according to Homer it was not. In fact, his extended wandering and many adventures have led to the use of the word "odyssey" as a synonym for a long (and eventful) trip. So what does Ulysses have to do with modern applied mathematics? Not much; but not long ago one of the authors sat in a cozy Paris restaurant and explained to his loving companion why mathematics and computers have fascinated him for the better part of his adult life. Of course, the couple talked about the *traveling salesman problem* and here is a one-line description of this notoriously difficult problem: Given n cities and distances between them, find a *roundtrip of minimum length* that visits every city exactly once. "But this is what Ulysses should have done after the Trojan war," remarked his companion, drawing an unexpected analogy to the classic epic, "he would have been back to his Penelope so much earlier." True. Mankind, however, might have been deprived of a cherished tale – and the authors would have had the trouble of looking for a different *leitmotif* for the story they are about to tell.

Introduction

In this article we report on the theoretical underpinnings of the enormous progress that has been made in the past 15 years in applying discrete mathematics to a multitude of economic, industrial and engineering problems. Before going into a finer description of some of these practical problems we will outline the issues that are involved in bringing a modern day Ulysses home to his beloved Penelope as quickly as possible.

Suppose that Ulysses is a top manager of our time and call him the *modern Ulysses* or *Ulysses 2000*. This Ulysses wants to visit the 16 tourist spots without wasting any travel time. Of course, he has a helicopter at his disposal. To plan the trip he needs, first and foremost, some quantitative data. Table 1 lists the polar coordinates of the 16 locations on the globe that he wishes to visit. Given the coordinates he can calculate the distance between any two points along the earth's great circles which is roughly proportional to the minimum flying time between two points. The distances, rounded to the nearest kilometer, are listed in Table 2. On the basis of this information he knows that Homer's Ulysses must have traveled more than 9,913 km as he followed the whims of the sea's currents. Now comes the manager's real problem. Just *how* should *he* travel? After all, time is money. Evidently, Homer's Ulysses wasted much time by crisscrossing the Mediterranean – against his own will according to Homer. So Ulysses 2000 decides to do like the traveling salesman: he opts for a roundtrip.

It is not difficult to come up with "possible" roundtrips; see Figures 5a, 5b, 5c, 5d below. But how many are there? We show in the box "Counting" how the number of roundtrips can be computed. It turns out that Ulysses 2000 has exactly 653,837,184,000 distinct choices. We have generated *all* of these roundtrips and computed their lengths. To do so took us 92 hours on a powerful 28 MIPS workstation, i.e., almost 4 days. Wow! The shortest tour has length 6,859 km and is shown in Figure 1b.

The classical mathematical approach to optimization problems involving *finitely* many choices or, in technical jargon, to *combinatorial optimization problems*, has always been the brute force enumeration technique described above. Indeed, this simplistic approach to "solving" finite choice problems still lingers on, even in some (dusty) mathematical minds. However, problems arising in industrial and engineering applications are far too complex for this kind of an approach because they are much "bigger" than Ulysses' problem. One of the reasons why we have chosen Ulysses' odyssey as an illustration is the fact that 16 cities is just about the limit to size that existing computers can handle by enumeration. The small increase to 25 cities puts the problem's solution out of reach for even more clever forms of enumeration. Indeed, a supercomputer of the next generation, such as the much talked about teraflop machine, will never permit us to *enumerate* all roundtrips in its lifetime.

On the research front of combinatorial optimization, however, practical problems coming from industry and having over 2,000 cities have reportedly been solved to optimality, and recently the optimization of a problem having 4,461 cities has been announced! Problems of this and much larger size arise daily, e.g., in the production of printed circuit boards (PCBs), in VLSI design, X-ray crystallography and other areas.

In Figure 2a we show a rectangular piece of material into which 2,392 holes must be drilled (by means of a mechanical drill or a laser gun). The drilling is but one step in a multistage production process for a mass produced PCB of the kind you see in washing machines, transistor radios, TVs, computers, etc. Evidently, output per hour is proportional to the time it takes to complete the drilling of one board. Just like Ulysses 2000, the drill has to visit every hole (city) exactly once, drilling time per hole is constant, and savings in the total processing time result from minimizing the travel time of the drill. In Figure 2 we have assumed that the travel time between two points is proportional to the Euclidean distance of the two points in the plane. Like in Ulysses' problem, finding *some* roundtrip for the drill is easy. Figure 2b shows the solution produced by a professional scheduler in industry and – taking a closer look at this odyssey – the question that comes to one's mind is: can one not produce, by means of a mathematical algorithm, a better or even an optimal solution to such a problem automatically in reasonable computing time?

Mathematical Modeling

The first step towards solving instances of such large sizes is to find a mathematical formulation of the combinatorial problem such that every solution of the real problem corresponds to a solution of the mathematical model, and vice versa. In our case, this is easy. We describe the traveling salesman problem by means of mathematical structures known as graphs. A graph consists of points and lines connecting pairs of points. The graph associated with a traveling salesman problem is constructed as follows. Each city (or hole of a PCB etc.) is declared to be a point. We connect a pair of points by a line if there is a direct link between the associated cities. Let us assume that, as in the case of Ulysses' problem, every two points are connected by a line and thus in Ulysses' problem there are exactly 120 lines for the 16 points. The resulting graph is called a *complete* graph. Moreover, with each line we associate a *length* which corresponds to the time (or distance) it takes to travel between the two points that the line connects. Every roundtrip of the cities corresponds to some subset of the lines. A set of lines corresponding to a roundtrip is called a tour or a Hamiltonian cycle in graph theory. Every tour has a length which is nothing but the sum of the lengths of the lines in the roundtrip. In graph theoretical terms, the traveling salesman problem (TSP) is the task of finding a shortest tour in the complete graph. Note that different tours may have the same length; the length of the shortest tour, however, is a unique value. In combinatorial optimization we want to find some tour that has this value. A shortest tour of the 2,392 city drilling problem is shown in Figure 2c. It was computed by Manfred Padberg and Giovanni Rinaldi in December of 1986 on a CYBER 205 computer in about 27 hours of CPU time. We will outline the approach they took in this article.

Having a mathematical framework to work with, let us first try to find out why the TSP is referred to as a problem of notorious difficulty. As we have seen above the TSP has an enormous number of tours. Does this fact imply that the problem *must be* difficult? Surprising as it may seem, this is not the case, i.e., the number of possible solutions is *not* an appropriate measure of difficulty of a combinatorial problem.

To prove our point we shall consider the following different combinatorial problem. We have, like in Ulysses' case, 16 cities and want to construct a *communication network* such that every city can communicate with every other city, possibly via some other cities. The construction of a direct communication link, which might be a fiber optic cable between two cities, costs a certain amount that depends on the distance between the two cities. In principle we can build a direct link between every pair of cities which results in a complete graph and a costly network. To build a network that costs less we can proceed as follows. We begin with the graph containing all lines corresponding to all possible direct communication links, order the lines by decreasing cost and remove, beginning with the most expensive line, the lines one by one while insisting that the resulting sparser graph remains connected. We stop to "thin out" the graph when the removal of any one of the remaining lines results in a disconnected network. The resulting communication network is shown in Figure 3a for the cities of Ulysses' problem. It has a length of 4,540 km. As simple as this algorithm is, it is a mathematical fact that the algorithm finds an *optimal*, i.e. a cost minimal, solution to the problem! Note that this algorithm processes at every stage the most expensive line and it is therefore called a *greedy algorithm*.

Suppose the initial graph has n points. Then the graph that results from applying the algorithm has the property that it has $n-1$ lines and that it contains no cycle. Such graphs are called *spanning trees*. A very old result of graph theory found by Arthur Cayley(1821-1895) in 1889 says that the complete graph on n points contains exactly n^{n-2} different spanning trees. For Ulysses' problem with 16 cities this number equals 72,057,594,037,927,936 which is finite, but substantially bigger than the number of tours of Ulysses' roundtrip problem. In general, the number of tours of the TSP is by many orders of magnitude smaller than the number of spanning trees. Yet a minimum cost spanning tree for 16 cities, see Figure 3a, can be found in a matter of milliseconds on a personal computer.

So, what is it that makes the TSP difficult? In truth, the answer is that we do *not* know. There is, however, a mathematical "theory of complexity", that provides a framework for distinguishing between "easy" problems like the minimum cost spanning tree problem and "hard" problems like the TSP. Without going into the details of this theory we remark simply that besides the TSP a plethora of other combinatorial problems of tremendous practical interest fall into the class of "hard" problems.

We shall see instantly that a minor variation of a problem – or so it may look on first sight – can render an "easy" problem a "hard" combinatorial optimization problem. The greedy algorithm for finding a minimum cost spanning tree provides us with a communication network such that every city is connected to every other city in the network at minimum cost. Connectivity of networks is, however, but one consideration that communication network designers have to take care of. Consider an adversary, e.g. nature, that could destroy parts of the network. This may happen due to an earthquake, a fire, an attack by terrorists or simply because an excavator cuts a cable. If in the spanning tree shown in Figure 3a the adversary destroys the communication link between Messina and Corfu, then the six eastern cities can no longer communicate with the ten western cities. So considerations relating to the "survivability" of the communication network enter. The obvious possibility of constructing a network of maximum survivability calls for a complete graph and thus a very costly solution. That means that we are faced with the problem of *trading off* survivability and the cost of constructing and maintaining the network. To be able to do so, we need to set priorities on the communication links between the cities or on the cities themself. The beauty of combinatorial approaches to problem solving is that such considerations can be worked into the mathematical model of practical problems — but this additional feature also makes the resulting new problem just as hard as the traveling salesman problem in the general case.

From a survivability point of view one would thus like to protect a network against the failure of a number of links or nodes or both. This is not always done in reality. For example, during the process of replacing the standard copper wires of the LATA (Local Area Telecommunication Access) networks in the United States by fiber optic cables, survivability considerations were given a secondary role compared to cost effectiveness, maybe due to the high cost of the new technology. As has been reported in major newspapers all around the United States, this neglect brought about tremendous disasters like the complete breakdown of the Chicago telephone network in 1988. The aftermath of this event was a truly costly reconstruction and reconsideration of the network design rules.

At present, communication networks are (roughly) designed as follows. In the beginning a set of locations to be connected is chosen. In a second step all direct links between pairs of locations are determined that can possibly be constructed and their construction cost is estimated. Then the locations are grouped by their relative importance. For our purpose, we assign numbers 0, 1, 2, ... called *types* to the locations. To be considered survivable the network to be designed must have the following property: for any two locations of type i and j, respectively, the network must contain a path connecting the two locations, even if a number of direct links equal to the minimum of i and j fail. Moreover, among all networks having this property we wish to find and construct one of minimum cost. Networks can be protected in a similar manner against the failure of nodes or the common failure of nodes and links.

Let us consider the problem of connecting the 16 cities in the Mediterranean from Ulysses' problem by a survivable communication network. We declare, for instance for historical reasons, Ithaca (1), Troy (2), Djerba (5), Circeo (10) and Messina (13) "important" cities and call them type 1 locations, while all other cities are of type 0. The task then is to find a minimum cost network such that in case any single direct link fails, each of the important five cities is still connected to all the other important ones by a communication path. Assuming that the construction cost of a direct link is proportional to the length of the link we obtain the network of length 5,842 km shown in Figure 3b. In fact, the optimum solution is such that except for the cities 9, 11 and 12 all other cities are better connected to each other, although this was only required for the five important cities. By "playing" with the parameters of the network such as the node types, network designers nowadays try to find a reasonable balance between additional network construction costs and increased customer protection against network faults.

Communication networks that are highly vulnerable due to an aggressive environment, e.g. those of a military vessel, have to provide a much higher degree of survivability. For instance, if we assign type 3 to each node of our Mediterranean problem we obtain the network of minimum cost shown in Figure 3c. Its length is 16,112 km. Here, even the failure of three direct communication links at the same time does not destroy the functioning of the network. Moreover, if any one of the nodes of the network fails (in practice this occurs, for instance, due to a fire in a network hub) the remaining nodes are still mutually connected by a communication path. If, however, the nodes 8 (Zakinthos) and 16 (Corfu) fail simultaneously, then the communication network will be disconnected.

Figure 4 shows the cost minimal solution of a real LATA network in the eastern part of the United States, where the nodes of type 0 are depicted by full circles and the nodes of type 1 by little squares. This communication network connects 116 hubs.

By looking at variations of Ulysses' problem we have arrived at other problems of significant practical importance. In fact, we have not told the full story of modeling real-world network (and other) design problems. Even in the (already quite complicated) design of survivable networks further technical aspects may have to be considered that lead to additional side constraints on the design. Besides network design problems, there is a host of further relevant practical problems in engineering, management, etc. that are treated and solved by means of combinatorial optimization. Given the state of affairs it is impossible to comprehensively survey all applications of combinatorial optimization. Problems of this type arise in the planning of public and private transportation, in operations management, in the financial planning of large organizations, in the planning and design of logistics and distribution systems, in the area of network design and operation of all kinds, etc; see the box "Applications of Combinatorial Optimization". The interested reader will find many articles on these and related topics in the Operations Research and Management Science literature as well as in the Applied Mathematics/Engineering literature.

Methodology

Let us now address the question: How does one find good or optimum solutions and how does one provide "quality guarantees" for solutions? Clearly, every specific combinatorial problem has its particularities that can and must be exploited in the solution process. However, the design of solution algorithms follows in general some fundamental principles.

Due to the finite nature of combinatorial problems it is clear that there is an optimum solution having a certain optimum value. To make matters concise let us assume that we have a minimization problem and thus the optimum value is a minimum possible value. Any feasible solution to the combinatorial problem, that is, a solution that satisfies all side constraints, provides an *upper bound* for the optimum value. Like in the case of Ulysses 2000, finding feasible solutions is frequently not a difficult task. Algorithms that construct feasible solutions, and thus upper bounds for the optimum value, are called *heuristics*. Heuristic algorithms can broadly be classified into two categories: *construction heuristics* attempt to find a "good" feasible solution in a single attempt, while *improvement heuristics* are designed to iteratively modify and eventually improve some given starting solution.

The trouble with "heuristic" approaches to combinatorial problem solving – when used on a standalone basis – is that, there is no quality guarantee about the closeness of the upper bound to the optimum value when the algorithm stops. Sometimes the upper bound may very well be equal to the optimum value, but it may also be very far off. One simply does not know. This is unsatisfactory from a practical point of view. To estimate the deviation of the upper bound from the optimum value a *lower bound* on the optimum value is needed. If the upper and lower bound coincide, a proof of optimality of the solution giving the upper bound is achieved. If not, a conservative estimate of the true relative error of the upper bound compared to the optimum value is provided by the difference of the upper bound and the lower bound divided by the lower bound. When multiplied by 100, this measure tells us by at most how many percentage points the value of the best known solution is above the value of an optimum solution.

So the next "real" question is how does one find a lower bound on the optimum value. The correct answer is that we will have to construct an appropriate *mathematical formulation* of the lower bounding problem. Indeed, the upper and the lower bounding techniques can be combined in an iterative fashion to find provably optimal solutions to hard combinatorial optimization problems. This combination is the thrust of the recent developments that have advanced numerical problem solving in combinatorial optimization so substantially.

To demonstrate these two general principles let us return to the problem of Ulysses

2000 and discuss upper and lower bounding techniques that have been developed for the traveling salesman problem. The story for other hard problems differs from this one essentially only in technical detail, the mathematics of which can be quite arduous, though.

Upper Bounding Heuristics

Upper bounding heuristics are best described by adapting the greedy idea. We pick a starting city and then we apply the following myopic rule: If we are in some city we always pick the nearest neighboring city that has not yet been visited. If no city is left, then we return to the starting point. This method is called the *nearest neighbor heuristic.* If we start in Ithaca, we first go to Zakinthos, from there to Corfu, to Messina, etc. Finally, we get the tour of length 9,998 km shown in Figure 5a. Amazingly, this tour has about the same length as Ulysses' odyssey. On the other hand, starting in Gibraltar we obtain the tour of length 8,225 km shown in Figure 5b.

These two figures teach us two lessons. First, the result of the heuristic strongly depends on the starting point. Second, it is easy to see that a better solution can be constructed, for instance, by "uncrossing" some of the crossing edges. The first observation holds true for most of the heuristics that are in use, and it is the second observation that gives rise to the idea of "improvement" heuristics that manipulate given solutions to obtain better ones.

Another class of construction heuristics, called *insertion heuristics*, works as follows. In an initial step some cycle on a few cities, say 3, is chosen arbitrarily. In a general step, one has a cycle of length l, say, and some cities not yet in the cycle. One of the cities outside is chosen, say city k. One tries all l possibilities to remove an edge ij of the present cycle and replace it by the path from i to k to j. City k is inserted between those two nodes i and j for which the length of the new cycle is as small as possible. The insertion heuristic terminates when the cycle contains all cities.

There are various proposals in the literature for the choice of the next node to be inserted. For instance, if we choose the city farthest away from (nearest to) all the cities in the present cycle, we obtain the farthest (the nearest) insertion heuristic, respectively. Practical experiments have revealed that the farthest insertion heuristic provides reasonable tours, in general much better than the nearest neighbor or nearest insertion heuristic. The solution obtained by farthest insertion, starting with the cycle formed by the cities 1, 5 and 9 has length 7,023 km and is shown in Figure 5c. A whole host of further construction heuristics can be found in the relevant technical literature.

Let us now discuss improvement techniques. The basic rule for such procedures is always similar to the idea of "uncrossing edges" mentioned above. To give an example we outline the so-called 2-*opt method.* It works as follows. Given a tour, consider all pairs of edges hi and jk of the tour, where all four cities h, i, j, k are distinct. Removing the two edges hi and jk from the tour, there is a unique way of reconnecting the two remaining paths such that a new tour is obtained. If the new tour is shorter, then it replaces the old tour and the procedure is repeated until no such improvement can be made. A tour with that property is called locally optimal (with respect to this 2-exchange). This basic idea is easily extended in various ways and gives rise to improvement heuristics that exchange various numbers of lines and nodes. Of course, the more possibilities we consider to choose a line or a node for either removal or reconnection of the resulting pieces, the longer the running time will be. The empirical record of the best of these improvement heuristics is surprisingly good with a typical relative error of 2% to 5% for the resulting "local optimum". For large problems the running time may, however, be excessive unless sophisticated reduction and data handling techniques are employed. We have applied the 2-*opt* method to the three tours shown in Figures 5a, 5b, 5c. In the first case we obtained a tour of length 6,974 km, in the second case one of length 6,890 km and in the third case a tour of length 6,973 km. Thus the 2-*opt* method considerably improves the nearest neighbor tours, but there is not much length reduction if we apply it to tours obtained by the farthest insertion heuristic. The application of the nearest neighbor and the farthest insertion heuristic and the improvement by the 2-*opt* method has resulted in the tour of length 6,890 km shown in Figure 5d. How good is this tour? How can we judge its quality?

To enhance the heuristic techniques described above further one frequently combines them with *random sampling* schemes in order to reduce the chance that the method might get "trapped" in some local optimum too early during the calculation. The basic idea is to permit the algorithm to select, as the next solution for the improvement heuristic, a solution that has a worse value than the present one. The choice of selecting a worse solution is simply made on the basis of a sequence of probabilities that change during the execution of the algorithm and that tend to zero. Methods that employ these principles go by the name of Monte Carlo Methods, Simulated Annealing, Genetic Algorithms, Evolution Methods, etc. The improvement heuristics are computationally expensive already; so these "randomized" procedures often require excessive computation times and are therefore not applicable when approximate solutions have to be calculated for large problem instances within limited computation time. But they can be expected to perform better in terms of the quality of the solution obtained.

The programming effort for these methods for the heuristic solution of medium sized traveling salesman problems is rather moderate. As a result, they have become the playground of many amateurs, which – unfortunately – has produced grossly exaggerated claims as to the goodness and practicality of such methods. On the other hand, limitations of current technology make heuristic algorithms the workhorse of practical combinatorial problem solving and they also help the exact optimization methods in several ways.

Lower Bounding Techniques

Whatever "tricks" you employ to find upper bounds, the question of finding a quality guarantee remains. For this we need a lower bound on the optimum value. The principal idea behind all lower bounding techniques is the following. We "embed" the set of all feasible solutions, in Ulysses' case this is the set of all tours, into a larger set that contains all feasible solutions and possibly further elements. The minimum value of the objective function over all elements of the larger set is clearly a lower bound for the minimum value over all elements of our original set. The challenge thus is to find an "embedding" such that we can minimize over the larger set rather easily and such that the gap between the two respective minima is not too large. If both criteria can be met, good lower bounds can be computed quickly.

There are several general modeling techniques that yield suitable lower bounds. We will discuss two such approaches. One will be called *combinatorial relaxation*, the other one *linear programming relaxation.*

Combinatorial Relaxations

Combinatorial relaxations depend on the availability of easily solvable combinatorial optimization problems that contain the given hard one. Such easy problems are not always at hand, but if they are they may be quite useful. In the case of the traveling salesman problem there are several combinatorial relaxations such as perfect 2-matchings, perfect assignments, and 1-trees that do the job. Let us discuss the last example here.

We shall assume that the traveling salesman problem is stated on a complete graph having n nodes with arbitrary lengths on the lines – just like we have discussed it above. We observe that every tour has the following properties:

1. *It contains exactly one cycle.*
2. *It contains exactly n lines.*
3. *It is connected.*
4. *Every node is contained in exactly two lines.*

If we want to find a subset of the lines of the graph of minimum weight that satisfies some subset but not all of these four requirements we arrive at a *combinatorial relaxation* since every tour will also satisfy these fewer constraints. Clearly, there may be other subsets of the lines, which are not tours, that satisfy such a reduced list of requirements. For instance, if we drop conditions 1 and 3, the feasible solutions are the so-called perfect 2-matchings. If we drop condition 4 we obtain the set of 1-trees. In fact, one can reduce the size of the set of 1-trees slightly by making the following requirement 5 instead of 4.

5. *An arbitrarily chosen specific node is contained in the cycle and in exactly two lines.*

Such an 1-tree of minimum weight can be computed easily as follows. Choose the special node. Delete the special node from the complete graph. Compute a minimum length spanning tree of the reduced graph (e.g., by the greedy algorithm described above). Add the two lines of shortest length that contain the special node to the spanning tree. This set of lines is a shortest 1-tree, and its length is obviously a lower bound for the optimum value of the TSP. If the modern Ulysses chooses Gibraltar as the special node he obtains the 1-tree shown in Figure 6 of length 6,044 km.

This value is not really a superb lower bound. The same, unfortunately, holds true for most other combinatorial relaxations. But there are general techniques such as *Lagrangian relaxation* with which these lower bounds can be improved – sometimes – considerably. These techniques involve methods of nondifferential nonlinear optimization and cannot be explained here.

At this point we have computed a best upper bound of 6,890 by means of the nearest neighbor heuristic combined with the 2-opt method and a lower bound of 6,044 using the 1-tree relaxation for the problem of Ulysses 2000. These bounds guarantee that the tour shown in Figure 5d is at most 14% above the true optimum. Note that the current best tour length is in fact only 0.5% above the optimum value 6,859 km. But (disregarding the result of the enumeration program) we have no proof of this fact yet.

Linear Programming

Linear programming is undoubtedly one of the most important techniques of broad applicability that post World War II applied mathematics have produced. It is most frequently interpreted as the problem of optimally allocating economic activities to scarce resources and today it is used daily in the planning activities of airlines, oil refineries, banks, car manufacturers, etc. Not surprisingly, linear programming is also used to solve combinatorial optimization problems; indeed, the most successful lower bounding techniques for such problems all utilize linear programming. To explain the underlying idea – which requires us to think in terms of the geometry of vector spaces – we will again resort to the problem faced by Ulysses 2000.

Since we have 16 cities there are exactly 120 direct links between all of them. Ulysses 2000 needs to choose 16 links from these possible 120 ones that form a tour. So let us view a tour as an array or – as we shall say – as a vector of 120 "variables" that can assume the values zero or one according to whether or not Ulysses chooses to travel a particular route.

Each variable corresponds to a direct link and let us – somehow – index the 120 variables sequentially 1, 2, ..., 120. Every tour of Ulysses defines a vector of "length" 120 having 16 variables equal to one – namely those corresponding to the links in the tour – and the remaining 104 variables are equal to zero. On the other hand, not every such vector with 120 variables corresponds to a tour; so we will have to impose some constraints on the vector's variables. Geometrically, every tour of Ulysses can now be viewed as a point in the vector space of 120 dimensions. The fact that we cannot "picture" more than the vectors of the three-dimensional space in which we live does not bother us. Ever since René Descartes (1596-1650) wrote his famous treatise "La Géométrie" in 1637, mathematicians have worked analytically in spaces of any dimension. Now remember that there are exactly 653,837,184,000 tours for Ulysses 2000. So we have exactly this many points in the 120-dimensional space. Among these points we wish to find one such that the sum of the lengths corresponding to the variables that assume the value one is as small as possible. Since the remaining variables for this point are zero we can express the objective function as a linear function of the 120 variables. Mathematically, we thus wish to minimize a linear function over the set of 653,837,184,000 points in the 120-dimensional vector space. Now let us connect any two of the points by a line of the 120-dimensional space – just like we would do in the plane or in the 3-dimensional space. Since the objective function is linear it is clear that one of the two endpoints is "best". None of the points in the "interior" of the line can be "best" unless they all are equal in terms of the linear objective function, which corresponds itself to a "line" in some plane. Extending this operation of connecting points systematically to all triplets, quadruplets etc. of the 653,837,184,000 points in question we get a *polytope* in the 120-dimensional space. Now it is a fact that was known already to mathematicians of the 19th century that every polytope can be described as the solution set of finitely many linear equations and inequalities. Inequality is to be understood in the weak sense of "less-than-or-equal-to" rather than in the sense of strict inequality. The problem faced by Ulysses 2000 is thus the problem of optimizing a linear objective function of 120 variables over some finite set of linear equations and inequalities in the same set of 120 variables. But this is exactly what linear programming is all about and thus Ulysses 2000 can solve his problem – in principle, at least – using the techniques of linear programming; see the box "Convex Hull".

To apply the techniques from linear programming one must have – at least in theory – a complete *list* of the inequalities that define the associated polytope. For the problem with 16 cities such a complete list is not known to date. Indeed, *complete* descriptions may never be found – which is another way of saying that the traveling salesman problem is a notoriously difficult problem. Explicit linear programs for traveling salesman problems are only known for up to 8 cities. More precisely, the polytope associated with the 8-city traveling salesman problem has 8 equations and 194,187 inequalities in the 28 variables corresponding to the 28 direct links that are possible between 8 points. The corresponding numbers for the polytope associated with 7 cities are 7 equations and 3,437 inequalities in 21 variables. Thus there is an enormous growth in the number of inequalities that describe the polytope when we go from 7 to 8 cities. For Ulysses' 16 cities the number of inequalities is astronomically large. Yet far larger traveling salesman problems than Ulysses' are today optimized using linear programming techniques. How is this done?

One of the principles that mathematicians concerned with practical calculations have utilized for centuries is the principle of *relaxation*. This principle means that one replaces the "true" problem by a larger one – just like we did when we discussed "combinatorial" relaxations of our problem. In the linear programming context this means that we are going to ignore the "integrality" requirement of zero or one and almost all of the astronomically many constraints that are necessary to describe the "true" polytope of Ulysses' problem. Instead, we will start with only a very small subset of the constraints and permit the variables to assume *any real* values between zero and one; see the boxes "Integer Programming" and "Geometric Representation". Since every city must be met by two links we can start with 16 equations in the 120 variables expressing the fact that the variables corresponding to the links meeting a city must sum to two. Dropping the requirement of zero or one and replacing it by the requirement that all 120 variables may assume any value between zero and one (including zero and one) we get 240 inequalities and 16 equations in 120 "real" variables. Solving the corresponding linear program for Ulysses' problem we get a minimum value of 6,113 km for the objective function which is a *better* lower bound than the value of 6,044 km that the 1-tree relaxation produced. So we now have a proof that the length of the currently best tour of Figure 5d is at most 12.7% above the optimum value.

The linear program solver not only provides us with the minimum objective function value of 6,113 km but also with a solution vector, i.e., with 120 values for the variables that satisfy the equations and inequalities that were fed into it. If the nonzero components of the solution vector corresponded to a possible tour for Ulysses 2000, then we could stop right there because we would have found an optimal solution for Ulysses 2000 and thus solved his problem. But they do not. Given the fact that an astronomically large number of additional inequalities was ignored it is not surprising that they do not. By ignoring all those constraints we have in fact embedded the "true" polytope into a "larger" polytope in the same space of 120 dimensions. Since the true polytope for Ulysses' problem is contained in this larger one there must be some way to "separate" the solution vector obtained in the first step from the true polytope. This is done by inequalities of the less-than-or-equal-to variety. If we require an inequality to hold with equality then we have an equation. Geometrically, an inequality divides the 120-dimensional space into two halfspaces while the corresponding equation gives the "hyperplane" that is common to both halfspaces – just like a line in the ordinary plane which divides the plane into two halves. The separating inequalities that are of interest to us contain the true polytope in one halfspace whereas the point given by the solution vector of the linear program lies on the other side, the wrong side of the hyperplanes given by the corresponding equations. So these separating inequalities *cut off* the point given by the linear programming solution from the polytope and are for that reason called *cutting planes*; see the boxes "Cutting Plane Algorithm" and "Separation". In fact, many such cutting planes exist and among those that do the job there are "best possible" ones. These are those that intersect with the true polytope as much as possible and define the *facets* of the polytope. Much theoretical work has focussed in the past twenty years on identifying cutting planes that define facets for numerous combinatorial optimization problems that are of practical importance.

In Figures 7a, 7b, 7c we display the optimal solutions to several linear programs that we have solved to find an optimal solution to the problem that Ulysses 2000 faces. The first optimal solution consists of four short cycles or "subtours" such as, e.g., the one from Troy to Maronia to Malea and back to Troy; see Figure 7a. This is not what Ulysses 2000 wants; he wants a single roundtrip. To this end he can choose at most two of the three direct links connecting the cities Troy, Malea and Maronia. So mathematically, the sum of the corresponding three variables out of the total of 120 variables must be less than or equal to two, while for the first linear programming solution it is three. The corresponding inequality is one of the huge number of inequalities of the true polytope that we have left out. All that we have to do is to add it to our linear program as a cutting plane. Doing likewise for the remaining three subtours we have four inequalities that are added to the linear program, which gives us a second linear program to be solved. Indeed, all that needs to be done is to "reoptimize" the first linear program after adding the four new inequalities that all cut off the first solution vector – which can be done fast and efficiently with contemporary software packages.

Solving the second linear program we get an optimal objective function value of 6,228 km and a solution vector that still does not correspond to a tour for Ulysses 2000 – see Figure 7b; but the algorithmic idea is clear now, for all we have to do is to *iterate*. We "activate" another very small subset of the astronomically many inequalities that we have ignored so far. Doing so and solving a third linear program we get an optimal objective function value of 6813.5 km and a solution vector that still does not correspond to a tour for Ulysses; see Figure 7c. Indeed, the linear programming solution assumes values of 1/2 for some of the 120 variables – which is what we permit for the "intermediate" linear programs to happen and which are depicted by the "broken" lines of Figure 7c. Iterating again by adjoining some more inequalities and reoptimizing the corresponding fourth linear program we obtain an objective function value of 6,859 km, a solution vector that corresponds to the optimal tour of Ulysses 2000 displayed in Figure 1b and we are done!

The iterative procedure that we have described above is called a *cutting plane algorithm*. If the possibly astronomically large number of inequalities describing the polytope is fully known, then it is clear that the cutting plane algorithm will always stop after a finite number of steps – like in the case of Ulysses' problem where we iterated four times. But we know at present all of these necessary inequalities *neither* for the traveling salesman problem *nor* for most of the other combinatorial optimization problems of practical interest. Much research activity focuses on accumulating such knowledge about more and more combinatorial problems – knowledge that permits to push computational limitations way beyond what most scientists in our field considered attainable only a few years ago. On the other hand, the "hardness" of the traveling salesman and other combinatorial optimization problems may prove to forbid a *complete* knowledge of the astronomically large list of necessary inequalities forever – we just do not know – and at present we clearly do not know them all. So different from the case of the problem of Ulysses 2000, the iterative cutting plane procedure may come to a halt before an optimal tour is found.

What are the options now? We may stop because we have obtained a reasonable lower bound for the optimum tour length. We may also enter an enumerative procedure by *branching* on some variable x that is positive and not equal to one in the linear programming solution vector when the iterative procedure stops. Branching simply means that we create two new problems from the current one where in one of the two problems we force the link corresponding to such a variable x into the solution by setting it equal to one and in the other one we force it out of the solution by setting it equal to zero. Both problems are then again subjected to the cutting plane procedure. This leads to a tree search procedure that goes by the name of *branch and cut* and which is related to, but much more powerful than the *branch and bound* procedure developed in the 1960s for the solution of combinatorial problems.

At present, the most successful algorithms that solve large-scale combinatorial optimization problems to optimality or that provide very good quality guarantees are based on the cutting plane procedure outlined in this article. Of course, they are also enhanced by various heuristic and enumerative techniques. As should be clear from our description, this modern approach requires a lot of thorough mathematical analysis, knowledge of several fields related to optimization and computer science, knowledge of efficient data structures, data handling and computer programming, because to "prove" its value for numerical problem solving one must necessarily experiment on real computing machinery. Indeed, members of our research teams and researchers at numerous other institutions have successfully implemented branch and cut algorithms for many practically relevant combinatorial optimization problems – substantially more than the ones discussed in this short article. The TSP examples illustrated here were solved by the code developed by Padberg, and Rinaldi of IASI-CNR in Rome, the survivable network examples by a code developed by Martin Grötschel, Clyde Monma of Bell Communication Research in Morristown, N.J., and Mechthild Stoer of the Konrad-Zuse-Zentrum für Informationstechnik in Berlin. The solution times for the various problems on 16 cities are a matter of milliseconds on a modern workstation – including the solution of the problem faced by Ulysses 2000. The latest version of the Padberg-Rinaldi code solves the drilling problem for the printed circuit board (PCB) with $2,392$ cities of Figure 2 in about 1 hour 40 minutes on a time-shared Sun-Sparc 10 workstation, i.e. on a computing machine that is somewhat less powerful than the one we used to *enumerate* all possible tours for the 16-city problem of Ulysses 2000 – which took 92 hours or about 4 days of uninterrupted computing. Think about that and more importantly, think about what parallel processors, vector processors, teraflop machines, and beyond, promise to do in terms of exact problem solving by their sheer computing power! Add sound applied mathematics and clever software engineering into the equation – *exact* problem solving will become a reality for problems of substantially larger scale than we all might consider attainable at present.

To close our story on combinatorial problem solving, let us return to the question of why the traveling salesman problem is called a notoriously difficult problem. As you must have noticed, we have repeated several times that there are *astronomically* many inequalities needed to describe the associated polytope completely. That much we know to be a fact today. So the next question is simply: Does this fact explain the difficulty of the problem? In part this is probably so, but – as astounding as it may be – it does not explain fully the difficulty of the problem either. Indeed, there are combinatorial problems that are solved easily – both in theory and in practice – but whose polytopes require *far more* inequalities than the one associated with the traveling salesman problem. Of course, we talk about polytopes in spaces of equal dimension and the polyhedron associated with the minimum cut problem that we need in order to solve the *separation problem* for subtour elimination constraints is one such example. Space limitations forbid us to elaborate on this point, but then there is a sort of mathematical "dualism" at work. Extreme points and facets are just the two faces of a coin. So we are back full circle. If it is *neither* the number of possible solutions *nor* the number of inequalities of a complete description for a combinatorial problem that makes it difficult, what is it that makes it "difficult", what is it that makes a problem "easy"? Nobody knows an answer to these questions at present, but the *computational* record of combinatorial problem solving is good nevertheless.

Epilogue

We hope that you have enjoyed this journey into the world of combinatorial problem solving. While researchers around the globe have made enormous progress to tackle such problems, many open problems persist, the satisfactory solution of which will require many years of hard work. Practical problems of truly large scale can be solved to optimality already today, but the fundamental question of tractability of hard combinatorial problems promises to remain a challenge for many years – perhaps generations – to come. As we have tried to explain, considerable efforts are on the way to deal with the complexity of such problems. So just like Ulysses who was struggling in ancient times against misfortunes and bad winds in search of his way home to his beloved Penelope, dedicated mathematicians and computer scientists are searching today for ways of coping with the presumed intractability of hard combinatorial optimization problems of practical relevance.

transportation:	airline scheduling: crew and fleet
	mix of trucks of a transportation company
	routing of trucks and driver assignment
	cost-minimal garbage collection
mass transit:	timetable optimization
	bus and subway circulation scheduling
	drivers' scheduling
VLSI design:	placement of components on chips
	routing of nets
	layer assignment
production:	automatized operations: control of CNC-machines, control of the moves of stacker cranes, control of driverless vehicle systems, etc
	assembly line balancing
	job sequencing
	location of storage space
	assignment of items to storage locations
	flexible manufacturing
finance:	portfolio selection
	arbitrage optimization
	capital budgeting
networks:	capacity planning of communication networks
	database design and management
	extension and management of centralized heating networks
	power distribution in electrical networks
	design of networks that survive catastrophic events
location and logistics:	location of distribution or sales centers and their management

Applications of Combinatorial Optimization are too numerous to be surveyed here in detail. The above outlines some of the numerous areas where methods from combinatorial optimization are used today to aid human problem solving. Given the tremendous growth in exact and heuristic problem solving capabilities – due to both intensive mathematical research and vastly improved computing machinery – the areas of application of exact and heuristic methods for combinatorial optimization are bound to increase substantially in the years to come.

References

A Book on Ulysses' Voyage

E. Bradford, "Ulysses Found", Harcourt, Brace & World, New York, 1963.

Recommended technical Books

G. B. Dantzig, "Linear Programming and Extensions", Princeton University Press, Princeton, 1963.

M. R. Garey, D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman, San Francisco, 1979.

M. Grötschel, L. Lovász, A. Schrijver, "Geometric Algorithms and Combinatorial Optimization", Springer, Berlin, 1988.

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys (eds.), "The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization", Wiley, Chichester, 1985.

G. L. Nemhauser, L. A. Wolsey, "Integer and Combinatorial Optimization", Wiley, New York, 1988.

M. Padberg, "Linear Programming: Lecture Notes", New York University, New York, 1993.

M. Stoer, "Design of Survivable Networks", Lecture Notes in Mathematics, Vol. 1531, Springer Verlag, Berlin, 1992.

Press Articles on the Vulnerability of Communication Networks (sample)

New York Times (1988), "Phone system feared vulnerable to wider service disruptions", *New York Times* (May 26, 1988).

New York Times (1989), "Experts say phone system is vulnerable to terrorists", *New York Times* (February 8, 1989).

Newark Star Ledger (1987), "Damage to fiber cable hinders phone service", *Newark Star Ledger* (September 22, 1987).

Newark Star Ledger (1988a), "Cable snaps, snags area phone calls", *Newark Star Ledger* (February 26, 1988).

Newark Star Ledger (1988b), "Phone snafu isolates New Jersey; long-distance cable snaps", *Newark Star Ledger* (November 19, 1988).

"Keeping the phone lines open", IEEE Spectrum, June 1988.

Press Articles on Traveling Salesman and related Problems (sample)

A. Sagnelli, "Short-circuiting the traveling salesman problem", *New Scientist*, 27 June 1992, page 16.

"Auf dem schnellsten Wege um die Welt", *GEO*, Nr. 9, September 1987, 168 - 170.

Barry A. Cipra, "Spotlight on Three Invited Speakers", SIAM News, Vol. 24, No. 5, September 1991, pages 13 and 20.

Barry A. Cipra, “The 1993 SIAM Annual Meeting”, SIAM News, Vol. 26, No. 6, October 1993, pages 1 and 10 - 12.

Gina Kolata, “Math Problem, Long Buffling, Slowly Yields”, New York Times, March 12, 1991 (with Correction in New York Times, May 24, 1991, page A30).

M. Padberg, “Out of Place”, Letter to the Editor, OR/MS Today, December 1991, pages 11 - 12.

Recommended Technical Articles

H. Crowder, M. Padberg, “Solving Large-Scale Symmetric Traveling Salesman Problems to Optimality”, Management Science, 26 (1980) 495 - 509.

H. Crowder, E. L. Johnson, M. Padberg “Solving Large Scale Zero-one Linear Programming Problems”, Operations Research, 31 (1983) 803 - 834.

M. Grötschel, C. Monma, M. Stoer, “Computational Results with a Cutting Plane Algorithm for Designing Communication Networks with Low-Connectivity Constraints”, Operations Research, 40 (1992) 309 - 330.

M. Grötschel, O. Holland, “Solutions of Large-Scale Symmetric Traveling Salesman Problems”, Mathematical Programming, 51 (1991) 141 - 202.

M. Grötschel, C. Monma, M. Stoer, “Polyhedral Approaches to Network Survivability”, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 5, AMS 1991, 121 - 141.

K. L. Hoffman, M. Padberg, “Solving Airline Crew-Scheduling Problems by Branch-and-Cut”, Management Science, 39 (1993), 657 - 682.

K. L. Hoffman, M. Padberg “Improving LP-representations of Zero-one Linear Programs for Branch-and-Cut”, ORSA Journal on Computing, 3 (1991) 121 - 134.

M. Padberg, “Covering, Packing and Knapsack Problems”, Annals of Discrete Mathematics, 4 (1979), 265 - 287.

M. Padberg, G. Rinaldi, “A Branch-And-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems”, SIAM Review, 33 (1991), 60 - 100.

T. Van Roy and L. Wolsey, “Solving mixed integer programming problems using automatic reformulation”, Operations Research, 35(1) (1987), 45 - 47.

1	Ithaca	38.24N	20.42E
2	Troy	39.57N	26.15E
3	Maronia	40.56N	25.32E
4	Malea	36.26N	23.12E
5	Djerba	33.48N	10.54E
6	Favignana	37.56N	12.19E
7	Ustica	38.42N	13.11E
8	Zakinthos	37.52N	20.44E
9	Bonifaccio	41.23N	9.10E
10	Circeo	41.17N	13.05E
11	Gibraltar	36.08N	5.21W
12	Stromboli	38.47N	15.13E
13	Messina	38.15N	15.35E
14	Taormina	37.51N	15.17E
15	Birzebbuga	35.49N	14.32E
16	Corfu	39.36N	19.56E

Table 1. Polar coordinates of the 16 locations in the Mediterranean.

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	509	501	312	1019	736	656	60	1039	726	2314	479	448	479	619	150
2		126	474	1526	1226	1133	532	1449	1122	2789	958	941	978	1127	542
3			541	1516	1184	1084	536	1371	1045	2728	913	904	946	1115	499
4				1157	980	919	271	1333	1029	2553	751	704	720	783	455
5					478	583	996	858	855	1504	677	651	600	401	1033
6						115	740	470	379	1581	271	289	261	308	687
7							667	455	288	1661	177	216	207	343	592
8								1066	759	2320	493	454	479	598	206
9									328	1387	591	650	656	776	933
10										1697	333	400	427	622	610
11											1838	1868	1841	1789	2248
12												68	105	336	417
13													52	287	406
14														237	449
15															636

Table 2. The distance table for Ulysses 2000.