INF431

Branch-and-Bound for the Travelling Salesman Problem

Topic proposed by Leo Liberti

1 Introduction

We consider here the travelling salesman problem.

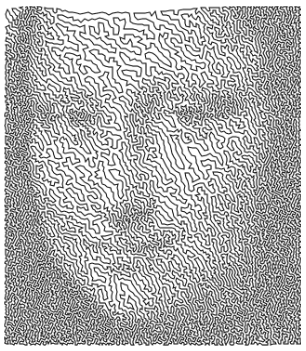
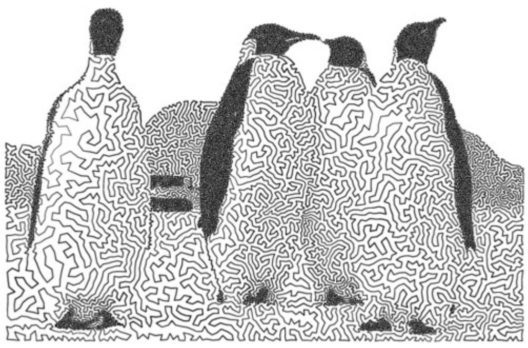
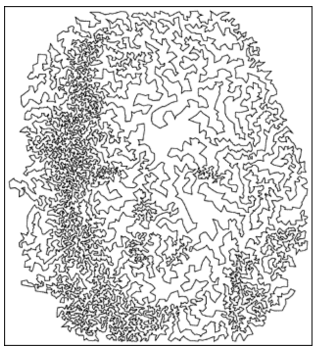
TRAVELLING SALESMAN PROBLEM (TSP)

Given a complete simple digraph $G = (V, A)$ with arc cost function $d : A \to \mathbb{Q}_+$, find a tour of minimum cost.

Notation: a *circuit* is a sequence of nodes $(v_1, \ldots, v_k)$ in V, without repetitions, such that $(v_k, v_1) \in A$ and $(v_i, v_{i+1}) \in A$ for all $1 \leq i < k$. A circuit is *Hamiltonian* if $\{v_1, \ldots, v_k\} = V$. A *tour* is a Hamiltonian circuit. Within the TSP context, circuits are sometimes also called *subtours*.

Since a complete digraph is uniquely identified by the number of nodes $|V|$, the instance is here a pair (n, D) where $n = |V|$ and $D = (d_{ij})$ is an $n \times n$ matrix with entries in $Q_+ \cup \{\infty\}$: $d_{uv} = d((u, v))$ for all $(u, v) \in A$ and $d_{vv} = \infty$ for all $v \in V$. If S is a subset of arcs we extend d to S by defining $d(S) = \sum_{(u,v) \in S} d_{uv}$. Furthermore, if H is the subgraph of G induced by S, we extend d to H by setting $d(H) = d(S)$.

The TSP has countless practical applications[1]. One very interesting application, albeit perhaps less practical than others, is TSP-art: scatter nodes of the graph in a square containing a given image, with a node density per surface unit which is proportional to the image colour wavelength; the cost of each arc is then the Euclidean distance between two nodes on the square. Minimum cost tours then look like the following images[2]:

Another nice property of the TSP problem is that it has been used as a classic testbed for new methods in combinatorial optimization and integer programming for several decades. The TSP has helped shape new methods

[1]See http://en.wikipedia.org/wiki/Travelling_salesman_problem.

[2]Taken from http://wiki.evilmadscience.com/s3/eggbot/tspart/mona-lisa-tsp.png, http://www.dominoartwork.com/images/monroe_tsp_small.gif and http://www.cgl.uwaterloo.ca/~csk/projects/tsp/.

and improve existing methods more than any other problem in the literature. New advances on the TSP always attract a lot of a attention in the optimization community.

Question 1 By enumeration, find the optimum tour for the TSP instance given by $n = 4$ and

$$(d_{ij}) = \begin{pmatrix} \infty & 2 & 4 & 8 \\ 2 & \infty & 16 & 32 \\ 4 & 16 & \infty & 64 \\ 8 & 32 & 64 & \infty \end{pmatrix}.$$

How many tours did you need to enumerate for this instance in order to find the optimum tour? Generalize this to arbitrary n and arbitrary distance matrices, giving the minimum number of tours to enumerate as a function of n. What if the matrix is symmetric? $\diamond$

2 Integer programming

A tour can be represented in several ways. In the lecture, we considered tours as subsets of arcs of cardinality n. In the following, we represent a tour as an automorphism of the set $[n] = \{1, \ldots, n\}$, i.e. a bijection $\mathtt{t}$ from $[n]$ to itself. Specifically, if the automorphism maps i to j then (i, j) is an arc in the tour.

Question 2 Is it true that every automorphism of $[n]$ is a tour?

Question 3 Prove that every automorphism of $[n]$ represents a set of subtours. $\diamond$

We describe the (unknown) optimal tour by means of an array of *decision variables* x_{ij} for $i, j \leq n$ with $i \neq j$. We constrain such variables to take values in $\{0, 1\}$ and denote the array by $x = (x_{ij})$. The **intended meaning** of these variables is as follows:

$$(*) \qquad \forall i \neq j \leq n \quad x_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ is in the tour} \\ 0 & \text{otherwise} \end{cases},$$

Whereas the instance of a problem can be seen as its input, decision variables provide an abstract way to describe its output. Decision variables can be:

- binary, if they take values in $\{0, 1\}$;
- integer, if they take values in $\mathbb{Z}$;
- discrete, if they take values in a discrete set;
- continuous, if they take values in $\mathbb{R}$.

In order to enforce the intended meaning of $(*)$, we impose some constraints on x:

$$\forall i \leq n \quad \sum_{\substack{j \leq n \\ j \neq i}} x_{ij} = 1 \tag{1}$$

$$\forall j \leq n \quad \sum_{\substack{i \leq n \\ i \neq j}} x_{ij} = 1 \tag{2}$$

$$\forall \varnothing \subsetneq U \subsetneq [n] \quad \sum_{\substack{i \in U \\ j \in [n] \smallsetminus U}} x_{ij} \geq 1. \tag{3}$$

Notice that every tour is consistent with (3): if this were not the case, then $\mathtt{t}$ would define a strict subset U of $[n]$ disconnected from $[n] \smallsetminus U$, implying that the circuit is not Hamiltonian, against the assumption. Constraints (3) are also called *subtour elimination constraints*.

Question 4 Prove that if we define $\mathtt{t}$ by $\mathtt{t}(i) =$ the unique j s.t. $x_{ij} = 1$ and x satisfies (1)-(3), then $\mathtt{t}$ represents a tour. ◇

Now, in order to find the tour of minimum cost, we shall consider the objective function:

$$\min_x \sum_{i\le n} \sum_{\substack{j\le n\\ j\ne i}} d_{ij} x_{ij}. \tag{4}$$

A *mathematical programming formulation* for a given optimization problem is a set consisting of an objective function and possibly some constraints, all expressed as mathematical expressions of the data in problem instance and of a set of decision variables. The formulation of the TSP is:

$$\left.\begin{array}{rrcl} \min_x & \sum\limits_{i\le n} \sum\limits_{\substack{j\le n\\ j\ne i}} d_{ij} x_{ij} & & \\ \forall i \le n & \sum\limits_{\substack{j\le n\\ j\ne i}} x_{ij} & = & 1 \\ \forall j \le n & \sum\limits_{\substack{i\le n\\ i\ne j}} x_{ij} & = & 1 \\ \forall U \subsetneq [n] & \sum\limits_{\substack{i\in U\\ j\in [n]\smallsetminus U}} x_{ij} & \ge & 1 \\ \forall i \le n, j \ne i & x_{ij} & \in & \{0,1\}. \end{array}\right\} \tag{5}$$

Since all decision variables in Eq. (5) are binary and all mathematical expressions are linear in the decision variables, (5) belongs to the class of Binary Linear Programs (BLPs), which is a subclass of the Mixed-Integer Linear Programs (MILPs), i.e. problems with linear objective and constraints, and with a mixture of integer and continuous decision variables. A word of warning: optimization with respect to continuous variables makes the search domain potentially infinite. This may or may not be a problem according to the mathematical properties of the problem. For MILPs, we remark that fixing the integer variables yields a Linear Program (LP), for which it can be proved that the search domain can be restricted to be discrete (see e.g. the simplex method [3]).

The interest of Mathematical Programming (MP) as a formal language is that it allows to describe almost all optimization problems, and that there exist several *generic solvers* that target specific problem classes. In particular, MILPs are solved to optimality using a Branch-and-Bound (BB) algorithm.

In general, in combinatorial optimization problems we consider a set $\mathcal{F}$ of mathematical structures of interest (e.g. tours). After the optimization process acting on the corresponding mathematical programming formulation, the values assigned to the decision variables are supposed to encode elements of $\mathcal{F}$.

Definition 1 An assignment of values to the decision variables is a **feasible solution** if it satisfies all the constraints. ◇

Definition 2 A mathematical programming formulation describing elements of a set $\mathcal{F}$ is **valid** if every feasible solution encodes an element of $\mathcal{X}$ and if to every element of $\mathcal{X}$ there corresponds a feasible solution of the formulation. ◇

Definition 3 The **complexity** of a MILP formulation is given by the number of variables plus the number of linear constraints. ◇

Question 5 What is the complexity of the TSP formulation (5)? ◇

Since BB is a worst-case exponential algorithm, applying it to solve problem formulated with a more than polynomial complexity yields impractical CPU times. It is customary to only apply BB to formulations with polynomial complexity. Formulations with exponential complexity can sometimes be treated by more sophisticated techniques, such as dynamic row or column generation (if there are no integer variables), or Branch-and-Cut, Branch-and-Price or Branch-and-Cut-and-Price (if there are integer variables). See [3] for more details.

An alternative MILP formulation for the TSP [1], which we shall denote by (†), can be obtained from (5) by introducing continuous variables $y_1, \ldots, y_n$ and replacing (3) with:

$$\forall i, j \in \{2, \ldots, n\} \text{ with } i \neq j \quad y_i - y_j + (n-1)x_{ij} \quad \leq \quad n-2. \tag{6}$$

We remark that formulation (†) has the same objective as (5) but the minimization takes place with respect to both x and y variables.

Question 6 Prove that formulation (†) is valid for the TSP. ◇

Question 7 What is the complexity of the TSP formulation (†)? ◇

3 Branch-and-Bound

Formulation (†) is a MILP formulation that can be solved using BB. The binary variables x make a good candidate for branching. As in the lecture, select an arc $(i, j) \in A$ for branching: either (i, j) is in the tour, or it is not. In the lecture, however, the absence of a formulation made this disjunction relatively awkward to model: we had to resort to two sets of "forced" and "forbidden" arcs that grew during the search. In the case of MILP with binary decision variables, this is easier to do: simply consider the disjunction,

$$x_{ij} = 1 \quad \vee \quad x_{ij} = 0. \tag{7}$$

In the MILP setting, a solution is an assigment of values to the variables x, y of Formulation (†). A solution is feasible if it satisfies all the constraints, and optimal if it yields a minimum of the objective function. Let $\mathcal{F}$ be the set of feasible solutions of (†). A BB node is given by the list of variables whose value was fixed by branching on a disjunction (7), with the corresponding values.

As was explained in the lecture, certain BB nodes do not need to be explored further. Let $\mathtt{t}^*$ be the best tour found so far. If it can be determined in advance that no tour better than $\mathtt{t}^*$ (with cost d^*) can be reached by following the branches of the disjunction of the current nodes, then the BB node can be discarded. This is the case if we can compute a lower bound $\bar{d}$ to the optimal value of the tours compatible with the requirement of the current BB node, i.e. a subset of binary variables are set to 1 (those corresponding to arcs in the tour) and a subset to 0 (correponding to arcs not in the tour). Then, if $\bar{d} \geq d^*$, the current BB node can be discarded.

Question 8 Suggest a technique for computing a lower bound to the optimum of (†) compatible with the partial assignment of values to variables defined at each BB node. ◇

3.1 Combinatorial bounds

In the lectures we described a BB algorithm for the TSP which did not involve any MP formulation. Such BB methods are sometimes termed *combinatorial* BB, and the bound is a *combinatorial* bound. The strategy we employed in the lecture was the simplest possible: since any tour is also a map $[n] \to [n]$, we considered the map of minimum cost as a relaxed solution.

Since every tour in G is also a spanning subgraph of G containing a spanning tree, the minimum cost spanning tree (MST) yields another relaxed solution for tours.

Question 9 Point out a way to tighten the MST bound.
Hint: a tour has n arcs, whereas an MST has $n-1$ edges.

3.2 Finding good incumbents

The efficiency of BB method can be improved by pruning more nodes earlier on in the search. Since pruning occurs when the bound at a node is worst than the best feasible solution so far (i.e. the *incumbent*), finding good quality incumbents is important. Feasible tours of reasonably low cost can be found by heuristic algorithms. When one is able to guarantee a quality bound over the solution found by a heuristic, then the algorithm is also called an *approximation algorithm*.

Definition 4 An algorithm for a minimization problem yielding a solution with cost f' is a **k-approximation algorithm** if $f' \leq kf^*$, where f^* is the value of an optimal solution. $\diamond$

We are now going to discuss a $\frac{3}{2}$-approximation algorithm for the TSP, called *Christofides' heuristic*. This algorithm combines a MST and a matching to produce a tour. In order for Christofides' heuristic to be a $\frac{3}{2}$-approximation algorithm, we must assume that the arc cost function $d : V \to \mathbb{Q}_+$ defines a symmetric metric on $G = (V, A)$, i.e. for all $u, v, w \in V$ we have $d_{uv} + d_{vw} \geq d_{uw}$ and $d_{uv} = d_{vu}$.

Christofides' heuristic proposes a solution method for an **NP**-hard problem such as the TSP by exploiting two well-known combinatorial optimization problems that can be solved in polynomial time, namely the MST and the matching of minimum cost. The relationship between the TSP and the MST is that the latter is a relaxation of the optimum tour, as remarked in Question 9. As for the minimum cost matching, every tour of an even set of nodes can be decomposed into two disjoint matchings (see below).

We shall need the following definitions.

Definition 5 A **matching** in a graph $G = (V, E)$ is a subgraph $M = (V, F)$ of G such that each vertex in V is incident to at most one edge in F. A matching $M = (V, F)$ is **perfect** if every vertex of V is incident to exactly one edge in F. $\diamond$

Since our input digraph $G = (V, A)$ is complete and symmetric, we also consider it as a complete undirected graph (V, E). We informally define a **multigraph** as a graph where we allow more than one edge between pairs of vertices.

Definition 6 A **walk** in a multigraph $\mathcal{G}$ is a sequence of vertices $W = (v_1, \ldots, v_k)$ in $\mathcal{G}$ (possibly with repetitions) such that $\{v_i, v_{i+1}\}$ is an edge in $\mathcal{G}$ for all $i < k$. We let $\mathcal{E}(W) = \{\{v_i, v_{i+1}\} \mid 1 \leq i < k\}$ be the multiset of edges in the walk. A walk is **closed** if $v_1 = v_k$. A walk W is **Eulerian** if is it closed and each edge in the multiset of edges of $\mathcal{G}$ appears in $\mathcal{W}$ exactly once. $\diamond$

A word of caution: since a multigraph $\mathcal{G}$ has a multiset of edges, if an edge e appears twice in the edges of $\mathcal{G}$, then it must appear twice in the edge multiset $\mathcal{E}(W)$ of every Eulerian walk W of $\mathcal{G}$.

Let $W = (v_1, \ldots, v_k)$ be a Eulerian walk in a multigraph $\mathcal{G}$ whose set of vertex V is the same as that of the complete graph $G = (V, E)$ which is part of the TSP instance.

Question 10 Prove that any subsequence H of W which consists of distinct vertices and lists all vertices in V induces a tour of G with $d(H) \leq d(W)$.

In general, to a given Eulerian walk W of $\mathcal{G}$ there may correspond several tours H of G. Tours obtained this way are said to be *embedded* in W.

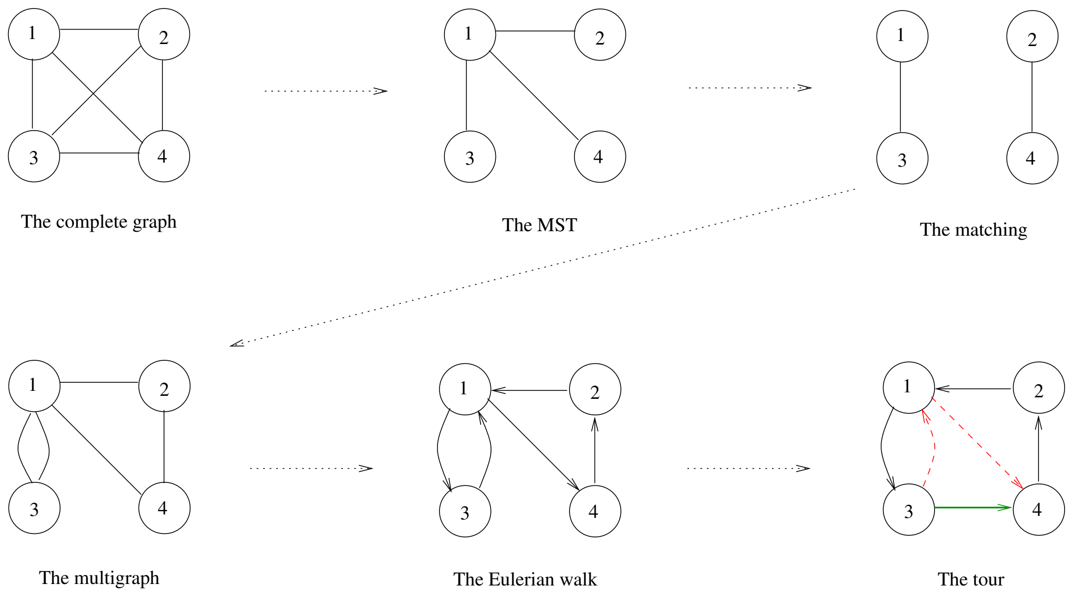
Here is Christofides' heuristic: notice it can be implemented to run in worst-case polynomial time.

1. Let $T = (V, S)$ be the minimum cost spanning tree of G (this can be done in time $O(n^2)$ in the worst case).
2. Let O be the set of vertices of V adjacent to an odd number of edges in S.
3. Let $M = (O, F)$ be a minimum cost perfect matching of O in G (this can be done in time $O(n^3)$ in the worst case, see Problem 14, Chap. 11 in [2]).

4. Let $\mathcal{G}$ be the multigraph on V obtained by considering the edge multiset $S\dot{\cup}F$, listing all edges in $S \cup F$, with all edges in $S \cap F$ appearing twice.

5. Let W be a Eulerian walk in $\mathcal{G}$ and H be a tour embedded in W (this can be done in time $O(|E|)$ in the worst case, see p. 412-413 in [2]). Return H.

Christofides' algorithm is shown graphically below.



Question 11 Prove that M exists, i.e. that the subgraph of G induced by the vertex subset O contains a perfect matching. $\diamond$

Question 12 Prove that W exists, i.e. that $\mathcal{G}$ contains a Eulerian walk. $\diamond$

Question 13 Prove that $d(H) \leq \frac{3}{2}d^*$, where d^* is the cost of the optimal tour t*.
Hints: the MST is a relaxation of the optimal tour, and every tour on a set of even cardinality decomposes into two disjoint matchings. $\diamond$

References

[1] C. Miller, A. Tucker, and R. Zemlin. Integer programming formulation of the traveling salesman problems. *Journal of the ACM*, 7:326–329, 1960.

[2] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, New York, 1998.

[3] L.A. Wolsey. *Integer Programming*. Wiley, New York, 1998.