Algorithmic Modeling and Complexity **Fall 2003**

Lecture 1: September 2, 2003

Lecturer: J. van Leeuwen *Scribe: S.R. van der Schuyt*

1.1 Overview

This first lecture started with some organisational details (not mentioned in these notes). This was followed by an introduction to the field of Algorithmic Modeling and a summary of the goals of this course. The last part of the lecture dealt with the Vehicle Routing Problem, *viz.* the Traveling Salesman Problem and the multiple Traveling Salesmen Problem as modeling challenges.

1.2 Introduction to Algorithmic Modeling

Algorithms are solutions to problems. But what are the associated problems? And how do we get from a problem to a solution? The answer is through algorithmic modeling and design, which is an important part of the *development cycle* of any system.

Algorithmic modeling is applied to a multitude of problems. All systems in computer science are 'algorithmic systems'. In this course the emphasis will lie on problems in the domain of programming and optimising *business processes*, in which field there are many challenging problems. Algorithmic modeling applied to business processes is widely applied in domains like *management science* and *operations research*.

Algorithmic modeling and design uses a more or less standard approach to solve algorithmic problems. In this approach there are different phases:

1. model the problem,
2. analyze the model,
3. predict the quality of the feasible solutions,
4. implement the solution of choice, and
5. integrate and/or scale the solution.

The result of this approach should be an effective, implemented (component of an) algorithmic system. In this course we will deal only with the first three phases.

Algorithmic systems is a prominent area of research world-wide. Many journals and conferences deal with this topic and the results of research are applied in many disciplines. Some examples:

- web modeling and the design of effective multi-media retrieval mechanisms,
- geographic information systems and their use in e.g. car navigation systems,
- discovery of patterns in large information bases ("data-mining"),
- techniques for the analysis and assembly of DNA strings,
- distributed handling and coordination of processes in ad-hoc mobile networks,
- planning and scheduling of activities, for example the planning of transports along a supply chain,
- optimal design of (large-scale) business processes,
- computational techniques for the support of management decisions,
- etc.

As one can see, not just computer scientists apply algorithmic modeling.

1.2.1 Objectives

The course has the following objectives:

- see the typical modeling tools of algorithmic design such as: networks, integer and mixed integer programming, logical formulae, distributed processes, transition systems, and Petri nets. The first four of these frameworks will certainly be dealt with in this course.
- learn to apply these tools on a variety of complex problems and use them in deriving optimal (or approximate) solutions which can be found in a reasonable amount of computational effort.
- learn to do this yourself.
- become acquainted with some areas of current research in algorithmic systems.
- learn to transfer knowledge of advanced research results.

1.3 The Vehicle Routing Problem in logistics

The problem of how to transport goods efficiently from producer to customer is ancient, but it was first studied as a problem in mathematics and computer science by Dantzig and co-workers in the 1950's. In 1959, Dantzig & Ramser for example published papers on *The Truck Dispatching Problem* and on the *Optimum Routing of Gasoline Trucks*. The general problem they studied is now widely known as the Vehicle Routing Problem (VRP). Some modeling tools will be demonstrated using this case.

1.3.1 Modeling the problem

The VRP consists of several components such as:

- *Depots.* At a depot, goods are picked up for delivery. There may be one or more of these and they have specific locations.
- *Vehicles.* There may be one or more vehicles. Each vehicle may have specific characteristics like a cost (for deploying it) and a 'capacity'.
- *Customers.* Customers all have an address or location. This will be a place where goods have to be picked up or where goods must be delivered. Customers may have demands (request for a certain amount of goods) but also goods may have to be picked up for delivery to the depot or to another customer. There may be time windows when goods can be picked up or delivered.
- *Routes and their costs.* This may include a specification of the allowable routes and such things as the costs associated with a route or the time it takes to travel it. Even stochastic information may come in like the expected travel time of a certain (part of a) route.
- *Constraints.* Vehicles may have a limited 'capacity', routes may be limited by driver schedules which may have to conform to legal requirements, customer demands may always have be met in full, time windows may be hard (no delivery outside tim window allowed) or soft (a penalty is incurred when delivery is outside time window), etcetera.

In addition, there are one or more *objective functions* associated with an instance of the VRP. This may include objectives like: minimize the total cost, minimize the total distance traveled by the vehicles, minimize the travel time, realize all demands and minimize the number of vehicles used, employ a fixed number of vehicles, etc.

In the VRP the notion of 'vehicle' is an abstraction of any moving entity: a truck, a passenger car, a robot, a person, a drill on a plate of metal, etc.

1.3.2 The underlying network

All locations (depots and customers) together form the nodes of a *network*, with edges representing the shortest/cheapest way of traveling from one node to another. We assume that for all locations i and j we know the cost c_{ij} of traveling directly from i to j ($c_{ij} \geq 0$). We also assume that there are no edges from any node i to itself. We can then distinguish different cases:

- the unrestricted or *asymmetric* case. Costs c_{ij} may take any value.
- the *symmetric* case. $c_{ij} = c_{ji}$, $1 \leq i \leq n$, $1 \leq j \leq n$, $i \neq j$, where n is the number of locations in the network.
- the *metric* case. The triangle equality ($c_{ik} \leq c_{ij} + c_{jk}$) holds for any locations i, j and k in the network.

- the geometric or *Euclidean* case. The locations are embedded in the 2-dimensional plane and the cost c_{ij} is equal to the Euclidean distance between i and j.

We remark that all inputs to a VRP (or any other algorithmic problem) must be *finitely specified*, e.g. as reals in finite precision or as elements of $\mathbb{Q}$.

Exercise. Argue that a Euclidean network is symmetric and metric.

We will now examine some special cases of the VRP.

1.4 The Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is the VRP with only a single depot and a single vehicle (with unrestricted capacity). In addition, for the TSP the goal is to design a tour of least cost, such that the tour starts and ends at the depot and each location in the network is visited exactly once (with the exception of the depot). Tours which satisfy these constraints are called "feasible" tours.

It would seem logical to enumerate all possible tours and choose the best one, but to enumerate all the tours one must consider $(n-1)!$ tours where n is the number of locations. Since $(n-1)!$ grows extremely quickly as $n \to \infty$ it is in most practical cases impossible to enumerate all the possibilities. Striving to find a good approximate solution is more effective.

Exercise. Show that $n! \geq (\frac{n}{e})^n$. (Hint: use that $e^x = 1 + x + \ldots + \frac{x^i}{i!} + \ldots$.)

The question then arises how different cases of the TSP are different and what they have in common.

Proposition 1.1 (Kanellakis and Papadimitriou, 1980) *Any asymmetric TSP on n locations can be reduced to a symmetric TSP on $3n$ locations.*

Proof: Let i and j be nodes in an asymmetric network G and c_{ij} and c_{ji} the cost to travel from i to j and from j to i respectively. We will create a network G' as follows. With every node i in G we create three nodes in G': i_1, i_2 and i_3, which are connected as follows:

$$i_1 — i_2 — i_3$$

For all i and j, the (directed) edge (i, j) with cost c_{ij} in G is replaced by a (undirected) edge (i_3, j_1) with cost c_{ij} in G'. The edges between i_1 and i_2 and between i_2 and i_3 are given associated cost zero. It is not desirable that the other edges incident to i_2 are ever used so they are assigned a cost that is so large that the edge will never appear in an optimum solution. For example n^2 times the largest cost of any edge in the original asymmetric TSP.

Thus, for every node i in G, all ingoing edges of i 'correspond' to edges now ending in i_1, all outgoing edges of i 'correspond' to edges now starting in i_3, but all going both ways (symmetric property of the new network).

Any feasible TSP tour of finite cost through the symmetric network G' now corresponds to a feasible TSP tour through the asymmetric network G (and vice versa) once we orient it so i_1 is visited before i_3 for some i. (Note that it follows by construction that then j_1 is visited before j_3 for every j.) Nodes i_2 are used to make sure that any route of finite cost that passes through i_1 must subsequently pass straight on to i_3 and vice versa, because this is the only opportunity to visit i_2. In this way the connection between nodes i and their counterpart in G' is enforced in the TSP tours. ■

Theorem 1.2 (Kumar and Li, 2002) *Any asymmetric TSP on n locations can be reduced to a symmetric TSP on $2n$ locations.*

It is open whether this result is tight.

The significance of this result is that if an efficient algorithm can be found for the symmetric case, then we will automatically have found one for the asymmetric case and vice versa. A disadvantage may be that in the translation the number of nodes increases, which may be very undesirable for keeping the running time of the algorithm within limits.

Exercise. Show that any (asymmetric or symmetric) TSP on n locations and no infinite costs can be transformed to a metric TSP on the same n locations. (Hint: replace costs c_{ij} by $c_{ij} + \Delta$ for Δ large enough.)

1.4.1 The m - TSP

There is a related problem to the one described above, namely the *multiple* Traveling Salesmen Problem (m-TSP). In this problem there is still only one depot, but there are m ($m \geq 1$) vehicles $v_0, \ldots, v_{m-1}$, where each v_i has a cost c_i. We assume that $c_0 \leq \ldots \leq c_{m-1}$.

For $r \leq m$, a set of r tours will be called 'feasible' if all tours in the set start and end at the depot, the tours only overlap at the depot and are otherwise disjoint, and each location (other than the depot) is visited exactly once (by one of the tours). Given a feasible set of r tours, the locations can be visited by dispatching r vehicles in parallel. In fact, it will be cheapest to dispatch $v_0, \ldots, v_{r-1}$. This motivates the following definition.

Definition 1.3 *The cost of a feasible set of r tours is equal to $\sum_{i=0}^{r-1} c_i$ plus the sum of the costs along the tours.*

The m-TSP asks to determine a feasible set of r tours (for some $r \leq m$) that has *least (total) cost*. Clearly, a solution to the m-TSP implies a way to dispatch a subset of the available vehicles simultaneously and supply all customers at least total cost.

The following fact is due to Bellmore & Hong (1974) for the asymmetric case and to Rao (1980) for the symmetric case.

Theorem 1.4 *Any m-TSP on n locations can be reduced to a standard TSP on $n + m - 1$ locations. (If the initial multiple Traveling Salesmen Problem has a symmetric underlying network then so will the resulting TSP.)*

Proof: We only consider the case of symmetric networks and assume w.l.o.g. that $m \geq 2$. To prove the result we will use 'implicit partitioning' of a TSP, by noting visits to (up to) m special locations.

Let G be the network associated with an m-TSP. Let G have $n-1$ locations ('the network') and a depot h_0. We now create a network G' as follows. Keep the network of locations but duplicate the depot $m-1$ times so that there are now m 'depot nodes' h_0, ..., h_{m-1}. Connect $h_1, \ldots, h_{m-1}$ to the network by duplicating the connections like h_0 has them. Finally, connect $h_0, \ldots, h_{m-1}$ in a 'backbone':

$$h_0 \text{ — } h_1 \text{ — } \ldots \text{ — } h_i \text{ — } \ldots \text{—} h_{m-1}$$

A feasible set of r tours $T_0, \ldots T_{r-1}$ for the m-TSP in G can easily be translated into a TSP tour in G', e.g. as follows. The tour starts at the original depot h_0, follows T_0 but instead of returning to h_0 the tour steps to h_{m-1}. The tour then goes up the backbone to h_{r-1}, enters the network following T_{r-1}, and then continues by returning to h_i and going back into the network following T_i, returning to h_{i-1} and so on, for i from $r-2$ down to 1. After traversing T_1 the tour returns to h_0 (rather than h_1) again. Clearly, there are many other ways of circling T_0, ..., T_{r-1} in G', by varying the points on the backbone where one returns.

Every time the corresponding TSP tour leaves the backbone and goes into the network again, a different vehicle is used in the m-TSP and the related cost must be charged. If the TSP tour passes through a number of h-nodes in succession, either the old or the new vehicle may be used but no extra cost should be incurred. These costs of using the vehicles need to be represented in the network G' so they are accounted for in the TSP tour. This is done as follows:

- add $\frac{1}{2}c_i$ to the cost of all edges leaving/entering a h_i into the network ($0 \leq i \leq m-1$), and
- assign to the edge between h_i and h_{i+1} ($0 \leq i \leq m-2$) a cost of $\frac{1}{2}c_i - \frac{1}{2}c_{i+1}$.

Claim 1.5 *Any feasible set of r tours for the m-TSP in G can be transformed into a TSP tour in G' of the same cost.*

Proof This follows from the given transformation. Note that the cost of entering with T_0 into h_{m-1} and moving up to h_{r-1} equals

$$\frac{1}{2}c_{m-1} + (\frac{1}{2}c_{m-2} - \frac{1}{2}c_{m-1}) + \ldots (\frac{1}{2}c_{r-1} - \frac{1}{2}c_r) = \frac{1}{2}c_{r-1}$$

and this is exactly half the cost for using v_{r-1}. Another $\frac{1}{2}c_{r-1}$ is picked up on the first edge leaving into the network following T_{r-1}. For the other vehicles v_i the cost is picked up when entering and leaving the backbone at h_i. For vehicle v_0 a cost of $\frac{1}{2}c_0$ is picked up at the beginning of the tour and another $\frac{1}{2}c_0$ is picked up at the end.

Conversely, any TSP tour in G' can be factored into a feasible set of r tours $T_0, \ldots T_{r-1}$ of the 'network', assuming the TSP comes into (or leaves) the backbone r times and all backbone-visits are collapsed as visits to h_0. (Note that all nodes of the backbone must eventually be visited in a TSP tour of G'.)

Claim 1.6 *Any TSP tour in G' can be transformed into a feasible set of r tours for the m-TSP in G of the same cost.*

Proof The cost of the r tours $T_0, \ldots, T_{r-1}$ in G is clearly the same as in G'. We only need a policy of assigning vehicles so their cost adds up to the cost the TSP incurs in G' for entering/leaving/traversing the backbone.

Orient the TSP tour so the first step away from the depot h_0 leads into the network. Do this using vehicle v_0: a cost of $\frac{1}{2}c_0$ is already picked up, and the remaining cost of $\frac{1}{2}c_0$ will be accounted for at the end of the tour. Proceeding inductively, suppose at any time in the tour we return into the backbone at some (unvisited) node h_i, using some vehicle v. Upon entering the backbone at node h_i a cost of $\frac{1}{2}c_i$ is picked up. Now three possibilities can occur:

Case 1: the TSP tour immediately steps back into the network again, to traverse the next 'subtour' of the network. Do this using vehicle v_i: the remaining cost of $\frac{1}{2}c_i$ is picked up when leaving into the network.

Case 2: the TSP tour moves up to a h_j (some $0 \leq j < i$) and steps into the network again from there (unless $j = 0$). Observe that the cost for entering into h_i and moving to h_j adds to:

$$\frac{1}{2}c_i + (\frac{1}{2}c_{i-1} - \frac{1}{2}c_i) + \ldots (\frac{1}{2}c_j - \frac{1}{2}c_{j+1}) = \frac{1}{2}c_j.$$

Do this using vehicle v_j. Also use it when leaving from h_j into the network: with the additional cost of $\frac{1}{2}c_j$ that is picked up, v_j is fully accounted for. If $j = 0$, the cost of moving to h_0 completes the remaining cost of $\frac{1}{2}c_0$ that was needed to account for v_0.

Case 3: the TSP tour moves down to a h_j (some $i < j \leq m-1$) and steps into the network again from there. Observe that the cost for entering into h_i, moving to h_j and stepping into the network adds to

$$\frac{1}{2}c_i + (\frac{1}{2}c_i - \frac{1}{2}c_{i+1}) + \ldots (\frac{1}{2}c_{j-1} - \frac{1}{2}c_j) + \frac{1}{2}c_j = c_i.$$

Do this using vehicle v_i!

This completes the dispatching strategy and proves the claim. Note that no vehicle is dispatched more than once.

The two claims together prove the theorem.

■

Observe that in the transformation one must apparently allow edges with negative cost. In a construction due to Hong and Padberg (1977) this is circumvented, at the expense of having $n + m + 4$ rather than $n + m - 1$ locations in the equivalent standard TSP.

Exercise. Consider the 'strict' m-TSP in which it is required that all m vehicles are used. Modify the given transformation to show that also the strict m-TSP on n locations can be reduced to a standard TSP on $n + m - 1$ locations. (Hint: do not connect the nodes in the backbone.)

Conclusion: if the standard TSP has a good solution, then this will translate to a good solution for the m-TSP. Unfortunately, there is *no known polynomial-time algorithm* that solves the TSP problem exactly.

References

[1] M. Bellmore and S. Hong. Transformation of multisalesmen problem to the standard traveling salesman problem. *J.ACM* 21 (1974) 500-504.

[2] S. Hong, M.W. Padberg. A note on the symmetric multiple traveling salesman problem with fixed charges. *Oper. Research* 25 (1977) 871-874.

[3] P-C. Kanellakis and C.H. Papadimitriou. Local search for the asymmetric traveling salesman problem. *Oper. Research* 28 (1980) 1086-1099.

[4] R. Kumar and H. Li. On asymmetric TSP: transformation to symmetric TSP and performance bound. Submitted to *J. Oper. Research* (2002).

[5] M.R. Rao. A note on the multiple traveling salesmen problem. *Oper. Research* 28 (1980) 628-632.