Algorithmic Modeling and Complexity **Fall 2003**

Lecture 2: 4 September

Lecturer: J. van Leeuwen *Scribe: M. Mahabiersing*

2.1 Overview

The Vehicle Routing Problem is a classic problem in algorithmic modelling. The problem is computationally very hard and is often approached using faster, but inexact approximation algorithms. This lecture was to get acquainted with different *qualities of approximation.* Two special cases of the VRP were discussed: the Euclidean TSP and the m-TSP with capacities.

2.2 Performance Ratio

Consider instances I of an optimization problem like the VRP. Let the optimum solution of an instance I have value $Opt(I)$. Suppose we have an algorithm $\mathcal{A}$ that produces feasible solutions with value $Sol(I)$.

Definition 2.1 *The performance ratio of $\mathcal{A}$ is (any bound on) $\alpha(n) = \max_{|I|=n}\{\frac{Sol(I)}{Opt(I)}, \frac{Opt(I)}{Sol(I)}\}$.*

For a minimization problem like the TSP, an approximation algorithm has a performance ratio of (say) 2 if it always delivers feasible solutions that are within a factor of 2 from optimum: $Sol(I) \leq 2 \cdot Opt(I)$. If optimum solutions are hard to compute, one would like to hope for an efficient approximation algorithms that have a performance ratio $\alpha(n) \to 1$ for $n \to \infty$.

Unfortunately, for the general TSP and thus for the general VRP, *there is NO known efficient i.e. polynomial time-bounded algorithm that solves the problem within a performance ratio bounded by a constant*, in fact not even with a performance ratio bounded by any polynomial in n, the number of locations.

In special cases of the VRP there are ways of approximating the optimum solutions with certain performance guarantees.

2.3 The symmetric, metric TSP

Consider the TSP problem on a symmetric, metric network, i.e. assume $c_{ij} = c_{ji}$ and the triangle inequality. Let the optimum TSP-tour in a given instance have length OPT.

Observation: Deleting one edge from an optimum TSP-tour gives a spanning tree with a cost $\leq OPT$.

This observation can be used in a 'fast' approximation algorithm for solving the TSP as follows.

Algorithm S

Step 1. Determine a minimum spanning tree T with the depot as the root.

Step 2. Make a depth-first traversal of the tree T and when backtracking over visited nodes, 'shortcut' the traversal to the first unvisited node in the traversal order.

Step 3. Stop when the traversal ends, at the root.

Step 1 follows easily with a standard low polynomial-time algorithm. Step 2 (and step 3) can be implemented in linear time.

Proposition 2.2 *Algorithm S computes a feasible tour and this tour has a length* $\leq 2 \cdot OPT$.

Proof: Feasibility is clear. By the observation, T has length $\leq OPT$. Without shortcuts, the traversal has a length $\leq 2 \cdot OPT$. The metric property implies that shortcuts can only shorten the tour. ■

The proposition shows that the symmetric, metric TSP can be solved with a constantly bounded performance ratio: algorithm S achieves a performance ratio ≤ 2.

Theorem 2.3 (Christofides, 1976) *The symmetric, metric TSP can be solved in polynomial time within a performance ratio* $\leq \frac{3}{2}$.

The $\frac{3}{2}$ is still the best known result in the symmetric, metric case. We note that the best performance ratio for the *a*symmetric, metric TSP is essentially $\log n$, shown by Frieze *et al.* in 1982. Improvements appear in Kumar and Li [7].

2.4 The Euclidean TSP

In the important case of the Euclidean TSP ('the TSP in the plane') it appears possible to obtain a better result. We show that in this case one can get a performance ratio 'very close' to 1.

2.4.1 Preliminary bound

Scale the problem so that all n locations fit in the unit square. We will not rely on computing euclidean distances between locations, as this would require infinite precision computing.

Lemma 2.4 *The n locations in the unit square can be visited in a feasible tour of length* $\leq \sqrt{2} \cdot \sqrt{n} + \frac{\sqrt{2}}{2} + 2$.

Proof: Divide the unit square in $\frac{\sqrt{n}}{c}$ horizontal strips of thickness $\frac{c}{\sqrt{n}}$, for some c to be determined later. Now design two tours as follows.

Tour I. Starting a distance of $\frac{c}{2\sqrt{n}}$ below the top left corner of the square, follow the midline of the first strip from left to right. While doing so, visit each location in the strip that is passed by moving at most $\frac{c}{2\sqrt{n}}$ up or down and back. At the end of the first strip, move down $\frac{c}{\sqrt{n}}$ down, and traverse the midline of the second strip, this time moving from right to left. Continue sweeping back and forth through the strips until the lowest strip has been traversed. Then go back in the straight line to the starting point, either along the leftmost edge of the unit square or along the diagonal (if the traversal ended in the bottom right corner). The visits to the locations are not left as side-steps in the traversal but shortcut to straight lines. This only shortens the tour. The length of tour I is: #strips · horizontal distance + vertical distance + distances to locations and back + return to starting point $\leq \frac{\sqrt{n}}{c} \cdot 1 + 1 + n \cdot 2 \cdot \frac{c}{2\sqrt{n}} + \sqrt{2}$.

Tour II. The same as the first, but this time the tour starts at the top left corner, and we traverse the top-edge of each strip. Again locations are visited as they are passed within a distance of at most $\frac{c}{2\sqrt{n}}$ by moving the appropriate distance down or up and back again. At the end one needs to traverse the bottom edge of the lowest strip, to properly complete before return ing to the top left corner. The length of tour II is: #strips · horizontal distance + vertical distance + distances to locations and back + traversal of bottom edge + return to starting point $\leq \frac{\sqrt{n}}{c} \cdot 1 + 1 + n \cdot 2 \cdot \frac{c}{2\sqrt{n}} + 1 + \sqrt{2}$.

Note that the length of returning to the starting point (defined as $\sqrt{2}$ here) is actually 1 in one of the tours. Which tour that is depends on the number of strips.

By adding the length of the tours, we can save on the estimate for visiting the locations. Since the return length of 1 tour is 1, the return length of the 2 tours together is at most $\sqrt{2}+1$. Also if one tour visits a location 'from below', the other tour will visit it 'from above' and vice versa. Thus $2 \cdot \frac{c}{2\sqrt{n}}$ is is actually a bound on the *sum* of the two terms in tour I and II, for each location. Hence:

$$\text{tour I} + \text{tour II} \leq \frac{\sqrt{n}}{c} \cdot 2 + 2 + n \cdot 2 \cdot \frac{c}{2\sqrt{n}} + \sqrt{2} + 1 + 1 = (\tfrac{2}{c} + c)\sqrt{n} + \sqrt{2} + 4.$$

For $c = \sqrt{2}$, the average length of the two tours is: $\sqrt{2} \cdot \sqrt{n} + \frac{\sqrt{2}}{2} + 2$, which means that one of the tours must be shorter than this. ■

2.4.2 Karp's heuristic for the Euclidean TSP

Karp's *partitioning scheme* now works as follows. We continue with the instance of n locations in the unit square. Take a value s such that $s! \leq n$. (Show that one can take e.g. $s = \frac{\log n}{2 \log \log n}$.)

Algorithm K

Step 1. Draw $\sqrt{\frac{n}{s}}$ vertical strips containing $\sqrt{ns}$ points each. Also draw $\sqrt{\frac{n}{s}}$ horizontal strips *within* each vertical strip, with each strip containing s points. (This partitions the unit square into $\frac{n}{s}$ 'blocks' of s points each.)

Step 2. Solve the TSP problem within each block of s points exactly e.g. using an algorithm that enumerates all $s!$ tours.

Step 3. Choose an arbitrary point in each block (thus $\frac{n}{s}$ points in total). Connect these points in a tour as given in Lemma 2.4. Combine this tour with the tours in the blocks, shortcutting other traversal to obtain a feasible tour.

By the choice of s, algorithm K runs in *polynomial time.* Let the resulting tour be T and let its length be: $c(T)$. Let the optimum TSP-tour through the n points have length OPT.

Theorem 2.5 (Karp, 1977) *The tour T computed by algorithm K satisfies: $OPT \leq c(T) \leq OPT + O(\sqrt{\frac{n}{s}})$.*

Proof: Let U be any optimum TSP-tour, $c(U) = OPT$. Consider any individual block B_i. Let U_i be the (length of the) segments of U inside B_i. Let OPT_i be the (length of the) optimal tour through the points in block B_i as computed by algorithm K. Also, let $P(B_i)$ be the perimeter of B_i.

Claim 2.6 $OPT_i \leq U_i + \frac{3}{2}P(B_i)$.

Proof We complete U_i to a closed tour through the s locations inside B_i as follows. Let $x_1, \ldots, x_k$ be the points in clockwise order around the perimeter of B_i where U_i enters/leaves B_i. Note that k is *even.* Consider the graph consisting of $x_1, \ldots, x_k$ and the locations inside B_i, with edges as in U_i. Make the graph connected by adding the the segments between the consecutive x_i's along the perimeter of B_i as edges. The locations all have degree 2, but the x_i's all have degree 3.

Now observe that there are k edges along the perimeter and thus the x_i's can be matched in pairs so the total length of the $\frac{k}{2}$ matched edges is $\leq \frac{1}{2}P(B_i)$. Add the $\frac{k}{2}$ matched edges to the graph, thus 'duplicating' these edges. It means that now the x_i's all have degree 4, i.e. even degree as well. The total length of the edges in the graph is now: $U_i + P(B_i) + \frac{1}{2}P(B_i) = U_i + \frac{3}{2}P(B_i)$.

Because all nodes in the graph have even degree, the graph admits an Eulerian cycle (a cycle that traverses all edges exactly once). Doing the Eulerian traversal in the graph but shortcutting over nodes that were already visited and over the nodes $x_1, \ldots, x_k$ leads to a feasible tour of length $\leq U_i + \frac{3}{2}P(B_i)$ *inside* B_i visiting all locations. The claim follows.

Now estimate $c(T)$ as follows, using Lemma 2.4 and the claim.

$$c(T) \leq \sum_{i=1}^{\frac{n}{s}} OPT_i + \sqrt{2} \cdot \sqrt{\frac{n}{s}} + O(1) \leq \sum_{i=1}^{\frac{n}{s}} U_i + \frac{3}{2} \sum_{i=1}^{\frac{n}{s}} P(B_i) + \sqrt{2} \cdot \sqrt{\frac{n}{s}} + O(1)$$
$$\leq c(U) + \frac{3}{2}(2\sqrt{\frac{n}{s}} + 2\sqrt{\frac{n}{s}}) + \sqrt{2} \cdot \sqrt{\frac{n}{s}} + O(1) = OPT + O(\sqrt{\frac{n}{s}}),$$

where we note that $\sum_{i=1}^{\frac{n}{s}} P(B_i)$ amounts to counting twice the full width and height of the unit square for each strip worth of (horizontal c.q. vertical edges of) $\sqrt{\frac{n}{s}}$ blocks. ∎

Theorem 2.5 says at best that the tour computed by algorithm K is within a distance of $O(\sqrt{n})$ from optimum. It does not yet say something about the performance ratio.

Theorem 2.7 (Karp, 1977) *The 'expected' performance ratio of algorithm K converges to 1 for $n \to \infty$.*

Proof:

It can be shown that the 'expected' length of a TSP through n uniformly distributed locations in the unit square is $\geq \beta \cdot \sqrt{n}$, for some constant β. Thus

$$\frac{c(T)}{OPT} \leq (OPT + O(\sqrt{\frac{n}{s}})/OPT \leq 1 + \frac{\gamma}{\sqrt{s}}$$

for some constant γ. This proves the result. ■

2.4.3 Arora's algorithm scheme for the Euclidean TSP

Karp's result was the best result, and a remarkable one, for the Euclidean TSP for almost twenty years. It still stands as a powerful technique. More recently the partitioning scheme was considerably refined so as to achieve a guaranteed performance ratio as close to 1 as one would want to have it, still with a polynomial time algorithm.

Theorem 2.8 (Arora, 1996) *There is an algorithm $\mathcal{A}$ operating on pairs I, ϵ such that for any fixed $\epsilon > 0$, $\mathcal{A}$ solves Euclidean TSP instances I within a performance ratio $1+\epsilon$, in time polynomial in $n = |I|$.*

Arora's algorithm typically has a running time in the order of $n^{\frac{1}{\epsilon}}$.

2.5 The m-TSP

Unfortunately in many instances of the VRP, it is considerably harder to achieve approximations with very good performance guarantees. In this case one often resorts to *heuristics*: algorithms that do well in practice but for which we have no absolute guarantees.

We examine the 'capacitated' m-TSP, with 1 depot and m vehicles with 'capacity' Q. The problem is to devise $\leq m$ feasible tours that cover all locations with least total cost, where a tour is called feasible if it can be serviced by a vehicle of capacity Q. We do not assume symmetry of the costs for traversing edges or anything.

2.5.1 Clarke-Wright savings heuristic

Clarke and Wright (1964) proposed an approach that can be adapted to various versions of the VRP. In the case of the m-TSP, the Clarke-Wright heuristic maintains a set R of tours that satisfies the following *invariant*:

- every tour in R is feasible,
- the tours in R overlap *only* in the depot and are otherwise disjoint, and
- the tours in R jointly cover all locations.

An initial set R is easily constructed. Start with the empty set, and for each location i, add the tour that goes from the depot to i and back. If we can combine tours so eventually $|R| \leq m$, then the first moment this happens (if it happens at all) one has at least a feasible solution to the m-TSP! The quality will depend on the *combine-rule* for tours.

Let h_0 denote the depot. Consider two disjoint, feasible tours: $h_0 - i_1 - I - i_p - h_0$ and $h_0 - j_1 - J - j_q - h_0$. The tours can be combined to e.g.

$$h_0 - i_1 - I - i_p - j_1 - J - j_q - h_0$$

with a *savings* in cost of $c_{i_p h_0} + c_{h_0 j_0} - c_{i_p j1}$. Similar cross-overs at the depot between the first and the second tour can be made, with a similar calculation of the savings.

> *Combine rule:* Given two disjoint feasible tours I and J, combine them into one of more new tours $T = [I, J]$ as above *provided* T is feasible. The savings of a new feasible tour T is the difference in edge cost between $I + J$ and T.

In addition to the set R, the Clarke-Wright heuristic maintains the set S of all combinations of tours in R as given by the combine-rule. Every combined (feasible) tour is specified in terms of the two constituent tours, the way they are combined, and the subsequent savings. Note that combinations that lead to infeasible tours (which may happen in case of e.g. finite Q), are not included in S.

Given the initial set R defined above, the set S is initialized to the set of all tours $h_0 - i_1 - j_1 - h_0$ and $h_0 - j_1 - i_1 - h_0$, with the corresponding savings marked over $h_0 - i_1 - h_0$ and $h_0 - j_1 - h_0$, respectively. (Note that we do not assume symmetry so order counts.) In the course of the algorithm, S may contain combinations of tours that have already been combined with other tours in the meantime. The following formulation of the algorithm takes this into account.

Clarke-Wright savings heuristic

Initialize R and S as above.

while $S \neq \emptyset$ **do**

- extract the potential combination $[I, J]$ from S with the maximum saving
- **if** I and J are still in R **then**
  - delete I and J from R
  - add all results of combining $[I, J]$ with a tour from R in a feasible way as in the combine-rule to R
  - add the combined tour $[I, J]$ to R
- **else** delete $[I, J]$ from S

until $|R| = m$

Lemma 2.9 *The Clarke-Wright savings heuristic always terminates and preserves the invariant for R.*

Proof: In each iteration of the **while**-loop, either $|S|$ or $|R|$ reduces in size by one. ∎

If upon termination one has $|R| = m$, then the tours in R form a feasible solution to the m-TSP that can be expected to be 'optimized' (but not necessarily optimal). The Clarke-Wright is reputed to do well in practice. To improve it, some implementations 'bias' the cost of T in the combine-rule by a factor > 1.

Lemma 2.10 *The Clarke-Wright savings heuristic can be implemented in* $O(n^2 \cdot \log n)$ *time.*

Proof: Implement S as a heap. S has $O(n^2)$ elements initially, and at most $O(n)$ elements can be added to it each time a new tour is formed. When a new tour is formed, R reduces in size by 1 and this can happen at most $n - m = O(n)$ times. Thus S will remain of size $O(n^2)$ throughout.

Extractions can be done in order $\log n^2$ thus $O(\log n)$ time. In the worst case all $O(n^2)$ potential element of S will eventually be extracted. The combine-rule is called at most n times, and each call involves combining a new tour with the $\leq n$ tours currently in R. This accounts for a total of $O(n^2 \cdot \log n)$ time. ∎

2.5.2 Clarke-Wright savings heuristic for the TSP

Taking $m = 1$ and $Q = \infty$, the Clarke-Wright heuristic become a heuristic for the ordinary TSP.

Theorem 2.11 (Ong and Moore, 1984) *The Clarke-Wright savings heuristic applied to the symmetric, metric TSP has a performance ratio of* $\log_2 n + 1$.

References

[1] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In: *Proc. 37th Ann. IEEE Symposium on Foundations of Computer Science* (FOCS'96), 1996, pp. 2-11.

[2] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. *Techn. Report* 388, Grad. School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1977.

[3] G. Clarke and J.W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Research* 12 (1964) 568-581.

[4] A.M. Frieze, G. Galbiati, and F. Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks* 12 (1982) 23-39.

[5] C. de Jong, G. Kant, and A. van Vliet. Efficient implementations of the savings method for the vehicle routing problem with time windows. *Manuscript*, Department of Computer Science, Utrecht University , 1996.

[6] R.M. Karp. Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane. *Math. Oper. Research* 2 (1977) 209-224.

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

[8] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (Eds). *The traveling salesman problem - A guided tour of combinatorial optimization.* John Wiley & Sons, Chichester, 1985.

[9] H.L. Ong, J.B. Moore. Worst-case analysis of two travelling salesman heuristics. *Oper. Res. Lett.* 2 (1984) 273-277.