Solution of a Min-Max Vehicle Routing Problem

David Applegate
Algorithms and Optimization Department
AT&T Labs – Research

William Cook*
Applied and Computational Mathematics
Princeton University

Sanjeeb Dash
Applied and Computational Mathematics
Princeton University

André Rohe
Forschungsinstitut für Diskrete Mathematik
Universität Bonn

August 14, 2001

We use a branch-and-cut search to solve the Whizzkids'96 vehicle routing problem, demonstrating that the winning solution in the 1996 competition is in fact optimal. Our algorithmic framework combines the LP-based traveling salesman code of Applegate, Bixby, Chvátal, and Cook, with specialized cutting planes and a distributed search algorithm, permitting the use of a computing network located across Rice, Princeton, AT&T, and Bonn. The 1996 problem instance was developed by E. Aarts and J. K. Lenstra, and the competition was sponsored by the information technology firm CMG and the newspaper De Telegraaf.

(Combinatorial Optimization; Vehicle Routing; Integer Programming; Distributed Computing)

1 The Test Instance

A wide range of practical problems involve sets of customers that need to be served by vehicles located at a common depot; models that optimize the selection of routes to the customers are referred to as *vehicle routing problems.* To capture different aspects of routing applications, a variety of models have been studied in the literature, starting with the work of Dantzig and Ramser (1959). Surveys of vehicle routing research can be found in the books by Golden and Assad (1988), Ball, Magnanti, Monma, and Nemhauser (1995), and Toth and Vigo (2001).

*Supported by ONR Grant N00014-01-1-0058 and Texas ATP Grant 003604-0034-1999.

We consider a vehicle routing problem proposed by E. Aarts and J. K. Lenstra as part of the Whizzkids'96 (1996) mathematics challenge, sponsored by the information technology firm CMG and the newspaper De Telegraaf. We refer to the problem as the *newspaper routing problem*; the Whizzkids'96 announcement discusses the delivery of newspapers to customers. The goal of this problem is to ensure that all customers are served as soon as possible. Assuming that travel-time is proportional to distance traveled, and that a negligible amount of time is spent at a customer location, the above goal is achieved by minimizing the length of the longest route, starting from the common depot, taken by a vehicle. This contrasts with the more typical provider-centric objective of minimizing the total distance traveled by the fleet.

The newspaper routing instance specified in the Whizzkids'96 competition consists of a set of 120 customers to be served by 4 vehicles. The locations of the customers and the depot are given as (x, y)-coordinates, and the travel-time from a point (x_1, y_1) to a point (x_2, y_2) is proportional to the Manhattan distance $|x_1 - x_2| + |y_1 - y_2|$ between the points. A solution to the problem consists of 4 paths starting at the depot such that each of the 120 customer locations is on one of the 4 paths; the objective is to minimize the maximum of the lengths of the 4 paths. In graph-theory terms, the instance can be specified by the complete graph on 121 nodes, with a distinguished depot node and with edge lengths given by the Manhattan distance between the corresponding points; a solution consists of 4 internally disjoint simple paths starting at the depot and covering all nodes in the graph.

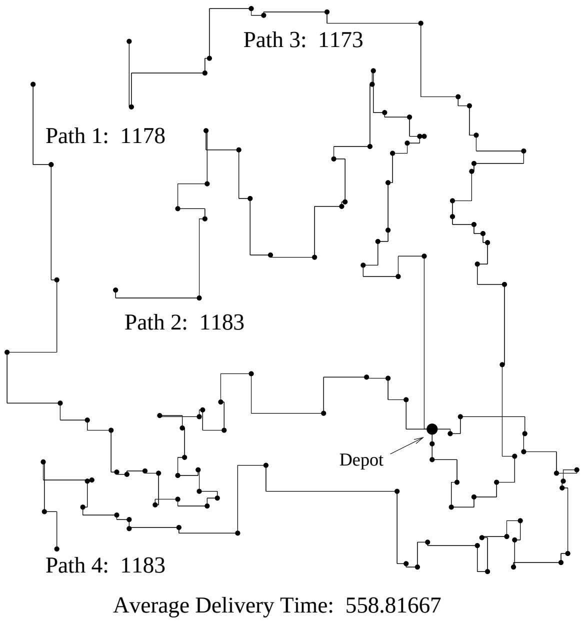

Figure 1: A Routing Solution

The min-max objective function in the newspaper routing problem can permit considerable freedom in the selection of the shorter of the paths. Keeping in the spirit of a customer-service model, a secondary objective in the Whizzkids'96 competition was to minimize the average delivery time to the 120 customers; this secondary objective was used as a tie-breaking rule. The winning solution was obtained by T. Hemel, S. van Erk, and P. Jenniskens (1996) from Technische Universiteit Eindhoven using a simulated annealing algorithm. Their solution is illustrated in Figure 1; it has maximum path-length 1,183 and average delivery time 558.81667.

After the Whizzkids'96 competition, a 5,000 Dutch Guilder prize was offered by CMG for the first group to produce either a solution with objective value less than 1,183 or a proof that 1,183 is optimal. In this paper we describe a branch-and-cut search that we use to verify that 1,183 is indeed the optimal value. We also verify that the 558.81667 average delivery time is optimal among all 1,183-valued solutions. CMG was kind enough to deliver the 5,000 Guilder prize in March 2001, based on a preliminary report of our work.

The paper is organized as follows. In Sections 2 and 3 we describe the linear programming (LP) model we use to obtain lower bounds for the newspaper routing problem. In Section 4 we present the details of the computation used to prove that 1,183 is the optimal value for the challenge problem, and in Section 5 we describe how we extended the search algorithm to prove that 558.81667 is the optimal average delivery time. Finally, in Section 6 we give some remarks on the general methods used in the study.

We assume that the reader is familiar with discrete optimization solution methods, including cutting-plane algorithms and branch-and-bound. A treatment of this area can be found in the books of Nemhauser and Wolsey (1988) and Wolsey (1998).

2 TSP with Side Constraints

The most well-studied model in vehicle routing is the *traveling salesman problem* (TSP), where we have a single vehicle that must visit all customers and return to the depot, with the objective being to minimize the total cost of travel. A treatment of the TSP can be found in the book Lawler, Lenstra, Rinnooy Kan, and Shmoys (1985).

A common strategy in the design of solution methods in vehicle routing is to take advantage of the large body of work which has gone into the TSP. This is often accomplished by transforming the problem into a TSP-like problem by splitting the depot into several copies and allowing the separate routes of the vehicles to be combined into a single tour. We will employ this technique in our study of the newspaper routing problem.

As a first step, considering the problem in terms of graphs, we add two dummy nodes that are joined to each of the customers and to the depot, setting the cost of all edges meeting the dummies to 0. Next we create a copy of the depot which is joined to each of the customers and to the two dummy nodes with edges having the same cost as the corresponding edges joining the original depot node to the customers and dummies. Let G denote this expanded graph. Now any solution of the original newspaper routing problem can be extended to a TSP tour in G, as illustrated in Figure 2. Here the cost of the tour equals the sum of the lengths of the 4 paths in the newspaper routing solution. Notice, however, that there exist tours in G that do not correspond to solutions to the newspaper routing problem, as we indicate in Figure 3. We describe below a set of TSP side constraints that forbid these bad tours.

We will work with the following standard LP relaxation of the TSP. Let E denote the edge-set of G and let V denote the set of nodes. For each edge e, let c_e denote its cost and let x_e be a variable (with the interpretation that $x_e = 1$ if e is in the tour and 0 otherwise).

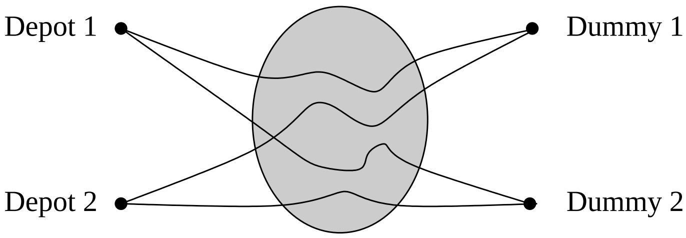

Figure 2: Reduction to Constrained TSP

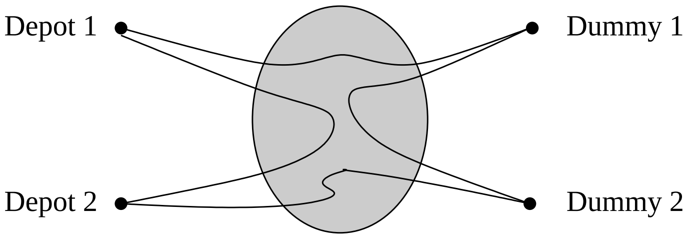

Figure 3: A Bad Tour in the Expanded Graph

For any subset S of nodes, let $\delta(S)$ denote the set of edges having one end in S and the other end not in S. The Dantzig, Fulkerson, and Johnson (1954) relaxation of the TSP is

$$\begin{gathered}\text{Minimize } \sum(c_e x_e : e \in E)\\ \text{subject to}\\ \sum(x_e : e \in \delta(\{v\})) = 2, \text{ for all } v \in V\\ \sum(x_e : e \in \delta(S)) \geq 2, \text{ for all } S \subset V,\ \emptyset \neq S \neq V\\ 0 \leq x_e \leq 1, \text{ for all } e \in E.\end{gathered} \tag{1}$$

The $0/1$ solutions of the constraint set of (1) are precisely the set of TSP tours through the nodes of the graph. The inequalities $\sum(x_e : e \in \delta(S)) \geq 2$ are called *subtour constraints*.

In our expanded newspaper graph G, we denote the depot nodes by d_1 and d_2 and we denote the dummy nodes by a_1 and a_2. The bad tours are those that include a subpath from d_1 to d_2, without passing through either of a_1 or a_2. These tours can be forbidden by including the additional constraints

$$\sum(x_e : e \in \delta(S)) \geq 4, \text{ for all } S \text{ with } \{d_1, d_2\} \subseteq S,\ \{a_1, a_2\} \cap S = \emptyset, \tag{2}$$

that is, for each set of nodes that contains the two depots and neither dummy, we require that at least 4 edges (in a $0/1$ solution) cross the boundary formed by the set.

The difficulty with this approach to the newspaper routing problem is that we cannot express the min-max objective with a linear function of the x_e variables. Hurkens (1997) showed, however, that we can directly handle the feasibility question of whether or not there exists a newspaper routing solution of value less than or equal to B, where B is any constant. This can be accomplished by adding the constraints

$$\sum(x_e : e \in \delta(S)) \geq 4, \text{ for all } S \in \mathcal{H}_B, \tag{3}$$

where $\mathcal{H}_B$ denotes the collection of all subsets S of customer nodes such that the shortest path starting at a depot and passing through each node in S has cost greater than B (one vehicle takes too long to serve S). These constraints forbid long paths from appearing in 0/1 solutions to the model.

The min-max objective

To handle the min-max objective, we describe a layered graph that permits us to distinguish between the 4 paths in the newspaper routing solution. For each customer v we create nodes v_i^{in}, v_i^{out} for $i = 1, \ldots, 4$, and for each pair of customers v, w we create edges (v_i^{in}, w_i^{out}) and (v_i^{out}, w_i^{in}) for $i = 1, \ldots, 4$, giving each edge the same cost as the original edge (v, w). For each customer v and for each $i = 1, \ldots, 4$, we add edges $(v_i^{out}, v_{i+1}^{in})$ (where the subscripts are taken modulo 4) and (v_i^{in}, v_i^{out}), both with cost 0; we fix the variables corresponding to each of the edges $(v_i^{out}, v_{i+1}^{in})$ to 1. We next create four copies of the depot node $d_1, \ldots, d_4$, along with four dummy nodes $a_1, \ldots, a_4$. For $i = 1, \ldots, 4$ and for each customer v, we create the edge (v_i^{in}, d_i) with the same cost as the original edge joining v to the depot, and we create the edge (v_i^{out}, a_i) with cost 0. For $i = 1, \ldots, 4$, we create the edges (d_i, a_i) and (a_i, d_{i+1}) (where the subscripts are taken modulo 4), each with cost 0. Finally, we fix the variables corresponding to the edges (a_i, d_{i+1}) to 1.

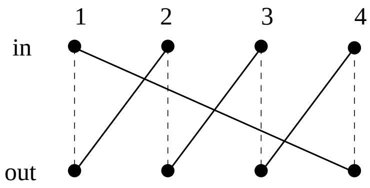

Figure 4: Edges for a Set S^v

In the layered graph we have constructed, each customer node v corresponds to a set of 8 nodes, which we denote by S^v; the edges internal to S^v are indicated in Figure 4, where the solid edges are fixed to 1. To complete the model we add the constraints

$$\sum (x_e : e \in \delta(S^v)) = 2, \text{ for all customer nodes } v. \tag{4}$$

Under (4), any tour in the layered graph corresponds to a collection of paths from the depots to the dummies, with one path p_i on each layer i, for $i = 1, \ldots, 4$. To see this, notice

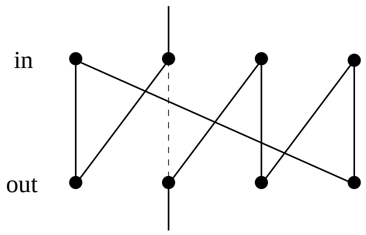

Figure 5: Route Through S^v

that if a tour contains an edge in $\delta(S^v)$ meeting the node v_i^{in}, then it must also contain an edge in $\delta(S^v)$ meeting node v_i^{out} (see Figure 5). So paths on a given layer go through pairs of nodes v_i^{in}, v_i^{out} until they reach a depot or a dummy.

For each $i = 1, \ldots, 4$, let E_i denote the set of edges having both ends in layer i of the graph. By adding the constraints

$$\sum(c_e x_e : e \in E_i) \leq \sum(c_e x_e : e \in E_4), \text{ for } i = 1, 2, 3, \tag{5}$$

we can force path p_4 to have the maximum cost among $p_1, \ldots, p_4$. Using these constraints, we can express the min-max objective as

$$\text{Minimize } \sum(c_e x_e : e \in E_4).$$

This completes the description of the layered model.

Implementing the models

In Hurkens' model of the feasibility problem and in the layered model of the optimization problem, the solution sets (that is 0/1 solutions to the LP problems) consist of special TSP tours through the node sets of the graphs. Therefore, in each case we can improve the LP relaxations by including any inequalities that are known to be valid for all TSP tours. Surveys of classes of such TSP inequalities can be found in Jünger, Reinelt, and Rinaldi (1995) and in Naddef (2001).

In our computations we use the *Concorde* TSP code of Applegate, Bixby, Chvátal, and Cook (1998), permitting Concorde to generate TSP inequalities from its library of cutting-plane routines. In the layered model, it is straightforward to incorporate the side constraints, since we can explicitly add the equations (4) to the initial LP relaxation created for the TSP. Fixing the appropriate variables to 1 is also easy. The feasibility model is more difficult, however, since the constraints (2) and (3) consist of an exponential number of distinct inequalities; these two classes of constraints will be treated implicitly, via a cutting-plane approach.

A *cut* in a graph is a set of edges of the form $\delta(S)$ for some proper non-empty subset of nodes S; the weight of a cut is the sum of the weights of its edges. To handle (2) and (3), we use minimum-cut algorithms where the weight on each edge e is the value of x_e in the current LP solution. In the case of constraints (2), we compute the minimum weight cut $\delta(S)$ with $d_1, d_2 \in S$ and $a_1, a_2 \notin S$ by shrinking the two depots into a single node d, shrinking the two dummies into a single node a, and then using a maximum-flow-minimum-cut algorithm with source d and sink a. If the cut $\delta(S)$ we find has weight less than 4, then we can add the corresponding constraint (2) as a cutting plane. Otherwise, all constraints (2) are satisfied by the LP solution.

For constraints (3), we use the minimum-cut algorithm of Karger (1993) to sample candidate sets S for possible violations, following the approach used by Cook and Rich (1999) in the context of vehicle routing with time-window constraints. This method is motivated by the result of Karger and Stein (1993) showing that Karger's random contraction algorithm on a graph with n nodes finds with high probability all cuts with weight within a multiplicative factor α of the minimum cut in $O(n^{2\alpha} log^3 n)$ time. Since Concorde includes routines to check for subtour constraints, we can assume that LP solutions to our model have no cuts of weight less than 2. Therefore, we can apply Karger's algorithm with $\alpha = 2$ in order to list sets S having $\sum(x_e : e \in \delta(S)) < 4$. Our computer code uses the implementation of Karger's algorithm given in Dash (2000).

If Karger's algorithm produces a cut $\delta(S)$ of weight less than 4 and the set S consists of only customer nodes, then we compute the shortest path starting at the depot and passing through all nodes in S, by converting this problem to a TSP. We solve the TSP using an implementation of the branch-and-bound algorithm of Held and Karp (1971). (On small instances this is more efficient than Concorde's general LP-based algorithm.) If the shortest path has cost greater than B, then S is contained in $\mathcal{H}_B$ and the constraint (3) can be added as a cutting plane. To speed up the computation, we call Concorde's version of the Chained Lin-Kernighan TSP heuristic (Martin, Otto, and Felten (1992)) as a preliminary step in the TSP solution procedure; if the heuristic returns a tour of value B or less, then we do not need to run the branch-and-bound algorithm.

Table 1: Comparison of Models

Model	Lower Bound	Time (Seconds, 500 MHz Alpha EV6)
Hurkens, bound 1,182	1143	55.6
Layered, no bound	1116	338.4
Layered, bound 1,182	1143	370.0

In the first two entries of Table 1, we compare Hurkens' feasibility model and the layered optimization model, where we use a bound of 1,182 in the feasibility model, that is, $B = 1182$ in (3) (all distances are integer valued, and 1,182 is 1 better than the value of the best solution found in the Whizzkids'96 competition). The much larger graph used in the layered model translates into a larger running time. Moreover, the constraints (3) based on the 1,182 bound are strong enough to improve the LP lower bound well beyond that given by the layered model (for the feasibility model, we divide the objective value by 4 to obtain a lower bound on the min-max objective). The layered model can be improved to match the feasibility bound by incorporating constraints of the form

$$\sum(x_e : e \in \delta(\cup(S_v : v \in S))) \geq 4$$

for sets $S \in \mathcal{H}_B$; Karger's algorithm can again be used to sample candidate sets. The third entry in Table 1 gives the running time required for this combined approach.

The results reported in Table 1 were obtained on a Compaq XP10000 workstation, with a 500 MHz Alpha EV6 processor. The Concorde TSP code was run using ILOG's CPLEX 6.5 LP solver.

Model selection

The layered model has the attractive feature that it deals directly with the min-max objective function, but in our study we did not discover a practical way to take advantage of this to produce better LP lower bounds for the problem (for example, by generating cutting planes derived from the constraints (5)). Making the assumption that the 1,183-valued winning solution in the Whizzkids'96 competition is close to being optimal, we decided to adopt Hurkens' feasibility approach in our computations. If the search produced a solution of value less than 1,183, the algorithm would have to be run again with the adjusted bound.

3 k-path Cutting Planes

The feasibility constraints (3) force 0/1 solutions to the newspaper routing model to use at least 2 paths through the specified set of customers S, with the justification that any single path would have length greater than that allowed by the objective bound B. The effectiveness of these simple constraints in increasing the LP lower bound (see Table 1) suggests the use of more general inequalities

$$\sum(x_e : e \in \delta(S)) \geq 2k \tag{6}$$

to force solutions to use at least k paths (for $k \leq 4$) through sets S, in cases where we can verify that $k-1$ or fewer paths would necessarily include one of length greater than the bound B. Inequalities of this form were studied in the context of vehicle routing by Laporte, Nobert, and Desrochers (1985) and dubbed *k-path* cutting planes by Kohl, Desrosiers, Madsen, Solomon, and Soumis (1997).

In searching for k-path cutting planes, we can again make use of Karger's random contraction algorithm to produce candidate sets S. A difficulty is that, in general, we need to solve a newspaper routing feasibility problem to verify that k paths are required through S. For $k = 2$ (that is, for the feasibility constraints (3)) we take advantage of the fact that the single-vehicle newspaper routing problem is easily reduced to an instance of the TSP. This does not work for $k = 3$ or 4, but the TSP does provide a relaxation that allows us to verify that k paths are required in certain cases.

For $k = 3$, we form a TSP instance by taking the nodes in S together with the depot, and joining each of them to a dummy node via an edge of cost 0. If the optimal solution to the TSP in this graph has length greater than $2B$, then it is not possible to visit all customers in S with two paths, each of length less than or equal to B. So in this case we can add the 3-path cutting plane. On the other hand, if the solution of the TSP is less than $2B$, we have the option of solving the newspaper routing feasibility problem on S to check if it still might be the case that it is not possible to visit all customers with two paths of length less than or equal to B. In our implementation, we make this second check by recursively calling our newspaper routing solver, but we only do so if we do not find a TSP solution that is significantly less than $2B$ (in our code we use $2B - 100$ as the cut off); our reasoning is that if there is a very short TSP tour, then it is likely that there exists a pair of short paths through S.

A similar approach is used for 4-path cuts. In this case we obtain a TSP relaxation by taking the customers S and two copies of the depot, joining each node to two dummies via edges of cost 0, and fixing at 1 the variable corresponding to one of the edges joining a depot and a dummy (this allows the tour to simulate 3 paths rather than 4). Again, if the optimal solution to the TSP is greater than $3B$, then we can add the 4-path cutting plane. Otherwise, if we do not find a TSP solution of value less than $3B - 150$, then we make a recursive call to the newspaper routing solver to try to establish a 4-path cutting plane.

In Table 2, we compare the lower bounds we obtain for the Whizzkids'96 problem, using just 2-path cuts, using just 2-path and 3-path cuts, and using 2-path, 3-path, and 4-path cuts; the runs were again carried out on a Compaq XP10000 workstation. The stronger

Table 2: Lower Bounds from k-path Cuts

k-path Cuts	Lower Bound	Time (Seconds)
$k = 2$	1,143	55.6
$k = 2, 3$	1,147	217.7
$k = 2, 3, 4$	1,151	194.5

inequalities cause a significant improvement in the LP relaxation, and a promising direction of research would be to study further classes of problem-specific cutting planes for this class of routing problems.

The results in Table 2 were obtained using 5,000 iterations of Karger's algorithm to produce candidate sets S during each run of the k-path cutting-plane search routine. This number is far below the target suggested by the result of Karger and Stein (1993), but it still provides a good sampling of the possible cuts in the graph which yield k-path cuts. To support this statement, we ran each of the three tests again using 50,000 iterations of the algorithm. The results for this second set of tests are reported in Table 3, and show

Table 3: Lower Bounds with 50,000 Karger Iterations

k-path Cuts	Lower Bound	Time (Seconds)
$k = 2$	1,144	214.8
$k = 2, 3$	1,146	439.4
$k = 2, 3, 4$	1,151	426.6

little improvement despite the extra computation time. (The decrease in the lower bound in the $k = 2, 3$ case is just an artifact of running a cutting-plane algorithm with heuristic separation routines.)

The root LP relaxation

The relatively large gap between the LP lower bound and the 1,183 value of the best known set of routes indicates that solving the Whizzkids'96 challenge problem may require a very large search tree. We therefore need to take care to tighten as much as possible the initial LP relaxation that will serve as the root of the tree. To this end, while working on the root LP, we set the tolerances in our solver to continue the search for cutting planes even after only very small improvements in the objective function, and we make multiple runs of the code, each starting with the LP produced by the previous run. The results of these

Table 4: Computation of the Root LP

Run	Lower Bound	LP Value	Time (Seconds)	Total Time (Seconds)
1	1,151	4600.89	715.6	715.6
2	1,152	4604.31	144.3	859.9
3	1,152	4604.91	382.5	1242.4

computations are reported in Table 4; the final LP has objective value 4,604.91, giving a lower bound of 1,152.

4 The Branch-and-Cut Search

The Concorde TSP code implements a branch-and-cut search, that is, a branch-and-bound search where the LP bound is enhanced by the addition of cutting planes at each node of the search tree. In our newspaper routing solver (a modified version of Concorde), we make use of Concorde's default TSP cuts, as well as the k-path cuts described in the previous section and the cuts (2). Concorde follows a best-bound strategy to grow the search tree, where the next search node to be explored is the one having smallest LP value. The branching mechanism uses an idea of Clochard and Naddef (1991), which can be viewed as branching on subtour constraints. At each node in the search tree, the associated subproblem is split into two new subproblems (the children) by choosing a proper subset of nodes S and adding the constraints $\sum(x_e : e \in \delta(S)) \leq 2$ and $\sum(x_e : e \in \delta(S)) \geq 4$, respectively. The sets S are chosen from a collection of sets that are maintained as part of a pool of cutting planes generated during the branch-and-cut run.

For the Whizzkids'96 challenge problem, we specify an upper bound of 4,729 in Concorde's initialization; whenever a subproblem attains a lower bound greater than 4,728 it is pruned from the search tree. Since 4,728 is equal to 4 times 1,182, any larger LP bound would imply that there is no collection of 4 routes with maximum length less than 1,183 (recall that all distances are integer valued).

Starting with the root LP at value 4,604.91, we need to push the bound of each leaf of the search tree above the 4,728 cut off value. In Figure 6 we display the branch-and-cut tree, up to depth 6. The vertical position of a search node corresponds to the value of its LP relaxation, indicated by the scale on the left-hand-side of the figure. The search tree is promising in that most branches force the bounds of the two children to improve significantly, but it is clear that a very large tree will be needed to establish the 4,729 target bound.

Number of active nodes

During a branch-and-cut run, an important statistic is the number of active search nodes in the tree, that is, the number of nodes that have not been pruned by the cut off value. In the Whizzkids'96 run, this active node count grew in a very predictable fashion when compared to the increase in the overall lower bound provided by the search tree, that is, the minimum LP value over all leafs of the tree. We plot this growth in Figure 7, using a log scale for the number of active nodes.

The plot indicates that the number of active nodes is roughly doubling for every increase of 5 in the lower bound. This rate of growth would overwhelm the computing resources if it were allowed to continue much further (in particular the memory required to store the

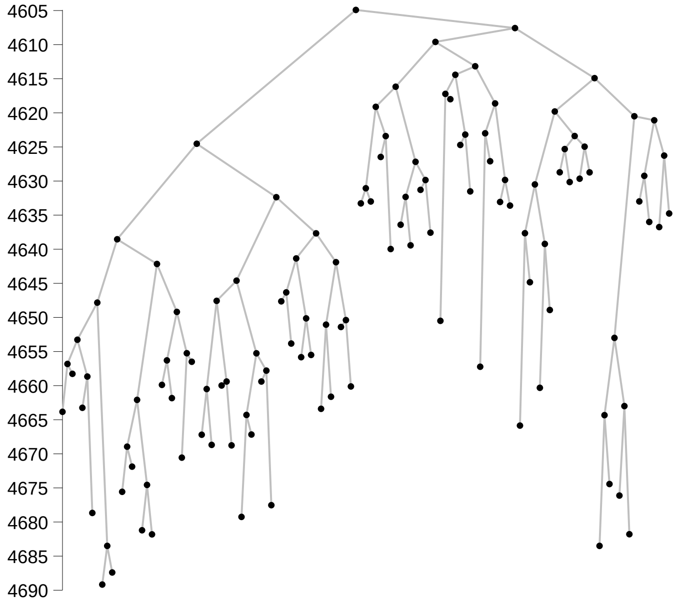

Figure 6: Search Tree at Depth 6

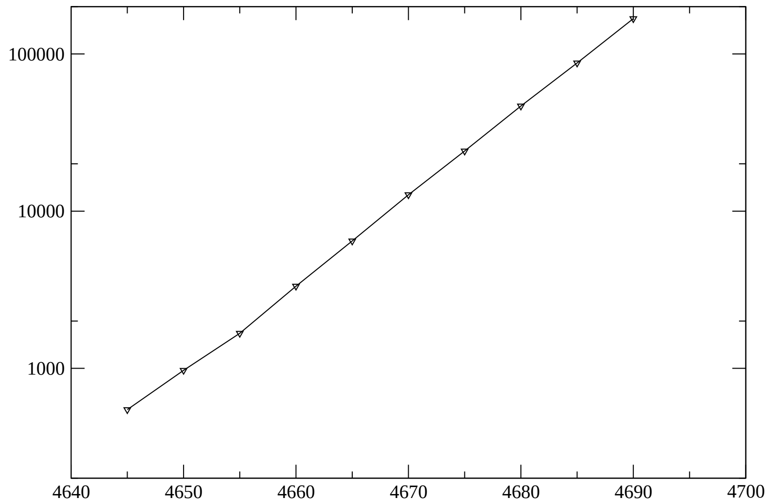

Figure 7: Number of Active Search Nodes versus Lower Bound

tree), so this initial phase of the solution process was terminated with 183,150 active nodes and a 4,690.6 bound.

Another important reason for terminating the branch-and-cut search at this stage was the observation that most of the improvement in the LP values came from branching as opposed to using cutting planes. It seemed desirable to spend less time per node in generating cutting planes; the evidence for this is discussed later on in Table 5.

The total running time used in the aborted branch-and-cut search was 7,000,027 seconds. The run was carried out in parallel on a cluster of Compaq ES40 servers having a total of 32 Alpha EV6 (500 MHz) processors; the 7 million seconds is the sum of the running times on the individual processors. Concorde's parallel implementation uses a master-worker framework, with communication carried out via NFS sockets. The wall-clock time for the run was roughly 7 days, so we achieved a utilization of factor of only 39%. The reason for this poor performance is twofold. Firstly, the computing cluster is used for several other research projects and our work often did not obtain a full share of each available processor. Secondly, the Concorde framework is not designed for branch-and-cut trees with a very large number of nodes and towards the end of the run the master was not able to process the work requests quickly enough to keep all machines busy (upon termination, we were only able to utilize approximately 50% of the available resources).

Distributed depth-first-search

To continue the Whizzkids'96 run, we modified Concorde's strategy for processing the branch-and-cut tree. With a large pool of active nodes, we decided to switch from a best-bound search to a distributed depth-first-search (DFS), where active nodes are assigned to individual processors and each processor runs a DFS search with its assigned node as the root. In this approach, a processor will not report to the master the intermediate status of the subtree that is grown, but only report either that the run terminated with a proof that there is no solution to the subproblem, or that an improved solution (one of value less than 1,183) has indeed been found. In the first case, the subproblem is removed from the active node list. In the second case, the entire Whizzkids'96 computation is terminated, since we will need to run the problem again using the new bound.

Distributed DFS has several advantages over a best-bound search, even if we had available a platform that could handle best-bound on very large search trees.

- With distributed DFS, the amount of computation expended at each node of the individual DFS trees can be varied according to the estimated difficulty of the assigned subproblem. This can be accomplished by initially pursuing an aggressive approach, using very limited cutting-plane searches before the branching steps. If we fail to solve the subproblem in this way, we can restart the DFS from scratch, using a less aggressive approach.

- Very little communication is necessary between the master and the workers, since assigned tasks are lengthy and they can be carried out with no intermediate information sent in either direction. This permits the search to be run on a low-bandwidth network, in our case utilizing a collection of machines at Rice University, Princeton University, AT&T Labs, and at the University of Bonn.

- The control structure of the distributed DFS permits a simple recovery mechanism when workers fail (for example, when the host machine is turned off by a user), since the active nodes do not need to be explored in any particular order. This feature facilitates the part-time use of desktop workstations during the computation.

The main disadvantage of distributed DFS is that we may explore a larger tree than necessary, if there exists a significantly better solution to the problem instance. In the Whizzkids'96 case, we viewed this as an unlikely event given the great effort that went into the heuristic computations during the challenge.

A second disadvantage of distributed DFS is that the cutting planes found during the individual DFS runs are not communicated to the master for re-use in runs on other processors. We attempted to limit the impact of this lack of communication by seeding each processor with a large pool of cutting planes that were accumulated during the initial best-bound run.

Implementing distributed DFS

We use a very simple framework to implement the distributed DFS. The data for the 183,150 active nodes is located at the site of the master, and the data for an individual search node is sent to a worker upon request. The master keeps a circular list of the active nodes, and when it receives a request for work it returns the next node on the list. If a worker reports that the processing on a node is complete, then the master removes the node from the list. If a worker aborts the run or if the worker fails, then the search node remains active and it will be sent to another worker on the next pass through the circular list. During each pass, we increase the amount of effort the workers are permitted to expend during their branch-and-cut search; this is controlled by a limit on the total time the worker can use on the subproblem, by the level of cutting planes the worker should employ, and by several parameters that control the time spent in the selection of the branching subtours. In the initial pass through the active nodes, we permitted 10,000 seconds for the solution of the subproblems; this was gradually increased to allow 100,000 seconds in the final pass.

Solving subproblems

To determine a suitable combination of cutting planes to use at nodes of the depth-first-search runs, we selected 10 search nodes from the list of 183,150 to use as a test bed. One of the important components of the solution procedure is the level of k-path cuts that are employed, determined both by k and by the number of Karger iterations permitted in selecting the candidate sets S. In Table 5 we report the average results over the 10 test problems, using 3 different choices for k-paths. The number of search nodes in the

Table 5: Ten Subproblems

Karger Iterations	k-path Cuts	Search Nodes	Time (seconds)
10	$k = 2, 3$	698.9	719.2
500	$k = 2, 3$	439.6	1178.6
5,000	$k = 2, 3, 4$	368.2	3477.1

trees grows as we decrease the level of cuts from our default of 5,000 Karger iterations and $k = 2, 3, 4$, but the total solution time drops significantly. Therefore, in the initial pass we used only 10 iterations of Karger's algorithm and we did not use 4-path cuts (in later passes we returned to our default settings).

The Whizzkids'96 run

The final DFS run on the Whizzkids'96 instance took a total of 10 days, running on a geographically-distributed network of 188 processors. The number, type, and location of the components of the network are given in Table 6. The sum of processing times, scaled to a 500

Table 6: The Computer Network

Number	Machine	Location	Subproblems
40	Compaq Alpha EV6, 500 MHz	Rice	61,723
59	Intel Pentium III, 450-600 MHz	Princeton	30,427
30	AMD Athlon, 800 MHz	Rice	30,032
12	Compaq Alpha EV67, 667 MHz	Rice	29,822
12	Compaq Alpha EV5, 400 MHz	Rice	9,036
4	Compaq Alpha EV6, 500 MHz	AT&T	6,367
14	SGI MIPS R10k, 196 MHz	AT&T	6,308
6	Intel Pentium III, 933 MHz	Bonn	5,212
5	Intel Pentium Pro, 200 MHz	Rice	1,670
2	Intel Pentium II, 300 MHz	Rice	923
1	Intel Pentium III, 600 MHz	Rice	651
1	Sun 300 MHz UltraSparc II	Rice	513
2	Sun 147 MHz UltraSparc I	Rice	466

MHz Alpha EV6 processor, was approximately 72 million seconds, giving a combined time of 79 million seconds for the best-bound and DFS phases of the computation. The scaling factors for the various machine types are based on the average number of internal DFS search nodes that were processed per second; these figures are given in Table 7. The combined

Table 7: Relative Speed of Machines

Machine	Nodes per Second
Compaq Alpha EV67, 667 MHz	1.85
Compaq Alpha EV6, 500 MHz	1.36
Intel Pentium III, 933 MHz	0.95
AMD Athlon, 800 MHz	0.94
Intel Pentium III, 600 MHz	0.61
Compaq Alpha EV5, 400 MHz	0.58
SGI MIPS R10k, 196 MHz	0.43
Sun 300 MHz UltraSparc II	0.42
Intel Pentium II, 300 MHz	0.38
Intel Pentium Pro, 200 MHz	0.26
Sun 147 MHz UltraSparc I	0.21

search tree used a total of 98,331,581 nodes. The run did not produce a new solution, and therefore the run established that 1,183 is the optimal value for the Whizzkids'96 problem.

5 Average Delivery Time

In the introduction we described the secondary objective function adopted in the Whizzkids'96 competition, namely to minimize the average delivery time to each of the customers. This tie-breaking rule appears to be a difficult objective to handle using a TSP relaxation, since we cannot easily capture the distance from the depot to a given customer. We therefore decided to simply enumerate all 1,183-valued solutions, and select the one with the optimal value of the secondary objective. This approach can be adapted to any problem class where the set of optimal solutions is tightly enough constrained to make a complete enumeration feasible.

To begin the enumeration run, we set the feasibility bound B to 1,183 and we initialize Concorde with the bound 4,733 (that is, 1 greater than 4 times 1,183). We also need to supply Concorde with a branching rule to handle the case where the LP solution is a 0/1 vector that corresponds to a solution of total length less than 4,733, since the standard code would accept this as a new integer solution and prune the tree after adjusting the 4,733 bound.

In handling 0/1 branching, we restrict ourselves to selecting a *branching edge* e, and creating two new subproblems by adding the constraints $x_e = 0$ and $x_e = 1$ (this is a special case of the subtour branching used in Concorde). If we choose a branching edge corresponding to a t-valued variable (where $t = 0$ or $t = 1$), then it is likely that the subproblem obtained by adding $x_e = 1 - t$ will be pruned due to an increase in the LP bound, while the subproblem obtained by adding $x_e = t$ will need to be explored further. To limit the depth of the tree, it is therefore much better to choose a branching edge corresponding to a 1-valued variable rather than a 0-valued variable, since there is a much lower limit on the number of variables that can be forced to 1 before we violate the bound. We carry this argument further, and branch on an edge e that corresponds to a 1-valued variable that lies on the longest path in the solution. The idea is that if we end up fixing all variables to 1 along a path of length greater than 1,183, then we can prune the search since no 1,183-valued solution can occur in the subtree. (Notice that it may not be possible to eliminate such paths with the feasibility cutting planes (3), since after branching the path may not be a shortest route through its set of customers.)

With these changes, a second run of the Whizzkids'96 problem established that 558.81667 is the minimum delivery time over all 1,183-valued solutions. Remarkably, this solution was also obtained by the winning team of Hemel, van Erk, and Jenniskens (1996); it is illustrated in Figure 1.

The second run of the Whizzkids'96 problem took 154,020,343 search nodes (so roughly 50% more than the original run); we switched from from best-bound to distributed DFS when we reached 50,000 active search nodes. The cumulative time was roughly 62 million seconds, scaled to a 500 MHz Alpha EV6 (the time per search node is smaller due to the fast branching rule that was used when enumerating integer solutions to the LP).

It would be interesting to know the number of distinct 1,183-valued solutions of the Whizzkids'96 instance, but we elected not to spend the additional computation time that would be needed to obtain this statistic.

6 Conclusions

Distributed branch-and-bound (or cut) has long been discussed as a computational tool in discrete optimization. The solution of the Whizzkids'96 problem follows recent successes in this area, including the solution of the nug30 quadratic assignment problem by Anstreicher, Brixius, Goux, and Linderoth (2001) and the solution of the d15112 TSP by Applegate, Bixby, Chvatal, and Cook (2002). An immediate benefit of distributed computing is that it offers researchers the capability to explore algorithmic ideas that are beyond the scope of currently available sequential hardware (but perhaps not out of the reach of future computing devices). Ongoing work by Eckstein, Phillips, and Hart (2001), Ralphs and Ladányi (2001), and others is aimed at increasing the scope of applications of distributed computing by providing general frameworks that can be adapted to a wide variety of discrete optimization problems. The Whizzkids'96 work indicates that distributed DFS may be a useful tool in these general computing platforms.

Although the Whizzkids'96 instance was solved, the size of the search tree (nearly 100 million nodes) makes it frightening to consider the solution of much larger examples in newspaper routing. This is a common feature of vehicle routing problems outside of the domain of the TSP, for example, in capacitated vehicle routing there are currently unsolved test instances having as few as 50 nodes (see Ralphs, Kopman, Pulleyblank, and Trotter (2001)). It is not clear how much of the TSP success can carry over to the more general classes of problems, but it is certainly the case that further work on cutting-plane separation routines in vehicle routing can have an immediate impact on solution methods. Promising recent work in this direction has been carried out by Augerat, Belenguer, Benavent, Corberán, and Naddef (1998), Blasum and Hochstättler (2000), Ralphs et al. (2001), and others.

Acknowledgements

We would like to thank Cor Hurkens and Jan Karel Lenstra for very helpful discussions on solution techniques for the Whizzkids'96 problem. We would also like to thank Compaq and AMD for their generous support of high performance computing at Rice University, where this work was begun.

References

Anstreicher, K., N. Brixius, J.P. Goux, and J. Linderoth. 2001. Solving large quadratic assignment problems on computational grids. *Mathematical Programming.* To appear.

Applegate, D., R. Bixby, V. Chvátal, and W. Cook. 1998. On the solution of traveling salesman problems, in *Documenta Mathematica Journal der Deutschen Mathematiker-Vereinigung, International Congress of Mathematicians*, pp. 645–656.

Applegate, D., R. Bixby, V. Chvátal, and W. Cook. 2002. To appear.

Augerat, P., J.M. Belenguer, E. Benavent, A. Corberán, and D. Naddef. 1998. Separating capacity constraints in the CVRP using tabu search. *European Journal of Operations Research* **106** 546–557.

Ball, M.O., T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, eds. 1995. *Network Routing.* Elsevier Science, Amsterdam.

Blasum, U., and W. Hochstättler. 2000. Application of the branch and cut method to the vehicle routing problem. Technical Report zpr2000-386, Zentrum für Angewandte Informatik Köln, Köln, Germany.

Cook, W., and J.L. Rich. 1999. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Computational and Applied Mathematics, Rice University, Houston, TX.

Dantzig, G., R. Fulkerson, and S. Johnson. 1954. Solution of a large-scale traveling salesman problem. *Operations Research* **2** 393–410.

Dantzig, G.B., and J.H. Ramser. 1959. The truck dispatching problem. *Management Science* **6** 80–91.

Dash, S. 2000. http://www.caam.rice.edu/$\sim$sanjeebd/software/karger.tar.gz.

Golden, B.L., and A.A. Assad, eds. 1998. *Vehicle Routing: Methods and Studies.* Elsevier Science, Amsterdam.

Eckstein, J., C.A. Phillips and W.E. Hart. 2001. PICO: an object-oriented framework for parallel branch and bound, in *Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications* (D. Butnariu, Y. Censor, and S. Reich, eds.), *Studies in Computational Mathematics* **8**, Elsevier Science, pp 219–265.

Held, M., and R.M. Karp. 1971. The traveling salesman problem and minimum spanning trees: part II. *Mathematical Programming* **1** 6–25.

Hemel, T., S. van Erk, and P. Jenniskens. 1996. The Manhattan Project. http://www.win.tue.nl/whizzkids/1996/tsp.html.

Hurkens, C. 1997. Presented at the *1997 INFORMS National Meeting in Dallas, Texas*, in a tutorial lecture by J.K. Lenstra, Session SE28.

Jünger, M., G. Reinelt, and G. Rinaldi. 1995. The traveling salesman problem, in *Handbooks in Operations Research and Management Science, Volume 7* (M.O. Ball, T. Magnanti, C.L. Monma, and G. Nemhauser, eds.), Elsevier Science, pp. 225–330.

Karger, D.R. 1993. Global min-cuts in $\mathcal{RNC}$ and other ramifications of a simple mincut algorithm, in *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM-SIAM, pp. 84–93.

Karger, D.R., and C. Stein. 1993. An $\tilde{O}(n^2)$ algorithm for minimum cuts, in *Proceedings of the 25th ACM Symposium on the Theory of Computing*, ACM Press, pp. 757–765.

Kohl, N.J., J. Desrosiers, O.B.G. Madsen, M.M. Solomon, and F. Soumis. 1997. k-path cuts for the vehicle routing problem with time windows. Technical Report IMM-REP-1997-12, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark.

Laporte, G., Y. Nobert, and M. Desrochers. 1985. Optimal routing under capacity and distance restrictions. *Operations Research* **33** 1050–1073.

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, eds. 1985. *The Traveling Salesman Problem*. Wiley, Chichester.

Martin, O., S.W. Otto, and E.W. Felten. 1992. Large-step Markov chains for the TSP incorporating local search heuristics. *Operations Research Letters* **11** 219–224.

Naddef, D. 2001. Polyhedral theory and branch-and-cut algorithms for the symmetric TSP. To appear.

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

Ralphs, T.K., L. Kopman, W.R. Pulleyblank, and L.E. Trotter Jr. 2001. On the capacitated vehicle routing problem. To appear.

Ralphs T.K., and L. Ladányi. 2001. Computational experience with parallel branch and cut. To appear.

Toth, P., and D. Vigo, eds. 2001. *The Vehicle Routing Problem*. SIAM, Philadelphia. To appear.

Whizzkids '96. 1996. http://www.win.tue.nl/whizzkids/1996/index.html.

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