Carnegie Mellon University
Research Showcase @ CMU

Tepper School of Business

5-2008

Connections in Networks: A Hybrid Approach

Carla P. Gomes
Cornell University

Willem-Jan van Hoeve
Carnegie Mellon University, vanhoeve@andrew.cmu.edu

Ashish Sabharwal
Cornell University

Follow this and additional works at: http://repository.cmu.edu/tepper

Part of the Economic Policy Commons, and the Industrial Organization Commons

Recommended Citation
Proceedings of the Fifth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2008), LNCS 5015, 303- 307.

This Conference Proceeding is brought to you for free and open access by Research Showcase @ CMU. It has been accepted for inclusion in Tepper School of Business by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

Connections in Networks: A Hybrid Approach

Carla P. Gomes[1] Willem-Jan van Hoeve[2] Ashish Sabharwal[1]

[1] Department of Computer Science, Cornell University, Ithaca NY 14853, U.S.A.
{gomes,sabhar}@cs.cornell.edu
[2] Tepper School of Business, Carnegie Mellon Univ., Pittsburgh PA 15213, U.S.A.
vanhoeve@andrew.cmu.edu

Abstract. This paper extends our previous work by exploring the use of a hybrid solution method for solving the connection subgraph problem. We employ a two phase solution method, which drastically reduces the cost of testing for infeasibility and also helps prune the search space for MIP-based optimization. Overall, this provides a much more scalable solution than simply optimizing a MIP model of the problem with Cplex. We report results for semi-structured lattice instances as well as on real data used for the construction of a wildlife corridor for grizzly bears in the Northern Rockies region.

In recent work [2], we investigated the *connection subgraph* problem, which seeks to identify a cost bounded connected subgraph of a given undirected graph connecting certain pre-specified terminal nodes, while maximizing the overall utility. Here costs and utilities are non-negative numbers assigned to each node of the graph, and the cost (or utility) of a subgraph is the sum of the costs (utilities, resp.) of the nodes in it. This problem is a variant and generalization of the familiar Steiner tree problem, and occurs in natural settings such as wildlife conservation and social networks.[3] Our experimental results [2] identified an interesting easy-hard-easy pattern in a pure optimization version of the problem. They also brought out some surprising issues with respect to the hardness of proving infeasibility versus the hardness of proving optimality. Specifically, using a mixed integer programming (MIP) model for the problem and solving it to optimality using Cplex 10.1 [3] revealed that in median terms, Cplex took orders of magnitude longer to prove infeasibility of infeasible instances than it took to find optimal solutions to the feasible instances. This naturally raises the question, can one do better on infeasible instances?

This paper reports our results obtained using a hybrid technique for solving the connection subgraph problem, beginning with results on certain semi-structured grid graphs also considered previously. We use a two phase solution method. The first phase employs a minimum Steiner tree based algorithm to test for infeasibility and to produce a greedy (and often sub-optimal) solution for feasible instances. This phase runs in polynomial time for a constant number of terminal nodes. The second phase refines this greedy solution to produce an

[3] Due to lack of space, we refer the reader to our previous paper [2] for a formal definition and detailed discussion of the problem.

optimal solution with Cplex, also using shortest path information generated by the first phase to prune the search space significantly (often by 40-60%). With this hybrid approach, the time to test for infeasibility is drastically reduced, and in fact becomes negligible compared to the cost of running Cplex on feasible instances (the runtime for which is also significantly reduced due to the starting solution and pruning). The hardness profiles still show a clear easy-hard-easy pattern in the feasible region.

We also apply this technique to the original resource economics problem that motivated this work—designing a "wildlife conservation corridor" in the Northern Rockies for preserving grizzly bears. The scale of this real-world problem precludes computing optimal solutions in well over a month of CPU time, even with our hybrid approach. We therefore introduce a *streamlined* model, where we seek to compute the optimal (i.e., highest utility) solution which is restricted to include all nodes that form part of a minimum cost solution, which is also computed in the first phase. We are able to solve this "extended-mincost solution" problem significantly faster, and to near optimality within a month of CPU time on the real wildlife corridor data.

The extended-mincost solution is interesting to compute only if it does not dramatically limit the utility one might achieve in the end. To obtain further insights into this, we study how the extended-mincost solution compares in quality (i.e., attained utility) against the true optimal solution for a given budget, for both grid graphs and coarse granularity (and thus easier) versions of the actual corridor construction problem. We show that the *utility gap* between the optimal and extended-mincost solutions itself follows a fairly narrow low-high-low pattern as the budget increases, indicating that for a large range of budgets, solving the streamlined extended-mincost problem yields a fairly good approximation to the true optimal solution.

The Two Phase Approach. In Phase I, we compute a minimum cost Steiner tree for the terminal nodes of the graph, ignoring all utilities. While there are fixed parameter tractable (FPT) algorithms for computing a minimum cost Steiner tree, we used a simpler "enumeration" method (see, e.g., [4]) based on computing all-pairs-shortest-paths with respect to vertex costs. The idea behind this algorithm, which runs in polynomial time for a constant number of terminal nodes, is to compute a minimum Steiner tree for the "complete shortest distance graph" using the fact that in such a graph, there exists a minimum Steiner tree all whose non-terminal nodes have degree at least three, thereby limiting the total number of nodes in the tree. A minimum Steiner tree of the complete shortest distance graph yields a minimum Steiner tree for the original graph as well, by replacing edges by shortest paths.

The computation of the Steiner tree either classifies the problem instance as infeasible for the given budget or provides a feasible (but often sub-optimal) "mincost" solution. In the latter case, we use a very efficient *greedy method* to improve the quality of the solution by using any residual budget as follows. We consider those nodes that are adjacent to the current solution and have cost lower than the residual budget, and identify one whose *gain*, defined as the

utility-to-cost ratio, is the highest. If there is such a vertex, we add it to the current solution, appropriately reduce the residual budget, and repeat until no more nodes can be added. This process often significantly increases the solution quality. We call the resulting solution an *extended-mincost solution*. We will also be interested in computing the optimal extended-mincost solution, by "freezing" the vertices in the mincost solution to be part of all solutions of Phase II.

After Phase I, which always took almost negligible time compared to Phase II on our problem instances, we either know that the instance is infeasible or already have a greedily extended feasible solution. In the latter case, Phase II of the computation translates the problem into a MIP instance (see [2] for details of the encoding), and solves it using Cplex. Solving using Cplex is the most computationally-intensive part of the whole process. The greedy solution obtained from Phase I is passed on to Cplex as a starting solution, providing a major boost to its efficiency. Further, the all-pairs-shortest-paths matrix computed in Phase I is also passed on to Phase II. It is used to statically (i.e., at the beginning) prune away all nodes that are easily deduced to be too far to be part of a solution (e.g., if the minimum Steiner tree containing that node and all of the terminal vertices already exceeds the budget). This significantly reduces the search space size, often in the range of 40-60%. Overall, Phase II computes an optimal solution (or the optimal extended-mincost solution) to the utility-maximization version of the connection subgraph problem.

Experimental Results. For a varying budget, we investigate the computational hardness of the problem with respect to computing the optimal solution or the optimal extended-mincost solution. Our experiments were conducted on a 3.8 GHz Intel Xeon machine with 2 GB memory running Linux 2.6.9-22.ELsmp. We used Cplex 10.1 [3] to solve the MIP problems in Phase II.

For the first set of experiments, we make use of semi-structured lattice graphs of order m, with 3 terminal vertices, and with uniform random costs and utilities (see [2] for details). In Figure 1, each data point is based on 500 random instances for $m = 10$; similar results, peaking at identical x-axis values, were obtained for $m = 6$ and 8 as well, and are available from the authors. The hardness curves are represented by median running times over all instances per data point. In order to normalize for the small but non-negligible variation in the characteristics of various randomly generated instances with the same parameters, we use for the x-axis of most of our plots the 'budget slack percentage', rather than simply the budget, computed as follows. For every instance, we consider its *mincost*, the cost of the cheapest solution. The *budget slack %* with respect to mincost is defined as: $100 \times (budget - mincost)/mincost$. In other words, we consider computational hardness and other measured quantities as a function of the extra budget available for the problem beyond the minimum required.

In the left half of Figure 1, we show the hardness profiles for the lattices, which exhibit an easy-hard-easy pattern, the peak of which is to the right of the mincost point (shown as 0 on the relative x-scale). As one might expect, computing the optimal extended-mincost solution (lower curve) is significantly easier than computing the true optimal solution (upper curve). How much "better" are the

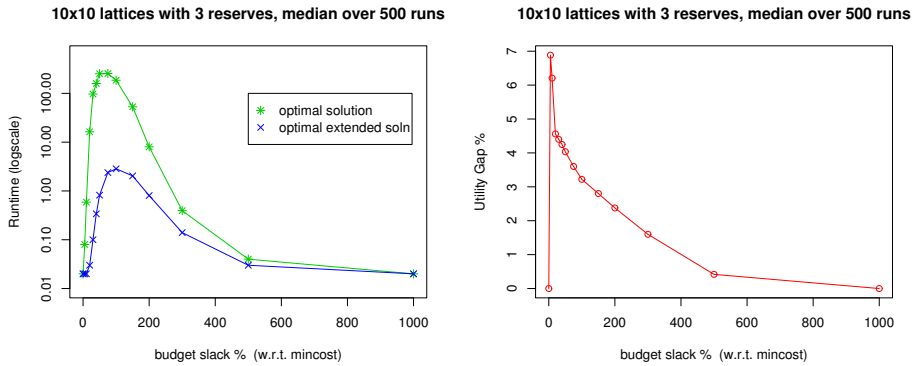

Fig. 1. Left: Hardness profile (runtime, log-scale) for lattices of order 10 with 3 terminal nodes; upper curve: optimal solution; lower curve: optimal extended-mincost solution. Right: Percentage gap in the utility of optimal and extended-mincost solutions.

true optimal solutions compared to the easy-to-find extended solutions? The right half of the figure shows the *relative utility gap %* between the solution qualities (i.e., attained utilities) in the two cases, defined as $100 \times (optimal - extended)/optimal$. We see that when budget equals mincost, both optimal and extended solutions have similar quality. The gap between the qualities reaches its maximum shortly thereafter, and then starts to decrease rapidly, so that the extended solution at 100% budget slack is roughly 3.2% worse than the optimal solution for order 10 grids, and at 500% budget slack, only around 0.4% worse.

For the second set of experiments, we used real data for the design of a wildlife conservation corridor for grizzly bears in the Northern Rockies, connecting the Yellowstone, Salmon-Selway, and Northern Continental Divide Ecosystems in Idaho, Wyoming, and Montana. To measure the utility of each parcel, we use grizzly bear habitat suitability data [1]. The cost is taken to be the land value estimate provided by the U.S. Department of Agriculture. We experimented with various granularities for the problem, going from County level regions down to 5 km $\times$ 5 km square grid regions. Going to finer granularities reduces the cost of the **cheapest corridor** from \$1.9 B for the County level, to \$1 B for a 40 km square grid, to as low as \$11.8 M for the 5 km grid. Using a 25 square km *hexagonal grid* allows for better connectivity than the 5 km $\times$ 5 km square grid, since each hexagonal parcel is connected to 6 other parcels rather than 4, and results in a further decrease in cost to only \$7.3 M. A hexagonal grid also yields a wider corridor on average. As the granularity of the parcels is increased, the problem size grows rapidly. For example, while the County level abstraction has only 67 parcels, the 40 km square grid already has 242 parcels, and the 25 square km hexagonal grid has 12,889 parcels. As a result, solving the connection subgraph model in a naïve manner (as in [2]) using the Cplex solver quickly becomes infeasible: in fact, Cplex even had difficulty finding *any* feasible solution at all for a 40 km square grid or finer.

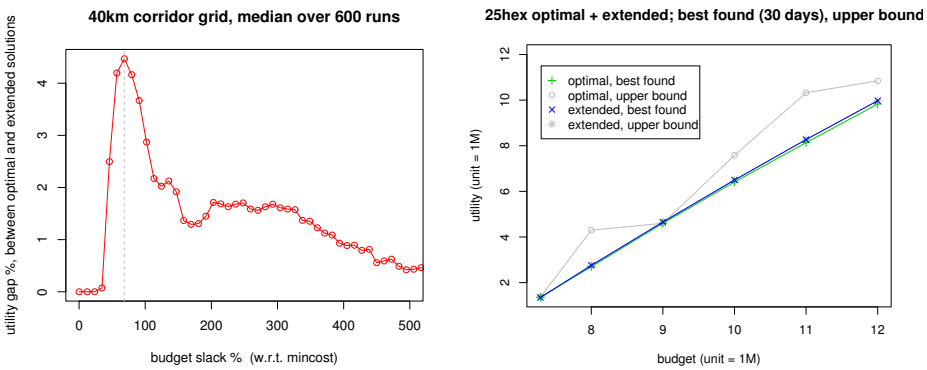

Fig. 2. Left: Utility gap % of optimal and extended-mincost solutions for 40 km grid. Right: Best found optimal and extended-mincost solutions for the 25 sq. km hexagonal grid, 30 day cutoff. Upper bound computed from the optimality gap reported by Cplex.

The left half of Figure 2 shows the relative gap between the optimal and extended solution utilities for the 40 km square abstraction (both were solved optimally). The relative gap is under under 5% when it is at its peak, and is usually within 2% of the optimal. This suggests that for this problem, one does not lose too much by solving only for the extended-mincost solution.

The right plot in Figure 2 depicts results on our best grid: the 25 square km hexagonal grid. This grid is significantly harder to solve. While the County level and the 50 km square grid were solved to optimality within seconds, even the extended-mincost solution for the hexagonal grid could not be solved optimally in over 10 days. Fortunately, the eventual optimality gap for the best extended-mincost solutions found after 30 days was only 0-0.07% (the "best found" curve for extended solutions is visually right on top of the corresponding "upper bound" curve). The best true optimal solutions, on the other hand, had an optimality gap of up to 27% (in one case 59%), as seen from the top curve. Interestingly, the best extended solutions found in this case were in fact of better quality than the best optimal solutions found (the green line is slightly *lower* than the blue line). This is in line with the concept of streamlining, where restricting the problem to only extended-mincost corridors allowed Cplex to compute better quality solutions within a limited amount of computation time.

References

[1] CERI. Grizzly bear habitat sustainability data, 2007. Craighead Environmental Research Institute, Bozeman, MT.

[2] J. Conrad, C. P. Gomes, W.-J. van Hoeve, A. Sabharwal, and J. Suter. Connections in networks: Hardness of feasibility versus optimality. In 4th *CPAIOR*, vol. 4510 of *LNCS*, pp. 16–28, Brussels, Belgium, May 2007.

[3] ILOG, SA. CPLEX 10.1 reference manual, 2006.

[4] H. J. Prömel and A. Steger. *The Steiner Tree Problem: A Tour Through Graphs, Algorithms, and Complexity.* Vieweg, 2002.