The Resource-Constrained Elementary Shortest Path Problem with Hotstarting

Luis J. Novoa
CIS & BSAN, James Madison University, Harrisonburg, VA 22807, novoadlj@jmu.edu,

Ahmad I. Jarrah
Department of Decision Sciences, The George Washington University, Washington, DC 20052, jarrah@gwu.edu,

Jeremy Ezell
CIS & BSAN, James Madison University, Harrisonburg, VA 22807, ezelljd@jmu.edu.

Motivation

The resource-constrained elementary shortest path problem (RCESPP) is an NP-hard one (Dror (1994)) that often arises as a subproblem for column-generation (CG) algorithms seeking to solve different classes of vehicle, aircraft, and ship routing and scheduling optimization models as well as other non-transportation ones (see, for example, Desaulniers, Desrosiers, Solomon (2006)). Solving the RCESPP more efficiently would directly translate into more tractable CG-based solution algorithms, and is, thus, an area of great interest for researchers and practitioners using CG techniques.

We build upon a solution framework comprised of decremental state-space relaxation (DSSR) (see Boland et al. (2006) and Righini and Salani (2008)), backward-forward dynamic programming with joining (see Righini and Salani (2006)), and 2-cycle elimination (see Irnich and Villeneuve (2006) for an overview) by identifying hotstarting notions that can be used to significantly speed up each DSSR iteration given the latest information from the prior one. We comprehensively discuss the underlying principles, detail how each notion can be efficiently implemented, and report the major resultant speedups observed via comprehensive computational tests.

Problem Statement

We limit our focus to solving the RCESPP in the context of the capacitated vehicle routing problem with time windows (VRPTW), whose objective is to find the minimum-cost set of routes visiting a set of customers exactly once using identical vehicles of capacity *vcap*. The problem can be defined (see Baldacci, Mingozzi, and Roberti 2011) on a directed graph $G = (\{SN, EN\} \cup \mathbb{O}, A)$, where SN and EN are the source and end nodes, respectively, corresponding to the depot, $\mathbb{O}$ represents the set of customers, and A is the set of arcs. For each arc $(i, j) \in A$, we associate a traveling cost c_{ij} and a nonnegative traveling time $\hat{\tau}_{ij}$, which generally includes the service time at vertex i. For each vertex $i \in \mathbb{O}$ we associate a demand v_i and a time window $[a_i, b_i]$, where a_i and b_i represent the earliest and latest time, respectively, at which service can begin at node i.

We build upon the paradigm of DSSR-based label correcting algorithms (Righini and Salani (2008)) that relax the set of *unreachable nodes* for a given label, which is the set of all nodes that *cannot* be reached from the label including, in particular, those nodes visited in the path leading to the label. The *state-space relaxation* is achieved by iteratively including, in a *restricted set of unreachable nodes* (RUN), only unreachable nodes belonging to the set of *critical nodes*, which are nodes that previously appeared in a cycle in the optimal route at the end of *any* iteration. This is done to circumvent the exponential explosion in the number of labels that would result otherwise. Within this paradigm, each feasible partial path from the source node (SN) to node i, can be represented by a generic label $l_i = (RUN_i,\ T_i,\ V_i,\ RC_i)$ capturing resource consumption, where:

RUN_i : Restricted set of unreachable nodes from node i,
T_i : Elapsed time up to node i,
V_i : Capacity consumed up to, and including order i, and
RC_i : Calculated reduced cost up to, and including order i.

Defining $\mathbb{P}(l_i)$ as the partial path corresponding to label l_i, (g,h) as a pair of successive nodes in the path, and π_h as the dual variable at node h, we have $RC_i = \sum_{(g,h)\in\mathbb{P}(l_i)}(c_{gh} - \pi_h)$. Instead of generating complete paths from the source to the sink, Righini and Salani (2006) demonstrate the significant computational advantage of using a "forward" dynamic program (DP) to generate partial routes in the forward direction (starting from the source; i.e., node SN), and a "backward" DP to generate partial routes in the backward direction (starting from the sink; i.e., node EN), and then merging the partial paths to form complete routes. This reduces the number of visited nodes and the number of their associated labels. Forward extension functions determine the attributes for the new label l_j when extending label l_i at node i forward to node j, while backward extension functions determine the attributes for the new label l_i when extending label l_j at node j backwards to node i. Forward and backward extensions of untreated labels are only allowed as long as the capacity consumed for label l_i does not exceed *half* of the vehicle's capacity, or, more precisely, as long as $V_i \leq \left\lceil \frac{vcap}{2} \right\rceil$.

Finally, while performing extensions to build partial paths, routes with cycles of length two are not uncommon, especially when the time windows are wide and/or there is no relatively tight resource constraint. The approach requires classifying each label l_i as either *strong* (S), *semi-strong* (SS), or *weak* (W). An S label is an efficient one whose corresponding partial path cannot be feasibly extended back to the predecessor node of the label, while an SS label is one for which such an extension is feasible. A W label is one that is dominated by an SS label but that is retained to account for the possibility that the W label may actually result in attractive paths that the SS label cannot produce because it cannot be extended to its predecessor node without generating a cycle of length two. See Welz (2010) for more details.

In this research, our objective is demonstrate the effectiveness of incorporating hotstarting notions into the above-described solution framework by:

- Establishing the fundamental underlying principles,
- Identifying the necessary information and its required maintenance between iterations,
- Developing the updated logic flow for the updated solution framework, and
- Empirically demonstrating the speedups that can be expected.

Approach

Before discussing our approach for incorporating hotstarting into the solution framework, the following proposition can be used to enhance the efficiency of the joining procedure regardless whether hotstarting is used or not.

Proposition 1: *At the completion of the forward and backward DPs, if label l_i (l_j) in the forward (backward) network can be feasibly joined with label l_j (l_i) in the backward (forward) network, then l_i (l_j) can be feasibly joined with all the labels in the partial path $\mathbb{P}(l_j)$ ($\mathbb{P}(l_i)$) corresponding to label l_j (l_i).*

The hoststarting techniques developed will now be explained via a series of propositions and corollaries.

Proposition 2: *For the optimal DP solution obtained at the end of iteration t, any label l_i that corresponds to an elementary route remains efficient in all subsequent iterations. On the other hand, any label l_i that corresponds to a non-elementary route remains efficient in subsequent iterations until the label's possible deletion.*

The permanent efficiency of the elementary routes can be directly justified using the *relaxation principle*, while the transient nature of non-elementary labels arises from the fact that as the set of critical nodes expands, labels with cycles involving critical nodes get deleted. The important implication of the proposition is that iteration t+1 can be hotstarted with all the labels that survive at the end of iteration t. Based on the proposition, the following corollaries can be justified and exploited for improved efficiency within hotstarting.

Corollary 2.1: *The algorithm terminates at the end of iteration t if either (i) the optimal route is elementary, (ii) the reduced cost of the non-elementary route obtained is equal to that of the best elementary route found, or (iii) a non-elementary route with a positive reduced cost is obtained indicating that no elementary route with negative reduced cost exists.*

Corollary 2.2: *If the extension of l_i to node j during iteration t is proven infeasible, then the extension of l_i to node j remains infeasible, and, hence, should be avoided in subsequent iterations.*

Corollary 2.3: *If the extension of l_i to node j during iteration t, resulting in label l_j^1, is dominated by label l_j^2 at the same node, then the extension of l_i to node j should be avoided in subsequent iterations as long as the relationship $RUN_j^2 \subseteq RUN_j^1$ holds.*

Corollary 2.4: *The extension of l_i to node j resulting in l_j during iteration t should be avoided in subsequent iterations.*

Corollary 2.5: *If the joining feasibility conditions are satisfied for label l_i at node i in the forward network and label l_j at node j in the backward network, then feasibility testing for these pair of labels should be avoided in subsequent iterations.*

Corollary 2.6: *If any of the joining feasibility conditions is not satisfied for label l_i at node i in the forward network and label l_j at node j in the backward network, then these pair of labels should not be considered for joining in subsequent iterations.*

A critical aspect of effective hotstarting is the ability to use the solution at the end of iteration t to identify *bounds* that can be used to control the growth in the number of labels required for the subsequent iteration, and, ultimately, to identify the optimal solution. At the completion of iteration t, a byproduct of the forward and backward DPs followed by joining is the identification of the least costly route for reaching the end node, *EN*, from any label l_i in the forward network. Let α_{l_i} denote the corresponding *forward completion cost*. Analogously, let β_{l_j} denote the least *backward completion cost* identified by the procedure for reaching the start node, *SN*, from any label l_j in the backward network.

Proposition 3: *At the conclusion of iteration t, the forward (backward) completion cost α_{l_i} (β_{l_j}) for label l_i (l_j) in the forward (backward) network, serves as a forward (backward) completion lower bound for l_i (l_j) for all subsequent iterations.*

Based on Proposition 3, the following corollary holds.

Corollary 3.1: *Given an upper bound, ub, on the cost of the optimal resource-constrained elementary shortest path, if $RC_{l_i} + \alpha_{l_i} > ub$ ($RC_{l_j} + \beta_{l_j} > ub$), then label l_i (l_j) can be deleted from the forward (backward) network, and the extension of the predecessor label of l_i (l_j)to node i (j) should be avoided in subsequent iterations.*

The completion bounds for labels can be used to determine completion bounds at the node level.

Proposition 4: *At the conclusion of iteration t, a forward (backward) completion lower bound for node i (j) in the forward (backward) network can be determined as the lowest forward (backward) completion bound among all the labels at node i (j) that have either been deleted due to non-elementarity or had their RUN resource modified.*

Based on Proposition 4, the following corollary holds.

Corollary 4.1: *Given an upper bound, ub, on the cost of the optimal resource-constrained elementary shortest path, and a forward (backward) completion lower bound κ_i (κ_j) for node i (j) in the forward (backward) network obtained at the end iteration t-1, if $RC_{l_k} + (c_{ki} - \pi_i) + \kappa_i > ub$ ($RC_{l_k} + (c_{kj} - \pi_k) + \kappa_j > ub$), then the extension of label l_k to node i (j) should not be allowed in iteration t.*

The following proposition can be used to altogether avoid extensions to certain nodes in the network.

Proposition 5: *At the conclusion of iteration t-1, if for node i (j) in the forward (backward) network no labels are deleted due to non-elementarity and none of their RUN resources are updated, then node i (j) is classified as "unmodified" and no forward (backward) extensions should be allowed to it in iteration t.*

The question of when the completion bounds should be updated (tightened) is answered by the following proposition.

Proposition 6: *The forward (backward) completion bound α_{l_i} (β_{l_j}) for label l_i (l_j) in the forward (backward) network remains current if none of its successor labels in its forward (backward)*

completion path in the forward (backward) network is deleted and the first label in its forward (backward) completion path that belongs to the backward (forward) network is not deleted.

One last aspect of the approach worth mentioning is that at the end of iteration t, and depending on which labels get deleted, it may be necessary to reclassify the surviving W labels for the purposes of 2-cycle elimination. In particular, a W label whose associated SS label is deleted may get reclassified as either SS or S, and an SS label might be reclassified as an S one due the changes in the RUN sets.

Results and Conclusions

The new hotstarting algorithm is tested on various networks from the Solomon dataset (Solomon (1995)), and using randomly generated dual variables as in Righini and Salani (2006) and Righini and Salani (2008). Preliminary results indicate that run-time reductions of upwards of 75% can be expected as compared to not using hotstarting. However, we noted that the run-times (for both with and without hotstarting) are highly dependent on the set of duals used. Accordingly, we plan to run the algorithm on a large sample of duals in order to obtain statistically reliable results for the run-times. Detailed tables summarizing the results from the runs and the variability will be presented.

This research establishes and exploits various novel hotstarting notions in order to significantly improve the efficiency of solving the RCESPP using a solution framework based on DSSR, forward and backward label correcting dynamic programming algorithms followed by joining, and 2-cycle elimination. An interesting future extension would be to conduct a study that thoroughly benchmarks our hotstarting approach with a similar but modified one that uses the "ng" relaxation introduced by Baldacci, Mingozzi, and Roberti (2011) and further experimented with by Martinelli, Pecin, and Poggi (2014), as well as with the implicit enumeration approach of Lozano, Duque, and Medaglia (2015). Such a study would be very helpful in highlighting the strengths and shortcomings of leading state-of-the art algorithms for solving the challenging RCESPP, and the findings would be highly relevantly when it comes to devising CG-based algorithms.

References

Baldacci R, Mingozzi A, Roberti R (2011) New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem. *Operations Research.* 59(5):1269-1283.

Boland N, Dethridge J, Dumitrescu I (2006) Accelerated Label Setting Algorithms for the Elementary Resource Constrained Shortest Path Problem. *Operations Research Letters.* 34(1):58-68.

Desaulniers G, Desrosiers G, Solomon M eds. (2006) *Column Generation* (Springer).

Dror M (1994) On the Complexity of the Shortest Path Models for Column Generation in VRPTW. *Operations Research.* 42(5):977-978.

Feillet D, Dejax P, Gendreau M, Guegen C (2004) An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints: Application to Some Vehicle Routing Problems. *Networks.* 44(3):216-229.

Irnich S, Villeneuve D (2006) The Shortest Path Problem with Resource Constraints and K-Cycle Elimination. *INFORMS Journal on Computing.* 18(3):391-406.

Lozano L, Duque D, Medaglia A (2015) An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints. *Transportation Science.* 50(1): 348-357.

Martinelli R, Pecin D, Poggi M (2014) Efficient Elementary and Restricted Non-Elementary Route Pricing. *European Journal of Operational Research.* 239: 102-111.

Righini G, Salani M (2006) Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization.* 3(3):255–273.

Righini G, Salani M (2008) New Dynamic Programming Algorithms for the Resource Constrained Elementary Shortest Path Problem. *Networks.* 51(3):155–170.

Solomon M, (1995) VRPTW Benchmark Problems. http://w.cba.neu.edu/ msolomon/problems.htm

Welz W (2010) "Route planning for robot systems", Diplomarbeit Technische Universitat Berlin.