On the Reformulation of Vehicle Routing Problems and Scheduling Problems*

J. Christopher Beck[1], Patrick Prosser[2], and Evgeny Selensky[2]

[1] 4C, University College Cork, Ireland cbeck@4c.ucc.ie
[2] Department of Computing Science, University of Glasgow, Scotland. pat/evgeny@dcs.gla.ac.uk

Abstract. We can reformulate a vehicle routing problem (VRP) as an open shop scheduling problem (SSP) by representing visits as activities, vehicles as resources on the factory floor, and travel as set up costs between activities. In this paper we present two reformulations: from VRP to open shop, and the inverse, from SSP to VRP. Not surprisingly, VRP technology performs poorly on reformulated SSP's, as does scheduling technology on reformulated VRP's. We present a pre-processing transformation that "compresses" the VRP, transforming an element of travel into the duration of the visits. The compressed VRP's are then reformulated as scheduling problem, to determine if it is primarily distance in the VRP that causes scheduling technology to degrade on the reformulated problem. This is a step towards understanding the features of a problem that make it more amenable to one technology rather than another.

1 Introduction

In the capacitated vehicle routing problem with time windows, m identical vehicles initially located at a depot are to deliver discrete quantities of goods to n customers. Each customer has a demand for goods and each vehicle has a capacity. A vehicle can make only one tour starting at the depot, visiting a subset of customers, and returning to the depot. Time windows define an interval for each customer within which the visit must be made. A solution is a set of tours for a subset of vehicles such that all customers are served only once and time window and capacity constraints are respected. The objective is to minimise distance travelled, and sometimes additionally to reduce the number of vehicles used. The problem is NP-hard [2].

An $n \times m$ job shop scheduling problem (JSSP) consists of n jobs and m resources. Each job consists of a set of m completely ordered activities. Each activity requires a resource and has an execution duration on that resource. The complete ordering defines a set of precedence constraints, such that an activity cannot begin execution until the preceeding activity has completed. Activities that require the same resource cannot overlap in their execution and no pre-emption is allowed. The problem is then to decide if all activities can be

* This work was supported by EPSRC research grant GR/M90641 and ILOG SA.

scheduled within a given makespan, while respecting the resource and precedence constraints. The job shop scheduling decision problem is NP-complete [2].

What are the characteristics of vehicle routing problems that make them more amenable to local search techniques allied to construction methods? What properties of scheduling problems make them more suitable to systematic search and powerful constraint propagation? In this paper, we take a first step toward answering these questions. We present reformulations between the VRP and factory shop scheduling problems[1] (SSP's), and identify three significant differences between VRP's and SSP's. We then propose a pre-processing transformation of VRP's that reduces one of these differences, namely the ratio of processing times to transition times. We then investigate the behaviour of this transformation using standard VRP and JSSP benchmarks.

2 Transformations: VRP $\leftrightarrow$ SSP

VRP $\rightarrow$ SSP: We reformulate the VRP into a SSP as follows. Each vehicle is represented as a resource, and each customer visit as an activity. The distance between a pair of visits corresponds to a transition time between respective activities. Each activity can be performed on any resource, and is constrained to start execution within the time window defined in the original VRP. Each activity has a demand for a secondary resource, corresponding to a visit's demand within a vehicle. For each resource R there are two special activities $Start_R$ and End_R. Activities $Start_R$ and End_R must be performed on resource R. $Start_R$ must be the first activity performed on R and End_R the last. The transition time between $Start_R$ and any other activity A_i corresponds to the distance between the depot and the i^{th} visit. Similarly the transition time between End_R and A_i corresponds to the distance between the depot and the i^{th} visit. The processing time of $Start_R$ and End_R is zero. We associate a consumable secondary resource with every (primary) resource to model the capacity of vehicles. Consequently a sequence of activities on a resource corresponds to a vehicle's tour in the VRP. In the resultant SSP each job consists of only one activity, each activity can be performed on any resource, and there are transition times between each pair of activities. The problem is then to minimise the sum of transition times on all machines and maybe also to minimise the number of resources used.

SSP $\rightarrow$ VRP: We have for each resource a vehicle, and for each activity a customer visit. The visits have a duration the same as that of the corresponding activities. Each visit can be made only by the vehicles corresponding to the set of resources for the activity. Any ordering between activities in a job results in precedence constraints between visits. Transition times between activities correspond to travel distances between visits. The deadline D imposes time windows on visits. Assuming we have m resources, and therefore m vehicles, we have $2m$ dummy visits corresponding to the departing and returning visits to the depot.

[1] We will refer to the jobshop and the open shop scheduling problems as the shop scheduling problem, i.e. SSP.

A vehicle's tour corresponds to a schedule on a resource. For the $n \times m$ JSSP we have a VRP with $m(n+2)$ visits and m vehicles. Each visit can be performed only by one vehicle. Since there are no transition times in the JSSP, there are no travel distances between visits, but visits have durations corresponding to those of the activities. There are precedence constraints between those visits corresponding to activities in a job. The decision problem is then to find an ordering of visits on vehicles that respects the precedence constraints and time windows.

In [3] Selensky investigated the two extreme cases identified above, i.e. the reformulation of benchmark VRP's as SSP's (and their subsequent solution using ILOG Scheduler) and the reformulation of JSSP benchmarks as VRP's (solved using ILOG Dispatcher). The VRP's used were the 56 Solomon benchmarks [4] and jobshop problems of size 6×6 [6] and 15×15 [5]. The VRP instances reformulate to strange open shop problems, having essentially single activity jobs, a vast selection of resources for each activity, vanishingly small durations, and comparatively enormous transition times. Conversely the JSSP instances map to VRP's where a visit can be performed only by one vehicle and there is no travel! Not surprisingly, the performance of the tools was greatly degraded as the problems were reformulated.

While these results do not come as a great surprise, they do lend support to the belief that there are characteristics of vehicle routing and scheduling problems that make them more suitable to their standard resolution technology. Three differences stand out. The transformed VRP have many more alternative resources than typical scheduling problems. The SSP has more complex temporal relations than the typical VRP. The activity duration is small and transition is large in VRP, whereas in SSP it is the converse. In this paper we focus on just one of these, the ratio of activity duration to transition time. We present a transformation that attempts to reduce the difference between the transition time and activity duration in VRP's and investigate its effect on problem solving.

3 Compressing the VRP

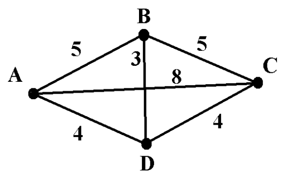
The basic idea of this transformation is to reduce travel within a VRP by adding a portion of travel costs to the costs of visits. Consequently, when the VRP is reformulated as a SSP that SSP will have activities with relatively large durations and small set up cost and look more like a *normal* SSP than a VRP. First, we show how to compress a travelling salesman problem (TSP). We then take into consideration time windows, and show how to apply our transformation to the VRP.

Compressing the TSP: Consider a travelling salesman problem where nodes have costs as well as edges. The cost of visiting a node is then the cost of the edge entering the node plus the cost of the edge exiting the node, plus the cost of the node itself. We can transform this cost such that the node cost is increased and the costs on the entering and exiting edges are reduced.

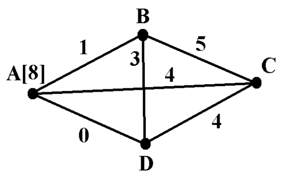
Consider node j, with entering edge (i,j) and exiting edge (j,k), with costs C_j, $C_{i,j}$ and $C_{j,k}$ respectively. Let $C_{min} = min(C_{i,j}, C_{j,k})$. We can reduce the cost of both edges by C_{min}, such that one of these becomes zero, and add to the node cost $2 \times C_{min}$. This preserves the cost of entering, visiting, and exiting j. More generally, for a TSP we can process each node i as follows

1. let C_{min} be the cost of the cheapest edge incident on node i;
2. for each edge incident on node i subtract C_{min} from the edge's cost;
3. add a cost of $2 \times C_{min}$ to C_i, the cost of node i;
4. the node now has at least one incident edge of zero cost.

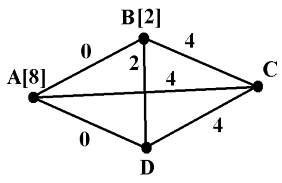
We need only process each node once, i.e. after processing, a node has at least one zero cost incident edge and re-processing will have no effect. Figure 1 shows the sequence of transformations of a four-node clique. All nodes start with zero cost and are processed in alphabetic order. Note that if we processed the vertices in a different order we might end up with a different final graph, i.e. the transformation is order dependent.



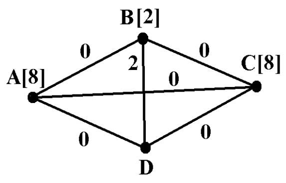
(a) Original graph



(b) Step 1. Processing node A



(c) Step 2. Processing node B



(d) Step 3. Processing node C

Fig. 1. Transformation of a 4-node clique. New node costs are shown in square brackets.

Compressing the VRP: In a Hamiltonian path we have two distinguished nodes, i.e. the start node s and end node e, and the transformation is then modified as follows. Since s is not entered and e is not exited, we do not add to those vertices twice the cost of its minimum incident edge. Instead, we add the minimum cost once only, and process all other vertices as above.

When time windows are associated with nodes the compression can change the lower and upper bounds on the time of entering and exiting a node. Consider the problem of finding a Hamiltonian path from s to e, where each node i has a time window $[ES_i, LS_i]$, i.e. we must arrive at node i no earlier than ES_i and no later than LS_i. Assume we have a graph with edges (s, i), (s, j), and (i, j) where $C_{s,i} < C_{s,j}$ and $C_{i,j} < C_{s,j} - C_{s,i}$. Further assume that we process nodes in the order s, then i, then j. On processing start node s, the transformed costs of node s becomes $C'_s = C_s + C_{s,i}$, and transformed edges $C'_{s,j} = C_{s,j} - C_{s,i}$, and $C'_{s,i} = 0$. Node i is then processed, and this has no effect since it has a zero cost incident edge $C'_{s,j}$. On processing node j we get the transformed costs $C'_j = C_j + 2 \times C_{i,j}$, $C''_{s,j} = C'_{s,j} - C_{i,j}$, and $C'_{i,j} = 0$. Note that via substitution $C''_{s,j} = C_{s,j} - C_{s,i} - C_{i,j}$. Consequently, on the transformed graph the cost of travelling directly from s to j is $C_s + C_{s,j} - C_{i,j}$, i.e. we arrive earlier on the transformed graph and might now have to wait before entering node j. A symmetric argument holds for leaving the node.

Theorem 1. *On transforming an arbitrary node i, with time window $[ES_i, LS_i]$, its earliest start time becomes $ES_i - C_{i,j}$ and its latest start time becomes $LS_i - C_{i,j}$, where $C_{i,j}$ is the cost of the cheapest edge incident on node i.*

Proof. We need to prove that on travelling from i to j, or from j to i in the transformed graph we maintain any slack that existed in the original graph. The proof is by cases and is presented in full in [1]. *QED*

Order Dependence: The transformation is order dependent. We will investigate three transformations. The first, and most obvious, uses a lexicographic ordering of vertices. We will refer to this as a *lex* ordering. The second, we call *maxMin*; when selecting a node i to process next we choose a node such that its cheapest incident edge is a maximum. For example, in Figure 1 this would initially select node A or node C, as their cheapest incident edges are largest, with a cost of 4. The intuition behind this is that it will attempt to make the biggest reduction in edge costs. The third ordering, is *minMin*. This might be thought of as the anti-heuristic, selecting to process a node i with smallest minimum incident edge cost. In Figure 1 this would initially choose node B or D.

4 An Empirical Study

After the VRP has been compressed, we can then reformulate the VRP as a SSP and solve it using scheduling technology. We now attempt to determine if any of these compressions of VRP benchmarks result in better solutions, with or without the reformulation to a SSP.

The experiments were performed on the 56 Solomon benchmarks [4]. These problems fall into six classes: C1, C2, R1, R2, RC1, and RC2. All problems in a class have the same set of customer visits, i.e. there is one set of 100 x/y coordinates corresponding to the customer visits. What differentiates problems within a class is the distribution of time windows and demands. In C1 and C2 visits are clustered, and in C2 visits have larger time windows and increased vehicle capacity, i.e. C2 is a less constrained set than C1. The locations of visits in R1 and R2 are random, and again, each R2 instance has larger time windows and increased vehicle capacities. The RC set is made up of clustered groups of randomly generated visits, and again RC2 is a less constrained set than RC1.

The 56 benchmark VRP were transformed as above, using the lex ordering, maxMin ordering, and minMin ordering. This gives us a total 224 problems, i.e. the original problems and each problem transformed using the three orderings. The problems were then solved using ILOG Dispatcher, each instance being allowed 10 minutes cpu time. The purpose of this was to determine how VRP solving technology is influenced by the amount of travel within the problem. That is, will Dispatcher perform worse or better as we compress problems, transforming travel between nodes into the cost of processing those nodes? The same set of 224 problems were then reformulated as scheduling problems and solved using ILOG Scheduler, and again given 10 minutes cpu time per instance. Would the transformation result in an improvement in Scheduler's performance on the reformulated VRP's?

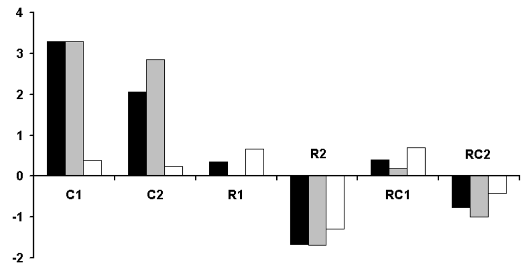


Fig. 2. Percentage increase in cost as a result of transforming VRP's and solving with Dispatcher

Solving with Dispatcher: Figure 2 shows the percentage change in cost as a result of the transformation when solving the VRPs with Dispatcher, i.e. without reformulation. The black bars represent lex ordering, the grey bars maxMin ordering, and minMin ordering is in white. Results are expressed as the average percentage change in cost compared to the original problem solved with Dispatcher. We can see that for clustered problems all of the orderings result in an increase in cost with a maximum of over 3%. An analysis of the first solutions found, prior to improvement with local search, showed that the first solutions were not considerably affected by the transformations (the largest increase in cost of the first solutions being less than 0.3%). Unfortunately, the results for problems with a random element in customer distributions are not so clear. However, this still demonstrates the sensitivity of the VRP solving technology to the mix of travel and processing time. The maxMin ordering attempts to make the largest compression of the problem, and this corresponds to our worst performance of Dispatcher. This tends to suggest that the VRP technology degrades as the VRP becomes more like a SSP. This might also suggest that an inverse transformation, one that was able to *stretch* the VRP by converting the cost of processing a visit into additional travel, might improve the VRP technology.

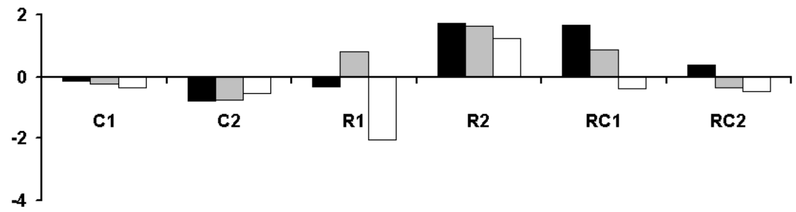


Fig. 3. Percentage increase in cost as a result of transforming VRP's, reformulating them, and solving with Scheduler

Solving with Scheduler: Each of the 224 problems were then reformulated as SSP and solved using ILOG Scheduler. In Figure 3 we see again the percentage change in cost as a result of the transformation (again with black bars for lex ordering, grey for maxMin ordering, and white for minMin ordering). In the clustered problems, C1 and C2, we see a reduction in cost, and a comparatively larger improvement in the relaxed C2 problems. In the random problems, R1 and R2, there tends to be an increase in cost, even greater in the less constrained R2 problems. This might suggest that the transformations do not fare well in unstructured problems. In the RC problems it appears that the transformations reduce cost but only when the problems are less constrained (i.e. the RC2 problems).

Distance appears to be a crucial factor when solving VRP's with VRP technology. When reformulated as scheduling problems, transition times (i.e. travel time) have more of an impact when problems have structure.

5 Conclusion and Future Work

We have presented reformulations between VRP's and SSP's. By solving the reformulated problems with domain specific technologies we have demonstrated that the vehicle routing technology appears to be well suited to VRP's and not at all suited to reformulated SSP's [3]. The converse also appears to hold. However we must add the caveat, that this is no surprise because the benchmark problems used are extreme cases and should indeed be well fitted to their solving technologies.

We have presented a transformation process for the VRP, which can be used as a pre-processing step before solving a problem, or reformulating and solving a problem. This transformation attempts to *compress* the VRP by adding an element of travel into the processing of each node. This was done in the hope that the reformulated problem would appear to be more like a SSP than a VRP, i.e. duration of activities would be increased and transition times would be decreased. Our experiments showed that the transformations can degrade the VRP technology, suggesting that indeed travel is an important problem feature. When reformulated as SSP it was less clear. Although the activities now have increased duration and decreased transition times, they are still peculiar problems: they remain as single activity jobs that can be performed on any resource.

In our ongoing studies we are producing VRP's with more structure, and structure that we can control. This is done by gradually varying time windows, vehicle capacity, heterogeneity of the fleet, sequencing constraints between visits, etc. By gradually increasing the richness of these VRP's we expect to reach a point where SSP technology competes with VRP technology. Our study will attempt to determine just what features bring about this competition. In addition, we plan to investigate the inverse transformation i.e. rather than compress a VRP we might stretch it. Might this be a pre-processing step that will improve VRP solving? Also, when there are set up costs in the SSP, might a further compression improve solving?

Could the transformations and reformulations be of any real use? We believe so. As we add more constraints to the VRP, such as sequencing constraints between visits, restrict time windows, and specialise visits to a subset of the fleet, VRP's will tend to be more like SSP's. Similarly, as the tooling on the shop floor becomes more flexible, such that machines can perform many functions, and the cost of re-configuring tools increases, the SSP will also tend to have features similar to the VRP. Therefore on a spectrum that has the VRP at one end and the JSSP at the other, we expect that as problems become richer it will be less clear as to just what technology is most appropriate. At that time we might expect that transformations and reformulations will become an important tool.

Acknowledgements

This work was done while the first author was employed by ILOG, SA.

References

1. J. Christopher Beck, Patrick Prosser, and Evgeny Selensky. On the reformulation of vehicle routing problems and scheduling problems. Technical Report APES-44-2002, APES Research Group, February 2002. Available from http://www.dcs.st-and.ac.uk/ apes/apesreports.html.
2. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.
3. Evgeny Selensky. On mutual reformulation of shop scheduling and vehicle routing. In *Proceedings of the 20th UK PLANSIG*, pages 282–291, 2001.
4. M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints. *Operations Research*, 35:254–365, 1987.
5. Jssp benchmarks, 15 by 15. http://www.dcs.gla.ac.uk/pras/resources.html.
6. Jssp benchmarks, 6 by 6. http://www.ms.ic.ac.uk/jeb/pub/jobshop1.txt.