Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)

The application of Multilevel Refinement to the Vehicle Routing Problem

Demane Rodney, Alan Soper, and Chris Walshaw

***Abstract*—We discuss the application of the multilevel (ML) refinement technique to the Vehicle Routing Problem (VRP), and compare it to its single-level (SL) counterpart. Multilevel refinement recursively coarsens to create a hierarchy of approximations to the problem and refines at each level. A SL heuristic, termed the combined node-exchange composite heuristic (CNCH), is developed first to solve instances of the VRP. A ML version (the ML-CNCH) is then created, using the construction and improvement heuristics of the CNCH at each level. Experimentation is used to find a suitable combination, which extends the global view of these heuristics. Results comparing both SL and ML are presented.**

***Keywords**: Heuristics, metaheuristic, combinatorial optimization, vehicle routing, multilevel refinement, aggregation techniques*

I. Introduction

DUE to the practical importance of combinatorial optimization (CO) problems in a number of industries, work into finding algorithms for such problems continues to be an active area of research. With this type of problem, the goal is to find an optimal solution from a finite or numerable infinite set of possible solutions [2].

Vehicle routing has emerged as an important CO problem because of its practical significance to a number of industries. In addition, it is of interest in a theoretical context because of its research components being closely related to the Travelling Salesman Problem (TSP – one of the most widely studied CO problems) and the Bin Packing Problem (BPP).

The VRP has many variants with different side constraints [10]. In this paper we consider the capacity vehicle routing problem. The VRP requires the creation of a set of vehicle routes originating and ending at a depot D, serving the demands $d_i > 0$ of n customers, for $i = (1,2,3....n)$. The demand of the depot D is zero. A non-negative cost is defined between any two customers $i \neq j$ as $c_{ij} = c_{ji}$ and between any customer and D. The depot holds V identical vehicles of capacity Q. The total demand of customers on a route should not exceed an upper capacity Q. The cost of a route: given by the sum of the costs between customers on the route and the cost to and from D should not exceed an upper cost M (perhaps relating to the maximum distance a vehicle can travel). The solution then seeks to minimise the total cost of the routes.

The VRP is known to be NP-hard [13] and problem instances solved to optimality normally consist of less than 100 customers. However, since there are real world instances of the problem consisting of thousands of customers [4], there is a need for suitable heuristic approaches capable of producing solutions of usable quality in an acceptable runtime. Laporte et al. [12] provide a survey of heuristic approaches to solving the VRP, while Funke et al. [9] provide a survey of local search techniques used for the VRP.

Walshaw found success in extending multilevel techniques to the TSP [26] where it outperformed many of the traditional methods. The similarities between the two problems (i.e. the TSP and the VRP) made it a logical decision to ask:

How would a ML algorithm for the VRP perform compared to an equivalent single-level version?

How would a ML algorithm compare with other metaheuristics in the field?

These questions provided the motivation for the current research.

A. The Multilevel Refinement Technique

The multilevel technique (ML) [1], [21] has been used for a number of years with proven effectiveness across varying problem areas. These include clustering [11], grid computing [14], graph partitioning, graph colouring and the TSP [25], [26]. Additionally metaheuristics from simulated annealing through genetic algorithm to tabu search have been incorporated into effective multilevel implementations as reported by Walshaw [25].

Coarsening: This forms the construction phase of a ML algorithm where aggregate modelling [17] techniques are applied to the problem. The aggregation process filters the search space creating a simplified model. The model retains the characteristics of the original problem ensuring solutions of the model are applicable to both.

Refinement: This is the process of changing the current solution so that the value of the problem attribute(s) being controlled changes 'favourably' with respect to the cost

Manuscript received October 31, 2006. This work is supported by the University of Greenwich London. The application of Multilevel Refinement to the Vehicle Routing Problem.

Demane Rodney is with the School of Computing and Mathematical Sciences, University of Greenwich, Greenwich, London England SE10 9LS (phone: +44 (0)20 8331 8454; e-mail: d.o.rodney@gre.ac.uk).

Alan Soper is with the School of Computing and Mathematical Sciences, University of Greenwich, Greenwich, London England SE10 9LS (e-mail: a.j.soper@gre.ac.uk).

Chris Walshaw is with the School of Computing and Mathematical Sciences, University of Greenwich, Greenwich, London England SE10 9LS (e-mail: c.walshaw@gre.ac.uk).

function. By favourably we mean that for a minimisation problem the value of the measured quantity is being reduced by this process and vice-versa for a maximisation problem.

A *level* is a period in the solution defined during the coarsening process. In the coarsening phase a level demarcates an approximation to the problem, while in the refinement phase a level demarcates one solution to the problem.

Due to the complexity of CO problems, composite heuristics [8], [19] are the preferred method to find a good solution in reasonable time. Typically composite heuristics consist of two phases. During a construction phase a construction algorithm is applied to create an initial solution. An improvement phase, utilising local search algorithms, is then deployed to improve the solution obtained from the construction phase.

The multilevel refinement technique follows a similar but somewhat different approach. The solution process consists of a coarsening and refinement phase. The initial feasible solution is created at the end of the coarsening phase, which also constructs a hierarchy of approximations to the problem. A refinement phase analogous to the composite heuristics' improvement phase occurs recursively on each approximation in reverse order [20]. Not only does this mean that at each level a feasible solution exists, making the multilevel heuristics faster to run, but also that the hierarchical view of the problem seems to impart a much more global view than can be seen by single-level local search heuristics [25].

0 *set level counter* $i := 0$
1 *set problem* $= P_i$
2 *while* (P_i *can be coarsened*)
3 $P_{i+1} = coarsen(P_i)$
4 $i := i + 1$
5 *end*
6 *Set initial solution* $S_i = P_i$
7 *while* ($i \geq 0$)
8 $i := i - 1$
9 $S_{temp} = extend(S_{i+1})$
10 $S_i = refine\ (S_{temp})$
11 *end*

Fig. 1. Multilevel Algorithm [25].

The multilevel algorithm is shown in Fig. 1. Lines 2 to 6 outline the coarsening phase while lines 7 to 11 encapsulate the refinement phase. Line 10 constitutes the improvement phase of an iterative improvement algorithm. The differentiating feature of the ML algorithm however, is that this is repeated at each level of refinement.

A *segment* is a section of a proposed route having a cost, demand, two end points and a fixed edge. At the start of the solution process, referred to as level zero, a segment represents a single customer (or a vertex). The demand is equal to the customer's demand. The cost of the segment is zero as the end points are the same.

Segments at the upper levels (excluding level zero) are created by fixing an edge of least cost between a pair of existing segments. The edge connects two of the four available end points such that the cost of the edge is minimised. The two unconnected end points become the end points of the created segment. The new segment is then represented by its two end points and a *fixed edge*. The refinement algorithms treat segments of level zero (single customers) and segments of the upper levels (group of customers) in the same manner as the internal structure of a segment is not accessible.

A *fixed edge* is the section of a route within the end points of a single upper level segment. Fixed edges form the internal structure of segments and could represent multiple segments. However segments represented by fixed edges demarcate sections of a route that is not currently available for merging during the coarsening phase or for improvement during the refinement phase.

For the VRP we define the graph $P = (S, E)$. S defines a segment set given by, $S = \{s_0, s_1 \ldots\ldots s_n\}$. Segment s_0 is the depot and the other segments represent customers. E represents the set of edges between the segments given by $E = \{(s_i, s_j) : s_i, s_j \in S, i \neq j\}$. The edges defined by the set E have a non-negative cost, C_{ij} and are termed *free edges*, referring to the fact they can be acted upon by edge-exchange heuristics [9]. At the upper levels where the segments' end points are different, an edge in E defines the edge of least cost that can be used to connect an end point of s_i to an end point of s_j.

Fig. 2 shows the multilevel algorithm solution process. Coarsening the problem achieves two things. Firstly, constructing a solution to the problem, by fixing edges (represented by solid lines) between segments that will hopefully be served on the same route in a final solution of high quality. Secondly, reducing the level of detail to be considered when the problem is refined at each level. When an edge is fixed at a level it is freed when the level is revisited in the extension phase [25]. The refinement phase optimises the free edges (represented by dotted lines) in the solution. Segments that contained fixed edges are treated as single nodes, capable of being optimised by node exchange heuristics [9] however, the internal structure is not considered while fixed.

This means a minimal level of detail is presented to the refinement algorithms at the start of the refinement process. However, the free edges correspond to the edges fixed at the higher levels of the coarsening process and potentially of greater cost as the edges fixed at the lower levels tend to be edges between segments that are 'closer' together. Hence most of the improvements found by the ML algorithm will occur early in the refinement phase.

When the optimisation is completed at a level the solution in place is projected to the level below, the extension process then frees the edges fixed at that level. The refinement process now has a problem of greater detail, but improvements found earlier in the process help make the search faster.

For the example shown in Fig. 2, no improvement is found at levels 3 and 2. For some problems no improvement will be found at the highest levels of the refinement process. However there is refinement at levels 1 and 0, with the refinement at level 1 being done on segments, giving an indication of the gains found from coarsening the problem.

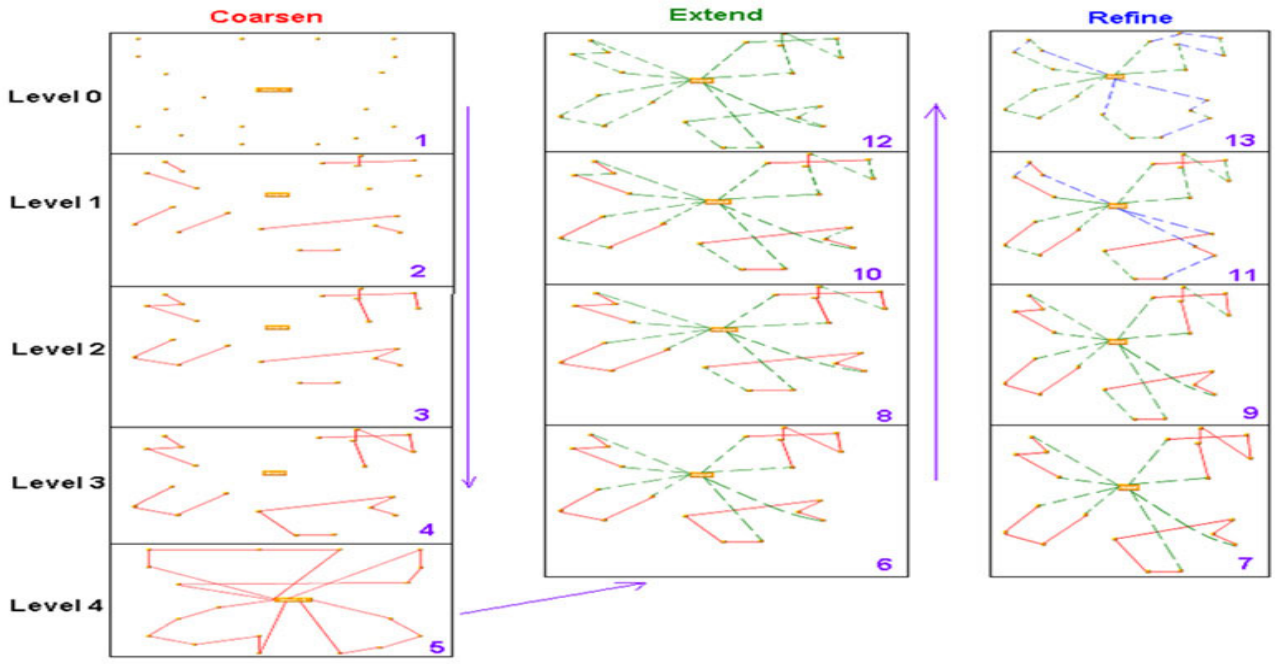

Fig. 2. ML refinement applied to a VRP showing the stages of coarsening, extension and refinement. Continuous lines show fixed edges and broken lines show free edges.

II. Coarsening

The coarsening process is shown on the left of Fig. 2. At the first level (level zero) a starting segment is chosen at random and an edge is fixed between it and another unmatched segment. The process continues while there are pairs of unmatched segments at the current level. When all the matches satisfying the above requirements have been completed the created segments are moved to the following level (level 1) and the process is repeated. When no new segments can be created the last level of coarsening is reached. Segments are matched once at a given level and the created segments should respect the problem capacity and cost constraint.

The Clark Wright saving [6] algorithm was used to select the segments to be matched at each level during the coarsening phase. For the set of segments $C = S \setminus s_0$ we define an ordered set of free edges given by $C_E = \{(s_i, s_j): s_i, s_j \in C, i \neq j, C := C \setminus (s_i, s_j)\}$ .Assume C_E is ordered by decreasing savings in which case s_i and s_j are segments yielding the greatest savings when merged based on the Clark Wright saving algorithm.

The coarsening process at each level defines the sets C and C_E and match segments corresponding to edges resulting in the greatest savings. The set C at each level corresponds to the segments created at the last level excluding the depot. A match consists of fixing an edge between a chosen pair of segments creating a new segment. Edges are fixed between the end points of segments where the greatest saving were found. Checks are done on the segments created to ensure they do not exceed the constraints on the problem. When all allowed matches are done at a given level, a level counter is incremented indicating a new level. The segments created from the last level are used to repeat the process while new segments can be created respecting the problem constraints.

The coarsening process is similar to the one used for the TSP [27]. However the depot is never aggregated during the coarsening process. At the end of the coarsening phase segments are formed having low average values with regard to the objective function. These segments are then joined to the depot to form the initial VRP routes.

III. Refinement

Using the initial solution created from the coarsening phase, the refinement process seeks to iteratively improve the quality of the solution by reducing the total cost. The process uses a combination of inter and intra route optimisation heuristics to explore the neighbourhoods

accessible from the initial solution by the refinement algorithm of Fig. 3.

```
0 set level counter i := last level of coarsening
1 set initial solution Si
2 do
3   expand segments coarsened at level i
4   run Split procedure
5   do
6     refine routes with 3-opt lambda-shift
7     perform cyclic transfer on routes
8   while (improvement found)
9 i := i - 1
10 while (i ≥ 0)
```

Fig. 3. Refinement algorithm.

The refinement process works through the solution levels in the reverse order to that experienced during the coarsening phase, starting at the highest level. As the process descends though the levels the segments in each route are checked to determine if they were coarsened at the present level. If so, they are expanded to reveal their constituting segments, which are used to replace the old segments in the route. The improvement stage is then executed while the cost of the solution can be reduced. The 3-opt exchange is used for intra-route optimisation.

A. Split Procedure

Given a Giant Tour [24]: a Hamiltonian cycle linking all the segments and not the depot, it is possible to find the best way to partition it into sections which are attached at their ends to the depot so as to form routes. This can be done by solving a set-partitioning problem [3] to select the optimal combination of routes [18]. We refer to this procedure as the Split Procedure that can be performed in $O(n^2)$ [16], [7]. The routes it produces respect the capacity and cost constraints of the problem.

Prins [16] has used this procedure to find improvements to VRP solutions. He first removes all the edges from routes which are incident at the depot creating sub-routes. The sub-routes are then joined to form a Giant Tour by connecting the sites at the ends of different sub-routes. Details of how Prins chose the connecting edges were not published. We have joined sub-routes by starting with an arbitrary sub-route end as the start, then connecting the other end of the sub-route to the nearest (using the least cost edge) unconnected sub-route end which is not yet part of the Giant Tour. We repeat the procedure for this extended sub-route and so on. Eventually only the start will be available to connect to, completing the tour. The Split Procedure is then applied to this Giant Tour to recover a VRP solution satisfying any constraints and hopefully of lower cost.

This procedure has the potential to introduce large changes to a solution, for example it has sometimes been seen to produce large 'rotations' of the routes. It has also been useful for implementing rebalancing. A VRP solution that violates the constraints can be used to form a Giant-Tour as above, and then the Split Procedure is applied, but now with the constraints tightened. The formation of a Giant Tour followed by the split can be performed at any level.

B. Lambda – Shift

This is a heuristic created by Osman [15] that searches for improvements by moving vertices between a given pair (p, q) of routes. Osman considers all possible insertions in the route q of a vertex from route p and vice -versa. He also considers exchanges where two vertices on different routes exchange places i.e. each is inserted into the other route with the same neighbours as the removed vertex. Additional transfers involving groups of neighbouring vertices are considered.

Since at higher levels segments represent groups of vertices and the ML algorithm operates on segments, we restrict our attention to the case where only one segment is removed from a route i.e. Lambda equal to one. Most transfers executed at solution levels greater than zero, will be equivalent to a shift of lambda greater than one for a non-ML scheme.

If a transfer leading to an improvement is found, it is immediately accepted and the 3-opt exchange is applied to the affected routes. All pairs of routes are searched.

C. Cyclic Transfers

It has been found that most known low-cost solutions to the VRP are composed of routes that are close to capacity. Thus when searching for inter route improvements, the transfer of a customer from one route to another will normally require the removal of one or more customers from the route into which it has been inserted. The ejected customers will have to be inserted into another route, if the solution is likely to be one of low cost, and so on.

This has lead to the construction of cyclic transfer algorithms, in which the sets of allowed transfers between routes forms a cycle – the customers ejected from the last route are inserted into the first.

Two parameters influence the complexity of a cyclic transfer algorithm; these are the number of customers ejected from each route and the number of routes forming the cycle (cycle depth). The cyclic transfer algorithm implemented follows Thompson [22] but is restricted to the case where only one segment (a multilevel customer/group of customers) is transferred at a time. The number of routes in the cycles is varied during the algorithm execution with the first improvement found accepted.

The least cost insertion point of a customer in another route, for each possible ejection is used. This can be efficiently pre-calculated [22]. Unlike the lambda–shift heuristic the segment can be inserted in a position in the route that is different from the position of the ejected

segment. The search for cyclic transfers that reduce the cost of a solution utilises techniques for finding elementary circuits in graphs [23].

IV. SINGLE-LEVEL

In analysing the effect of the ML philosophy on a given problem it helps to compare the performance of the single-level (SL) counterpart with the ML version being analysed.

0 *set problem = P*
1 *construct initial solution S for P*
2 *While* (*improvement found*)
3 *refine* (*S*)
4 *end*

Fig. 4. Single-level Algorithm.

The SL algorithm shown in Fig. 4 for the implemented combined node-exchange composite heuristic (CNCH) can be classed as an iterative improvement algorithm. Algorithms of this kind find it difficult to escape local optima. A first improvement approach is used in the improvement phase of the CNCH SL and ML versions. This is possibly the least effective approach to use. Unsurprisingly the results of the overall SL approach are poor, however the ML version designed on the same principles produces results comparable to most metaheuristics in the field.

For the SL version of the CNCH coarsening as seen from a multilevel perspective is best viewed as a construction process, in which the initial routes are created in one pass of the algorithm being used. The side constraints in place are satisfied at level zero, hence the results returned from the SL coarsening phase is inline with standard implementation of the Clark Wright saving algorithm.

The removal of lines 0, 2, 3, 9 and 10 from the refinement algorithm of Fig. 3 yields the refinement algorithm for the SL version. For the SL algorithm there is no concept of levels and hence no extension process is required.

The SL algorithm implemented uses the Clark Wright saving algorithm, the Split procedure, 3 - opt exchange, lambda - shift and cyclic transfer algorithms. To convert the CNCH into its ML counterpart consisted of a process of modifying the CWS algorithm so that coarsening became an iterative process across the levels, the implementation of an extension algorithm, and then running the improvement phase of the SL algorithm at each level.

V. BALANCING AND OVERLOADING

Problem constraints prevent moves that would otherwise lower solution costs. This is particularly restrictive at higher levels in ML algorithms when segments have for example in the VRP, demands which are a substantial fraction of the maximum route capacity. For the VRP inter-route moves are prevented by these capacity constraints. With this in mind we have implemented two changes to our algorithm that were found to work well together, segment balancing and route overloading. It was suggested by the results that these heuristics are particularly effective in the cases where the problems are clustered.

A. Segment Balancing

Segment balancing is the process of creating segments during the coarsening phase of approximately the same demand and cost. It was found that inter-route optimisation heuristics reported improved results when using balanced segments. The lambda-shift is used to outline the potential gain from segment balancing

For a given pair (p, q) of routes it is assumed that the demand and capacity of the routes are equal to the maximum values allowed by the problem side constraints. Additionally a lambda-shift exchange is attempted in transferring a segment from p to q and one from q to p. In the case where the segments are balanced if the transfer results in a solution of lower cost the transfer would be successful. In the unbalanced case however this transfer would most likely be prevented by the side constraints.

B. Route Overloading and Balancing

Overloading is the process of gradually relaxing the capacity constraints at each level during coarsening with a view of creating routes with improved solution cost. For the VRP overloading can be thought of as temporally ignoring the bin-packing element of the problem, while obtaining the best TSP solution possible for each route.

Route balancing is the complementary strategy to overloading and is done during refinement. It gradually brings the constraints back inline with the original values imposed by the problem while attempting to preserve the improvements found with regard to the cost function during the overloading process. A feasible solution is found when the constraints are sufficiently tightened to be inline with the problem specified values. The Split Procedure is used to implement route balancing at each level.

VI. RESULTS

The algorithms described above have been tested extensively with a number of standard test cases used by other authors for benchmarking VRP algorithms [5].

As part of this testing, many experiments were performed to choose optimal parameter settings and algorithmic configurations. For example, two heuristics were implemented for the coarsening phase, namely the Clark Wright Savings (CWS) algorithm and the Nearest Neighbour (NN) algorithm. The results indicated that the time spent during coarsening was a negligible percentage of the overall solution time and since the CWS method generally returns marginally better costs (about 0.5%) than the NN it is the method used in the results produced below.

However, it is of interest to look at how different heuristic approaches impact on the results and so in subsection A the

most significant findings are given. Then in subsection B the best results are presented, both in terms of a fixed set of parameters and following other authors' (e.g. [15]) individual parameter settings for each problem.

A. Heuristic Approach Testing

The results in this section present the findings for tests carried out using algorithmic configurations that feature different combinations of segment balancing and route overloading. These factors seem to be most significant in the experimental testing of the algorithms.

Note that when segment balancing is employed, a level demand must be specified. This is a factor governing the targeted demand of segments created at each level and is based on the average demand of customers in the problem. A level demand of 1.2 indicates that the initial level demand is 1.2 times the average demand of customers in the problem. The initial level demand value is then doubled at each level.

Similarly, when route overloading is in use an adjusted capacity factor must be set. Thus the maximum allowed demand for a route is increased at each level in the coarsening phase up to a predefined limit. The maximum allowed demand value is given by the adjusted capacity factor times the problem stated capacity.

Finally, during testing of such algorithms it is important to note that some evaluation of the trade-off between solution quality and run-time must be made. Thus, for the following results, costs are normalized with respect to the best-known values for each problem and averaged over the number of problems in the test suite.

TABLE I. A COMPARISON OF THE DIFFERENT HEURISTICS APPROACHES

Segment balancing	Level demand	Route overloading	Adjusted capacity factor	Normalised average cost
0	-	0	-	1.114
0	-	1	1.6	1.087
1	1.4	0	-	1.085
1	1.2	1	1.2	1.076

Table I shows a comparison of the four different configurations (a zero in the segment balancing and route overloading columns indicates that the heuristic was not used). The normalised cost averaged over all problem instances is given in the final column and thus it can easily be seen that using either segment balancing or route overloading improves the results but that the best configuration occurs when both are combined.

Indeed, when route overloading is used in combination with segment balancing it has the effect of increasing the value of the segment balancing level demand at each level by a factor of the overloaded capacity allowed at that level.

In a situation where there is no overloading, at the start of the refinement phase the total capacity in the solution (equal to the problem capacity multiplied by the number of routes) is close in value to the total demand of the solution. This means that there is a high possibility inter route moves are rejected because they violate the capacity constraint on the routes. Where the solution is overloaded the total capacity of the solution at the start of the refinement phase exceeds the total demand of the solution. This spare capacity makes it possible to implement inter-route improvement moves at the upper levels. Finally, note that many different values for the level demand and adjusted capacity factor settings were tested but that the ones in the table gave the best results for this configuration.

B. Best Results

In this section the best results are given, firstly using a fixed set of parameters values: level demand = 1.2, adjusted capacity factor = 1.2 and cyclic depth = 2. Then following other authors (e.g. [16]) individual parameter settings for each problem where the three parameters above are varied. The algorithms are tested on the Christofides instances [5].

Table II shows a comparison of the results for the multilevel (ML) and single level (SL) versions. The first two columns give the problem number and size. Columns 3 and 4 compare the cost and runtime for the ML and SL algorithms using the fixed set of parameters. Columns 5 and 6 present the best solutions found by the ML algorithm.

As can be seen for a constant set of parameters the ML algorithm can get within an average of 7.13% of the best known with an average runtime of 310 seconds. The SL algorithm is considerably faster (by around a factor of 2, as predicted by Walshaw in a general discussion of multilevel schemes [25]) but gives much worse results. The SL algorithm, when allowed to run for the same time as the ML algorithm return an average cost of 18% above best known values.

The results are particularly interesting for the larger test cases. Prins results [16] while reporting better cost showed large increases in runtimes for the larger test cases. The ML algorithm presents a technique for dealing with larger problems.

In the best case, the ML framework returns an average cost 2.49 % above the best known results with an average runtime of 280 seconds. This is comparable to the improved petal heuristic [18] which reports an average cost of 2.43 above the best known values with an average runtime of 208 seconds.

The results indicated that where the problems are clustered (problems 11, 12, 13, 14) there are improvements to be found in the solution cost from using route overloading and segment balancing. For the problems where the segments are not clustered, overloading provides marginal improvements in some cases.

Whilst more extensive testing needs to be done using larger datasets of clustered problems the results above appear likely to be further validated after increased testing for the following reasons. Firstly, when inter-route heuristics

transfer balanced segments between routes, an improvement is found once the costs of the new edges are less than the costs of the edges being replaced in the affected routes. We assume for a clustered problem that a region of clustered segments exists requiring a given pair (p, q) of routes to serve the demand within the region. New edges generated by transfers of segments between the routes should be similar in cost and similar to the cost of old edges as the segments of each route are close to each other. On average, therefore, most potential moves attempted should return an average cost approximately equal to zero (no improvement or deterioration in the cost function).

In some cases there may be slight differences in the cost of the new edges compared to the old edges. In the cases where this corresponds to a gain the solution is changed. This allows the algorithm to explore areas of the search space that might not have been possible in the case where the problems are not clustered.

The second possible reason for the improvements found could be due to the mechanics of route overloading as applied to clustered regions. Route overloading is most effective where the actual route capacity is increased. An improving move is found when additional customers can be added to the route that will respect the increased capacity constraints without violating the cost constraints. It is not unlikely, therefore, that in a clustered region (where the average cost of the edges to any proposed new customer should be lower than that in a non-clustered region) the algorithm would have a greater likelihood of finding improving moves compared to a non-clustered region.

VII. CONCLUSIONS

It can be seen from the results that the ML version greatly improves the SL version and always returns an improved cost for the equivalent runtime. It is also important to note the differing experiences found from using the ML algorithm on clustered VRP problems and using it on Graph Partitioning Problems and Graph Colouring Problems of high density. As was shown once overloading and segment balancing were introduced the ML algorithm performed well on the clustered problems. The ML algorithm did not perform similarly on graphs of high density. While this requires more investigation the results suggest that the technique once allowed suitable relaxation of the constraints bounding the objection function, does possess the ability to refine problems of high density.

There are a number of areas that the framework could address to achieve further improvements, namely investigating the effect Node Ejection Chains [9] would have in improving the handling of inter–route optimisation. This is one heuristic that should benefit from the ML algorithm, as at the upper levels ejecting a segment would correspond to an entire section of a route being ejected, hence the potential for achieving improvement with limited runtime and ease of implementation.

We also plan to investigate how much influence the clustering within a problem has on the results especially since many real life problems are clustered.

TABLE II. COMPARISON OF THE ML AND SL FRAMEWORK FOR CONSTANT PARAMETER SETTINGS. BEST KNOWN VALUES TAKEN FROM [16]

Problem	N	Percentage Cost above best known for constant set of parameters		Runtime for constant set of parameters (s)		Percentage Cost above best known individual parameter setting for each problem	Runtime using individual parameter setting for each problem (s)	Best known
		ML	SL	ML	SL	ML	ML	
1	50	7.28	8.29	27	27	0.48	29	524.61
2	75	7.86	14.68	76	55	3.40	69	835.26
3	100	3.27	15.81	230	149	1.97	210	826.14
4	150	7.17	22.42	562	237	4.56	442	1028.42
5	199	8.09	28.41	1499	687	6.28	1558	1291.45
6	50	1.92	10.54	26	16	1.38	19	555.43
7	75	5.99	8.78	42	44	3.01	48	909.68
8	100	11.08	13.34	91	86	3.79	161	865.94
9	150	12.08	10.30	318	181	3.83	385	1162.55
10	199	11.27	15.04	618	411	1.42	119	1395.85
11	120	5.96	47.43	391	353	0.93	518	1042.11
12	100	2.83	17.43	117	187	0.27	106	819.56
13	120	10.12	25.84	244	167	3.45	163	1541.14
14	100	5.03	19.04	106	102	0.20	101	866.37
Average		7.13	18.38	310	193	2.49	280	

REFERENCES

[1] Brandt, A. (1977) Multilevel Adaptive Solutions to Boundary Value Problems, Math. Comp, vol.31, pp.333-390.

[2] Blum, C. and Roli, A. (2003) Metaheuristics in combinatorial optimization overview and conceptual comparison, ACM Computing Surveys, Vol. 35, No. 3, pp. 268 - 308.

[3] Boctor, F.F. and Renaud, J. (2000), The column-circular subsets-selection problem: complexity and solutions, Computers & Operations Research, vol.27, pp. 383 – 398.

[4] Caseau, Y. and Laburthe, F. (1999), Heuristics for Large Constrained Vehicle Routing Problems, Journal of Heuristics, vol.5, pp. 281-303.

[5] Christofides, N. Mingozzi, A. and Toth, P. (1979) The Vehicle routing problem. In Combinatorial Optimization. Pp. 315-338. Wiley, Chichester.

[6] Clark, G. and Wright, J.W. (1964) Scheduling vehicles from a central depot to a number of delivery points. Oper. Res. vol. 12, pp – 568 – 581.

[7] Cormen, T. Leiserson, C. and Rivest, R.919900 Introduction to algorithms. Cambridge, MA: MIT Press.

[8] Framinan, J. M. Leisten, R. and Ruiz- Usano, R. (2005) Comparison of heuristics for flowtime minimisation in permutation flowshops. Computer and Oper. Res. vol. 32, pp. 1237 – 1254`

[9] Funke, B. Grunert, T. and Irnich, S. (2005) Local Search for Vehicle Routing and Scheduling Problems: Review and conceptual integration, Journal of Heuristics, vol. 11, pp. 267 -306.

[10] Kilby, P. Prosser, P. and Shaw, P. (2000) A Comparison of Traditional and Constraint-based Heuristic Methods on Vehicle Routing Problems with Side Constraints, Constraints, vol,. 5, pp. 389–414.

[11] Karypis, G. Han, E. and Kumar,V. Multilevel refinement for hierarchical clustering. Technical Report TR-99-020, Department of Computer Science, University of Minnesota, Minneapolis,1999.

[12] Laporte, G. Gendreau, M. and Potvin, J.Y. (2000) Classical and modern heuristics for the vehicle routing problem. International Transactions in operational Research, vol.7, pp.285- 300.

[13] Lenstra, J. K. and Rinnooy, K.(1981) Complexity of vehicle routing and scheduling problem. Networks, Vol. 11, pp. 221-227.

[14] Moulitsas,I. and Karypis, G (2001) Multilevel Algorithms for Generating Coarse Grids for Multigrid Methods, sc, ACM/IEEE SC 2001 Conference (SC'01), pp. 15-24.

[15] Osman, I.H. (1993), Metastrategy simulated annealing and tabu search algorithms for vehicle routing problem, Annals of Oper. Res. vol. 41 pp. 421– 451.

[16] Prins, C. (2004) A simple and effective evolutionary algorithm for the vehicle routing problem, Computers & Operations Research, vol.31,pp. 1985–2002.

[17] Rogers, D.F. Plante, R.D. Wong, R.T. and Evans, J.R.(1991) Aggregation and disaggregation Techniques and Methodology in Optimization, Oper. Res. vol.39, No. 4, pp. 553–582.

[18] Renaud, J. Boctor, F.F. and Laporte, G. (1996), An Improved Petal Heuristic for the Vehicle Routing Problem, Journal of Operational Research Society, Vol. 47, pp. 329–336.

[19] Renaud, J. Boctor, F.F. and Laporte, G. (1996) A fast composite heuristics for the symmetric travelling salesman problem. INFORMS Journal on computing, vol. 8, no.2, pp.134 – 143

[20] Silver, E. (2004) An overview of heuristic solution methods, Journal of the Operational Research Society, vol.55, pp. 936–956.

[21] Teng, S. (1999). Coarsening, sampling and smoothing: elements of the multilevel method. In: Heath MT, Ranade A and Schreiber RS (eds). Algorithms for Parallel Processing. IMA Volumes in Mathematics and its Applications, Vol. 105,Springer Publishing Co. Inc., New York, pp 247–276.

[22] Thompson, P.M. and Psaraftis, H.N. (1993), Cyclic transfer Algorithms for Multivehicle Routing and Scheduling Problems, Oper. Res. vol. 41, No.5, pp. 935-946.

[23] Tiernan, J.C. (1970) An efficient search algorithm to find elementary circuits of a graph, Communications of the ACM, Vol.13, No. 12, pp. 722-726.

[24] Tore Grönert, B.F. and Irnich, S. (2005) Local Search for Vehicle Routing and Scheduling Problems: Review and Conceptual Integration, Journal of Heuristics, Vol. 11, pp. 267-306.

[25] Walshaw, C. (2004) Multilevel Refinement for Combinatorial Optimisation Problems, Annals of Oper. Res. vol. 131, pp. 325–3

[26] Walshaw, C (2002) A Multilevel Approach to the Travelling Salesman Problem, Oper. Res. vol.50, no. 5, pp.862–877.

[27] Walshaw, C. and Cross, M. (2000) Mesh Partitioning: A Multilevel balancing and refinement algorithm, SIAM J. SCI. COMPUT, Vol. 22, No. 1, pp. 63-80.