A Weight-Coded Genetic Algorithm For The Capacitated Arc Routing Problem

Matthew J. W. Morgan
School Of Computer Science
Cardiff University, Queen's Buildings
5 The Parade, Roath, Cardiff
CF24 3AA, United Kingdom
M.J.W.Morgan@cs.cf.ac.uk

Christine L. Mumford
School Of Computer Science
Cardiff University, Queen's Buildings
5 The Parade, Roath, Cardiff
CF24 3AA, United Kingdom
C.L.Mumford@cs.cf.ac.uk

ABSTRACT

In this paper we present a weight coded genetic algorithm (GA) based approach to the capacitated arc routing problem (CARP). In comparison to metaheuristic algorithms, simple constructive heuristic algorithms often produce poor quality solutions to the CARP. Using a novel weight coding model in conjunction with a series of standard CARP heuristics, acting as a solution engine, we demonstrate how these simple heuristic procedures can be 'duped' into producing better solutions to the CARP. The algorithm is tested on a set of problem instances drawn from the literature. Initial results for our GA show that it is possible to reliably produce an uplift in solution quality of between 7.3% and 14.3% above the standard heuristics, the GA identifying 47 optimum or best known solutions from the 57 problem instances tested.

Categories and Subject Descriptors

G.1 [**Numerical Analysis**]: Optimization; G.2.2 [**Discrete Mathematics**]: Graph Theory—*path and circuit problems*

General Terms

Algorithms

Keywords

Genetic Algorithms, Heuristics, Optimization, Transportation

1. INTRODUCTION

In this paper we present a GA based algorithm with fixed parameter settings, which works in conjunction with existing simple heuristic algorithms, to solve the Capacitated Arc Routing Problem (CARP). This is achieved by using a weight coding model, which essentially 'dupes' the heuristics into producing far superior quality solutions for CARP problem instances, than can be obtained from the heuristics, when used in their standard form. The algorithm does not use any additional local search or improvement procedures, resulting in a simple algorithmic implementation.

The transportation of goods and people continues to be a vital function in the world we live in today. The Energy Information Administration (EIA) predicts that over the next 25 years, world demand for liquid fuels and other petroleum is expected to increase more rapidly in the transportation sector than in any other end-use sector [1]. A portion of this increase comes about quite naturally through the economic expansion of poorer countries, however, a large proportion comes from the increasing demands of society as a whole.

Enormous sums of money are spent each year by businesses on logistics, namely on petrol, repairs and associated workforce costs. In addition to these measurable costs, there are other ethical issues that play more and more of a role today, most notably that of carbon emissions. Given the growing evidence of climate change, the effect of overusing transportation systems is potentially catastrophic.

Consider the refuse driver who departs each morning from his depot location to collect the refuse from the sub-network of streets within a city, returning to the depot once all collections have been made. The associated tangible and ethical costs of undertaking these collections can be limited through the derivation of a minimum length route which starts at the depot, traverses each street at least once and finally returns to the initial depot location. The problem that the refuse driver faces is known as an Arc Routing Problem (ARP)

For a small number of collection trips the savings may seem benign, but for the typical city council in the UK, the number of trips required to collect the refuse of its cities occupants can be substantial. This has been further compounded by the recycling targets allocated to councils, which if not met attract large financial penalties. Given the nature of recyclate material, which must not be mixed with food matter, different collection vehicles must be used, resulting in further associated costs.

The multitude of research already undertaken into such transportation problems, is largely due to their real world applicability. Numerous techniques proposed have been used in real world scenarios and many have proved successful in reducing associated transport and emission costs. However, there remains wide scope for new techniques. Through effective planning and the improvement of algorithmic techniques, the burden on our world can be helped.

ARPs can be traced back many years to the infamous

GECCO'09, July 8–12, 2009, Montréal, Québec, Canada.

Königsberg bridge problem. In their simplest form, they require the computation of a route across the edges of a graph network in order to minimise the cost of traversing that route, whilst adhering to any contraints that might exist. A specific problem in the subclass of ARPs is known as the Capacitated Arc Routing Problem (CARP) and places a capacity restriction on the quantity of goods that can be carried by any vehicle at a point in time.

Due to the very nature of vehicles, they can hold only a limited capacity, and typically more than one vehicle is required to service all the required edges (i.e. those with serviceable demands). A set of routes, one for each vehicle, with the aim of minimising total travelling distance must be computed.

The CARP can be modelled as two variants. The first variant of the CARP, where each edge has a positive demand, i.e. $d_{ij} > 0$, $\forall\{v_i, v_j\} \in E$, was first proposed by Christofides [9]. Consider a connected undirected weighted graph $G = (V, E)$ with associated edge costs c_{ij} and edge demands d_{ij}, $\forall\{v_i, v_j\} \in E$. A depot located at node $v_1 \in V$ serves as the base location for a quantity of vehicles, each with a holding capacity Q . The requirement is a set of m cycles in G, each starting/ending at the depot node, where each required edge is serviced exactly once, such that the cost of traversing all cycles is minimised and the total demand of each cycle does not exceed Q.

A second variant of the CARP was proposed by Golden and Wong [13] and has identical parameters to those in the Christofides model. The only difference between the two is the quantity of required edges within the graph. In contrast to the first variant, where every edge has a serviceable demand, only a subset $R \subseteq E$ of all edges present need to be serviced. The traversal of such non required edges is known as 'deadheading'.

Many simple and fast heuristics have been proposed for the solution of different combinatorial optimization problems. In general, the cost of such simplicity and efficiency is an inferior solution quality. To obtain superior quality solutions, the use of alternative methods such as Tabu Search and other metaheuristics has become the norm. In turn, these methods have become far more complicated from both an understanding and implementation perspective. To be truly effective, some require the use of very powerful computers not available to the average user.

The use of a weight coding model within a GA framework, allows existing simple heuristics to be utilised, can be run on standard computer hardware and is very easy to implement. The following sections introduce the notion of weight coding, outline two simple heuristic algorithms for the CARP, both tried and compared as heuristic engines in the weight coded GA, detail the different elements of the GA model and finally present a series of preliminary experiments and computational results, tested against a number of standard benchmark problem instances from the literature.

2. WEIGHT CODING

The application of weight coding has been demonstrated using a wide range of combinatorial optimization problems, such as the optimum communications spanning tree problem [22], the shortest common supersequence problem [6], the rectilinear Steiner tree problem [15], the minimum weight triangulation problem [7], the traveling salesmen problem [16], the multiple container packing problem [24], the multiconstraint knapsack problem [25] and the degree-constrained mimimum spanning tree problem [26].

Weight coding is a scheme used in conjunction with a GA. Chromosomes are encoded with weights, in the case of the TSP, each city is associated with a numeric weight. Each chromosome is used in conjunction with the distances for a given problem instance P, to produce a weight coded instance P'. A problem specific heuristic is then used to identify the solution for the instance P', which is then decoded using the original instance data P, to provide a solution and corresponding distance which is then used as the fitness function for the GA.

$$P = \begin{bmatrix} 4 & 5 & 2 & 6 & 7 & 8 & 3 \end{bmatrix}$$

Figure 1: Weight coded chromosome.

Figure 1 illustrates a typical chromosome for a weight coded GA to solve the TSP. The length of each chromosome is the same as the number of cities contained within a problem instance. Chromosomes are encoded from city 1 to n, in the case of this example, city 1 has the weight 4 associated with it and city 7, the weight 3.

$$C = \begin{bmatrix} 0 & \mathbf{21} & 35 & 36 & 21 & 25 & 16 \\ 21 & 0 & 21 & 32 & 29 & 16 & 18 \\ 35 & 21 & 0 & 18 & 30 & 11 & 22 \\ 36 & 32 & 18 & 0 & 21 & 16 & 21 \\ 21 & 29 & 30 & 21 & 0 & 20 & 11 \\ 25 & 16 & 11 & 16 & 20 & 0 & 11 \\ 16 & 18 & 22 & 21 & 11 & 11 & 0 \end{bmatrix}$$

Figure 2: Problem instance distance matrix C.

Each chromosome is used as a template to derive a new inter-city cost matrix from that of the original problem instance. Consider the cost matrix shown in figure 2. For each distance in the cost matrix C between 2 cities, the associated weights in the chromosome for both those cities are added to the original distance to produce a new weight coded distance matrix C', shown in figure 3.

For example, the weight coded integer values of 4 and 5 in chromosome P, for customers 1 and 2 respectively, are added to the distance of 21 between customers 1 and 2 in C, to produce a weight coded value of 30 in C'

$$C' = \begin{bmatrix} 0 & \mathbf{30} & 41 & 46 & 32 & 37 & 23 \\ 30 & 0 & 28 & 43 & 41 & 29 & 26 \\ 41 & 28 & 0 & 26 & 39 & 21 & 27 \\ 46 & 43 & 26 & 0 & 34 & 30 & 30 \\ 32 & 41 & 39 & 34 & 0 & 35 & 21 \\ 37 & 29 & 21 & 30 & 35 & 0 & 22 \\ 23 & 26 & 27 & 30 & 21 & 22 & 0 \end{bmatrix}$$

Figure 3: Weight coded distance matrix C'.

An outline of the framework for the solution of the CARP is shown in figure 4. The initial population of chromosomes

is constructed at random. Offspring that result after the application of a series of genetic operators are used in conjunction with the original cost matrix data, to produce a new altered matrix.

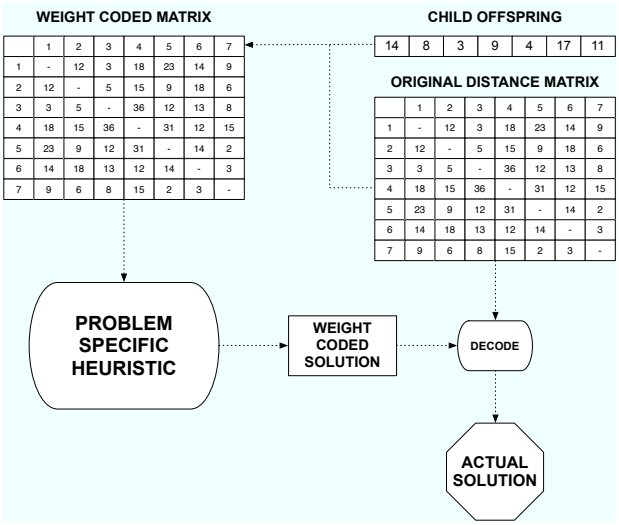

Figure 4: Overview of the solution process within the weight coded GA framework for the CARP.

Through each iteration of the GA, each offspring chromosome generated is used in conjunction with the original unaltered distance matrix C to derive a weight coded distance matrix C'. The newly created matrix is fed to a problem specific heuristic which generates a weight coded solution from the altered distances of the weight coded matrix.

The resulting solution, based upon the weight coded distances, is subsequently decoded using the distances in the original unaltered problem instance, resulting in a 'true' solution for the CARP instance.

The philosophy of weight coding is similar to that of the method of perturbation, which has been used under many guises and applied to a range of combinatorial optimization problems. The basic principle of of this approach is to introduce a number of perturbations into the problem instance data, providing a means to escape from a local optima. It was first outlined by Storer et. al. [27] and Charon and Hudry [8].

Codenotti et. al. [10] later applied these techniques within an ILS algorithm to solve the Traveling Salesman Problem (TSP). However, instead of perturbing the solution alone, the algorithm by Codenotti et. al., additionally perturbed the city coordinates in the problem instance itself. Valenzuela and Williams [28] and Bradwell et. al. [4] proposed variations of this method. The algorithms breed perturbed instance data, encoded as chromosomes within a Genetic Algorithm framework, to solve the TSP. A Nearest Neighbour and Karp heuristic algorithm are used to construct solutions from the chromosomes of perturbed coordinates within the population at any point in time.

A similar GA based algorithm, using perturbed instance data and a Clarke & Wright heuristic as a solution mechanism, was proposed by Morgan and Mumford [21] for the solution of the Capacitated Vehicle Routing Problem (CVRP).

3. CARP HEURISTIC ALGORITHMS

For $\mathcal{NP}$-Hard optimization problems such as the CARP, heuristic methods provide a mechanism for the production of, in the most part, good quality solutions, for large problem instances (allbeit generally not optimal), within realistic time frames. Numerous heuristic algorithms have been proposed for the CARP. The following section describes two of the most notable, providing a more in depth analysis for those algorithms that form the basis of the implementations described later.

3.1 Augment Merge Algorithm

The Augment Merge Algorithm (AMA) was originally proposed by Golden and Wong [13] and consists of 3 distinct stages. Its mechanisms are similar to the Clark & Wright algorithm used in the solution of the CVRP. The 3 phases are: initialisation, augmentation and merge.

Initialisation
The procedure begins by constructing a set of initial routes such that each route services precisely one servicable edge in the graph. For each edge selected, the shortest path from each of its endpoints back to the depot is calculated and a cycle constructed. All cycles generated are then tabulated in descending order based upon travelling distance, and each cycle assigned a numeric identifier. The longest cycle is labelled 1, the next in the sorted table 2, down to the last cycle n (initially equal to the total number of serviced edges present in the graph).

Augmentation
The process of augmentation then involves selecting each cycle in turn (the master route) starting at cycle 1 and evaluating the inclusion of each servicable edge in the shorter sibling cycles (i.e. all cycles in the table below the master cycle) into the currently selected master cycle. If the servicable edge in a sibling cycle can be serviced on the master cycle, whilst adhering to any vehicle capacity constraints, the master cycle is updated to include the service of that edge and the sibling cycle deleted.

Once all sibling cycles have been evaluated against a particular master cycle, the next cycle in the table becomes the master and it's sibling cycles are evaluated using the same procedure. The process continues selecting master cycles until the bottom of the table is reached and no further cycles exist. If at any point in time the total demand on the master cycle becomes equal to the capacity of the vehicles available, it is immediately set aside and the next cycle in the table is selected as the master.

Merge
The merge phase further refines the cycles derived during augmentation. The selection mechanism remains the same as that for augmentation. The first cycle in the table (not full to capacity) is selected as the master and each cycle following it in the table is selected in turn to take on the role of the sibling. Each master/sibling combination is then evaluated to assess the feasibility of a merger into a single cycle. A merger is only valid if the new cycle services every edge originally present in the master and sibling cycles. For each valid merger, the total distance saving achieved by combining the cycles is calculated as follows:

$$S_{ij} = l_i + l_j - m_{ij} \quad (1)$$

where:

S_{ij} = saving achieved from the merger of cycles i and j

l_k = length of cycle k pre merger

m_{ij} = length of post merger cycle resulting from combination of cycle i and j

3.2 Path Scanning Algorithm

In the Path Scanning Algorithm (PSA) of Golden et. al. [12] the procedure iteratively builds single cycles, each starting from the depot node, resulting in a final solution made up of a set of multiple cycles. It is run five separate times, each time using a different rule set to build solutions, after which the best overall solution is then selected from all of those generated.

Each rule set utilises a different selection criterion to obtain the next edge (i, j), along which to extend the current cycle route being built. Edges are chosen, subject to vehicle capacity constraints, using the following rule sets:

1. the cost/demand ratio c_{ij}/d_{ij}, where c_{ij} is the distance and d_{ij} the demand for edge i to j, is minimised.
2. the cost/demand ratio c_{ij}/d_{ij} is maximised.
3. the distance from node j (i.e. the end of the edge) back to the depot is minimised.
4. the distance from node j back to the depot is maximised.
5. if the vehicle is less than half full, use rule 4 to select next edge, else use rule 3.

The process begins by selecting the first rule and with a set R containing all required edges in that instance and an empty path P. The path P is then extended, one edge at a time, until the vehicle is full to capacity. For each edge extension, a set S containing all edges in R not exceeding the capacity of the route currently being extended, is generated. An edge is then selected from S using the chosen rule and P extended along that edge.

This process is further repeated for all remaining rules and the best of the solutions generations using the 5 different rules is chosen.

4. THE GA MODEL

The following section outlines the specific details of the different elements of the GA model. These include the chromosome representation, structure and initialisation of the population, selection procedure, crossover and mutation mechanisms and the method of generating and decoding solutions obtained from the heuristic procedure.

4.1 Chromosome encoding

The encoding scheme and manipulation of the distance matrix are identical to that described for the TSP in section 2. Each chromosome contains a sequence of customers, from 1 to n, with each position holding a weight coded integer value x_n corresponding to a customer vertex. The simple encoding scheme, illustrated in figure 5, allows standard genetic operators to be used without the worry of infeasible solutions being produced.

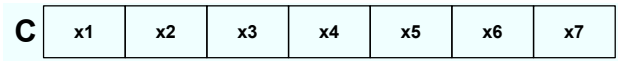

Figure 5: Weight coded chromosome encoding

4.2 Population structure and initialisation

The initial population consists of p chromosomes, where p equals the chosen size of the population. For each chromosome, a weight coded integer is randomly generated within a preset range and associated with each customer vertex. The process is then repeated until the required p chromosomes have been created.

Intensive investigation has shown a larger size of around 250 to be more suitable. It should be noted that this size is only relevant when the path scanning algorithm is used. For other heuristic algorithms with poorer scalability and running times, smaller population sizes may have to be used, if excessive times are to be avoided.

4.3 Selection, Crossover and Mutation

The process of parent chromosome selection and crossover involves the systematic selection of population members in turn which are paired with another randomly selected member of the population, through each single generation of the GA. A Crossover operation is further applied to each offspring. A 2 Point Crossover (2PX) operator is used throughout to produce child offspring.

Following crossover, a number of mutations are applied to each offspring chromosome. The mutation scheme involves the random selection from the chromosome of up to 2 weight coded integer values, each of which being replaced by a newly generated integer weight within the same predefined range.

4.4 Solution Mechanism and Decoding

Each offspring produced after selection, crossover and mutation has taken place, serves as the input to the chosen problem specific heuristic. The offspring chromosome is used to generate a weight coded distance matrix, altering the original distance matrix to reflect the values held within the chromosome. The new matrix is used by the heuristic to generate a solution, containing routes for all the vehicles, for the given problem instance. The solution obtained is decoded to produce its actual total cost using the original unaltered distance matrix.

The 'true' solution distance, which serves as the fitness of the chromosome, is then evaluated against that of the weaker of the two parent chromosomes used to create the offspring. If superior, the weaker parent chromosome in the population is replaced with that of the child offspring. Offspring whose 'true' distance is worse than both parents are discarded. Finally, the value of the best distance so far is updated, iff the decoded solution distance for the solution generated from the child offspring, is superior to that of the currently stored best overall solution distance.

5. COMPUTATIONAL EXPERIMENTS

A number of authors have published dataset instances for the CARP, namely DeArmon [11], Benavent et. al. [2], Li [19], Li and Eglese [20] and Kiuchi et. al. [17]. The DeArmon instances, presented in table 1 are called *gdb* and consist of 23 problem instances with a mixture of dense and sparse graph networks.

Problem Instance	$\|V\|$	$\|E\|$	$\sum d$	Q	Density
gdb1	12	22	22	5	33
gdb2	12	26	26	5	39
gdb3	12	22	22	5	33
gdb4	11	19	19	5	35
gdb5	13	26	26	5	33
gdb6	12	22	22	5	33
gdb7	12	22	22	5	33
gdb8	27	46	249	27	13
gdb9	27	51	258	27	14
gdb10	12	25	37	10	38
gdb11	22	45	225	50	19
gdb12	13	23	212	35	30
gdb13	10	28	240	41	63
gdb14	7	21	89	21	100
gdb15	7	21	112	37	100
gdb16	8	28	116	24	100
gdb17	8	28	168	41	100
gdb18	8	36	153	37	100
gdb19	8	11	66	27	20
gdb20	11	22	106	27	40
gdb21	11	33	154	27	60
gdb22	11	44	205	27	80
gdb23	11	55	266	27	100

Table 1: De Armon dataset analysis

Problem Instance	$\|V\|$	$\|E\|$	$\sum d$	Q	Density
val1	24	39	358	45-200	14
val2	24	34	310	40-180	12
val3	24	35	137	20-80	13
val4	41	69	627	75-225	8
val5	34	65	614	75-220	12
val6	31	50	451	50-170	11
val7	40	66	559	65-200	8
val8	30	63	566	65-200	14
val9	50	92	654	70-235	8
val10	50	97	704	75-250	8

Table 2: Benavent et. al. dataset analysis

The second set of instances from Benavent et. al., presented in table 2, are called *val* and comprise 34 problems, modelled on 10 sparse graph networks, with varying vehicle capacities for each network. The Li and Eglese instances are derived from real world data, relating to winter gritting in the county of Lancashire, UK and consist of 24 problems.

The *gdb* and *val* data instances include only required edges, in contrast to those from Li and Eglese where a mixture of both required and non-required edges are found. For the purposes of this paper, experimentation has been curtailed to those problem instances, namely *gdb* and *val*, containing only required edges.

5.1 Preliminary Experimentation

A series of preliminary experiments were undertaken to assess the effectiveness of using AMA and PSA as the heuristic engines in the GA.

5.1.1 Comparing AMA and PSA as heuristic engines for the GA

Initial experimentation was limited to using the PSA and AMA heuristics as the solution generation mechanism, with the following settings:

Population size:	250 (PSA) & 100 (AMA)
Crossover:	2PX
Mutation rate:	1 or 2 mutations
Stopping criterion:	1,500 generations

For all reported results, a statistic known as Relative Deviation (RD) is reported. This figure measures the RD from the best known or optimum solution for a problem instance, for each solution generated. It is stated for all individual solutions presented and also as an average over various problem instance sets. The calculation for RD is made using equation 2 and reported as a percentage value.

$$Dev\ \% = \left(\frac{\text{solution} - \text{optimum/best known}}{\text{optimum/best known}} \right) \times 100 \quad (2)$$

Each combination was run against a subset of the *gdb* and *val* problem instances and the results attained, for the different ranges, are presented in table 3. The average deviation % across the *gdb* and *val* instances tested are shown for the ranges 10%, 50% and 100%.

For each range, the mutation of a weight coded integer results in a change to the current weight coded integer within the chromosome by up to the particular preset percentage value. e.g. For a range of 50%, the mutation of a weight coded integer value of 10 would be ±5, producing a mutated value of between 5 and 15.

Method (instance subset)	Av Dev % 10%	Av Dev % 50%	Av Dev % 100%
AMA (gdb)	0.67	0.53	0.49
PSA (gdb)	0.25	0.21	0.24
AMA (val)	4.40	5.11	6.31
PSA (val)	3.12	2.89	3.41

Table 3: CARP preliminary results for different ranges of adjustment for weight coded integers.

As can be seen, the results are quite robust across all ranges evaluated, however, the runtime when using the path scanning method is superior when compared to augment merge. Although the augment merge method is capable of producing quality solutions to CARP problem instances given sufficient time, is does not lend itself as a valid solution mechanism with such poor scalability. In contrast, the path scanning approach scales well and offers superior solution quality. Given this fact, the solution mechanism for the CARP has been restricted to the path scanning approach.

5.1.2 Convergence and solution uplift

The uplift in solution quality obtained from the integration of the PSA and AMA heuristics into the weight coded GA framework in comparison to using these heuristics in their standard form is shown in figure 6.

Using the same settings are those described for the comparison of the AMA and PSA as heuristic engines, the weight coded GA has succeeded in substantially lifting the solution quality of the PSA and AMA heuristics by margins of

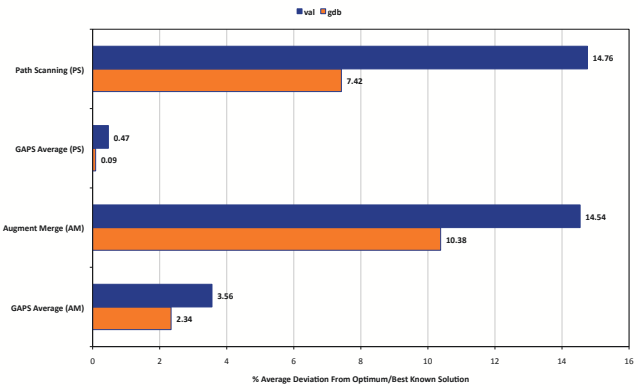

Figure 6: Uplift in solution quality using the weight coded GA in comparison to the standard heuristics for instance sets gdb and val.

between 7.3% and 14.3% across the *gdb* and *val* problem instance sets.

5.2 Computational Results

Utilising the same set of predefined parameters as those used for the preliminary experiments, the algorithm was run against two sets of problem instances, using the PSA heuristic and a range of 50% for all mutations. A total of 50 runs were carried out for each individual problem instance using a Pentium IV 2.8GHz computer, running a GNU/Linux Operating System.

The state of the art presented in the literature for the solution of the CARP are all based upon metaheuristic techniques. The results obtained from three of the most successful algorithms have been used as the basis of comparison against those obtained from the weight coded GA. These are a tabu search algorithm called "CARPET", a memetic algorithm "MA" and a deterministic tabu search algorithm called "TSAv2".

Another algorithm to solve the CARP with intermediate facilities was presented by Polacek et. al. [23]. The authors ran their algorithm on the basic CARP variant and detailed results for the *val*/*eglese* problem instances. The results attained by the procedure are on a par with those for the MA, but provide a superior run time performance in comparison.

Experimental results for the weight coded GA are shown in tables 4 and 5. The column headed "CARPET" give the results reported by Hertz et. al. [14], for an adapted version of TS called CARPET. The column headed "MA" show the results by Lacomme et. al. [18] using a memetic algorithm. The column headed "TSAv2" presents results for a deterministic tabu search algorithm by Brandâo and Eglese [5].

The CPU computing time in seconds reported for weight coded GA are achieved using the hardware described. The runtimes reported by other authors have been scaled in line with those reported by Brandâo and Eglese. The average deviation from the best known solution for each problem instance is presented and calculated using equation 2.

For the *gdb* set of problems instances, the average results for the weight coded GA are better than those of CARPET and very similar to those presented for TSAv2. However, the results for MA provide a slightly superior solution quality. With the exception of *gdb8* and *gdb9*, the weight coded GA has been able to identify the best known solutions for all *gdb* problem instances.

In contrast, the success of the weight coded GA for the *val* problem instance set is mixed, providing slightly inferior results, when compared to the results from the other algorithmic approaches. Clearly, good or in most cases best known solutions for the A and B variants of each problem are achieved. However, in the case of the C and D variants, only limited success has been achieved, the quality of solution degrading in line with an increase in problem size.

6. CONCLUSIONS

The trend amongst researchers over the last decade has been the development of more and more powerful algorithms for the solution of optimization problems. However, the result of these endeavours is often ever increasingly complex algorithms, which are typically both difficult to understand and implement.

Another side effect is often the desire of many authors to produce new best results for benchmark instances from the literature, resulting in algorithmic techniques which are not generic in nature and whose ability to provide good quality solutions for unseen instances is arguably questionable.

The key philosophy in the development of the weight coded GA framework is simplicity, making it is easy to both understand and implement. The experimental results for the weight coded GA, run against standard benchmark instances, clearly demonstrate the dramatic uplift in solution quality from using standard heuristic techniques alone, brought about through the integration of these problem specific heuristics within a genetic algorithm framework, in conjunction with a weight coded model.

Improvements over the standard heuristics are achieved for the CARP using a weight coded scheme. Over the same configuration of experimental runs, the weight coded GA has identified 47 optimum or best known solutions from the 57 problem instances tested. However, in the case of weight coding, it is evident that superior quality solutions are more easily obtained for smaller problem instances.

Given that the use of classical heuristics are still prevalent in commercial software today, largely due to the relative solution quality attained in relation to quick runtime execution, the substitution of a simple framework like the weight coded GA would arguably represent a realistic alternative method, providing consistently superior quality solutions, in realistic time scales.

7. FUTURE WORK

The extension of the present study to benchmark a wider range of larger problem instances is the most obvious course of action. Given that experimentation was curtailed to problem instances with only required edges, extension to encompass problem instances that include non required edges and those derived from real world data, would be the obvious priority.

Valid instances would be those by Beullens et. al. [3], based upon the road network in Flanders, Belgium, and by Brandâo and Eglese [5], derived from a winter gritting study in Lancashire.

It is also apparent that using the current scheme, all edge distances in the matrix are altered through weight coding, ir-

Problem	Best Known	CARPET Cost	CARPET CPU (s)	MA Cost	MA CPU (s)	TSA v2 Cost	TSA v2 CPU (s)	WC GA Cost	WC GA CPU (s)	WC GA Best
gdb1	316	316	2.4	316	0.0	316	0.0	316	4.8	316
gdb2	339	339	4.0	339	0.3	339	0.1	339	3.7	339
gdb3	275	275	0.1	275	0.0	275	0.0	275	0.2	275
gdb4	287	287	0.1	287	0.0	287	0.0	287	0.8	287
gdb5	377	377	4.3	377	0.1	377	0.1	379	14.6	377
gdb6	298	298	0.7	298	0.1	298	0.0	298	1.2	298
gdb7	325	325	0.0	325	0.1	325	0.0	325	1.4	325
gdb8	344	352	47.2	350	26.5	348	1.6	356	23.3	348
gdb9	303	317	41.8	303	4.7	303	26.1	309	96.4	303
gdb10	275	275	1.2	275	0.1	275	0.0	275	0.4	275
gdb11	395	395	1.8	395	0.9	395	0.1	395	21.3	395
gdb12	458	458	16.0	458	6.5	458	0.8	462	12.6	458
gdb13	536	544	1.9	536	4.9	540	4.8	539	17.4	536
gdb14	100	100	0.4	100	0.1	100	0.1	100	0.4	100
gdb15	58	58	0.0	58	0.0	58	0.0	58	0.1	58
gdb16	127	127	1.3	127	0.1	127	0.1	127	0.2	127
gdb17	91	91	0.0	91	0.1	91	0.0	91	0.1	91
gdb18	164	164	0.2	164	0.1	164	0.0	164	0.9	164
gdb19	55	55	0.2	55	0.0	55	0.0	55	0.1	55
gdb20	121	121	7.4	121	0.2	121	0.2	121	0.4	121
gdb21	156	156	0.9	156	0.1	156	0.0	156	2.3	156
gdb22	200	200	2.6	200	2.3	200	0.1	200	19.4	200
gdb23	233	235	26.6	233	34.1	235	22.3	234	15.2	233
Average Dev (%)		0.47		0.04		0.08		0.34		0.00

Table 4: Computational results for instance set *gdb*.

Problem	Best Known	CARPET Cost	CARPET CPU (s)	MA Cost	MA CPU (s)	TSA v2 Cost	TSA v2 CPU (s)	WC GA Av Cost	WC GA Av CPU (s)	WC GA Best
val1A	173	173	0.0	173	0.0	173	0.0	173	1.2	173
val1B	173	173	7.2	173	5.3	173	0.9	174	37.2	173
val1C	245	245	72.3	245	19.1	245	12.1	248	19.9	245
val2A	227	227	0.1	227	0.1	227	0.0	227	4.3	227
val2B	259	259	10.1	259	0.1	259	0.3	260	10.6	259
val2C	457	457	24.5	457	14.5	457	7.8	474	15.2	468
val3A	81	81	0.6	81	0.1	81	0.0	81	1.9	81
val3B	87	87	2.1	87	0.0	87	0.0	88	4.3	87
val3C	138	138	32.2	138	18.8	138	1.3	140	15.2	138
val4A	400	400	21.9	400	0.5	400	0.4	403	57.4	400
val4B	412	412	58.6	414	0.8	412	5.5	417	67.4	412
val4C	428	428	54.2	428	12.7	428	38.0	452	96.4	450
val4D	530	530	180.8	541	68.9	530	110.0	575	78.3	569
val5A	423	423	2.9	423	1.3	423	0.3	423	71.1	423
val5B	446	446	32.0	446	0.7	446	0.1	447	84.9	446
val5C	473	474	41.3	474	67.3	474	10.6	481	92.7	479
val5D	571	577	173.5	583	60.5	583	73.3	607	124.3	598
val6A	223	223	3.0	223	0.1	223	1.6	223	14.4	223
val6B	233	233	20.9	233	44.9	233	12.7	234	92.7	233
val6C	317	317	66.0	317	34.8	317	22.9	338	42.6	337
val7A	279	279	5.1	279	1.3	279	1.0	279	16.3	279
val7B	283	283	0.0	283	0.3	283	0.5	283	41.6	283
val7C	334	334	94.0	334	67.5	334	37.0	339	44.7	337
val8A	386	386	3.0	386	0.5	386	0.3	386	52.4	386
val8B	395	395	63.1	395	6.7	395	1.8	396	45.7	395
val8C	521	521	114.1	527	47.7	529	55.7	569	78.6	545
val9A	323	323	22.1	323	12.2	323	0.0	326	56.6	323
val9B	326	326	46.4	326	19.6	326	0.5	335	81.2	326
val9C	332	332	43.7	332	47.5	332	0.4	336	102.4	332
val9D	385	391	273.5	391	140.7	391	60.4	421	78.9	410
val10A	428	428	4.3	428	17.0	428	3.2	433	67.4	429
val10B	436	436	14.3	436	3.1	436	1.8	452	97.3	436
val10C	446	446	72.4	446	11.5	446	7.5	471	81.5	447
val10D	526	528	121.0	530	143.3	530	218.1	560	114.9	557
Average Dev (%)		1.86		0.23		0.15		2.45		1.38

Table 5: Computational results for instance set *val*.

respective of whether the edges are required or non-required. Perturbing edges distances for any traversals along non-required edges could potentially be detrimental. However, further investigation would be required in order to substantiate this theory.

8. REFERENCES

[1] E. I. Administration. *International Energy Outlook 2008*. Document No. DOE/EIA-0484(2008) at, 2008. URL: http://www.eia.doe.gov/oiaf/ieo/index.html.

[2] E. Benavent, V. Campos, A. Corber, and E. Mota.

The capacitated chinese postman problem: Lower bounds. *Networks*, 22(7):669–690, 1992.

[3] P. Beullens, L. Muyldermans, D. Cattrysse, and D. V. Oudheusden. A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research*, 147:629–643, 2003.

[4] R. Bradwell, L. Williams, and C. Valenzuela. Breeding perturbed city coordinates and 'fooling' a travelling salesman heuristic algorithm. In *Third International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA97)*, pages 241–249, 1997.

[5] J. Brandâo and R. Eglese. A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers and Operations Research*, 35:1112–1126, 2008.

[6] J. Branke and M. Middendorf. Searching for shortest common supersequences by means of a heuristic-based genetic algorithm. In *Second Nordic Workshop on Genetic Algorithms and their Applications Proceedings*, pages 105–113, 1996.

[7] K. Capp and B. Julstrom. A weight-coded genetic algorithm for the minimum weight triangulation problem. In *ACM Symposium on Applied Computing Proceedings*, pages 327–331, 1998.

[8] I. Charon and O. Hudry. The noising method: a new method for combinatorial optimization. *Operations Research Letters*, 14:133–137, 1993.

[9] N. Christofides. The optimal traversal of a graph. *Omega*, 1:719–732, 1973.

[10] B. Codenotti, G. Manzini, L. Margara, and G. Resta. Perturbation: an efficient technique for the solution of very large instances of euclidean ts. *INFORMS Journal on Computing*, 8:125–133, 1996.

[11] J. DeArmon. *A Comparison of heuristics for the capacitated Chinese Postman problem*. Master's thesis, University of Maryland, University of Maryland, College Park, MD, 1981.

[12] B. Golden, J. DeArmon, and E. Baker. Computational experiments with algorithms for a class of routing problems. *Operations Research*, 10(1):47–59, 1983.

[13] B. Golden and R. Wong. Capacitated arc routing problems. *Networks*, 11:305–315, 1981.

[14] A. Hertz, G. Laporte, and M. Mittaz. A tabu search heuristic for the capacitated arc routing problem. *Operations Research*, 48(1):129–135, 2000.

[15] B. Julstrom. Representing rectilinear steiner trees in genetic algorithms. In *ACM Symposium on Applied Computing Proceedings*, pages 245–250, 1996.

[16] B. Julstrom. Insertion decoding algorithms and initial tours in a weight-coded ga for tsp. In *Genetic Programming Proceedings*, pages 528–534, 1998.

[17] M. Kiuchi, Y. Shinano, R. Hirabayashi, and Y. Saruwatari. An exact algorithm for the capacitated arc routing problem using parallel branch and bound method. In *Spring National Conference of the Operational Research Society of Japan*, 1995.

[18] P. Lacomme, C. Prins, and W. Ramdane-Cherif. Competitive memetic algorithms for arc routing problems. *Annals of Operational Research*, 131(1-4):159–185, 2004.

[19] L. Li. *Vehicle Routing for Winter Gritting*. Ph.D thesis, Department of Management Science, Lancaster University, Lancaster, 1992.

[20] L. Li and R. Eglese. An interactive algorithm for vehicle routing for winter-gritting. *Journal of the Operational Research Society*, 47:217–228, 1996.

[21] M. Morgan and C. Mumford. Capacitated vehicle routing: perturbing the landscape to fool an algorithm. In *Congress on Evolutionary Computation*, pages 2271–2277, 2005.

[22] C. Palmer and A. Kershenbaum. Representing trees in genetic algorithms. In *1st International Conference on Evolutionary Computations Proceedings*, pages 379–384, 1994.

[23] M. Polacek, K. Doerner, R. F. Hartl, and W. Ramdane-Cherif. A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. *J. Heuristics*, 14(5):405–423, 2008.

[24] G. Raidl. A weight-coded genetic algorithm for the multiple container packing problem. In *IEEE Congress on Evolutionary Computation Proceedings*, pages 291–296, 1999.

[25] G. Raidl. Weight-codings in a genetic algorithm for the multiconstraint knapsack problem. In *IEEE Congress on Evolutionary Computation Proceedings*, pages 596–603, 1999.

[26] G. Raidl and B. Julstrom. A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. In *ACM Symposium on Applied Computing Proceedings*, pages 440–445, 2000.

[27] R. Storer, S. Wu, and R. Vaccari. New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, 38:1495–1509, 1992.

[28] C. Valenzuela and L. Williams. Improving heuristic algorithms for the travelling salesman problem by using a genetic algorithm to perturb the cities. In *Seventh International Conference on Genetic Algorithms (ICGA97)*, pages 458–464, 1997.