

A DISTRIBUTION-AWARE PARTITIONING ALGORITHM WITH VARIABLE-ORDER PERTURBATION TO APPROXIMATE OPTIMAL TOURS

Vic Grout

University of Wales, NEWI Wrexham, UK

ABSTRACT

Various existing techniques are discussed and combined to form a compound algorithm for the heuristic solution of large Travelling Salesman Problems. Improvements are then offered for two of its key components; one for general use and one for the Euclidean special case.

1. INTRODUCTION

In its general form, the *Travelling Salesman Problem* (or *TSP*) is expressed as follows.

Given an $n \times n$ matrix $C = (c_{ij}: 1 \leq i, j \leq n)$, find a cyclic permutation \mathbf{p} of the integers $1, 2, \dots, n$ such that

$$C_{\langle \mathbf{p} \rangle} = \sum_{i=1}^n c_{i\mathbf{p}(i)} \quad \mathbf{Y}$$

is a minimum.

The integers $1, 2, \dots, n$ are known as *nodes* or *cities*. The matrix C is referred to as the *cost* or *distance matrix*. A cyclic permutation on the integers $1, 2, \dots, n$ is known as a *tour*. Any cyclic permutation satisfying \mathbf{Y} (there may be more than one) is a *minimal tour*. The full terminology and history of the TSP and many of its applications are described in [1] and are not discussed here. However, if we consider each node (or city) as a location on a map (only truly representative for the final case in this section) then Figure 1 shows a general tour and Figure 2 a minimal tour.

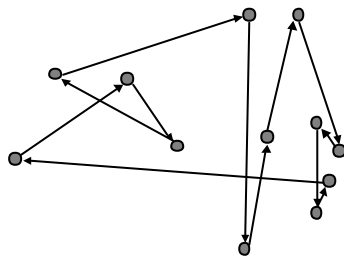


Figure 1. A tour of nodes/cities.

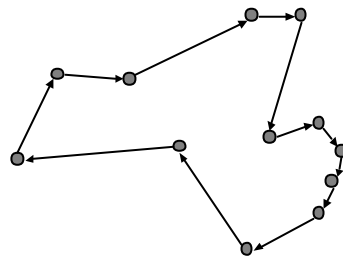


Figure 2. A minimal tour.

There are a number of variations of the general TSP, which are worthy of consideration. Each, in turn, places a constraint on the elements of the matrix C .

The *Symmetric Travelling Salesman Problem (STSP)* is a sub-class of the general TSP in which the matrix C is symmetric (about the leading diagonal). $c_{ij} = c_{ji}: 1 \leq i, j \leq n$. That is that the distance from one node to another is the same as the distance back again (Figure 3).

The *Triangulated Travelling Salesman Problem (TTSP)* is a (different) sub-class of the general TSP in which the triangle inequality holds for all elements of C . $c_{ik} \leq c_{ij} + c_{jk}: 1 \leq i, j, k \leq n$. This implies that the direct distance between two nodes can never be greater than the sum of the distances via any intermediate node (Figure 4).

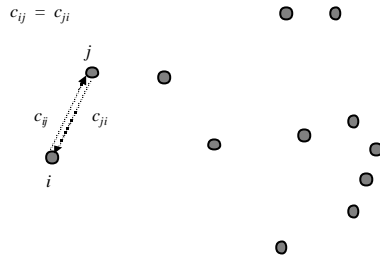


Figure 3. The symmetry rule.

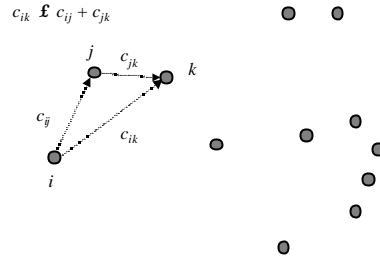


Figure 4. The triangle rule.

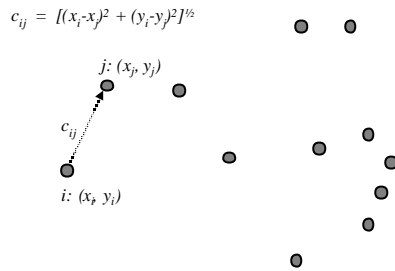


Figure 5. Euclidean distances.

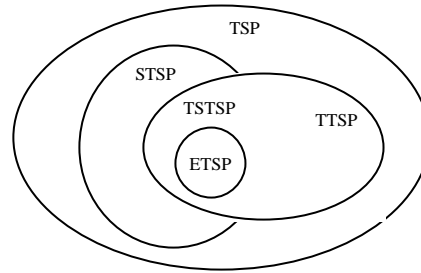


Figure 6. TSP classes.

Although the above restrictions are independent, both of course may apply simultaneously. This produces the *Triangulated Symmetric Travelling Salesman Problem (TSTSP)* with $c_{ij} = c_{ji}$ and $c_{ik} \leq c_{ij} + c_{jk} : 1 \leq i, j, k \leq n$.

The final restriction (of interest to us) is given by the *Euclidean Travelling Salesman Problem (ETSP)*. In this sub-class, nodes are defined by their position in the plane and the elements of C by the straight-line distance between them. If the co-ordinates of node $i : 1 \leq i \leq n$ are given by (x_i, y_i) then the elements of C are defined as $c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} : 1 \leq i, j \leq n$ (Figure 5). The ETSP is a special case of the TSTSP.

The complete relationship is shown in Figure 6. This categorization is necessary since, in the sections that follow, many techniques prove more appropriate for some classes and sub-classes than others. In discussing them, algorithms are described generally and intuitively. Details may be found in the source material where appropriate.

2. A COMPOUND SOLUTION

We mention complexity only briefly: a full discussion can be found in [1] and elsewhere. All classes of the TSP from the general case to the ETSP are NP-Complete [2]. In essence, for a TSP of n nodes, there are $n!$ possible tours. Thus any algorithm that attempts to solve the TSP by generating and comparing each tour in turn will have exponentially increasing complexity and will be an impractical approach for all but the smallest values of n . It is not known whether a significantly more efficient exact algorithm can exist.

In the absence of a better (exact) method of determining the optimum tour, our only recourse is to resort to heuristics - inexact, but hopefully reasonably accurate, techniques for approximating the minimum solution. Many approaches have been suggested over the years but, in general, these may be divided into three categories, summarised as: *simplify*, *construct* and *improve*. In fact, the following descriptions apply to most combinatorial optimisation problems.

- **Simplification/Reduction Methods**

If the essence of the difficulty lies with the scale of the problem and the number of potential solutions then an attempt to reduce its size or complexity may prove worthwhile. Small problems are often dealt with simply and large problems may yield to partitioning into a number of smaller problems or simplification to a single smaller problem. For example, if subsets of nodes can be considered independently or a single node made representative of a number of others then the scale of the problem may be reduced [3].

- **Approximation/Construction Methods**

Some optimisation problems yield entirely to a constructive method of solution. So-called *greedy algorithms*, for example will produce optimal solutions to certain problems such as that of the *Minimal Spanning Tree (MST)* [4] [5]. Although the TSP does not succumb in this fashion, a similar ‘shortest edge first’ approach can provide credible solutions [6]. This is particularly true if we remove the cyclic requirement that the last node be joined back to the first and the tour becomes merely a *walk* (see Figure 7, for example, in which the shortest edge has been chosen at each stage subject to the constraint that no subtours may be formed in advance of the complete walk/tour).

- **Perturbation/Improvement Methods**

Alternatively, it will always be possible to generate *some* solution to the problem - however bad. In most cases, for example (and including the TSP), simply joining nodes in labelled order, $1, 2, \dots, n$, will provide something with which to work. Now, looking to rearrange small numbers of edges, may offer a reduction in distance/cost. Figure 8, for example, shows an improvement over the tour from Figure 7 resulting from swapping two edges - still not optimal, but better. These perturbation techniques have been in use in a variety of fields for some time [7] [8].

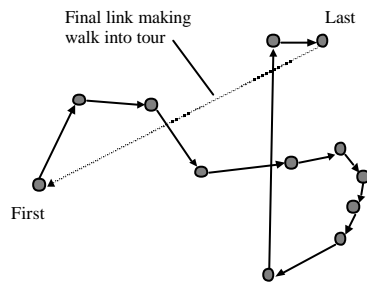


Figure 7. A ‘greedy’ walk/tour.

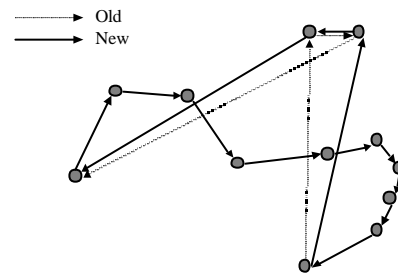


Figure 8. Improvement by perturbation.

It should be understood that none of these categories of heuristic solution is guaranteed to produce the optimum solution to the TSP - indeed no heuristic can. However, each individually offers an insight into the nature of the problem and, together, they suggest a more sophisticated line of attack. We can offer a general composite algorithm as follows.

- <1> Partition the n nodes into m subsets.
- <2> Construct a walk w_i or a tour t_i through the nodes of each subset $i: 1, 2, \dots, m$.
- <3> Combine the walks w_i or tours t_i into a tour T .
- <4> Apply perturbations to T to seek for improvements.

Steps <2> and <3> are the generic form of an algorithm in [9] and revisited in the next section. One, but not the only, option for step <3> would be:

- <3.1> Construct a tour T through the m subsets.
- <3.2> Patch each w_i or t_i into T .

Here, *patching* has the, hopefully intuitive, meaning suggested in Figures 9 and 10 - again illustrated in the ‘map’ form of the ETSP.

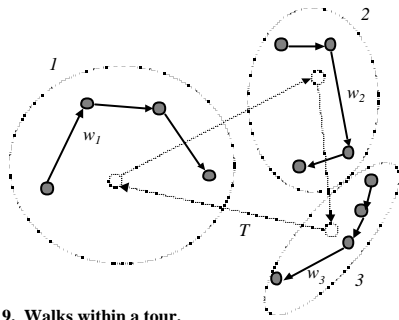


Figure 9. Walks within a tour.

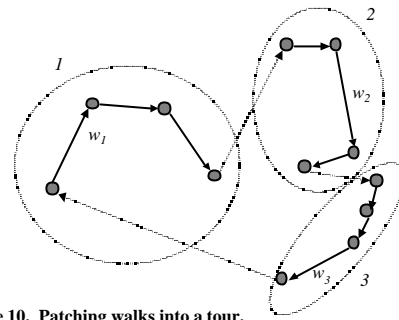


Figure 10. Patching walks into a tour.

There are, however, in the above algorithm, more ambiguities than the obvious in steps <2> and <3>. In fact, a number of questions can be raised at each point.

- 1 (a) How is m chosen?
- 1 (b) How are the m subsets chosen?
- 2 (c) Do we construct a walk or a tour?
- 2 (d) Do we use exact or heuristic algorithms? Which?
- 3 (e) How do we represent the subsets?
- 3 (f) How do we construct the tour?
- 3 (g) Are we patching walks or tours?
- 3 (h) How does the patching proceed.
- 4 (i) What form do the perturbations take?

It is the purpose of sections 3, 4 and 5 to discuss these questions and to evaluate different options. At this point it becomes necessary to consider the various classes of TSP since different classes are better served by some techniques than others. In some cases, it will even be necessary to modify the basic algorithm. Sections 6 and 7 then offer extensions.

3. PARTITIONING AND SUBTOURS

The ‘map’ representation in Figures 9 and 10 give a false impression for all classes of TSP other than the ETSP. The general TSP and most sub-classes have no such notion of locality. The elements of the matrix C are entirely unconstrained and will offer no natural groupings or subsets. However, there is an alternative to be found in considering another combinatorial problem, closely related to the TSP.

The *Assignment Problem* (or *AP*) is expressed as follows.

Given an $n \times n$ matrix $C = (c_{ij}; 1 \leq i, j \leq n)$, find any permutation r , of the integers $1, 2, \dots, n$ such that

$$C_{\langle r \rangle} = \sum_{i=1}^n c_{ir(i)} \quad \mathbf{F}$$

is a minimum.

This defines the general AP. The equivalent *Symmetric AP* (*SAP*), *Triangulated AP* (*TAP*), *Triangulated Symmetric AP* (*TSAP*) and *Euclidean AP* (*EAP*) may all be defined as with the corresponding TSPs from Section 1. The AP differs essentially from the TSP in that the required permutation need not be cyclic. The AP can be solved in polynomial time - an early method is given in [10]. A solution to the general AP will be a set of subtours of the n nodes (Figure 11, for example). These subtours may then be combined together in a step-by-step manner to produce a tour (Figure 12).

There are two distinct advantages to using the AP as a means of partitioning the nodes. Firstly, the node subsets already have a tour defined upon them (so that step <2> of the compound algorithm becomes unnecessary). Secondly, in the case of the general AP, the subsets, and hence the subtours, so

produced tend to be large [11], leaving a relatively small amount of work for step <3> and a reduction in the scope for error.

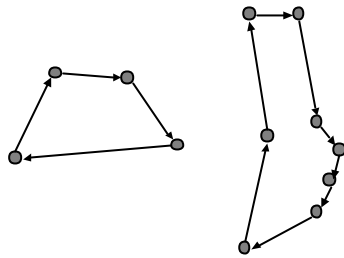


Figure 11. A solution to the AP.

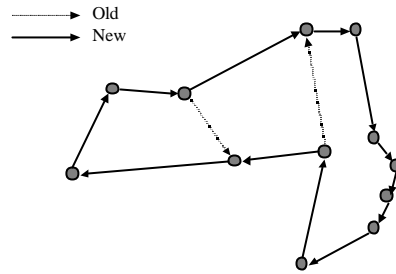


Figure 12. Combining subtours.

As the general problem is constrained, however, through the classes of SAP, TAP and TSAP, the subtours produced by solving the AP, on average, become smaller. The EAP produces the ultimate restriction with a typical solution consisting predominantly of subtours of length 2 and 3 [12]. Under these circumstances, step <3> is effectively working with a TSP problem of between one third and one half the size of the original - not the reduction in scale originally anticipated.

Fortunately, the special features of the ETSP that prohibit the effective use of the EAP as a means of generating initial partitions provide their own solution. The nodes of the ETSP *do* possess the concept of location identified at the start of this section. On this basis, nodes with similar location should be partitioned together. Figure 13 demonstrates the intuitive approach by which the plane is divided into regions and the nodes grouped accordingly. Possible subtours/subwalks are shown as an illustration. That this method of partition may not be the ideal is discussed in Section 7.

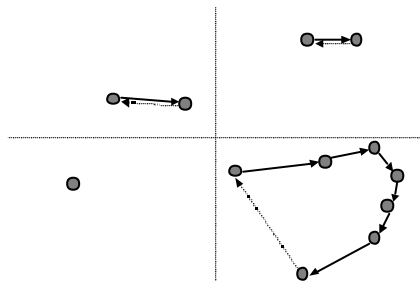


Figure 13. Partitioning nodes.

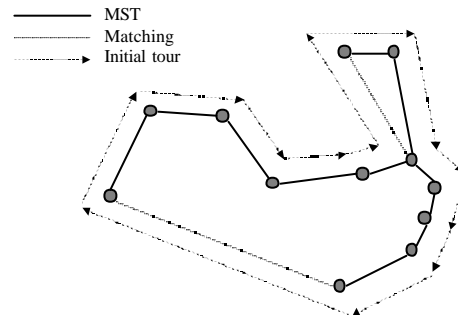


Figure 14. Christofides' Algorithm.

4. PATCHING WALKS AND SUBTOURS

There are various constructive techniques for approximating optimal TSP solutions, which may be used either in isolation or as part of a compound algorithm. Usually, in fact, these themselves are compound in their own right. Two examples are given here.

Christofides' Algorithm [6] can be expressed informally as:

- <C1> Find the MST on the n nodes.
- <C2> Find a minimum distance pairing (a *matching*) on the MST nodes of odd degree.
- <C3> Construct a closed walk combining the MST and the minimum distance pairing.
- <C4> Transform ('shortcut') the closed walk into a tour.

(There must be an even number of such nodes in $\langle C2 \rangle$ - see [13] for example). Christofides' Algorithm works for any TSP but can provide performance guarantees only for the TTSP (and ESTP). This, however, is a complete solution. If a set of m subtours or subwalks is already defined then Karp's Algorithm [9] offers an alternative:

- $\langle K1 \rangle$ Define an $m \times m$ distance matrix across the m regions as the shortest distance between any two points in different regions.
- $\langle K2 \rangle$ Using these distances, construct the MST over the m regions.
- $\langle K3 \rangle$ Construct a closed walk using every subtour/walk and passing from region to region using each edge of the MST twice.
- $\langle K4 \rangle$ Transform ('shortcut') the closed walk into a tour.

The essence of Christofides' and Karp's algorithms is given in Figures 14 and 15. In this example, both shortcut ($\langle C4 \rangle$ & $\langle K4 \rangle$) to the tour in Figure 16. Both, in practice, give good results. Neither, however, is optimal - nor can be expected to be. An alternative, already discussed, is to construct a (optimal or near optimal) tour across the m groups and patch each walk/tour into this.

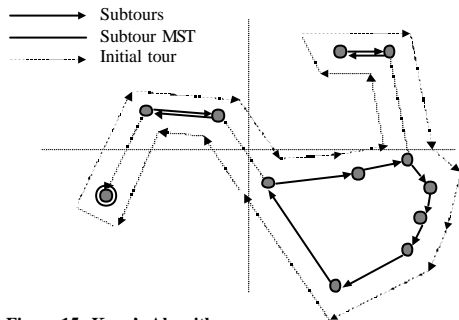


Figure 15. Karp's Algorithm.

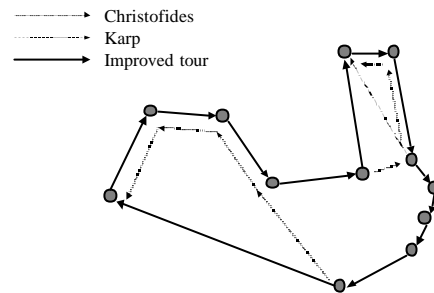


Figure 16. Applying shortcuts.

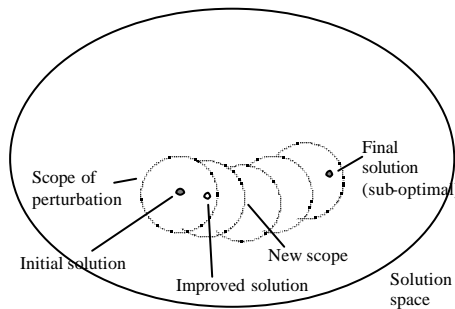


Figure 17. Perturbation of small order.

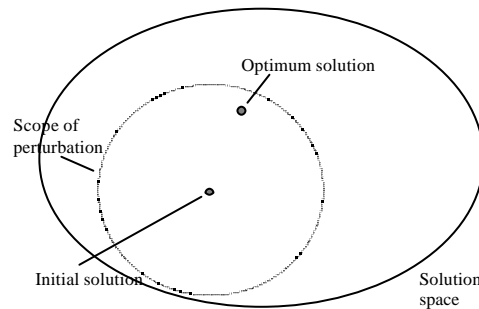


Figure 18. Perturbation of high order.

5. PERTURBATION AND DEGREES OF PERTURBATION

We have already mentioned perturbation briefly. A perturbation process seeks, through a sequence of experimental variations to find some local improvement to the current solution. An example is given in the improvement of Figure 8 over Figure 7. This is a typical application of simple perturbation. Two edges in the current solution have been removed and replaced by two others, giving an overall improvement. From all possible choices of removing two edges and adding two edges that still leave a valid tour, the combination leading to the largest improvement (if any) will be selected. This search may then be repeated, making perturbation a doubly iterative process. Seeking to 'perturbate' on two edges at a time is the lowest *order* of perturbation that can be allowed for the TSP. Replacing any single edge in the current solution by an edge currently not in the current solution can never leave a valid tour.

The problem with this form of perturbation of restricted order is that some improved solutions, and possibly the optimum itself, remain beyond the scope of the perturbation process. Figures 17, 18 and 19 model the situation. Figure 17 shows a limited order perturbation, which terminates with a sub-optimal solution. Figure 18 shows a perturbation process of larger order, which has the optimum solution within its initial scope. Figure 19 shows a perturbation process of moderate order, which reaches the optimum (or it could be a better sub-optimum) solution by having progressively better solutions within scope at each stage.

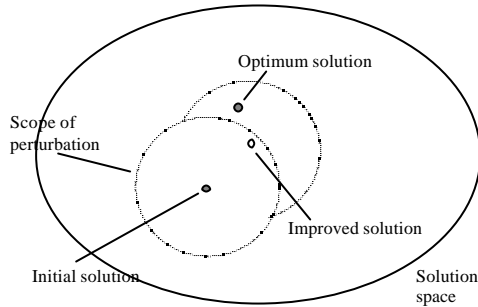


Figure 19. Perturbation of moderate order.

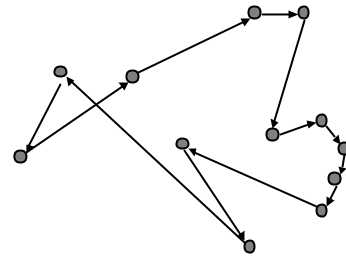


Figure 20. Three-edge perturbation (before).

In the case of the TSP, extending the scope/order of the perturbation process involves including larger numbers of edges (current and potential) in the experimental variation at each stage. Figures 20 and 21, for example, show a three-edge perturbation. The higher the order of perturbation - in this case, the larger the number of edges perturbed, the greater the chance of 'escaping' from a local minimum within the solution space and finding superior solutions. Of course, whilst extending the scope/order of the perturbation enhances the prospects of optimality, absolute or improved, it also increases the computational complexity of the underlying algorithm.

6. CONSTRUCTIVE PARTITIONING

The notion of location carried within the definition of the ETSP, and to an extent, though disguised, in the TTSP provides an intuitive method of partitioning - that is into areas of the plane. However, simply dividing the plane arbitrarily, into a grid say as in Figure 13, has two serious drawbacks:

- Arbitrary boundaries may cut through natural node clusters that should ideally be considered together. Nodes that would optimally be part of a single sub-tour will be split across two or more.
- Arbitrary regions will contain (possibly extremely) unequal numbers of nodes. Some sub-tours may prove hard to form on large numbers of nodes.

The arbitrary division method can be considered as a top-down approach. An alternative, bottom-up, form of partitioning is suggested here for the ETSP, in which the distribution of the nodes is taken into account in the forming of the sub-groups. It is applied through a process of *representative reduction*. We begin by assigning a weight, $w_i = 1$ to each node, i , which in turn is defined by its co-ordinates (x_i, y_i) . The following is then a single step/iteration.

- <R1> Find nodes i and j such that c_{ij} is a minimum.
 <R2> Replace i and j by k where $x_k = (w_i x_i + w_j x_j) / (w_i + w_j)$,
 $y_k = (w_i y_i + w_j y_j) / (w_i + w_j)$ and $w_k = w_i + w_j$.

Essentially, the closest two nodes are replaced by a single node that represents them in terms of both position and weighting. Replacement nodes are weighted toward the original node with the larger weight. If this process is repeated then the few nodes that remain will mark the centres of natural node clusters, which may then be used to form appropriate sub-tours or sub-walks. This reduction process continues until either:

- <T1> The number of remaining nodes is sufficiently small to permit an exhaustive search optimisation to find the minimal tour across the clusters

or

<T2> No further reductions can take effect without causing node clusters to become too large to permit exhaustive search optimisation to find the minimal tours within the clusters.

If the process terminates on condition <T2> there is in effect a gap in which the individual cluster sizes are threatening to become too large while the overall number of clusters is yet too large. There are different solutions to this problem:

<S1> Allow reduction to continue on non-minimal node pairs in clusters in which the weighting is sufficiently small to permit full optimisation

or

<S2> Perform heuristic optimisation within clusters

or

<S3> Perform heuristic optimisation across clusters.

The ideal, which occurs on terminating condition <T1>, will be to patch optimal cluster tours into an optimal larger tour. In either case, perturbation methods may then be used to search for improvements to this initial solution.

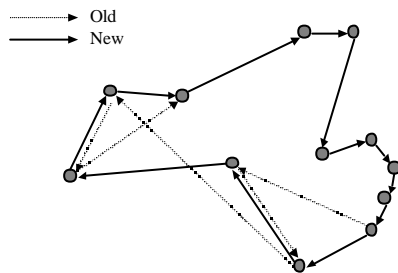


Figure 21. Three-edge perturbation (after).

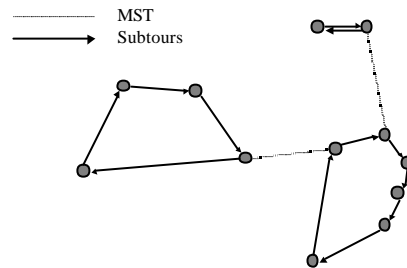


Figure 22. Subtours from reduction.

An example of sub-tours formed through reduction is given in Figure 22 together with the <K1><K2> MST. Transforming this partial solution into a closed walk, <K3>, then shortcutting, <K4>, gives the solution of Figure 12. A two-link perturbation process will be unable to improve upon this. However, a three-link perturbation will reach the optimum solution of Figure 2 (see Figure 23).

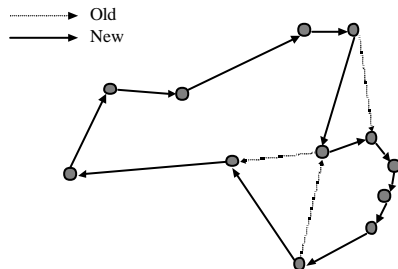


Figure 23. Shortcutting then perturbation.

| Nodes | Algorithm A | Algorithm B | | |
|-------|-------------|-------------------------|-----|------|
| | | Degree of perturbation: | | |
| | | 0 | 1 | 2 |
| 50 | 0 | 0.9 | 3.0 | 7.1 |
| 100 | 0 | 2.1 | 5.8 | 10.3 |
| 200 | 0 | 4.3 | 7.8 | 14.6 |

% improvement of Algorithm B over Algorithm A

Figure 24. Comparison of results.

7. RESULTS

A number of classes of the TSP together with a variety of algorithms and many variations have been discussed in this paper. Under these circumstances, it is essential that, in testing different approaches, we make appropriate comparisons. We also need to summarise in the interests of brevity. With this in

mind, from a range of test data, we offer the following compound algorithms for the ETSP as examples:

- **A: 'Standard' algorithm**
An arbitrary grid division of the plane followed by Karp's algorithm, $\langle K1 \rangle \langle K2 \rangle \langle K3 \rangle \langle K4 \rangle$.
- **B: 'Enhanced' algorithm**
A partition by representative reduction followed by Karp's algorithm, $\langle K1 \rangle \langle K2 \rangle \langle K3 \rangle \langle K4 \rangle$, followed by zero, one or two degrees of perturbation (i.e. no perturbation, two-link perturbation or three-link perturbation).

Sets of 50, 100 and 200 nodes were randomly generated in the plane and algorithms **A** and **B** applied to each. A comparison of results is given in Figure 24.

8. CONCLUSIONS

No algorithm, it is widely believed, can overcome the essential NP-Completeness of the TSP or its constrained derivatives. However, well-considered and appropriate methods may produce distinctly better results than poorly-applied ones. Two useful techniques are offered in this paper as part of a general, compound approach. Firstly, the order of perturbation may be raised in order to expand the search domain within the solution space, giving a greater chance of locating the optimum or at least a better solution. Although increasing the overall complexity of the process, it remains polynomial and thus preferable to the exponential nature of exhaustive search. (There is, in a sense, a 'random' aspect to this form of optimization – a comparable approach is to be found in [14]). Secondly, for the ETSP, using a representative reduction process to generate node groups and initial subtours provides a more adaptive approach to partitioning, which takes into account the original distribution of the nodes to be toured. The complexity of the reduction process, being a linear sequence of matrix searches, is bounded by $O(n^3)$, the same as for the AP, a less appropriate alternative.

REFERENCES

- [1] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. & Shmoys, D.B. *The Traveling Salesman Problem*. Wiley, 1985.
- [2] Papadimitriou, C.H., *The Euclidean Traveling Salesman Problem is NP-Complete*, Theoretical Computer Science, Vol. 4, pp237-244, 1977.
- [3] Grout, Vic, *Seeding a Broadband Distribution Network*, Second IMA Conference on Mathematics in Communications, Lancaster University, UK, 16th-18th December 2002.
- [4] Kruskal, J.B., *On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem*, Proceedings of the American Mathematical Society, Vol. 7, pp48-50, 1956.
- [5] Prim, R.C., *Shortest Connection Networks and Some Generalizations*, Bell Systems Technical Journal, Vol. 36, pp1389-1401, 1957.
- [6] Christofides, N. *Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem*, Report 388, Graduate School of Industrial Administration, CMU, 1976.
- [7] Rothfarb, B., Frank, H., Rosenbaum, D.M., Steiglitz, K. & Kleitman, D.J., *Optimal Design of Offshore Natural Gas Pipeline Systems*, Operations Research, Vol. 18, pp992-1020, 1970.
- [8] Kersenbaum, A. & Chou, W., *A Unified Algorithm for Designing Multidrop Teleprocessing Networks*, IEEE Transactions on Communications, Vol. COM-22, No. 11, pp1762-1772, 1974.
- [9] Karp, R.M., *The Probabilistic Analysis of Some Combinatorial Search Algorithms*, Algorithms and Complexity: New Directions and Recent Results, J.F. Traub (ed.), Academic Press, New York, pp1-19, 1976.
- [10] Kuhn, H.W., *The Hungarian Method for the Assignment Problem*, Naval Research Logistics Quarterly, Vol. 2, pp83-97, 1955.
- [11] Karp, R.M. & Steele, J.M., *Probabilistic Analysis of Heuristics*, The Traveling Salesman Problem [1], E.L. Lawler et al. (eds.), Wiley, 1985.
- [12] Grout, V.M., Sanders, P.W. & Stockel, C.T., *Reduction Techniques Providing Initial Groupings for Euclidean Traveling Salesman Patching Algorithms*, Applied Mathematical Modelling, Vol. 13, pp110-114, 1989.
- [13] Wilson, R.J., *Introduction to Graph Theory*, Longman, 1996.
- [14] Chandran, B., Golden, B. & Wasil, E., *A Computational Study of Three Demon Algorithm Variants for Solving the Traveling Salesman Problem*, in 'Computational Modeling and Problem Solving in the Networked World', H.K. Bhargava & N. Ye (eds.), Kluwer Academic, 2003, pp155-175.