Modelling and Solving Central Cycle Problems with Integer Programming

by L R Foulds, J M Wilson and T Yamaguchi

Business School

Research Series
Paper 2000:6
ISBN 1 85901 167 5

Loughborough University

Modelling and Solving Central Cycle Problems with Integer Programming

by

L R Foulds, J M Wilson and T Yamaguchi

Business School Research Series
Paper 2000:6
ISBN 1 85901 167 5

October 2000

Modelling and Solving Central Cycle Problems with Integer Programming

L R Foulds
Waikato Management School, University of Waikato, Private Bag 3105,
Hamilton, New Zealand
J M Wilson*
Business School, Loughborough University, Loughborough LE11 3NL,
United Kingdom
T Yamaguchi
Mathematical Science Division, Muroran Institute of Technology, Muroran, Japan

Abstract
We consider the problem of identifying a central subgraph of a given simple connected graph. The case where the subgraph comprises a discrete set of vertices is well known. However, the concept of eccentricity can be extended to connected subgraphs such as: paths, trees and cycles. Methods have been reported which deal with the requirement that the subgraph is a path or a constrained tree. We extend this work to the case where the subgraph is required to be a cycle. We report on computational experience with integer programming models of the problems of identifying cycle centres, cycle medians and cycle centroids, and also on a heuristic based on the first model. The problems have applications in facilities location, particularly the location of emergency facilities, and service facilities.

Keywords: cycle centre, cycle centroid, cycle median, graph, heuristic, integer programming, location.

* Corresponding author

Introduction

The problem of finding a circular route through a series of locations such that the distance of any point not on the route to its nearest point on the route is kept to a minimum occurs in a number of practical instances. An early operational research approach to a related problem appears in Hakimi (1964). Such problem instances we have just discussed are well known in the theory of facilities location, which is concerned with the location of one or more facilities that are to be sited so as to service a number of clients at given locations. We confine our attention in this paper to facilities on graphs or networks, rather than in the plane.

Here we extend the concept of what is to be located from a discrete set of facilities to a circular structure. In network terms the classic problems involve identifying a discrete set of network nodes, representing optimal facility locations. Our extension involves identifying an optimal cycle in the network.

The first problem that we wish to discuss is termed the *cycle centre problem*. It can be formulated in graph theoretic terms as follows.

The Cycle Centre problem

Given a connected simple graph $G = (V,E)$ with vertex set V and edge set E, identify a cycle C in G which minimises the maximum of the shortest distances between any vertex not a member of C to some vertex of C, such that C is of minimal length among all such cycles.

For the relevant graph theoretic notation and terminology see Foulds (1998).

The distance between any two vertices in G is defined as the least number of edges that need to be traversed, in any path in G between the two vertices. The resulting cycle C is termed a *cycle centre* of G. The problem is clearly a bi-criterion problem as the distance to non-cycle vertices from the cycle vertices and the actual length of the cycle are both criteria. The first of these dominates the second. A similar problem of identifying paths rather than cycles has been studied by Slater (1981,1982).

The most common applications of the cycle centre problem are with the location of emergency facilities, such as: health clinics, police stations, or fire stations. The objective is to minimise the greatest distance between any of the facilities and any of the clients. The classic problems from location theory with this objective are concerned with the identification of the optimal location of a single facility or a discrete set of facilities.

In addition to the work to be presented in this paper, related work on this problem was reported by Foulds and Yamaguchi (1998). In their paper a Tabu Search heuristic is described for the problem. The method starts by constructing a spanning tree to which a chord is added to create a cycle. The method iterates between different spanning trees and cycles, and records and updates the least cost cycle found so far. Schobel et al. (1999) have provided characterisations of the central cycle, for the special case where G is a grid graph.

A similar problem has been considered by Labbe' and Laporte (1986) who describe an integer programming model for the location of central post boxes in zones of a city. They divide the possible locations into n zones, allocate each of n boxes to a different zone, and then tour all zones. More recently, Current and Schilling (1994) have developed an integer programming model and heuristics for the median tour and maximal covering tour problems. Their problems have similar constraints to ours, but a different objective, namely minimising the total demand weighted travel distance between nodes on the tour that must be traversed in succession. Other related problems are described in a review by Mesa and Boffey (1996), who note the lack of consideration given to problems involving the location of cyclic structures. A further review of related problems can be found in Labbe' et al. (1998).

The Cycle Median problem

The second problem that we wish to discuss is termed the *cycle median problem*. It can be formulated in graph theoretic terms in a similar fashion to the previous problem, by replacing the word "maximum" by the word "sum". That is,
given a connected simple graph $G = (V, E)$, with vertex set V and edge set E, identify a cycle C of G which minimises the sum of the shortest distances between all vertices

not members of C to any vertex of C, such that C is of minimum length among all such cycles. The resulting cycle is termed a *cycle median* of G.

Applications of this objective include: the creation of express ring roads in urban environments, urban bus and rail routes, circular communication systems in organisational structures, and collection and distribution systems for public utilities such as special inorganic garbage collection, or parcel delivery.

The Cycle Centroid problem

The third problem we wish to describe is termed the *cycle centroid problem*. It can be formulated in graph theoretic terms as follows:
given a connected simple graph $G = (V, E)$, with vertex set V and edge set E, identify a cycle C of G which maximises the sum taken over all vertices v not in C of the differences between the number of vertices in $V \setminus C$ which are closer to at least one vertex in C than to vertex v and the number of vertices in $V \setminus C$ which are closer to vertex v than any vertex in C, such that C is of minimum length among all such cycles. The resulting cycle is termed a *cycle centroid* of G.

Applications of this objective include the location of special facilities, for example freeways, distribution networks, or communication networks with the aim of minimizing the distance from each client to each of the facilities.

The purpose of this paper is to discuss integer programming models of the three problems, to solve them exactly and to develop heuristic solution methods based on the models. The paper is laid out as follows. In the next three sections the problems are formulated as integer programming models. In the following section there is a discussion of the solution approaches based on the models. The final section presents some conclusions.

2. An integer programming formulation of the cycle centre problem

First we introduce some necessary notation, in the form of data that must be identified for any given graph G representing a numerical instance of the problem.

Let $|V| = n$.

Let c_{ij} = shortest distance from vertex i to vertex j, $i < j$, $i, j \in V$.

Let e_{ij} = 1 if there is an edge from vertex i to vertex j (denoted by (i,j)), $i < j$, $i, j \in V$

= 0 otherwise.

Note that $\{e_{ij}\}_{nxn}$ represents the adjacency matrix of G.

Let k_l be a lower limit on the number of vertices in the cycle centre C.

Some variables are now introduced.

Let y_i = 1 if vertex i is a member of C, $i \in V$,

= 0 otherwise.

Let d_i = shortest distance from vertex i, not a member of C, to any vertex in C .

Let p_{ij} = 1 if vertex j in C is the nearest vertex to vertex i not in C, $i \neq j$; $i, j \in V$,

= 0 otherwise.

Let x_{ij} = 1 if C includes edge (i,j), $i < j$; $i, j \in V$,

= 0 otherwise.

Note that $e_{ij} = 0 \Rightarrow x_{ij} = 0$; $i, j \in V$.

Let z = maximum of all values d_i , $i \in V$.

Let M be a relatively large constant. Typically $M = O(10n)$.

Note: It might seem arduous to obtain the c_{ij} data. However, if, as is likely in practical applications, it is required that the maximum of the shortest distances of all vertices not in C to some vertex in C is smaller than some constant k, say, then it will be sufficient to find exact vertex distances only if these are less than or equal to k. Distances known to be larger then k can be set to M. Thus when computing shortest distances to any vertex, it will be sufficient to consider only vertices within a distance

of k from that vertex. The cycle centre problem can now be stated formally as the following integer programming problem:

Objective

$$\text{Minimise } Z = Mz + \sum_{i=1}^{n} y_i$$

Constraints

$$\sum_{j=1}^{n} p_{ij} + y_i = 1 \ , \qquad i = 1,2, \ldots ,n. \tag{2.1}$$

$$\sum_{j=1}^{n} p_{ij} \le (n - k_l) y_j \ , \quad j = 1,2, \ldots ,n. \tag{2.2}$$

$$\sum_{j=1}^{n} c_{ij} p_{ij} = d_i \ , \qquad i = 1,2, \ldots ,n. \tag{2.3}$$

$$z \ge d_i \ , \qquad i = 1,2, \ldots ,n. \tag{2.4}$$

$$\sum_{j>i}^{n} x_{ij} + \sum_{j<i}^{n} x_{ji} = 2y_i \ , \ i = 1, \ldots ,n. \tag{2.5}$$

$$x_{ij} \le y_i \qquad i < j, \ i = 1,2, \ldots ,n, j = 1,2, \ldots ,n. \tag{2.6}$$

$$x_{ij} \le y_j \qquad i < j, \ i = 1,2, \ldots ,n, j = 1,2, \ldots ,n. \tag{2.7}$$

$$x_{ij} \varepsilon S \qquad i < j, \quad i = 1,2, \ldots ,n, j = 1,2, \ldots ,n. \tag{2.8}$$

$$x_{ij} \le e_{ij} \qquad i < j, \ i = 1,2, \ldots ,n, j = 1,2, \ldots ,n. \tag{2.9}$$

$$\begin{cases} y_i = 0,1 & i = 1,2,...,n \\ x_{ij} = 0,1 & i = 1,2,...,n; j = 2,...,n; i < j \\ p_{ij} = 0,1 & i = 1,2,...,n; j = 1,...,n; i \ne j \\ d_i \ge 0 & i = 1,2,...,n. \end{cases} \tag{2.10}$$

Constraint (2.1) ensures that a 'nearest' vertex on the path is selected only if a vertex is not on the cycle. Constraint (2.2) ensures that the vertex deemed 'nearest' to an off-cycle vertex is a member of C. Constraint (2.3) determines the distance between an off-cycle vertex and its 'nearest' neighbour in C. Constraint (2.4) ensures that z is the largest distance of any off-cycle vertex to its 'nearest' vertex in C. Constraints (2.5) ensure that all vertices of C are of degree 2. Constraints (2.6) and (2.7) are cuts

which are useful to add to the formulation since they strengthen (2.5). Constraints (2.8) are sub-tour elimination constraints (to be discussed in below). Constraint (2.9) ensures that *C* comprises only edges that are actually present in *G* and finally constraints (2.10) are the usual zero-one and non-negativity conditions.

2.1. Sub-tour elimination

The constraints used in (2.6) deserve further consideration. Because it will not be known in advance how many vertices will form the cycle, it is not possible to use either of the two common sub-tour elimination approaches from the Travelling Salesman Problem (TSP) (see for instance Lawler *et al*., 1985 and Orman and Williams, 1999). These approaches would introduce constraints of form,

$$u_i - u_j \le kr_{ij} \quad ,$$

where u_i and u_j are the sequence numbers of the vertices visited in the cycle, or constraints that partition the set of vertices into subsets to avoid sub-tours. In the former approach $k(=\sum_{i=1}^{n} y_i)$, is itself a variable quantity.

However, the latter approach may be adapted for the cycle centre problem and constraints of the following form may be adjoined to the formulation or used as cuts. Let J_1 and J_2 be the vertex sets of two disjoint connected subgraphs of $G = (V,E)$ such that $J_1 \cup J_2 = \{1,2, \ldots ,n\}$. Suppose J_1 and J_2 both contain at least one edge which has both its end points in that set. Then it follows that if

$$\text{either } \sum_{i,j \varepsilon J_1} x_{ij} = \sum_{i \varepsilon J_1} y_i \text{ or } \sum_{i,j \varepsilon J_2} x_{ij} = \sum_{i \varepsilon J_2} y_i \text{ , a sub-tour is present.}$$

Constraints to eliminate such sub-tours may be constructed as follows.
We introduce binary variables δ_{J_1}, δ_{J_2}, $\delta_{J_1 J_2}$ such that:

$$\delta_{J_1 J_2} \le \sum_{i \varepsilon J_1} y_i \le M\delta_{J_1} \quad , \quad \delta_{J_1 J_2} \le \sum_{i \varepsilon J_2} y_i \le M\delta_{J_2} \text{ and } \delta_{J_1} + \delta_{J_2} \le 1 + \delta_{J_1 J_2} .$$

Then to avoid sub-tours we require

$$\delta_{J_1 J_2} + \sum_{i,j \varepsilon J_1} x_{ij} = \sum_{i \varepsilon J_1} y_i$$

and $$\delta_{J_1 J_2} + \sum_{i,j \varepsilon J_2} x_{ij} = \sum_{i \varepsilon J_2} y_i$$

to hold. Note that these constraints will have considerable power in the linear relaxation of the problem because typically the number of non-zero vertices in each partition of the vertex set will be small. Thus the binary variable may well be tightly bounded and hence will cause restriction within the two equations when two discrete, but fractional, sub-tours are present.

Because the cycle will typically pass through only a small proportion of the vertices in the graph, it was found that generalisations for the cycle centre problem of the type of constraints described in Bauer (1997) (e.g. comb inequality) have little power in that problem. Accordingly they have not been used in the formulation or in any branch and cut format.

It is straightforward to show that $\max_{i,j} c_{ij}$ provides an upper bound on Z, the objective function value.

We now illustrate what we have discussed by using the above model to solve the following numerical example.

Example

Let $V = \{1,2, \ldots , 12\}$.

Values of e_{ij}

i/i	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	0	1	0	0	0	0	0	0
2		0	1	0	0	1	0	0	0	0	0	0
3			0	1	0	0	1	0	0	0	0	0
4				0	1	0	1	0	0	0	0	1
5					0	0	0	0	0	0	0	0
6						0	1	1	0	0	0	0
7							0	0	1	0	0	0
8								0	1	1	0	0
9									0	0	1	1
10										0	1	0
11											0	1
12												0

Table 1. The Upper Triangle of an Adjacency Matrix

Values of c_{ij}

i/i	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	2	3	4	1	2	2	3	2	4	4
2		0	1	2	3	1	2	2	3	3	4	4
3			0	1	2	2	1	2	2	4	3	3
4				0	1	2	1	3	2	4	3	1
5					0	3	2	4	3	5	4	2
6						0	1	1	2	2	3	3
7							0	2	1	3	2	2
8								0	1	1	2	2
9									0	2	1	1
10										0	1	2
11											0	1
12												0

Table 2. Distances Between Vertices

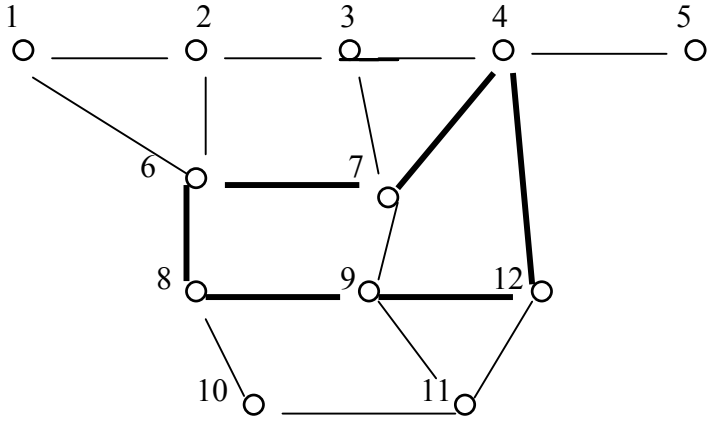

Figure 1 The cycle centre for the example problem

The cycle centre is <6,7,4,12,9,8,6> with $z = 1$, and is shown in Figure 1. If $M = 100$, then $Z = 100(1) + 6 = 106$.

Note: If the solution to any problem is such that $z = 0$ then a Hamiltonian cycle has been found. In general, it will not be expected that such a cycle exists in G. We now go on to modify the previous model so as to create a model for the cycle median problem.

3. A formulation of the cycle median problem

If constraint (2.4) is replaced by the constraint

$$z = \sum_{i=1}^{n} d_i \qquad (3.1)$$

then the problem becomes that of finding the *cycle median* of the graph *G*. Recall that the cycle median is the cycle *C* of minimum length that minimises the total of shortest distances of all vertices which are not members of *C* to some vertex in *C* .

The cycle median of the graph in Figure 1 is <1,2,3,7,4,12,9,11,10,8,6,1> and $z = 1$. This is shown in Figure 2. If $M = 100$, then $Z = 100(1) + 11 = 111$.

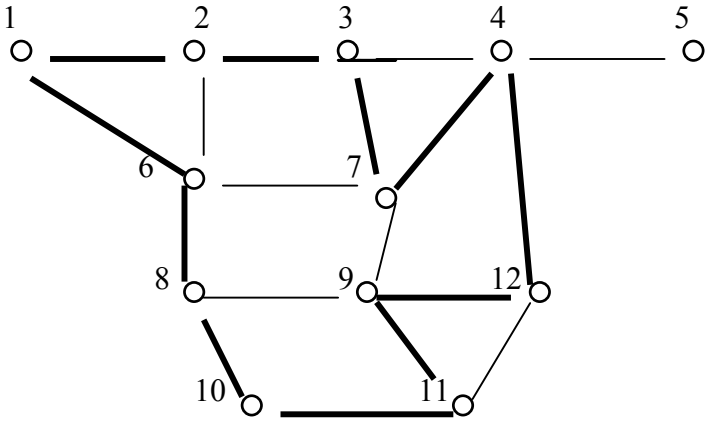

Figure 2 The cycle median for the example problem

4. A formulation of the cycle centroid problem

The variable *z* of section 2 is not used in the formulation of the cycle centroid problem, but all other variables defined in that section are used. We also introduce additional variables as follows.

Let v_{ij} = 1 if vertex *i* is not further, as measured by d_i, from the cycle than from vertex *j*, as measured by c_{ij} ($i \neq j$; $i, j \varepsilon V$)

= 0 otherwise.

Let u_{ij} = 1 if vertex *i* is not further, as measured by c_{ij}, from vertex *j* than from the cycle, as measured by d_i ($i \neq j$; $i, j \varepsilon V$)

= 0 otherwise.

Three further variables

f_j, g_j and h_j ($j \varepsilon V$)

are introduced to be defined by constraints (4.10) – (4.12).

The cycle centroid problem can now be stated formally as the following integer programming problem.

Maximise $Z = M\sum_{j=1}^{n} f_j - \sum_{i=1}^{n} y_i$

Subject to constraints (2.1) – (2.3), (2.5) – (2.10) as before, and

$$v_{ij} + y_j \leq 1 \qquad i = 1,2, \ldots ,n ; j = 1,2, \ldots ,n ; \ i \neq j . \tag{4.1}$$

$$v_{ij} + y_i \leq 1 \qquad i = 1,2, \ldots ,n ; j = 1,2, \ldots ,n ; \ i \neq j . \tag{4.2}$$

$$u_{ij} + y_j \leq 1 \qquad i = 1,2, \ldots ,n ; j = 1,2, \ldots ,n ; \ i \neq j . \tag{4.3}$$

$$u_{ij} + y_i \leq 1 \qquad i = 1,2, \ldots ,n ; j = 1,2, \ldots ,n ; \ i \neq j . \tag{4.4}$$

$$Mv_{ij} + My_i + My_j + d_i \geq c_{ij} + 1$$

$$i = 1,2, \ldots ,n ; j = 1,2, \ldots ,n ; \qquad i \neq j . \tag{4.5}$$

$$Mv_{ij} - My_i - My_j + d_i \leq c_{ij} + M$$

$$i = 1,2, \ldots ,n ; j = 1,2, \ldots ,n ; i \neq j . \tag{4.6}$$

$$Mu_{ij} + My_i + My_j - d_i \geq 1 - c_{ij}$$

$$i = 1,2, \ldots ,n ; j = 1,2, \ldots ,n ; \qquad i \neq j . \tag{4.7}$$

$$Mu_{ij} - My_i - My_j - d_i \leq M - c_{ij}$$

$$i = 1,2, \ldots ,n ; j = 1,2, \ldots ,n ; i \neq j . \tag{4.8}$$

$$d_j + My_i \leq c_{ij} + M \qquad i = 1,2, \ldots ,n ; j = 1,2, \ldots ,n ; \ i \neq j . \tag{4.9}$$

$$g_j = \sum_{i=1}^{n} v_{ij} \qquad j = 1,2, \ldots ,n \tag{4.10}$$

$$h_j = \sum_{i=1}^{n} u_{ij} \qquad j = 1,2, \ldots ,n \tag{4.11}$$

$$f_j = g_j - h_j \quad j = 1,2, \ldots ,n \tag{4.12}$$

$$\begin{cases} v_{ij} = 0,1 \qquad i = 1,2,...,n; \quad j = 1,2,...,n \\ u_{ij} = 0,1 \qquad i = 1,2,...,n; \quad j = 1,2,...,n \\ f_j \quad \text{is a free variable} \quad j = 1,2,...,n \\ g_j, h_j \geq 0 \qquad j = 1,2,...,n. \end{cases} \tag{4.13}$$

Constraints (4.1) and (4.2) ensure that if $y_i = 1$ or $y_j = 1$ then $v_{ij} = 0$ ($i = 1,2, \ldots ,n$; $j = 1,2, \ldots ,n$; $i \neq j$). Constraints (4.3) and (4.4) perform the role analogous to (4.1) and (4.2) for u_{ij}.

Constraint (4.5) ensures that if $y_i = 0$, $y_j = 0$ and $d_i \leq c_{ij}$ then $v_{ij} = 1$ ($i = 1,2, \ldots ,n$; $j = 1,2, \ldots ,n$; $i \neq j$).

Constraint (4.6) ensures that that if $y_i = 0$, $y_j = 0$ and $d_i \geq c_{ij} + 1$ then $v_{ij} = 0$ ($i = 1,2, \ldots ,n$; $j = 1,2, \ldots ,n$; $i \neq j$). Constraints (4.7) and (4.8) perform the role analogous to (4.5) and (4.6) for u_{ij}.

Constraint (4.9) ensures that that if $y_i = 1$ then $d_j \leq c_{ij}$ ($i = 1,2, \ldots ,n$; $j = 1,2, \ldots ,n$; $i \neq j$).

Constraint (4.10) defines g_j as the sum of vertices nearer the cycle than to vertex j.

Constraint (4.11) defines h_j as the sum of vertices nearer vertex j than to the cycle.

Constraint (4.12) defines f_j as the difference between g_j and h_j.

Finally, (4.13) gives the usual zero-one and non-negativity conditions defines f_j as a free (unconstrained) variable.

Using the formulation given above and the data of Section 2, the cycle centroid is given by the cycle <6, 7, 8, 9, 6>, as shown in Figure 3.

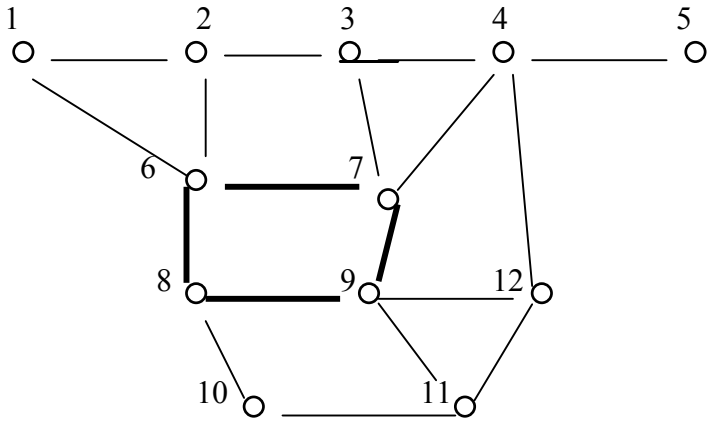

Figure 3 The cycle centroid for the example problem

5. Results using the IP model

To explore the models further, a set of test problems was generated as follows. A set of *n* vertices is chosen in the form of a rectangular grid with vertices evenly spaced. Vertices are numbered consecutively in vertical order, starting from the left. Each vertex is considered in turn, starting with the lowest numbered, and then three edges are introduced, each with probability equal to 0.33, to the vertex to its immediate right, to the vertex immediately below it, and to the vertex diagonally below it to the right. An appropriate adjustment is made at the perimeter of the grid. Grids that are not connected are rejected. The formulation given by (2.1)-2.10) for the cycle centre problem was then used and solved with the mathematical programming software XPRESS-MP (Dash Associates, Blisworth, Northamptonshire, England). Because of the structure of the grids, the constraints described in section 2.1 are straightforward to construct and are limited in number relative to the number of vertices.

Vertices		Nodes			CPU secs	
	min	mean	max	min	mean	max
16	10	53	79	0	0	0*
36	53	236	641	4	17	29
64	378	2316	32417	141	370	1674
100	850	4304	23794	1310	4595	7477

Table 3 Results for the cycle centre problem (averaged over 20 datasets)
* = too small to be recordable

Table 3 shows the number of branch and bound nodes required to solve the problems, and CPU time taken on an HP9000/800. As can be seen, the larger problems are slower to solve using the formulation, but the largest problems are still being solved to optimality comfortably. The authors plan to develop branch and cut methods to solve the problems with sub-tours being eliminated as and when required, rather than by the use of constraints (2.8).

Table 4 presents results for the cycle median problem. As the formulation of this problem is very similar to the cycle centre problem, it was not tested extensively. As can be seen from Table 4 the results are very similar to Table 3.

Vertices		Nodes			CPU secs	
	Min	mean	max	min	mean	max
36	101	226	324	7	16	20

Table 4 Results for the cycle median problem (averaged over 20 datasets)

Table 5 presents results for the cycle centroid problem. This is a much more difficult problem to solve than the cycle centre problem. The formulation given by (2.1) - (2.3), (2.5) - (2.10), (4.1) - (4.13) was used and solved with XPRESS-MP analogously with the results in Table 3. As can be seen from Table 5, even for fairly small

numbers of vertices the problems are slow to solve and both computational times and numbers of nodes generated grow rapidly as problem size increases.

Vertices		Nodes			CPU secs	
	Min	mean	max	min	mean	max
16	661	7452	43778	8	105	568
20	2283	20584	95400	46	637	2186
25	11515	45885	269900	568	2786	10099

Table 5 Results for the cycle centroid problem (averaged over 20 datasets)

7. Conclusions

We have introduced three bi-criterion undirected routing and location problems and developed integer programming models, and exact solution procedures for solving sizeable instances of them. This approach meets with considerable success for the cycle centre and cycle median problems, but with less success for the cycle centroid problem. Given the intractability of the problems it is unlikely that any exact method could be developed to solve very large instances of them in reasonable computational time. Thus the search for an efficient heuristic, particularly one incorporating bounds, is an important area for further research.

References

P Bauer (1997) The circuit polytope: facets. *Mathematics of Operations Research* **22**, 110-145.

J R Current and D A Schilling (1994) The median tour and maximal covering tour problems: Formulations and heuristics. *European Journal of Operational Research* **73**, 114-126.

L R Foulds (1998) *Graph Theory Applications*, 3rd printing, 404 pages, Springer-Verlag, New York.

L R Foulds and T Yamaguchi (1998) Cycle centres, medians and centroids in graphs. Memoirs Research Paper Series, Muroran Institute of Technology, Japan.

S L Hakimi (1964) Optimum location of switching centers and the absolute centers and medians of a graph. *Operations Research* **12**, 450-459.

M Labbe', G Laporte and I Rodri'guez – Martin (1998) Path, tree and cycle location. In T G Crainic and G Laporte (eds) *Fleet Management and Logistics*, Kluwer Academic Press, London.

M Labbe' and G Laporte (1986) Maximizing user convenience and postal service efficiency in post box location. *Belgian Journal of Operations Research, Statistics and Computer Science* **26**, 21-35.

E L Lawler, J K Lenstra, A H G Rinnnooy Kan, D B Shmoys (Eds) (1985) *The Traveling Salesman Problem*, John Wiley and Sons, Chichester.

J A Mesa and T B Boffey (1996) A review of extensive facility location in networks. *European Journal of Operational Research* **95**, 592 – 603.

A Orman and H P Williams (1999) A survey of formulations of the travelling salesman problem, Research Paper 101, Faculty of Mathematical Studies, University of Southampton, Southampton, UK.

A Schobel, H Hamacher and L R Foulds (1999) On central cycles in grid graphs, Research Report in Management Mathematics, University of Kaiserlautern, Germany.

P J Slater (1981) On locating a facility to service areas within a network. *Operations Research* **29**, 523-531.

P J Slater (1982) Locating central paths in a graph. *Transportation Science* **16**, 1-18.

Older material not being used

7. A branch and bound algorithm for finding the cycle centre

In this section a branch and bound algorithm for finding the cycle centre of a graph *G* will be given. This algorithm provides a systematic procedure for obtaining the cycle centre and provides an alternative to the integer programming approach.

In order to describe the algorithm, a *Hamiltonian Walk* will first be defined.

Definition A Hamiltonian Walk in a graph $G = (V, E)$ is a closed sequence of edges $(e_{i_1}, e_{i_2}, \ldots, e_{i_k})$ passing through every vertex *v* of *G* at least once.

For example, for the graph G given in Figure 3, a Hamiltonian walk *W* is <1,2,3,4,5,4,2,1>.

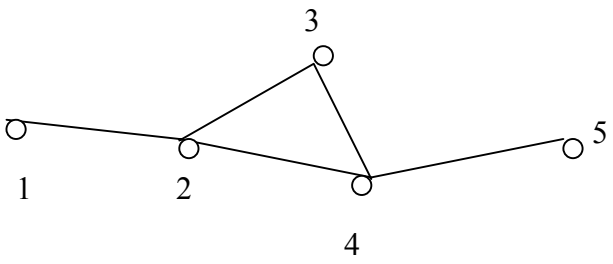

Figure 3

Note that as *G* is non-Hamiltonian, the walk contains repeated vertices.

Property Every vertex in the cycle centre *C* induces all vertices in the original graph *G* to have degree 2 or 0 with respect to *C*.

For example, in Figure 4 vertices 2,3,4 have degree 2 with respect to *C* and vertices 1,5 have degree 0 with respect to *C*.

The Algorithm

1. General step at each node *N* in the decision tree:
   Create a Hamiltonian walk *W* in the graph associated with *N*. If *W* is a cycle, without vertex repeats, it is a feasible solution to the original problem and *N* is fathomed by calculating *z* as defined in section 2. Otherwise:
2. Partitioning
   Ban part of the current walk *W*. Select a vertex *v* of smallest degree with respect to the walk. Select all edges of *W* incident with *v*. Ban each edge in an independent child node of *N*.
3. Branch from the lowest unfathomed node and set this as node *N*. Go to Step 1.

In the example of Figure 3, using the given walk, Step 2 will select vertex 2. Edges in *W* incident with *v* are <1, 2>, <2, 3>, <2, 4>. We then partition the feasible solutions in N as shown in Figure 4.

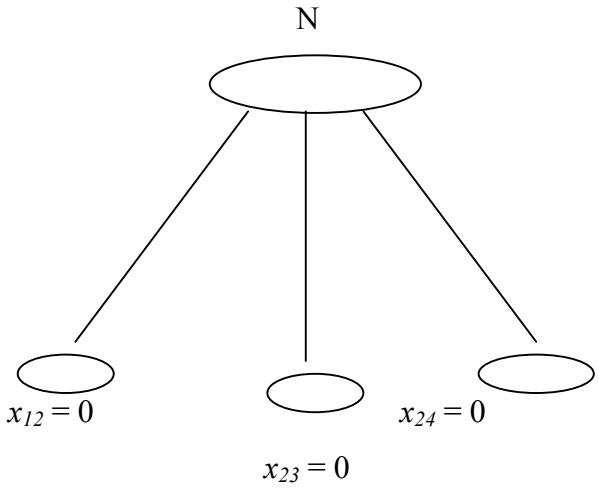

Figure 4

The above will provide a systematic procedure for obtaining the cycle centre of a graph. However, it is likely to be slow to solve sizeable problems. It has the advantage over the integer programming approach that the search may be terminated prematurely once (at least) one feasible solution has been obtained.

It is clear that it will not be practical to solve the models for numerical instances significantly larger than that shown in the table by either integer programming or the

branch and bound method, and hence there is a need for an efficient heuristic method to solve the problems. We now proceed to develop a heuristic method which is more effective than that given in Foulds and Yamagouchi (1998).

8. A heuristic approach

It is clear that the model given by (2.1) – (2.8) can be solved, but only for modest values of n. For larger values of n, a heuristic approach was developed, but its accuracy could at least be checked for smaller values of n where optimal solutions to the model were known.

The heuristic was developed from the observation that a vertex is likely to be in the optimal cycle if the variance of c_{ij} taken over $j = 1,2, \ldots ,n$ is relatively low. For the dataset described in the previous two sections the variances are as shown in Table 4 below.

Table 4 Calculated Variances

Vertex	Variance	Rank (Largest variance=1)
1	1.21	4
2	1.21	4
3	0.81	8
4*	1.00	7
5	1.44	2
6*	0.64	10
7*	0.36	12
8*	0.81	8
9*	0.64	10
10	1.69	1
11	1.44	2
12*	1.21	4

(* = vertex in cycle)

The steps of the heuristic are:

1. Calculate $V(i) = \sum_{j=1}^{n} (c_{ij})^2 - n(\sum_{j=1}^{n} c_{ij})^2$
2. Order the vertices i in increasing value of $V(i)$.
3. Select vertex i_1 lowest in order of previous step, which has not already been chosen.
4. Select vertex i_2 not previously chosen such that $e(i_1, i_2) = 1$ and i_2 next lowest in order of Step 2.
5. Progressively form a cycle by repeating Step 4 and backtracking where necessary, including Step 3.
6. Let cycle be $i_1, i_2, \ldots, i_k$. Calculate z (as defined in Section 2).
7. (Build) For any pair of vertices i_{j-1}, i_j in the cycle such that $e(i_{j-1}, i_j) = 1$ if there exists vertex i' not on the cycle such that $e(i_{j-1}, i') = e(i', i_j) = 1$ and z' for the augmented cycle is such that $z' \leq z$ then add i' to the cycle.
8. (Build) For any triple of vertices i_{j-1}, i_j, i_{j+1} in the cycle such that $e(i_{j-1}, i_j) = e(i_j, i_{j+1}) = 1$ and where there exists vertex i' not on the cycle and $e(i_{j-1}, i') = e(i', i_{j+1}) = 1$ then temporarily replace i_j by i' and recalculate z' for the revised cycle. If $z' < z$ replace i_j by i' and z by z'.
9. Repeat steps 7 and 8 until no further changes are made.
10. (Drop) For any triple of vertices i_{j-1}, i_j, i_{j+1} in the cycle such that $e(i_{j-1}, i_j) = e(i_j, i_{j+1}) = 1$ if $e(i_{j-1}, i_{j+1}) = 1$ and where z' for the cycle formed by dropping i_j is such that $z' \leq z$ then drop i_j from the cycle and replace z by z'.
11. Repeat step 10 until no further changes are made.

End

The whole procedure is now repeated $\sqrt{n}$ times, by varying the starting point of Step 3 and then a further $\sqrt{n}$ times with each choice in step 3 being subject to a probabilistic decision i.e. a simple form of Tabu search. The data used in Tables 1 and 2, with sets of 20 problems, is reconsidered in Table 5.

vertices	optimal	z too large	$\sum_{i=1}^{n} y_i$ too large
16	19	0	1 (1.0)
25	10	3 (1.0)	7 (1.4)

(mean deviation shown in parenthesis)

Table 5 Computational results for the heuristic for small problems

The second column of Table 5 shows the number of occasions when the heuristic solution was optimal, the third column shows the number of occasions when the heuristic solution produced a larger than optimal value for z and the final column shows the number of occasions when the heuristic produced a larger than optimal value for $\sum_{i=1}^{n} y_i$. As can be seen, the heuristic is very accurate on the 16-vertex problems. For the 25-vertex problems fewer optimal solutions are obtained but in a total of 17 problems the z value is correct. Accuracy of the z value will be a more important criterion than accuracy of the $\sum_{i=1}^{n} y_i$ value.

The heuristic was run on a set of 100-vertex problems, where optimal solutions are not known. Results for a set of 20 problems are given in Table 6.

Vertices		CPU secs			z/ub	
	min	max	mean	min	max	mean
100	1.9	7.4	4.0	0.17	0.39	0.28

Table 6 Computational results for the heuristic on larger problems

From Table 6 it is apparent that the heuristic is rapid and is producing solutions in line with the quality of the solutions (as measured by z/(upper bound)) on the smaller sized datasets. The heuristic is capable of producing results in reasonable time for

larger sized datasets. Problems with 200-500 vertices were solved in CPU times of between 50-100 secs.

For the cycle median problem, the heuristic developed earlier can be modified for use. Steps 1-6 are unchanged, Step 7 is modified and Steps 10 and 11 are dropped. The modified version of Step 7 is:

Step 7' (Build) (a) For any pair of vertices i_{j-1}, i_j in the cycle such that $e(i_{j-1}, i_j) = 1$ if there exists vertex i' not on the cycle such that $e(i_{j-1}, i') = e(i', i_j) = 1$ then add i' to the cycle.

(b) For any vertex i in the cycle if there exists vertices i', i'' not on the cycle such that $e(i, i') = e(i', i'') = e(i'', i) = 1$ then add i', i'' to the cycle.

(c) For any pair of vertices i_{j-1}, i_j in the cycle such that $e(i_{j-1}, i') = 1$ if there exist vertices i', i'' not on the cycle such that $e(i_{j-1}, i') = e(i', i'') = e(i'', i_j) = 1$ then add i'', i' to the cycle.

Using the example of Figure 1, Steps 1-6 build the cycle <7,6,8,9,7> with $z = 9$; Step 7 adds vertices 2,3 to vertices 6,7; vertices 4,12 to vertices 7,9; and vertices 10,11 to vertices 8,9. Vertex 1 is also added to vertices 2,6. The cycle is now <1,2,3,7,4,12,9,11,10,8,6,1> and $z = 1$. No further improvement is possible. This cycle is optimal and is shown in Figure 2.

For the cycle centroid problem an analogous modification can be made to the heuristic.

The heuristic for the cycle centre problem was also tested on problems where c_{ij} (given $e_{ij} = 1$) can take a value other than 1. A set of problems in the style of the 16-vetex problems of Table 3 was generated such that c_{ij} (given $e_{ij} = 1$) is given a value chosen at random from 1, 2 and 3, such that each value has equal probability. For this set of problems the performance of the heuristic was more erratic. However, as the heuristic was specifically designed for problems where $c_{ij} = 1$ this behaviour is not

unexpected and further research would be required to develop a more suitable heuristic for other variations.