Particle Swarm Optimization for Two-Connected Networks with Bounded Rings

Earl Brendan Foxwell, BSc. (Hons.)

Computer Science

Submitted in partial fulfillment

of the requirements for the degree of

Masters of Science

Faculty of Computer Science, Brock University St. Catharines, Ontario

Dedicated to my mother, Deborah Lee Foxwell

and to my father, David Foxwell.

"He smart. He make it go."

Abstract

The Two-Connected Network with Bounded Ring (2CNBR) problem is a network design problem addressing the connection of servers to create a survivable network with limited redirections in the event of failures. Particle Swarm Optimization (PSO) is a stochastic population-based optimization technique modeled on the social behaviour of flocking birds or schooling fish. This thesis applies PSO to the 2CNBR problem. As PSO is originally designed to handle a continuous solution space, modification of the algorithm was necessary in order to adapt it for such a highly constrained discrete combinatorial optimization problem. Presented are an indirect transcription scheme for applying PSO to such discrete optimization problems and an oscillating mechanism for averting stagnation.

Acknowledgements

I would like to thank the following people, without whom this thesis would not have been possible:

- **Dr. Beatrice Ombuki-Berman** - For guidance and supervision.
- **Dr. Brian Ross and Dave Bockus** - For endorsing me for the MSc program.
- **Vladimir Wojcik** - For advice and helpful comments.
- **Joseph A. Brown and Andrew Runka** - For insight, expertise, and assistance.
- **Ashley Reedy** - For putting up with me.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

This thesis undertakes the design of survivable and efficient networks using Particle Swarm Optimization (PSO).

1.1 Network Design

This thesis addresses a classic network design problem: determining the most effective and efficient connections to make between servers in a network. That is, if the locations of servers in a network are already known, then the goal is to make connections between those servers such that they can intercommunicate, with a design that affords both efficiency of resources and reliability of communication. Naturally, the most reliable network design, and the design with the shortest distances for signals to travel, would be to have all servers directly connected to all other servers, for point-to-point communications. However, in addition to being prohibitively expensive, it would also be impractical in terms of the high degree of intersecting cables, and the need to cut through numerous geographical and municipal boundaries repeatedly. Rather, it is more appropriate to intelligently select connections in a way that is both cost-efficient, and still reasonably reliable and resistant to failures.

The suggested use of the network design considered for this thesis is Wide Area Networks (WANs), which typically span very large areas, sometimes even continents, and often connect other networks together. Additionally, the designs are also suitable for Metropolitan Area Networks (MANs) and Campus Area Networks (CANs), which may be used for interconnecting several public or commercial networks together across a city, between several buildings within a financial district, or for connecting Local Area Networks (LANs) across a large campus or other similar institution. A common connection type is optical fibre, either entirely buried underground, or laid along the ground, sometimes in trenches, in the case of some underwater cabling connecting landmasses. However, this class of problem abstracts many real-world applications, such as logistics and transportation, computer networking [8], telecommunications [9], oil and gas lines [10], hospitals, universities, water and sewage systems.

1.2 Survivability

There are times when part of a network may cease functioning. This could be due to a server going offline, or it could be a result of a cable being broken. For example, trawling fishing vessels may damage an exposed submarine cable. If that were the only physical line of communication between two landmasses, then the only means of maintaining communication between the bodies would be to redirect traffic through satellites. However, as the volume of traffic increases, this becomes an increasingly infeasible solution [27]. Rather, it is a clear necessity to have a more *survivable* network; one that can retain the capacity to communicate with the other servers in the network, even in the event of a failure. There are different ways to judge survivability, but an accepted way is to gauge the connectedness of the network. That is, by examining the number of links or servers that can be removed without the remaining servers losing the ability to intercommunicate, the survivability of the network can be judged. Since the loss of a vertex (or server) is more disruptive than the loss of a link[1], this thesis only addresses vertex-connectivity.

If a network is sufficiently connected, then it may reroute communication through an alternate path. This is an underlying principle of modern network technologies such as *self-healing rings* [28]. So long as alternate paths exist, communication may be slowed, but will still be possible amongst the unaffected servers.

The design of survivable cost-effective networks is a hugely difficult problem since the number of potential topologies for even small networks is extremely large [1]. Furthermore, inefficient designs can fail to meet customer demands and inadequate service performance [21]. The Two-Connected Networks with Bounded Rings (2CNBR) problem [2][3][4][5] was examined. The 2CNBR problem is an NP-Hard combinatorial optimization problem that was first studied by Fortz et al. [2][3] and it involves designing a minimum cost network T satisfying two conditions:

1. T contains at least two node-disjoint paths between every pair of nodes. This is the*connectivity constraint* [6].

[1]The loss of a link can, at most, prohibit communication that would be routed between a pair of two vertices. The loss of a vertex prohibits *all* communication that would be routed through one of those vertices, including any coming from the other in that pair. Thus, it is equivalent to losing that link, as well as potentially several more.

2. Each edge of T belongs to at least one cycle whose length is bounded by a given constant K. This is the *ring constraint* [2].

The first condition defines 2-vertex-connectivity, henceforth referred to as *biconnectivity* or *two-connectivity.* As a general rule of thumb, a biconnected network is reasonably survivable. The second condition ensures that a rerouted communication will not be diverted through an unacceptably long path, wherein the network designer may choose the threshold of what constitutes a reasonable ring. Additionally, it adds a new dimension to the problem. Two-connected networks have already been studied at great lengths in the past, but the addition of the ring constraint is relatively new, and has still received insufficient attention. Two flavours of 2CNBR have been identified. The first defines the ring constraint in terms of Euclidean edge lengths, and the second requires that each edge belongs to a cycle using at most K edges. This thesis focuses on the former flavour.

1.3 Particle Swarm Optimization

Since finding the optimal solutions for such NP-Hard problems is computationally intractable [7], brute-force and other exact methods are not a realistic choice as the problem size increases. Instead, rather than pursuing optimal solutions, the goal becomes to find 'good' solutions within reasonable time. Such approximations are a natural application of metaheuristics. As Tabu[15] and Genetic Algorithms [16][17] have been applied to 2CNBR in the past, this thesis focuses on Particle Swarm Optimization.

Particle Swarm Optimization is a population-based metaheuristic inspired by flocking birds and schooling fish. Each particle continuously flies through some n^{th}-dimensional space, and the position in each of those n dimensions represents part of a complete solution. Over several iterations, the particles typically have some tendency to stay near to their individual best results, and some desire to move towards the best results found by other particles within the swarm. In general, the search is best used for problems that can be represented as vectors of floating point values, where there are no 'gaps' or other illegal values within the overall bounds of a dimension. They are also similarly not suited for problems where the value in one dimension constrains the legal values within other dimensions. Refer to section 2.3 for a more complete explanation of Particle Swarm Optimization, and Chapter 3 for a description of how it was applied to 2CNBR.

1.4 Objectives and Contributions

This thesis has three primary objectives. First and foremost, it aims to add to the body of work dedicated to the 2CNBR problem; to supplement the metaheuristic work previously done [15][16][17]. Furthermore, it devises methodologies for applying PSO to such highly-constrained problems as the 2CNBR by use of an indirect transcription scheme and introduces novel techniques for avoiding stagnation and improving upon the initial results. Finally, it attempts to combine elements of pheromones from Ant Colony Optimization with PSO.

Though network design, and even specifically the design of survivable networks, has already been explored in great depth, the addition of bounded rings introduced a new area of research for investigation, and this area has yet to receive sufficient attention. This thesis contributes to the relatively limited body of work devoted to the exploration of the Two-Connected Networks with Bounded Rings (2CNBR) problem. More specifically, this thesis expands upon the metaheuristic work on the 2CNBR problem, which has previously only consisted of Tabu search and Genetic Algorithms (refer to Chapter 2).

As such, this thesis, and its preliminary work published in [25], represent the first time that Particle Swarm Optimization (PSO) has been applied to this problem. PSO is normally suited for continuous spaces, not discrete optimization and the constraints and discrete nature of this problem would normally preclude the use of a traditional PSO. As explained in Chapters 2 and 3, this necessitated the implementation of a novel indirect transcription scheme to get past the natural limitations of PSO, as well as introducing oscillation to stave off stagnation. These innovations upon the standard PSO represent a significant contribution in and of themselves.

1.5 Overview

Chapter 2 provides a formal definition of the 2CNBR problem, as well as background information on particle swarms. Chapter 3 details the design and experiments of one application of PSO to 2CNBR, as well as an improved form thereof. Chapter 4 details the design and experiments of another variation that incorporates some qualities of ant colony optimization. Chapter 5 contains the final conclusions and discussions, as well as possible future work.

Chapter 2

Background

This chapter formally defines the Two-Connected Network with Bounded Rings problem, including the identification of solutions of its component constraints that are independent of this thesis's contribution. It then provides background information on particle swarms in general, again separate from the specific contributions of this thesis.

2.1 2CNBR Defined

We provide a mathematical formulation of the 2CNBR based on that derived and used by Fortz et al. [3]. Let $G = (V, E)$ be an undirected graph, where V represents a set of vertices, and E is the set of edges that represent possible pairs of vertices between which a direct link can be made. Each edge $e = (i, j) \in E$ (where i and j are any two vertices), has a non negative cost $C_e = C_{ij}$, and a length d_{ij}. The constant K defines the size of shortest cycle (ring) to which each edge belongs. Let the cost of a network $T = (V, E_T)$ (where $E_T \subseteq E$ is a subset of possible edges) be denoted by $c(E_T) = \sum_{e \in E_T} C_e$.

Given graph and $V' \subseteq V$, the edge set $\delta_G(V') = \{\{i, j\} \in E | i \in V', j \in V \backslash V'\}$ is called the *cut* induced by V'. Let $V - w = V \backslash \{w\}$ and $E - e = E \backslash \{e\}$ be the subsets as a result of removing one vertex or one edge from the set of vertices or edges. Thus, $G - w$ represents the graph $(V - w, E \backslash \delta(\{w\})$, as a result of removing a vertex w and its incident edges from G [6].

Each subset $E_T \subseteq E$ is associated with an incidence vector, defined as $y = (x_e)_{e \in E} E\{0, 1\}^{|E|}$ by setting $x_e = 1$ if $e \in E_T$, or $x_e = 0$ otherwise. On the other hand, each vector $y \in \{0, 1\}^{|E|}$ induces a subset $E_T = \{e \in E | x_e = 1\}$ of the edge set E. For any subset of edges $E_T \subseteq E$, $x(E_T) = \sum_{e \in E_T} x_e$ is defined.

Next, for each edge $e \in E$, ξ_e is defined as the set of cycles in G that includes edge e whose length is less or equal to the constant K. To differentiate which cycles are used in imposing ring constraints for a given edge, a new variable is introduced for each feasible network containing a given edge for all edges in the network. Hence, new binary variables Y_e^c, $c \in \xi_e$, $e \in E$, such that

$$Y_e^c = \begin{cases} 1 & : \text{ if cycle } c \text{ is in the solution } E_T \\ & \quad \text{and covers edge } e \\ 0 & : \text{ otherwise} \end{cases}$$

Now the 2CNBR can be mathematically formulated as follows [3]:

(1) $\min \sum_{e \in E} C_e x_e$
s.t

(2) $x(\delta(V')) \geq 2,\ V' \subset V,\ \emptyset \neq V' \neq V$

$x(\delta_G - w(V')) \geq 1, w \in V, V' \subset V \backslash \{w\},$
(3) $\emptyset \neq V' \neq V \backslash \{w\}$

(4) $\sum_{c \in \xi_e} Y_e^c \geq x_e, e \in E$

(5) $\sum_{c \in \xi_e : f \in c} Y_e^c \leq x_f, e \in E, f \in E \backslash \{e\}$

(6) $x_e, Y_e^c \in \{0, 1\}, c \in \xi_e, e \in E$

where inequalities (2) are called *cut inequalities* and they ensure that removing an edge preserves connectivity. Inequalities (3) are called *node cut inequalities* and they ensure that the resulting graph has no articulation vertex. Using inequalities (2) and (3) together with $x_e \in \{0, 1\}$, $e \in E$ based on (6), we obtain the formulation of the minimum 2-connected network problem as studied by Grotschel et al. [35]. In addition (4) and (5) are the ring constraints that extend the 2-connected network problem into the 2CNBR. Constraint (5) restricts the contribution of cycles that share some edges. Further details of this mathematical formulation of the 2CNBR problem is found in work done by Fortz et al. [3].

Some assumptions are made in dealing with this problem: each node location is given; bidirectional links are allowed, but no parallel edges are allowed; no link repair is considered, and each link cost is fixed and known. According to graph theory [6], for any graph representing a network to be termed as having two node-disjoint paths between every pair of nodes, the feasible condition must hold: for any two nodes in a network topology, there exists a cycle containing both of them.

2.1.1 Testing for Two-Connectivity

As previously mentioned, the first constraint in the 2CNBR problem is that there must be two node-disjoint paths between every pair of vertices within the network. This means that, for any candidate solution to the problem, there must be a reasonable way of testing if this property is present within a graph.

Actually doing a full search to find each separate path between each pair of vertices would be computationally prohibitive, not to mention logically unnecessary. Rather, it is preferable to address the problem in terms of *articulation points.*

Articulation Points

When considering biconnected graphs, one might also consider biconnected components [26]. Specifically, a biconnected component is an equivalence class of edges that lie within common cycles. However, what is important is that two biconnected components may have at most one common vertex, which is an *articulation point*, also known as a *cut-vertex.*

Aho et al. [26] describe an articulation point thus: Let $G = (V, E)$ be a connected, undirected graph. A vertex a is said to be an articulation point of G if there exist vertices v and w such that v, w, and a are distinct, and every path between v and w contains the vertex a.

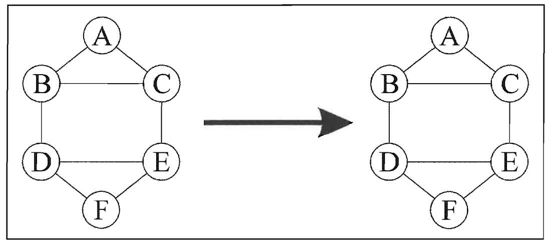
That is to say, if a is removed from G, then G will be split into at least two separate subgraphs. This means that, since a biconnected graph must possess at least two node-disjoint paths between every pair of vertices, a biconnected graph must possess no articulation points.

This means that there are two approaches that one may take to verifying that a graph is biconnected. One could find the biconnected components of a graph, and know that, if there is only one such component, then it is biconnected. Or, one could search for articulation points, verifying that it is biconnected if no such points exist. As it so happens, the methodologies for both techniques are virtually identical. For this work the algorithm used was a modification of Aho et al. [26](ch. 5)'s algorithm, which originally found biconnected components, identified by their articulation points.

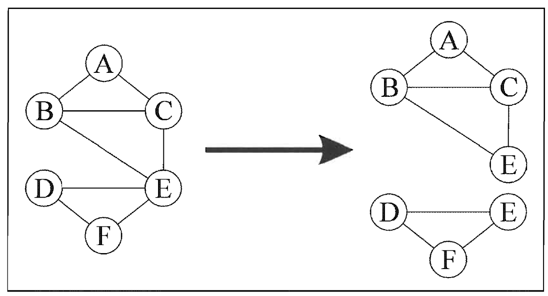
Identifying the Existence of Articulation Points

The graph seen in Figure 2.1a is biconnected, as it is identical to its biconnected component. However, the graph seen in Figure 2.1b is not biconnected, as it has two biconnected components. Alternatively, one can say that the graph in Figure 2.1b is not biconnected because it has a vertex, E, which is an articulation point.

Conceptually, the method proposed by Aho et al. [26] was to create a depth-first spanning tree of the graph. A vertex, a, is an articulation point if and only if [26]:



(a) one biconnected component



(b) two biconnected components

Figure 2.1: Two Graphs and Their Biconnected Components

1. a is the root and a has more than one son, or
2. a is not the root, and for some son s of a, there is no back edge between any descendent of s (including s itself) and a proper ancestor of a

This effectively means that a vertex is an articulation point if its removal would separate the tree into two components, which is analogous to the original problem. It is worth noting that it does not matter how the *root* is chosen for the tree. As the goal is to locate cycles, the end result will be the same irrespective of which vertex is chosen. Figure 2.2 depicts the depth-first spanning trees of the graphs shown in Figure 2.1. Again, we can see that, in Figure 2.2a, there is no vertex that would separate any son or descendent from its proper ancestors. We also see that, in Figure 2.2b, the removal of vertex E would separate D and F from their ancestors.

As stated, the purpose of Aho's algorithm was to actually produce the edge lists of biconnected components, identified by finding these articulation points. However, the only concern for this thesis was to identify two-connected networks, and thus actually locating the individual biconnected components of an incomplete network is of no use. As such, the algorithm was somewhat simplified to merely test for the existence of articulation points. The actual method used is shown in algorithm 1.

Note that the algorithm starts by assuming that the network is biconnected, until proved otherwise. This is because the graph is biconnected unless an articulation

Algorithm 1 Two-Connectivity Test

```
function checkTwoConnectivity
Input: A network (graph) to be tested
Output: A boolean, indicating if the biconnectivity constraint is satisfied
begin
  biconnectedFlag←true
  count←1
  for each vertex in vertices do
    vertex.flag←false //mark vertex as unvisited
  end for
  twoConnectivityDFS(rootVertex)
  for each vertex in vertices do //special case for disconnected vertices
    if vertex.flag=false then
      return false
    end if
  end for
  return biconnectedFlag
end

procedure twoConnectivityDFS
Input: A vertex, vertex
begin
  vertex.flag←true //mark vertex as visited
  vertex.dfn←count //assign an ID to this vertex in the tree
  vertex.low←count //identifies highest ancestor in this component
  count←count+1
  for each n in vertex.neighbours do //for each connected vertex...
    if n.flag=false then //if n is unvisited...
      n.father←vertex
      twoConnectivityDFS(n)
      if n.low≥ vertex.dfn then //if vertex is an articulation point...
        biconnectedFlag←false
      end if
      vertex.low←min(vertex.low,n.low)
    else if vertex.father≠ n then
      vertex.low=min(vertex.low,n.dfn)
    end if
  end for
end
```

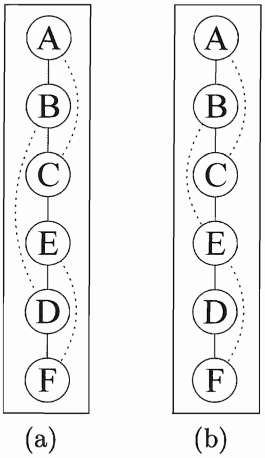


Figure 2.2: Depth-First Spanning Trees of Figure 2.1 Graphs

point is present (and eventually found) within it. The *rootVertex* is simply whichever vertex is chosen for the root of the tree. As mentioned earlier, it does not matter which vertex is chosen; in the examples shown in Figure 2.2, it was simply *A*.

The final implemented algorithm was, of course, slightly different, as it needed to include minor improvements for overall efficiency. For example, if the network being tested had not yet assigned at least two edges to each vertex, then there would be no possibility of the network being biconnected; as such, it would be pointless to bother with such a search. Similarly, if a network were to pass the two-connectivity test, but fail the bounded ring test (explained below), then, upon simply adding more edges, there would be no need to retest the biconnectivity constraint[1]. And, finally, when the algorithm found an articulation point, it did not actually continue running, but rather immediately returned *false*, as continuing to search for more articulation points, when only one is needed to declare the network invalid, would only have wasted computing time. However, none of these modifications change the actual functionality of the algorithm, merely the speed of execution, so, for the sake of clarity, they are not included in algorithm 1.

2.1.2 Testing for Bounded Rings

As previously mentioned, the second constraint in the 2CNBR problem is that every edge present must be part of some ring whose cost does not exceed the specified bound. It may, of course, *also* be present in other rings that exceed that upper

[1]Simply put, if a network is biconnected, then adding *more* edges cannot decrease the number of node-disjoint paths between vertices.

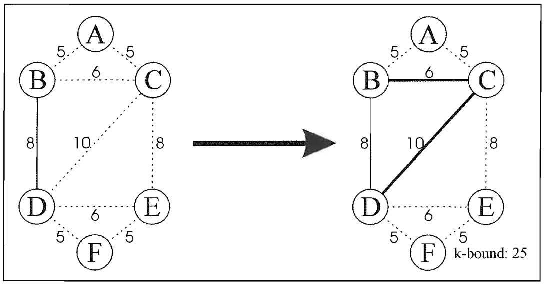
bound, as this will often be unavoidable. There are different approaches to testing this constraint. One naïve approach is shown in algorithm 2.

Algorithm 2 Naïve Bounded Ring Test

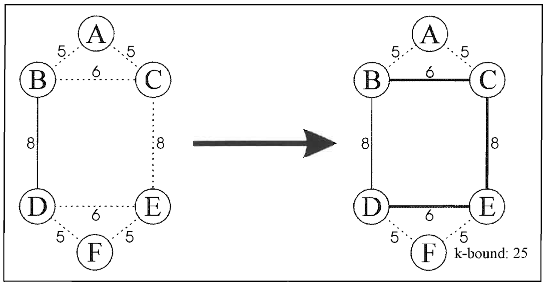
Input: A network (graph) to be tested
Output: A boolean, indicating if the ring constraint is satisfied
for each $edge_1$ in *edges* **do**
  for each $edge_2$ in *edges* **do**
    for each $edge_3$ in *edges* **do**
      if $edge_1$, $edge_2$ and $edge_3$ are all different **then**
        if $(|edge_1| + |edge_2| + |edge_3|) \leq K - bound$ **then**
          Flag $edge_1$, $edge_2$ and $edge_3$ as verified
        end if
      end if
    end for
  end for
end for
for each *edge* in *edges* **do**
  if *edge* is not flagged as verified **then**
    return *false*
  end if
end for
return *true*

However, this would be far too computationally expensive, considering how many times it would need to be run in an algorithm. The naïve approach is just that: naïve. An alternative approach would be to first test only edge *triangles* (as they seem to be most common for rings), and then only further test those edges that are not part of ring-satisfying triangles. However, in the event that a network were to consist of a large number of quadrilateral rings, it would devolve into the original problem with a naïve approach.

Another approach would be to first remove the edge to be tested, and then attempt to find an alternate path between the tested edge's vertices, wherein the cost of the new path plus the cost of the removed edge do not collectively exceed the specified bound. See Figure 2.3 for an illustration. The dashed lines represent edges present in the network being tested. The solid line represents the specific edge currently being tested. In the second halves of each subdiagram, the bold lines represent alternate paths to connect the vertices of the edge being tested. That is, if edge BD is being tested, then each subdiagram shows an alternate path from vertex B to vertex D. Since edge BD has a cost of 8, and the k-bound is 25, the alternate path must have a cost not exceeding 17 ($25-8=17$) if the network is to be verified as legal. However, in Figure 2.3b, there does not exist an alternate path that does not exceed the remaining allowable cost.



(a) legal alternate path exists



(b) no legal alternate path exists

Figure 2.3: Finding an Alternate Path When There Is and Is Not a Legal Ring Containing the Edge

The alternate path could be found using Dijkstra's algorithm [36]. Certainly, if the *shortest* alternate path between the two vertices yielded a cost which did not violate the ring constraint, then the ring constraint would be valid for that edge. And, similarly, if even the shortest alternate path between the vertices was still insufficient to satisfy the ring constraint, then the network could not be validated as a 2CNBR-compliant network. However, going so far as to find the shortest alternate path may be excessive. For example, if there are multiple alternate paths between the vertices that all satisfy the constraint, then it does not matter which of those paths is the shortest. It is the mere existence of *any* of them that contributes towards validating the network. Similarly, if no such alternate paths at all exist, then it is a fool's errand to continue trying to find the shortest path once it is apparent that any such path will violate the constraint anyway.

A Simple Depth-First Search

An alternative is to simply use a basic depth-first search algorithm to locate alternate paths between the vertices of the removed edge. Such an algorithm would merely return a *true* if it found an alternate path that did not violate the constraint, return a *false* if it could not find any alternate paths at all, and halt prematurely and return a *false* if it was not done searching but had already exceeded the allowable upper bound. This way, it does not invest unnecessary computation time into finding the shortest path when numerous other paths would do just as well. It also does not waste time pursuing paths after the bound is exceeded. In short, it acknowledges that it isn't the *actual* alternate path itself that matters, but simply the *existence* of such an alternate path.

Algorithm 3 shows how the depth-first search works. Note that it has a conditional before testing each edge, to check if that edge has already been verified. The reason this situation could arise is because of an efficiency trick. The basic methodology is to verify each edge in the network one-by-one. However, if an edge is verified by showing that it is part of some ring that doesn't exceed the k-bound, then that means that, naturally, all the other edges in that ring must also be part of such a ring. As such, they can all be marked as verified, thus eliminating the need to separately verify them later on. As with the case of the biconnectivity test, the actual implemented algorithm had other efficiency-increasing measures not shown in algorithm 3. For example, the edge verification flags are not actually set within the bounded-ring test's function. Rather, they are set externally. The reason is that, for algorithms that progressively add more edges and retest, there is no need to repeatedly re-test edges that have already been verified. If an edge is part of a bounded ring, then adding more edges to the network cannot change that quality. Additionally, code was added to prevent backtracking within the DFS, simply to encourage the algorithm to spread out to find the other vertex.

2.1.3 Evaluation

The actual goal of the algorithms, beyond simply satisfying the biconnectivity and bounded ring constraints, is to minimize the total *cost* of the network. As mentioned earlier, that *cost* can either be taken as the total Euclidean distances of all of the included edges, or it can simply be the total number of edges. For this thesis, the former was chosen. However, to allow direct comparison with the works of Ombuki, Ventresca and Fortz, it is necessary to use the precise same method of evaluation.

Specifically, the length of each edge is calculated, and then rounded to the nearest integer value. It is those integer lengths that are then summed for the total network cost. As such, all network costs listed and compared will always be integers. The

Algorithm 3 Bounded Ring Test Using Depth-First Search (DFS)

```
function checkKBound
Input: A network (graph) to be tested
Output: A boolean, indicating if the bounded-ring constraint is satisfied
begin
  kBoundFlag←true
  for each edge in edges do
    edge.flag←false //mark edge as unverified
  end for
  for each edge in edges do //for each edge in the network...
    if edge.flag=false then //if this edge has not been verified...
      from=edge.from //one vertex of edge
      to=edge.to //the other vertex of edge
      remove edge from edges
      if boundedRingDFS(from,to,edge.length) then
        add edge back to edges
        mark edge as verified
      else
        add edge back to edges
        kBoundFlag←false
        break
      end if
    end if
  end for
end

function boundedRingDFS
Input: Vertex from, vertex to, cost so far
Output: A boolean, indicating if an edge is verified
begin
  if from=to then //a stopping condition
    return true
  end if
  for each e in from.edges do
    if e.length+cost ≤kbound then
      if boundedRingDFS(e.other,to,e.length+cost) then
        e.flag=true
        return true
      end if
    end if
  end for
end
```

lower the cost, the better the algorithm performed.

2.2 Previous Work

Although designing a survivable two-connected network at minimum cost (so called low-connectivity) has been widely studied [8][9][10][12], and efficient methods for solving it are already available [13] (for a comprehensive survey of network design problems and their applications, see [14]), the extension of two-connected networks to include bounded rings was recently introduced by Fortz et al. [2][3]. Fortz et al. [2][3] proposed adding ring constraints to the two-connected network such that the shortest cycle to which each edge belongs does not exceed a given maximum length K. This is a relevant extension since the ring constraints limit the region of influence of the traffic which necessarily needs to be re-routed. Furthermore, a minimum cost two-connected network is often found to be a Hamiltonian Cycle. This means that if a connection is broken, the flow which was routed using such connection needs to be re-routed using all the edges of the network; this is an undesirable effect.

Fortz et al. [3] applied a branch and cut method for the two-connected network with bounded rings problem, and showed that the algorithm is only effective for small instances.Thus he presents a set of constructive heuristics and a Tabu search approach to solve this problem [2][15]. Ombuki *et al.* [16] introduced a genetic algorithm using permutation representation with a problem-specific crossover operator used to generate feasible solutions for to the $2CNBR$. Ventresca et al. [17] further expanded on the work in [16] by using a binary representation and incorporating a non-problem specific crossover operator. They demonstrate the GA's effectiveness with a comparative study with published Tabu search [15]. This thesis seeks to further contribute to the use of metaheuristics for 2CNBR by investigating the applicability of a simple particle swarm optimization algorithm for the problem.

Despite various literature reporting meta-heuristics applications to network design and optimization issues ([8][9][10][13], among others) the problem of designing two-connected network topologies with bounded rings using metaheuristics has not been fully investigated. The following are examples of previous work using population based meta-heuristics for related ring-based design problems. He et al. [18] introduced an evolutionary algorithm for ring-based SHD optical core networks. White et al. [1] introduced an efficient GA (for large problem spaces) with a hybrid bit and permutation representation for designing a ring based network. Armony et al. [19] present a genetic algorithm for solving SONET ring structure design problems. Chen and Zheng [20] present a GA for ring networks in order to balance the traffic loads on ATM rings and minimize the overall capacity requirement of the rings.

2.3 Particle Swarm Optimization

In this section we provide a brief overview of the general concept of particle swarm optimization. PSO [29][30][31] is a metaheuristic that was inspired by flocking birds. As birds scavenge for food, they fly over an area. Sometimes they will follow their own sense of smell, but they will also tend to follow other birds in the flock. This duality defines the balance that particle swarms attempt to strike between a particle doing its own thing, and it collaborating with the rest of the swarm.

PSO has been used for numerous types of optimization, including camera control [22], Job Shop Scheduling [23], and the training of Artificial Neural Networks [24]. In fact, by suitably modifying the PSO implementation, and choosing ideal parameters for the task, particle swarms can be applied to a wealth of different applications.

In normal Particle Swarm Optimization, a *particle* contains a complete solution to the problem being optimized. More specifically, the *position* of the particle in n^{th}-dimensional space represents a complete solution to the problem, where n reflects the number of values necessary to create a feasible solution to that problem[2].

A *swarm*, analogous to a population in Genetic Algorithms or other similar population-based search techniques, is the collection of all particles within the system. Thus, a swarm contains multiple candidate solutions to the problem being solved. Though each member particle within a swarm is an independent entity, there still may be some interaction between the particles within a swarm.

The particles are constantly moving through the n^{th}-dimensional space, with some *velocity* vector. This means that the candidate solutions are being continuously updated. It is by choosing a suitable mechanism to modify the velocities that the positions may approach 'good' solutions. Specifically, the velocity update rule should typically promote exploration and also afford the particles some degree of cooperation with the other particles within the swarm, to achieve a common goal.

As previously mentioned, particle swarms are suitable for a wide range of problems. However, there are certain qualities that will be present in problems for which particle swarms are suitable. First, for a given problem instance, the particle position will be of a fixed number of dimensions[3]. Additionally, since the particle's position is typically a vector of floating point values, appropriate problems will have solutions

[2] Compare this concept to *chromosomes* in Genetic Algorithms, wherein a chromosome represents a complete solution. In an example such as the weights of an Artificial Neural Network, a chromosome could be identical to a particle's position if both vectors contained floating point values representing the same solution.

[3] That is, if there are 100 dimensions at the beginning of a run, then each particle position will still have precisely 100 dimensions at the end of computation. However, there is no requirement that each dimension have influence over the transcribed solution, as such behaviour is strictly within the realm of the fitness evaluation function.

that can somehow be created from such vectors. Of course, even problems requiring integers can at least sometimes still be solved, by either truncating or rounding the decimals. However, an additional requirement is that the particles be permitted to move freely throughout the search space. That is, though there may be some upper and lower bound on each dimension, they must be continuous, with no 'holes' or 'gaps' within. Furthermore, the position of a particle within one dimension cannot limit the allowable positions within other dimensions. That is, it must be possible for the change in position within each dimension to be handled independently. This implies difficulty when attempting to use PSO for certain combinatorial optimization problems.

Though Particle Swarm implementations vary greatly from instance to instance, there are some basic commonalities nearly always found. First, there is typically some notion of *inertia*, such that a particle's velocity will tend to maintain some portion of that velocity across multiple iterations. Second, there is a *cognitive* aspect, such that a particle will tend to drift towards solutions that it has personally identified as being 'good'[4]. The third, and final, standard quality of Particle Swarms is the concept of *social* interaction; that is, the tendency to drift towards the best solution found by some other particle within the swarm[5]. Oftentimes [22], there is an additional *explorative* factor, which prompts the particle to accelerate independently of any past knowledge at all.

2.3.1 Basic PSO Algorithm

The basic particle swarm algorithm is fairly simple, and typically independent of the problem being solved. It is shown in algorithm 4.

Looking at the algorithm, one can see that there are two key points where innovation can be introduced. The fitness evaluation function, particularly by virtue of the mechanism by which the position is transcribed into a working solution, will have a great influence on the overall effectiveness and viability of the algorithm. And the manner in which the velocity and velocity update are handled will directly control the movement of the particles, and thus by extension the progress through the solution space. The velocity update rule is further explained later in this chapter.

---

[4]That is, a particle will tend to be attracted to the best solution it has personally found so far in that run.

[5]The particles may drift towards the best solution found by the entire swarm, or towards the best solution found by some neighbour, wherein that neighbour can be chosen in numerous different ways.

Algorithm 4 Basic PSO Skeleton

```
Randomize positions and velocities of particles
for iteration = 1 to MAXGEN do
  for all particles in swarm do
    Evaluate particle fitness
  end for
  for all particles in swarm do
    Calculate new velocity
    Update position based on velocity
  end for
end for
```

2.3.2 Canonical Particle Swarms

The original, canonical Particle Swarm velocity update mechanism is as follows:

$$\vec{v}\,' = \omega \cdot \vec{v} + c_1 \cdot \vec{r_1} \cdot (\vec{x}^b - \vec{x}) + c_2 \cdot \vec{r_2} \cdot (\vec{x}^{gb} - \vec{x})$$

where:

- ω is inertia.
- $\vec{v}\,'$ is updated velocity vector for the next iteration.
- $\vec{v}$ is the current velocity vector.
- $\vec{x}$ is the particle's current position.
- $\vec{x}^b$ is the position of the best solution this particle has found.
- $\vec{x}^{gb}$ is the position of the best solution the system has found.
- c_1 is a multiplier representing the particle's *cognitive* aspect.
- c_2 is a multiplier representing the particle's *social* aspect.
- $\vec{r_1}$ and $\vec{r_2}$ are random multipliers, with component values ranging from [0..1].[6]

To include the additional explorative factor, which has become common (and practical for avoiding premature convergence and stagnation), it becomes:

$$\vec{v}\,' = \omega \cdot \vec{v} + c_1 \cdot \vec{r_1} \cdot (\vec{x}^b - \vec{x}) + c_2 \cdot \vec{r_2} \cdot (\vec{x}^{gb} - \vec{x}) + c_3 \cdot \vec{r_3} \cdot \vec{z}$$

where:

[6]Note that $\vec{r_1}$ and $\vec{r_2}$ are vectors here, but many will choose to use scalar random multipliers instead.

- $\vec{z}$ is a random vector.
- c_3 is a multiplier representing the particle's *explorative* aspect.
- $\vec{r_3}$ is a random multiplier, with component values ranging from $[0..1]$.

For the rest of this thesis, whenever a 'canonical' particle swarm is mentioned, it will assume the presence of this explorative aspect.

As can be seen in the mechanism for the velocity update rule, the particles are intended to be able to float wherever they wish. As such, PSO is particularly well-suited for problems for which solutions can be translated from multiple floating point values, and where the positions of the particles in each dimension are not subject to specific constraints.[7]

2.3.3 Neighbours and Neighbourhoods in Particle Swarms

Since a major factor in PSO is the influence of social interactions, it begs the question of how the system chooses *which* other particles a given particle may interact with for social collaboration. The two issues that need to be defined are the number of other particles with which a particle may interact, and the method by which those other particles are selected. Though there are several options, there are two primary social mechanisms for dealing with particle cooperation.

Global social behaviour

The *global* social behaviour, used in the canonical PSO algorithm, is good for cooperation and quick convergence. It is seen in the canonical PSO velocity update rule as the term, $c_2 \cdot \vec{r_2} \cdot (\vec{x}^{gb} - \vec{x})$, where $\vec{x}^{gb}$ is the position corresponding to the 'best' solution found thus far by any particle in the system. It can be considered a special case of the *neighbourhood* behaviour, where the neighbourhood size is taken to be equal to the size of the swarm. That is, the *global* social behaviour can be replicated by implementing the *neighbourhood* social behaviour and simply ensuring that every particle lies within the neighbourhoods of all other particles within the swarm.

Neighbourhoods

With the neighbourhood behaviour, each particle is drawn towards the best solutions found by any of its neighbours. There are two factors which can determine how a particle's neighbours are chosen. The first is the size of the neighbourhood chosen.

[7] That is, one should assume that a particle will not have to 'skip' over any portions of a dimension; or have one dimension's position constricted dependent on the position in another dimension.

The second is the actual mechanism by which neighbours are identified, and there are two choices for this as well. The first choice is to use the proximity[8] of other particles to the particle being considered. If a particle is within a specified radius of the target particle, then it is considered to be a neighbour. The second option is to predefine each particle's neighbours before optimization even begins. Since the particles will naturally drift towards their neighbours, it can normally be expected that a particle will end up physically close to its neighbours anyways.

2.4 Preprocessing

In a simple approach to 2CNBR, the algorithm would assume that connections are permissible between any and all pairs of vertices, as this is part of the original problem definition. However, in practice, such an approach could present problems almost immediately, when one considers the K-bound. Recall that any edge included must be part of at least one ring wherein the ring's cost does not exceed a bound K. There are conceivably some problems where the length of some potential connections would negate the possibility of belonging to such a ring. In the (trivial case/worst-case scenario), a potential connection between two vertices could, by its own cost alone, be enough to violate the ring constraint.

Refer to Figure 2.4 for an example showing all of the possible connections between the vertices of a network. In this example, edge AF has a length of 16. The smallest ring that can include edge AF would have a total cost of 34. If the K-bound was 25 for this problem, then the decision to include edge AF would automatically guarantee that no subsequent network, irrespective of how many more edges were added, could ever be considered *legal*[9] until that edge was removed.

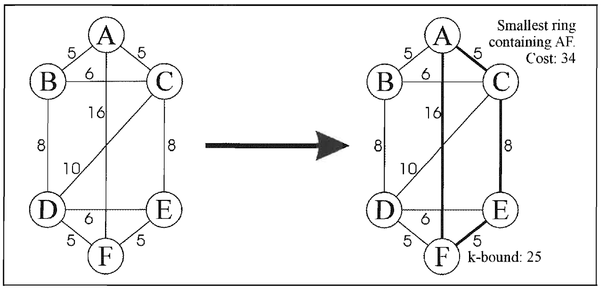


Figure 2.4: Example Depicting Mandatory Violation of the Ring Constraint

[8]In this case, there are different ways to determine the 'distance' between two particles, including the Euclidean distance or edit metrics across the dimensions of each particle.

[9]In this context, and for the remainder of this thesis, the term *legal* will be used to describe the quality of satisfying both the two-connectivity and ring constraints.

All such edges, wherein their inclusion would guarantee an illegal network, can be deemed *illegal edges.* This does not mean to say that the original problem definition explicitly forbids their inclusion, but rather that there is no possibility of their belonging to any legal solution. Algorithm 5 depicts a simple method of identifying edges as being either legal or illegal.

Algorithm 5 Illegal Edge Removal

Input: A set of all possible edges in a fully-connected graph
Output: A set of all legal edges in a graph
for each $edge_1$ in *edges* **do**
  for each $edge_2$ in *edges* **do**
    for each $edge_3$ in *edges* **do**
      if $edge_1$, $edge_2$ and $edge_3$ are all different **then**
        if $(|edge_1| + |edge_2| + |edge_3|) \leq K - bound$ **then**
          Flag $edge_1$, $edge_2$ and $edge_3$ as verified
        end if
      end if
    end for
  end for
end for
for each *edge* in *edges* **do**
  if *edge* is not flagged as verified **then**
    Remove *edge* from the set
  end if
end for

In order to do this preprocessing, the set of all edges in a fully-connected graph is the starting point. The illegal edges are then removed from that set, resulting in a set consisting solely of legal edges, which may be considered when creating the network. It should be noted that this algorithm is very similar to the naïve bounded ring test, algorithm 2. That algorithm was rejected for being computationally infeasible, but this algorithm is still computationally reasonable. The bounded ring test, for many algorithms, would need to be performed numerous times. It may need to be performed multiple times during the construction of a feasible solution, for subsequent evolved networks, and in the case of population-based algorithms, the number of tests would also be multiplied by the population size. As such, it the concern became the growing computational cost from multiplying an inefficiency several times over. On the other hand, this preprocessing need only be performed a single time prior to the application of a network construction algorithm. This is true even if multiple runs need to be performed on the same problem. A single application of this algorithm takes less than one second to perform for even the most difficult problems attempted in this thesis. As such, even though a comparable algorithm was rejected for use millions of times, this one has a negligible impact on total computation time.

Chapter 3

Priority-Based PSO

This chapter details the techniques by which PSO was used for the 2CNBR problem. It defines the initial issues that had to be resolved, and the results of preliminary experiments. Since those early results were unacceptable, it then continues to explore a modification to the original design that vastly improved the system's effectiveness, and includes experimental parameters and a more complete listing of results. Those results are then compared against past metaheuristic work.

3.1 Applying PSO to 2CNBR

The first and foremost problem with trying to use PSO for 2CNBR was in deciding how to represent a network in a particle's position. That is, how can one transcribe a vector of continuous floating point values into a legal network? The first representation considered was using the position in each dimension as a decision for including a corresponding edge. If $x \geq 0.5$, then the edge would be included, otherwise it would not. However, the first problem with this is that it could allow for illegal networks. Additionally, it did not seem as though the swarm would have had a good chance at improving solutions, as the positions would strictly define 'on' or 'off', and the velocities would not be expected to train positions that happened to remove costly edges while adding better edges simultaneously. To understand this idea, consider a genetic algorithm(GA). A GA can have two particularly useful strengths. First, it can use operators which ensure that the solutions are always legal. For example, crossover operators can have built-in mechanisms for ensuring that the child chromosomes also have feasible transcribed solutions. Mutation operators can remove one edge, but also add others, to guarantee that the constraints are still satisfied. Second, and similarly, it can connect the acts of addition or removal of some edges with the addition or removal of other edges. This may not sound like a significant quality, but it can be essential to an algorithm's ability to *improve* upon a solution. Oftentimes, for this type of problem, the only way that a network can have its cost reduced is if one or more edges are removed, and better edges are added in their place.

Compare this to the previous description of PSO. The position in one dimension should not have any direct impact on the position in another dimension. This means that the first considered methodology would not be appropriate, as it would almost certainly need to rely primarily on random chance to improve results. i.e. one edge drifts towards *off* while another edge, that happens to be able to satisfy the constraints, coincidentally drifts towards *on*. Indeed, this became as large a concern as the more basic problem of ensuring legal networks from each particle position. The solution to one turned out being the solution to both: use an indirect encoding scheme that both guarantees legal networks *and* allows edges to be able to 'swap' with each other, all without impeding the particle's ability to float freely without special restrictions.

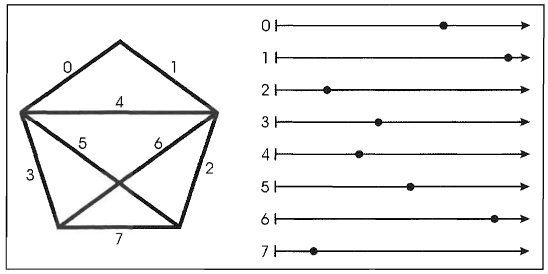
Algorithm 6 Priority-Based Particle Transcription and Evaluation

```
for all particles in swarm do
  QuickSort positions, from lowest to highest
  while Feasible network not yet constructed do
    Add edge with next lowest position
    Check two-connectivity and k-bound constraints
  end while
end for
```

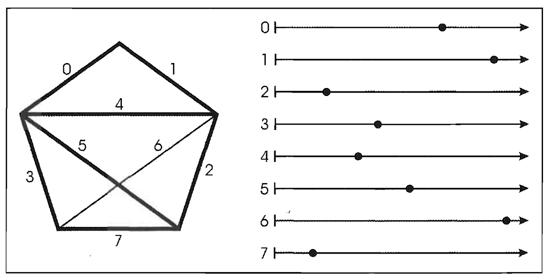
In the simplest of terms, the solution was to create a *priority list*. Each dimension in a particle's position corresponds to the *priority* or *desirability* of a *potential edge*[1]. The closer the particle's position is to the *origin* (i.e. to zero) within a given dimension, the more desirable the corresponding edge is. That is, if a particle's position in two dimensions, i and j, have values of, for example, 23.1 and 13.2, then the edge corresponding to dimension j would be added *before* the edge corresponding to dimension i.

The entirety of the transcription is shown in Algorithm 6. Note that, for a given position transcription, the constraints will be checked multiple times. This is why the bounded-ring test needed to be more efficient than the naïve approach. A simple transcription can be seen for two particle positions in Figure 3.1. Each ray in the figure represents one dimension. The black dot indicates the particle's position within that dimension. In this example, no numbers are provided. This is because it is only

[1]A *potential edge* is an edge that would be in a fully-connected network, but which has not already been precluded during the *preprocessing* stage. Refer to section 2.4 to see how these edges are identified.



(a)



(b)

Figure 3.1: Particle Transcription Illustration

the *relative* distances from the origin that matter. In Figure 3.1a, the edges are added in the order [7,2,4,3,5,0,6,1], resulting in a network containing all allowable edges. In Figure 3.1b, however, the algorithm stops after edge 1.

In general, there are two observations worth noting about this algorithm:

1. Since it stops once both constraints are satisfied, not all edges will necessarily be included.

2. If the position in one dimension increases, or the position in another decreases, then the effective priorities of two edges will be 'swapped'.

Of course, the former is an obviously desirable scenario. If the best solutions always included all edges, then there would be nothing to optimize. The important points are that the algorithm terminates at a reasonable time, and that it is a reasonable way to use a fixed number of dimensions to represent a variable number of edges. The latter is arguably far more significant. It provides a mechanism by which the status of one edge's dimension actually *can* affect the status of another edge in the transcribed solution. That is, it reclaims at least a portion of the flexibility of genetic operators. As such, a basic limitation of particle swarms with direct transcriptions can be avoided.

3.2 Preliminary Experiments

A first set of experiments was run to gauge the efficacy of the basic system. Since the results were to be compared against previous work by Fortz [15], Ventresca and Ombuki [17], the same datasets used in their work were also used here. Specifically, they were sets originally randomly generated by Bernard Fortz for his PhD thesis [2]. They all consist of some number of points (vertices) distributed in a two-dimensional plane, with coordinates in each axis ranging from 1 to 250. They are organized according to the numbers of vertices used (10, 20, 30, 40, or 50), with five instances of each vertex size classification. Furthermore, each dataset can be solved with more than one k-bound. Different datasets have different allowable k-bounds, which are included in the tables in Section 3.3 below. All of the k-bounds provided allow for at least one legal solution, so this system is guaranteed to find *some* solution[2], however efficient or inefficient it may be.

Problems with 10 vertices, particularly with the smallest k-bound, can be considered 'easy', since they are small enough to even be solved optimally by any number of different heuristics in reasonable time. Once the problem size grows to 30 vertices, particularly with larger k-bounds, it becomes substantially more difficult for the PSO. Such a problem may have up to 435 allowable edges, and well over 8×10^{130} possible networks through which to search. This was the target level of difficulty for making PSO competitive with other metaheuristics.

Problems with 40 or 50 vertices are extremely challenging, and represent an immense search space. Since they represent a new plateau of difficulty, and take substantially longer to process, these problems were included in a more limited form in the preliminary experiments, in that only ten runs were performed per experiment.

When initializing a particle swarm optimization system, there are some essential parameters and behavioural decisions which must be defined prior to starting. These include:

- *Maximum per-dimension position* (X_i^{max}): Each particle may only travel up to a certain distance within a dimension. If this value is exceeded, the particle simply *reflects* back within that dimension, with equal speed, in the opposite direction.

- *Maximum per-dimension velocity* (V_i^{max}): The velocity of a particle within any given dimension is capped, so that the rate of change (i.e. learning) may be controlled.

---

[2]Since only inherently illegal edges are dismissed, and since the problem itself is legal, the worst-case scenario is to simply accept *all* allowable edges, which is the guaranteed behaviour of the PSO transcription mechanism if it does not first find a 'better' solution.

- *# of iterations*: The higher this number, the more likely the system is to find a 'good' result. However, increasing it also increases computation time.

- *# of runs*: In order to know if the results from an experiment were reliable, it must be repeated for some number of *runs*. Thus, after some reasonable number of executions, both the *best* and *average* results can give insight into the system's efficacy.

- *Swarm Size*: A particle *swarm* consists of multiple particles, each with its own solution. A large swarm requires more computing time, but also may take longer to converge, and can potentially explore a wider range of solutions simultaneously.

- *Momentum*: Each iteration, some portion of a particle's velocity is retained from the previous iteration. The momentum reflects this portion, with values approaching 1 indicating that more of the velocity will be kept.

- *Cognitive Multiplier (c_1)*: This influences the tendency of a particle to drift back towards its personal best solution found. Refer to section 2.3.2 for details.

- *Social Behaviour*: For the social component (refer to section 2.3.2), a particle may be attracted to the position corresponding to the best solution found by the entire swarm, or by its neighbours. As such, the system may use a global social component, or a neighbourhood function, respectively.

- *Social Multiplier (c_2)*: This influences the tendency of a particle to drift towards the best solution found by some other particle, defined by the social component.

- *Explorative Multiplier (c_3)*: The tendency to follow a random vector, when included at all.

The best parameters for a vanilla PSO were determined empirically, and Table 3.1 reflects the experimental parameters for the best results obtained.

The maximum position (X_i^{max}) was set at 100,000, but this was somewhat arbitrary. Early tests showed that different values had little effect on the results. This is also to be expected, since it is only the *relative* per-dimension positions of a particle that influence the transcribed solution.

3.3 Results and Discussion

The results of the performed experiments are listed in tables 3.2 and 3.4. For each k-bound of each dataset, the best solution found in any run of that experiment is

Table 3.1: Parameters for Preliminary Experiments

Parameter	Value	Notes
X_i^{max}	100,000	Different values had little effect
V_i^{max}	5,000	
# of iterations	2,000	A traditionally common value and 'reasonable'
# of runs	20,10	20 for 10-30 vertex problems; 10 for 40-50 vertices
Swarm Size	200	
Momentum	0.3	This was the hardest to establish reliably
c_1	2	Traditional value, but also worked best
Social Behaviour	Global	Best for testing standard 'vanilla' system
c_2	2	Traditional value, but also worked best
c_3	1	Needed to be used sparingly

listed, as well as the average of the best results of each run. Tables 3.3 and 3.5 show how the particle swarm fared when compared to past works[3]. Values that are *italicized* indicate that the PSO matched or beat the results of the Stingy algorithm. Figure 3.2 shows the best solution costs and swarm average costs for the best run, for the 30-1 dataset with a k-bound of 200. A plot of that same best solution found by the algorithm for the same problem is shown in Figure 3.3a, and a plot of the worst solution found in the same experiment for the same problem is shown in Figure 3.3b. Notice that, in Figure 3.3a, there is quite a bit of 'overlapping' of rings. That is, if such a network were to be constructed with, say, cables, then those cables would crisscross each other several times. This would also intuitively be a sign of inefficiency. Of course, this pattern is far more prevalent in Figure 3.3b, which also clearly shows a great deal of redundancy.

Figure 3.4 consolidates the average performance of the PSO compared to the other techniques. The datasets and bounds were organized according to their numbers of allowable edges. Since the dimensionality of the problem dictates a problem's difficulty for this PSO algorithm, that was a more fair criteria for division than the number of vertices. Specifically, the sets were divided into four classes: $1 - 125 edges$ (containing 32 instances), $126 - 300 edges$ (containing 35 instances), $301 - 450 edges$ (containing 31 instances), and the remainders (containing 30 instances). For each class, the performance was measured by comparing the best results found by each

[3]Note: the values included are for a Stingy and Tabu search from Bernard Fortz's work [2][15], as well as for a GA from Mario Ventresca and Beatrice Ombuki-Berman's work [11]. However, this particular listing of values is taken from the latter, as it also included comparisons against Fortz's work.

technique against the PSO's best result. More precisely, the difference between the PSO's best result and another technique's result was then converted to a percentage. These percentages were averaged across all instances in that class.

$$score = \frac{\sum_{i=1}^{n} PSO_i^{best} - technique_i^{best}}{n} \times 100, \; where n = \# \; of \; instances$$

As such, the lower the calculated value, the better the PSO performed (plotted values below 0% would indicate that, on average, the PSO beat another technique for that class of problems).

It is readily apparent that this technique was certainly not competitive with past metaheuristic results, or even the Stingy algorithm. It was not able to solve the simplest (10-vertex) problems reliably enough, and fared far too poorly for the moderate to hard (30-vertex) problems. Indeed, it could arguably be considered a failure. The results were examined to identify flaws with the system, in the hope of exploring how the results could be improved.

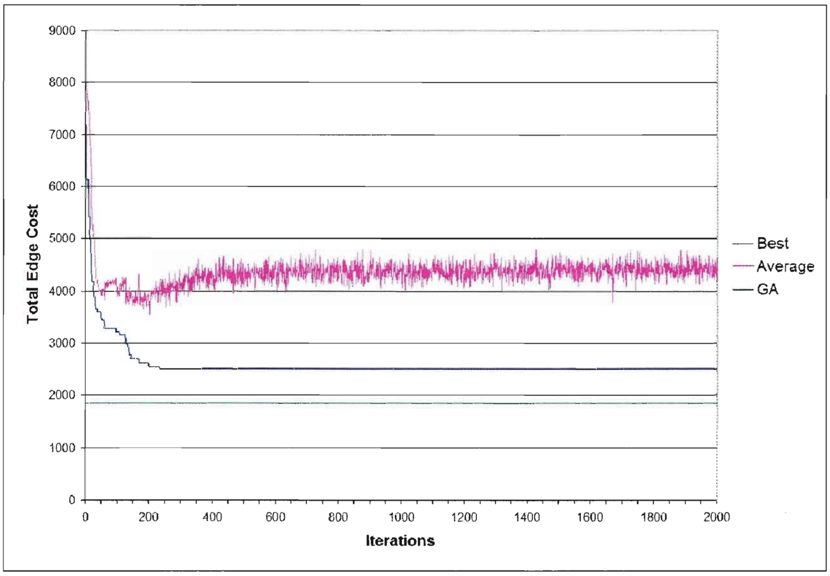
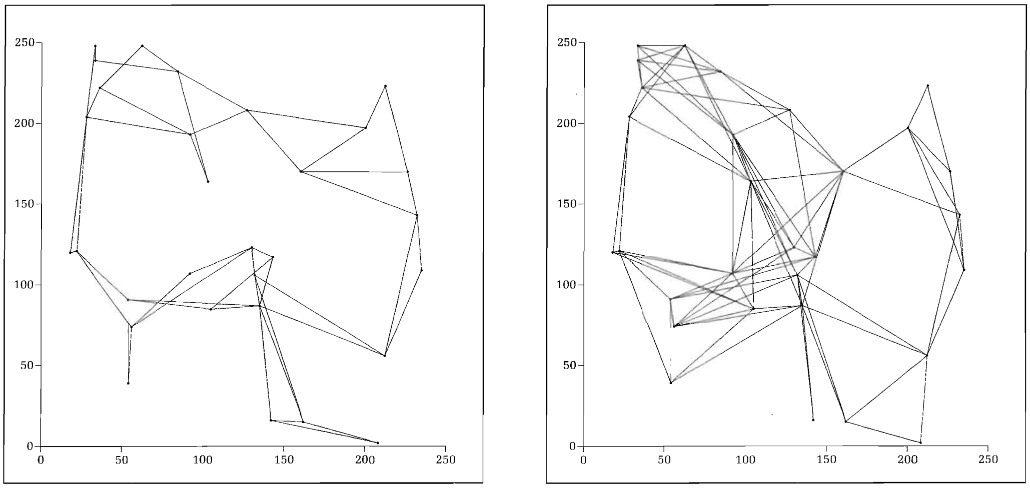
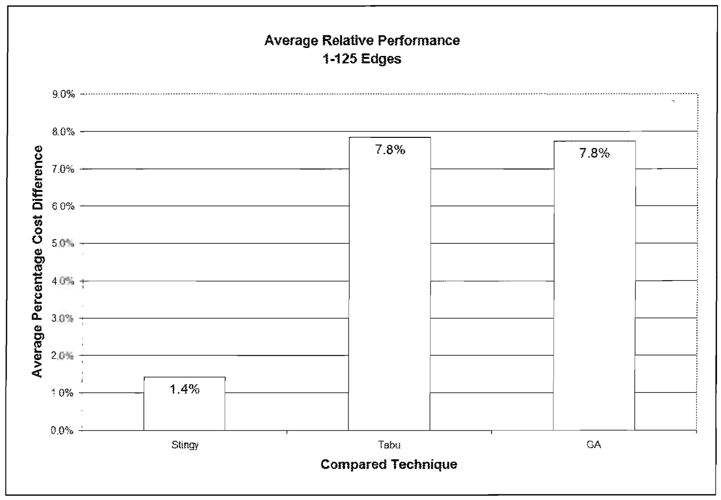


Figure 3.2: Preliminary Result: Training Curve for Best 30-1(200) Solution

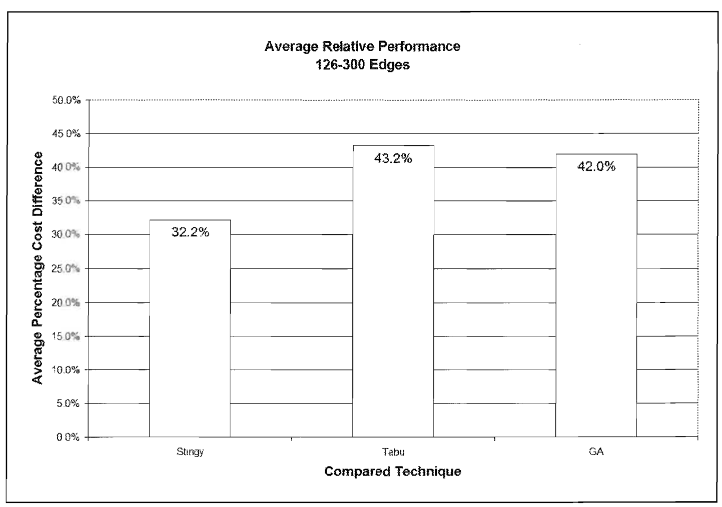


(a) Best Solution (b) Worst Solution

Figure 3.3: Preliminary Results: Best and Worst Solutions for 30-1(200) Plotted

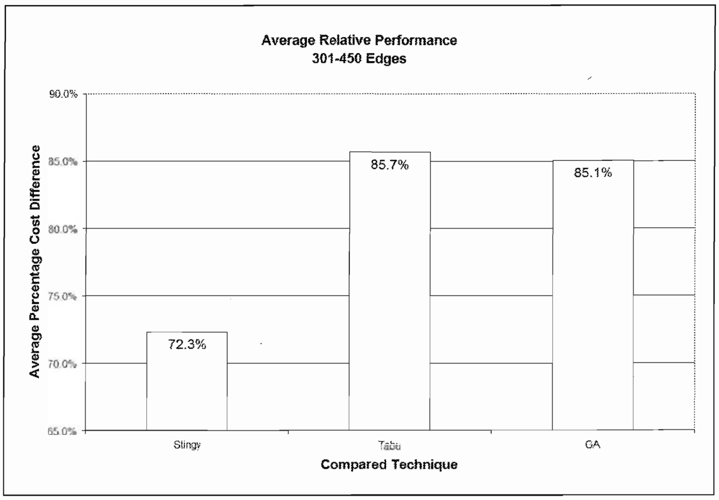


(a) 1-125 Edges

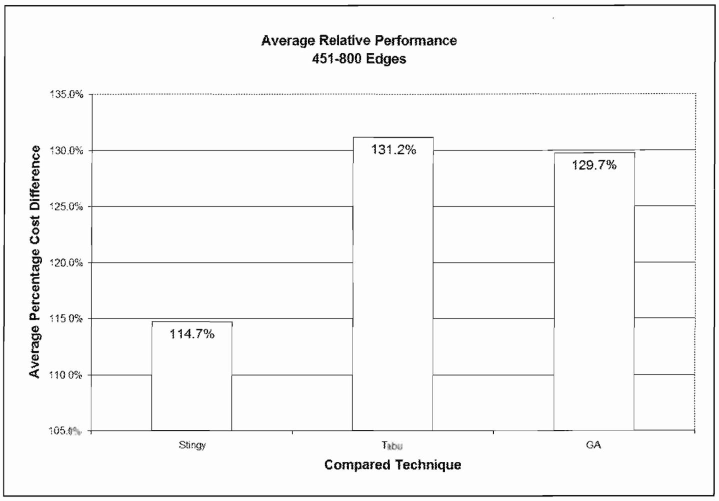


(b) 126-300 Edges

Figure 3.4: Preliminary Results: Comparison of Averages of Best Costs



(c) 301-450 Edges



(d) 451-800 Edges

Figure 3.4: Preliminary Results: Comparison of Averages of Best Costs (continued)

Table 3.2: Preliminary Results: Best and Average Cost Lengths

Data	K	Best	Average	Std. Dev.	Data	K	Best	Average	Std. Dev.
10-1	350	906	987.4	70.5	20-4	200	1971	2329	269.6
	400	934	1013.4	50.3		250	1822	2263.4	363.3
	450	896	1021.1	100.7		300	1822	2274.9	370.3
	500	919	1003.6	77.9		350	1653	2062.3	190.7
10-2	400	1146	1211	54.5		400	1682	2124	323.2
	450	1086	1251.2	117.9		450	1749	2106.2	230
	500	1093	1188.1	61.7		500	1737	2127.4	235.9
10-3	300	1269	1329.3	56.1	20-5	300	1612	2089.7	285.1
	350	1018	1113.1	89.9		350	1631	2168	399.2
	400	1058	1153	66		400	1673	2157.5	253.3
	450	952	1163.2	101		450	1640	2082.6	259.8
	500	988	1130.6	92		500	1676	2153.1	255
10-4	300	1391	1409.5	22.7	30-1	200	2502	3150.5	813.9
	350	1205	1323.8	91.8		250	2350	3422.8	711.6
	400	1135	1234.4	74.8		300	2357	3190.3	481.1
	450	980	1192.2	129.1		350	2400	3211.7	465
	500	1031	1121.4	87.6		400	2500	3196.9	404
10-5	350	1383	1431.7	54.3		450	2803	3303.1	346.8
	400	1260	1370.6	94.7		500	2416	3199.3	430.1
	450	1158	1281.9	72.3	30-2	300	2047	2783.6	365.9
	500	1106	1260.4	99.9		350	2225	2812.3	314.8
20-1	200	1645	1842	138.4		400	2601	3131.3	385
	250	1631	2111.2	424		450	2523	3115.4	385.9
	300	1786	2125.6	218.7		500	2671	3071.7	288.9
	350	1629	2195.4	424.8	30-3	250	2153	2901.1	464.1
	400	1714	2163.7	247		300	2266	2977.1	311.3
	450	1686	2189.2	215.1		350	2225	2964.4	536.1
	500	1541	2118.3	284.9		400	2173	2689.4	337.2
20-2	200	1646	1928	235		450	2457	2852.7	285.2
	250	1383	1769.1	272		500	2292	2733.5	320.9
	300	1422	1795	273.3	30-4	200	1962	2522.5	284.3
	350	1347	1813.3	272.1		250	2232	2779.9	285.1
	400	1351	1814.2	260.8		300	2082	2760.2	335.4
	450	1304	1645.4	249.2		350	2113	2851.8	401
	500	1375	1776.1	228.5		400	2225	2682.3	300.5
20-3	200	1716	1920.8	165		450	1825	2759	439.8
	250	1402	1636.3	160.2		500	2017	2669.4	346.2
	350	1414	1957.8	366.9	30-5	200	2677	3115.4	464.1
	400	1553	1967.6	266.4		250	2869	3507.3	400.6
	450	1413	2022.9	350.8		300	2671	3394	478.5
	500	1456	1828.9	248.3		350	2534	3335.9	370.7
						400	3000	3538.5	359.8
						450	2716	3688.9	557.3
						500	2853	3585.9	356.7

Table 3.3: Preliminary Results: Comparisons of Cost Lengths

Data	K	BB	Stingy	Tabu	GA	PSO	Data	K	BB	Stingy	Tabu	GA	PSO
10-1	350	906	906	906	906	***906***	20-4	200	1958	1962	1958	1958	1971
	400	898	976	898	898	*934*		250	1454	1524	1460	1460	1822
	450	896	955	898	896	***896***		300	1286	1442	1286	1286	1822
	500	854	880	854	854	919		350	1235	1334	1235	1235	1653
10-2	400	1140	1154	1140	1140	*1146*		400	1190	1259	1190	1205	1682
	450	1062	1062	1062	1062	1086		450	1164	1213	1164	1164	1749
	500	1031	1062	1031	1031	1093		500	1149	1213	1174	1164	1737
10-3	300	1269	1433	1269	1269	***1269***	20-5	300	1324	1419	1339	1332	1612
	350	1018	1089	1018	1018	***1018***		350	1251	1355	1251	1251	1631
	400	954	1025	954	954	1058		400	1211	1300	1213	1239	1673
	450	952	1025	952	952	***952***		450	1119	1162	1125	1119	1640
	500	952	1007	952	952	*988*		500	1082	1125	1082	1082	1676
10-4	300	1391	1445	1391	1391	***1391***	30-1	200	1726	1963	1898	1853	2502
	350	1205	1256	1205	1205	***1205***		250	1477	1689	1612	1655	2350
	400	1117	1186	1117	1117	*1135*		300	1413	1657	1474	1557	2357
	450	980	1149	980	980	***980***		350	1328	1622	1366	1406	2400
	500	980	1142	980	980	*1031*		400	1295	1464	1358	1333	2500
10-5	350	1383	1457	1383	1383	***1383***		450	1240	1382	1240	1248	2803
	400	1238	1456	1238	1238	*1260*		500	1213	1322	1221	1213	2416
	450	1143	1292	1143	1143	*1158*	30-2	300	1477	1607	1507	1488	2047
	500	1072	1213	1072	1072	*1106*		350	1381	1461	1386	1386	2225
20-1	200	1577	1859	1819	1577	***1645***		400	1319	1425	1322	1319	2601
	250	1445	1501	1455	1445	1631		450	1319	1386	1347	1319	2523
	300	1376	1430	1376	1383	1786		500	1295	1386	1302	1302	2671
	350	1253	1459	1253	1253	1629	30-3	250	1411	1711	1574	1613	2153
	400	1183	1416	1183	1183	1714		300	1325	1526	1329	1325	2266
	450	1144	1266	1144	1144	1686		350	1239	1250	1239	1239	2225
	500	1111	1185	1111	1111	1541		400	1198	1349	1198	1198	2173
20-2	200	1325	1360	1329	1390	1646		450	1152	1160	1152	1152	2457
	250	1094	1166	1098	1104	1383		500	1128	1151	1128	1128	2292
	300	984	1065	990	996	1422	30-4	200	1448	1612	1549	1577	1962
	350	953	974	953	953	1347		250	1250	1547	1327	1382	2232
	400	940	974	946	940	1351		300	1164	1484	1316	1391	2082
	450	919	959	932	929	1304		350	1134	1470	1143	1152	2113
	500	900	925	900	917	1375		400	1050	1215	1068	1068	2225
20-3	200	1449	1512	1449	1610	1716		450	1044	1206	1044	1097	1825
	250	1218	1272	1218	1218	1402		500	1044	1223	1079	1064	2017
	350	1100	1266	1100	1138	1414	30-5	200	2056	2282	2156	2080	2677
	400	1100	1239	1100	1103	1553		250	1759	2195	1915	1915	2869
	450	1100	1228	1101	1104	1413		300	1635	2097	1750	1742	2671
	500	1011	1219	1011	1011	1456		350	1562	1668	1562	1562	2534
								400	1493	1578	1497	1497	3000
								450	1452	1510	1459	1452	2716
								500	1424	1498	1425	1424	2853

Table 3.4: Additional Preliminary Results: Best and Average Cost Lengths

Data	K	Best	Average	Std. Dev.	Instance	K	Best	Average	Std. Dev.
40-1	200	3339	3958.3	673.1	40-4	200	3240	4203.9	824.2
	250	3763	4511.8	421.5		250	3268	4001.9	410.6
	300	3860	4501.4	421		300	3360	4030.8	469.8
	350	3678	4565	665.3		350	3655	4203.9	309.4
	400	4135	4913.7	526.7		400	4113	4714	515.6
	450	3602	4573.8	500.2		450	3514	4580.9	692
	500	3628	4670	644.9		500	3431	4350.8	559.2
40-2	300	3530	3903.4	324	40-5	200	2847	3336.5	386.9
	350	3592	4520.4	598.5		250	2850	3427.7	381.8
	400	3681	4542.2	549.2		300	3572	4006.2	363.9
	450	3315	4275.9	523.3		350	3101	3956.4	572.8
	500	3318	4358.1	729.2		400	3227	4032.4	498
40-3	200	3477	4557.5	1110.3		450	3177	4084.3	720.7
	250	3597	4203.6	532		500	3551	4113.8	379.5
	300	3617	4355.4	690.3	50-1	150	3584	3869.5	264.7
	350	4407	4823.4	352.6		200	3261	4369.6	547
	400	4046	4873.1	603.5	50-2	250	4434	5187	419.9
	450	4048	4818.7	594.6		300	4324	5217.4	611.9
	500	4245	5023.7	450.3	50-3	200	3898	4625	470.7
						250	4351	4945.9	272.3
					50-4	200	3743	4327.3	392.9
						250	4564	5291.6	668.2
					50-5	200	3667	4418.4	750.5
						250	3925	5075.3	667.1

Table 3.5: Additional Preliminary Results: Comparisons of Cost Lengths

Data	K	BB	Stingy	Tabu	GA	PSO	Data	K	BB	Stingy	Tabu	GA	PSO
40-1	200	2067	2549	2232	2320	3339	40-4	200	1894	2159	2069	2163	3240
	250	1800	2261	2031	2030	3763		250	1718	1877	1791	1795	3268
	300	1687	2213	1947	1854	3860		300	1610	1768	1706	1688	3360
	350	1616	1998	1691	1733	3678		350	1551	1664	1616	1616	3655
	400	1558	1699	1609	1627	4135		400	1503	1603	1552	1572	4113
	450	1533	1767	1571	1697	3602		450	1476	1590	1524	1536	3514
	500	1520	1751	1537	1589	3628		500	1458	1511	1492	1475	3431
40-2	300	1558	1737	1621	1617	3530	40-5	200	1626	1747	1720	1703	2847
	350	1496	1592	1514	1527	3592		250	1455	1727	1657	1676	2850
	400	1459	1544	1477	1535	3681		300	1393	1712	1607	1544	3572
	450	1434	1540	1462	1462	3315		350	1356	1699	1575	1607	3101
	500	1416	1505	1422	1416	3318		400	1315	1699	1546	1546	3227
40-3	200	2031	2421	2317	2355	3477		450	1266	1699	1422	1438	3177
	250	1821	2146	2077	2077	3597		500	1246	1699	1291	1246	3551
	300	1688	1897	1815	1815	3617	50-1	150	2165	2367	2250	2286	3584
	350	1620	1747	1654	1694	4407		200	1776	2036	1968	1968	3261
	400	1582	1649	1611	1649	4046	50-2	250	1884	2267	2200	2194	4434
	450	1561	1622	1575	1561	4048		300	1772	1914	1869	1881	4324
	500	1539	1622	1576	1539	4245	50-3	200	1877	2236	2053	2116	3898
								250	1777	2073	1896	1957	4351
							50-4	200	1852	2183	2090	2105	3743
								250	1709	2105	1822	1822	4564
							50-5	200	1777	2155	1960	2084	3667
								250	1650	1890	1835	1835	3925

3.4 Improving Effectiveness

The initial results were carefully studied, as was the actual particle swarm system itself, in the hope of finding areas for improvement. After some of those areas were found, modifications were made to the design in order to improve the capabilities of the system.

3.4.1 Analysis of Shortcomings

When examining the raw data, certain trends became apparent. The recurrent theme was *premature convergence*. The solutions tended to settle into local minima too quickly, and were then hesitant to explore new solutions. As part of this, the particles seemed to cluster together into the *same* solution. This was taken to be an indication of both excessive exploitation of individual past results, as well as excessive exploitation of swarm knowledge. As such, the solution would need to address how to limit both of these forms of exploitation.

Additionally, it seemed difficult to pin down a momentum (ω) that was 'best'. Numerous values were attempted, but none seemed to offer a consistent advantage. Additional consideration was given to this problem as well, as shown below.

3.4.2 Supersocial Particles

The first problem to address was the tendency to excessively exploit individual best past results. This behaviour is represented by the *cognitive* function of the velocity update rule. One option considered was reducing the cognitive parameter, the multiplier c_1. This improved results slightly, but what seemed to work better was to take this approach to its ultimate extent: to set it to 0, thus removing the cognitive function entirely. This means that, in the absence of a cognitive function, the only past knowledge that the particles would only be able to rely on would be that of neighbours. In theory, it likely would have been an option to reduce *either* form of exploitation, but some social behaviour was deemed necessary to prevent the algorithm from turning into nothing more than a parallel hill-climber. By removing the cognitive function, the particles are forced to rely primarily on social interaction, and are thus deemed *supersocial* in this thesis.

3.4.3 Neighbourhoods

Though it was decided that *supersocial* particles might be a good choice for the velocity update rule, that fact still did not change the concern over the particles clustering to each other (i.e. grouping to the same solution) too quickly. It was necessary to strike a balance between reducing the level of exploitation of swarm knowledge, but still retaining some component of the behaviour. The solution to this problem is far easier, as it is very common. As explained in chapter 2, section 2.3.3, a particle is not obligated to gravitate towards the position corresponding to the best solution found in the *swarm*. That is, the global social mechanism is not the only one available. Rather, it is permissible to use a *neighbourhood*, a smaller subset of the entire swarm that is associated with the particle being considered.

The idea is that, rather than having all the particles become immediately drawn towards a single globally-best solution, each particle will tend to be drawn towards a locally-best solution. The complementary explanation is that multiple different 'good' solutions will be attracting particles at the same time. Eventually, clustering will still occur. However, by distributing that attraction, it can prolong how long clustering will take, and in the meantime give the particles more opportunities to find different solutions before doing so.

3.4.4 Selecting a Momentum

As stated above, it was difficult to pinpoint an ideal momentum (ω) parameter. When the momentum was set very high, the particles were able to retain more of their velocities and avoid settling on solutions so quickly. However, this tended to prevent the ability to fine-tune solutions. On the other hand, a low momentum was very good for fine-tuning solutions, but tended to favour local minima. This is an example of the classic problem of *exploitation vs. exploration*. Should a system be permitted to coast broadly throughout the search space, in the hope of happening to stumble upon a good solution, or should it be forced to scrounge about within a smaller area, finding the best possible results within a smaller space? Obviously, neither solution was a reasonable option. The final decision was to attempt to somehow achieve both.

3.4.5 Variable Momentum

A technique that has been used with PSO for other problems in the past[22] is to simply have ω begin with a very high value, and slowly reduce over several iterations. The idea is to permit a period of vast, unfettered exploration, followed by a period of fine-tuning. This is very much analogous to Simulated Annealing, which is founded on this concept[4].

[4]In Simulated Annealing, the system starts with a high *temperature*, indicating that it is more prone to adopting other solutions that actually appear to be worse. Over time (iterations), the temperature *cools*, and the system becomes less and less likely to accept changes that appear 'bad' [37].

The only problem with implementing this type of behaviour is that, within a given run, there is only a single period of exploration and a single period of solution refinement. The obvious question is, what happens if it still has not found a decent solution? To address this solution, rather than implementing a momentum *curve*, I implemented a momentum *oscillation*.

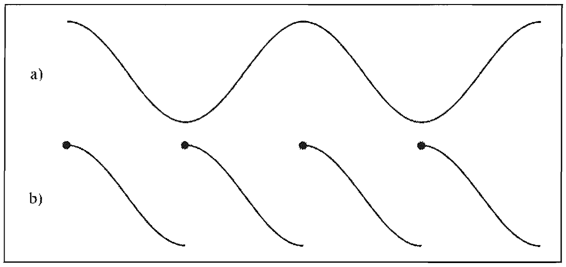


Figure 3.5: ω Scaling for Continuous Oscillation (a) and Pulsed Oscillation (b)

Continuous Momentum Oscillation

To allow for a behaviour that sometimes permitted the particles to accelerate, and sometimes forced them to decelerate, the momentum was patterned on a continuous cosine wave (refer to Figure 3.5a). The goal was to allow for multiple periods of alternating exploration and refinement. A new user-specified parameter was added: the wave's period. In doing so, I was able to choose the duration of the acceleration and deceleration phases. Similarly, I was able to implicitly choose how many such sessions would occur within a run.

The system's *iteration* (akin to a *generation* in several other techniques) is used as the 'time' unit for the progression of the wave. The wave is shifted and scaled to oscillate between zero and some user-defined maximum value (inclusive). The precise formula is:

$$\omega_i = \frac{(cosine(\frac{2 \cdot \pi \cdot i}{L}) + 1)}{2} \cdot \omega_b$$

where

i	is the current iteration
ω_i	is the momentum in iteration i
ω_b	is the maximum momentum to reach
L	is the period of the cosine wave

The maximum momentum value (ω_b) was always 1 for this thesis, since a value greater than 1 would artificially super-accelerate the particles independent of swarm knowledge, and also tend to be far too disruptive to the corresponding solutions.

By continuously changing the momentum, the result was to continuously change the potential for velocity. The hope was that the velocity would make minor corrections when the momentum was very low, and have the ability to build up great speed when the momentum was high.

Pulsed Momentum Oscillation

The primary concern with the continuous momentum oscillation approach was that, while although the top of the ω arc *allows* the velocity to build up very quickly, there is nothing guaranteeing that this will actually happen. If the particle's position is mired in what is believed to correspond to a 'good' solution, then it is only the random component of the velocity update rule that might pull the particle out for the purpose of exploration. I decided to also implement an alternate oscillation mechanism to allay this concern, and then to compare and see how each performed.

For the alternate mechanism, *pulsed ω oscillation*, only a portion of the cosine wave was used. Specifically, the descending portion was repeated. Furthermore, at the initial peak of each cycle, the velocities of the particles are reset; thus affording them an opportunity to explore new venues. It is important to note that it was the velocities that were repeatedly reset. If, instead, the particle positions had been constantly reset, then all of the progress achieved in each cycle would have been lost. However, by resetting only the *velocities*, the states of the solutions were retained; and it was only their directions and inclination to move which were affected.

3.5 Oscillating PSO Experimental Setups

The new experimental setups incorporated all of the changes detailed above. The swarms were switched to a cognitive-free, or supersocial, velocity update rule. The global social function was replaced with a neighbourhood function. And, both versions of oscillating momentum were attempted. After several preliminary runs, it was discovered that the two oscillation behaviours performed best with identical parameters, which are detailed below. However, there were first some basic elements that had to be defined even before anything else was empirically derived.

The datasets used were the same as those referred to in section 3.2, created by Bernard Fortz in his PhD thesis work[2]. For the same datasets used in the non-oscillating setup—10, 20, and 30 vertices—20 runs were performed for each experiment, for each oscillation style. As with the preliminary experiments, an additional batch of 10-run experiments were performed on the 40-vertex and 50-vertex datasets. The number of iterations was again set at 2,000 for all experiments, to allow direct comparison with the non-oscillating version. Additional parameters are found in Table 3.6.

Table 3.6: Parameters for Oscillating Experiments

Parameter	Value	Notes
X_i^{max}	100,000	
V_i^{max}	5,000	
Swarm Size	200	
Period	200	200 iterations per full oscillation
c_1	0	No cognitive behaviour
Social Behaviour	Neighbourhood	
Neighbourhood Size	12	
c_2	3	
c_3	2	

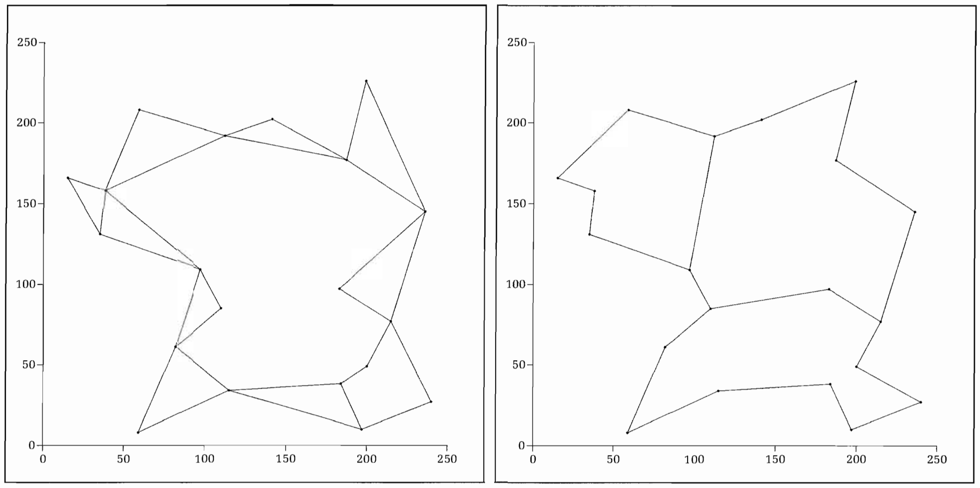
3.6 Oscillating PSO Results

Detailed below are the results of both experiment-sets. Some of these results have also been published in a paper co-authored by Dr. Beatrice Ombuki-Berman[25]. Tables 3.8, 3.10, 3.12, and 3.14 show the best (lowest) costs obtained by any particle in any run in the experiment, the average of the best cost per run for each experiment, and the sample standard deviation for each experiment.

Tables 3.9, 3.11, 3.13, and 3.15 show how the best results obtained by the PSO compare to the best results found by Fortz's Stingy algorithm, his Tabu search, and Ombuki and Ventresca's Genetic Algorithm. Values that are *italicized* indicate that the PSO matched or beat the results of the Stingy algorithm. Values that are **bold** indicate that the PSO matched or beat at least one metaheuristic (Tabu, GA, or both).

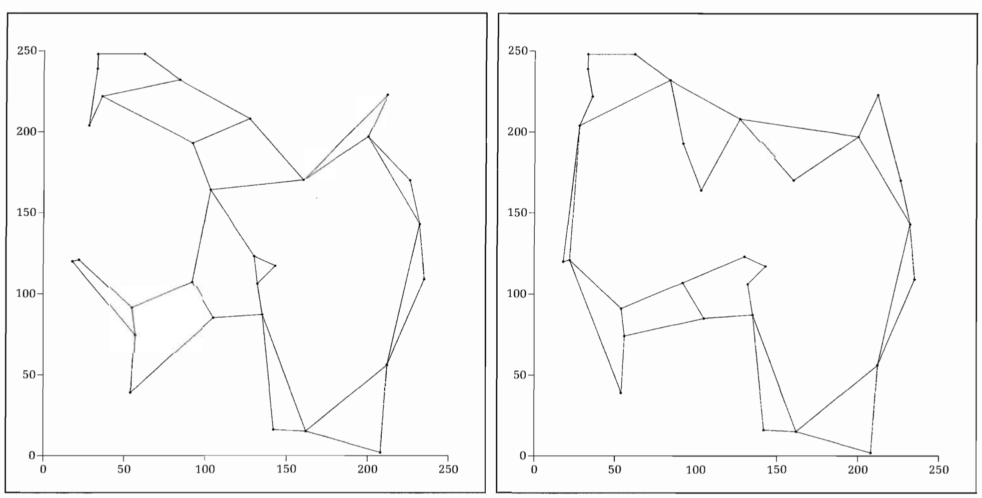
Figures 3.6a and 3.6b depict the optimum networks obtained for the 20-1 problem instance, with 20 vertices, using pulsed momentum oscillation. Figure 3.6a is for the problem using a k-bound of 200, and the network in Figure 3.6b has a k-bound of 500. As one would expect, as the bound increases, so does the flexibility for forming rings.

Figures 3.7a and 3.7b depict the results of using continuous and pulsed oscillation, respsectively, to solve problem instance 30-1, with 30 vertices, and a k-bound of 200.



(a) K-bound: 200 (b) K-bound: 500

Figure 3.6: Optimal Solutions for 20-1

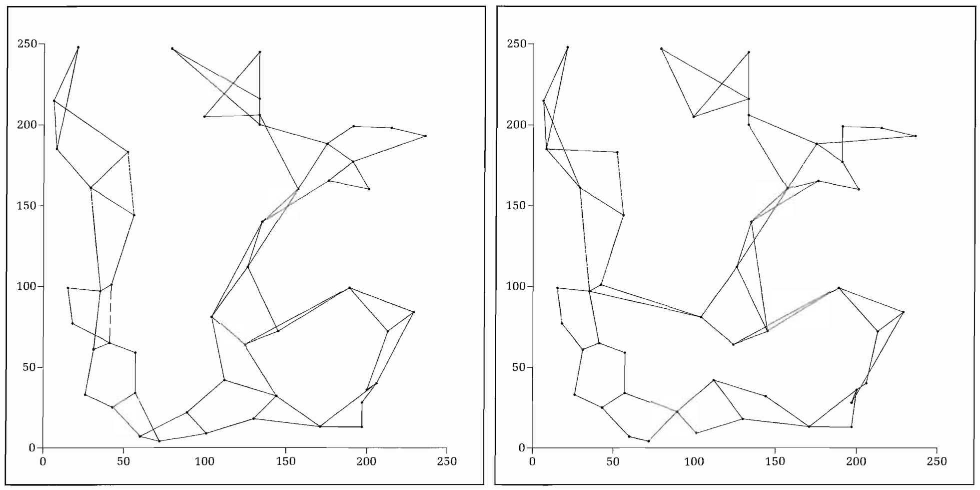


(a) Continuous Oscillation (b) Pulsed Oscillation

Figure 3.7: Best Solutions Found for 30-1(200) with Both Oscillations

Both networks represent new solutions to that problem, as far as metaheuristics are concerned, but they clearly ended up with significantly different solutions.

Similarly, 3.8a and 3.8b depict the results of using continuous and pulsed oscillation, but to solve the much more difficult problem instance of 50-1, with 50 vertices, and a k-bound of 150.



(a) Continuous Oscillation (b) Pulsed Oscillation

Figure 3.8: Best Solutions Found for 50-1(150) with Both Oscillations

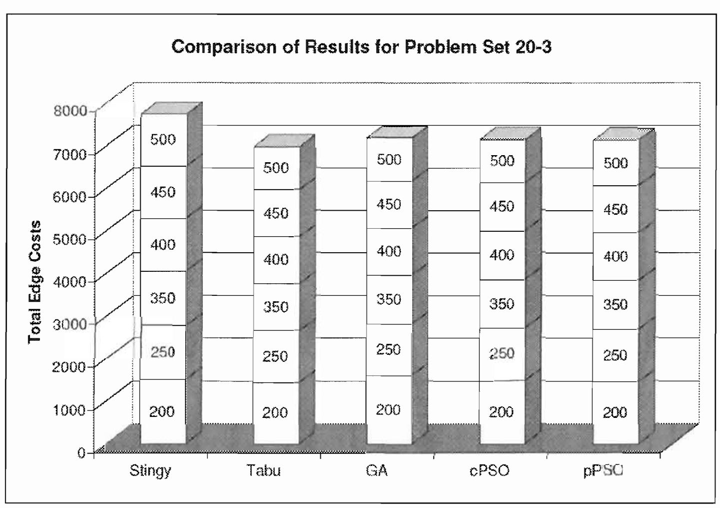


Figure 3.9: Comparison of Cumulative Performance for Problem 20-3

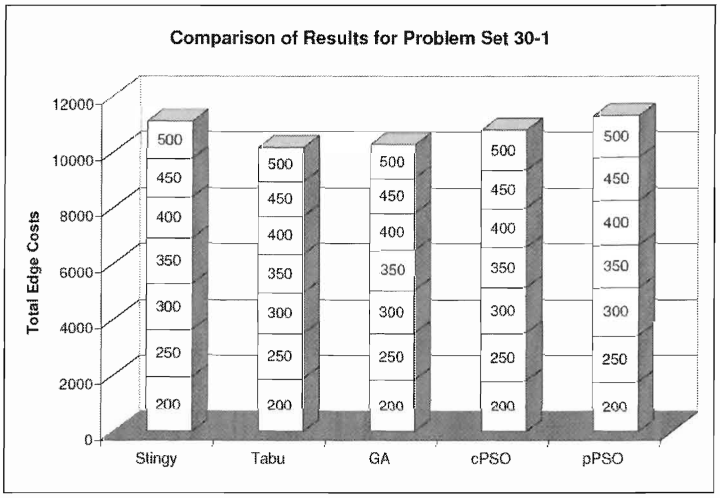


Figure 3.10: Comparison of Cumulative Performance for Problem 30-1

Figures 3.13 and 3.14 each consolidate the performances of the continuous and pulse oscillating PSOs, respectively, against the previously published results, and also against the non-oscillating PSO model. They were created in the same method as those found in section 3.3, and should be read in the same way. However, note that both oscillating models performed drastically better than the non-oscillating algorithm. While although both techniques performed the best overall for the first class (corresponding to the lowest numbers of edges), they were not able to beat Tabu or GA for the other three classes. Note, however, that the continuous oscillation model performed noticeably better than pulsed oscillation for the third and fourth classes. This implies that continuous oscillation is better suited to more difficult problems (or, at least, problems with higher dimensionality) than pulsed.

To further test this theory, statistical analysis was performed on the results of all continuously oscillating and pulse oscillating PSO results. An analysis of the data showed that, across all experiments performed, not all results conformed to a normal distribution. This was actually to be expected for at least some instances, as the oscillating PSO tended to frequently solve several of the simpler problems optimally. Since no results can be better than optimal, this meant that no results could fall below the optimality threshold. As such, symmetry of distribution was impossible for those instances, so long as the algorithm continued to perform well. Because the results could not be guaranteed to fall within a normal distribution, a nonparametric test was used instead. Specifically, a *Rank-sum test* [38] (also known as a *Wilcoxon Rank-sum test*, *Mann-Whitney U test*, or simply a *U test*) was used to determine if results from different oscillation models were actually from different populations. A confidence level of 95% was used for all tests.

Table 3.7: Statistical Comparison of Continuous and Pulsed Oscillation

	Tie	Cont. Won	Pulse Won
10's	20	0	1
20's	13	0	19
30's	10	20	2
40's	0	33	0
50's	1	9	0
Total	44	62	22

The first set of statistical tests performed compared oscillating particle swarms of both flavours against the non-oscillating version. In all 128 instances (25 datasets, with various different k-bounds), the improvement for oscillation was determined to be statistically significant. That is, within a 95% confidence level, the new supersocial, oscillating particle swarms (whether using continuous or pulsed oscillation) performed significantly better than the non-oscillating model.

The next set of tests were chosen to compare the two oscillation techniques against each other, to prove or disprove the previous theory. Table 3.7 confirms the suspicion that the continuous oscillation worked best overall. In particular, it showed that pulsed oscillation fared better primarily for relatively easy problems, and continuous oscillation performed better for harder problems.

The charts shown in Figures 3.9 and 3.10 represent the cumulative performances of the metaheuristics used. They graphically depict the costs of the best networks found for a given problem instance, across all supplied k-bounds. An interesting trend starts to become evident as the problem instances get more difficult.

Figure 3.9 shows the particle swarms (with both types of oscillation) performing comparably to Ombuki and Ventresca's Genetic Algorithm. However, even though the particle swarms did the best of all for the 30-1 dataset with the smallest k-bound, once the performance across all bounds were considered, it did not fare as well.

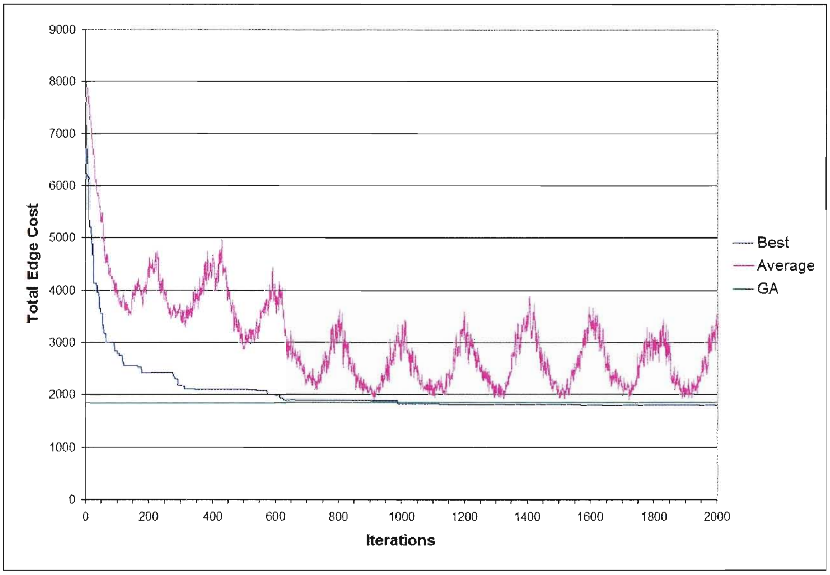


Figure 3.11: Continuous Oscillation: Training Curve for Best 30-1(200) Solution

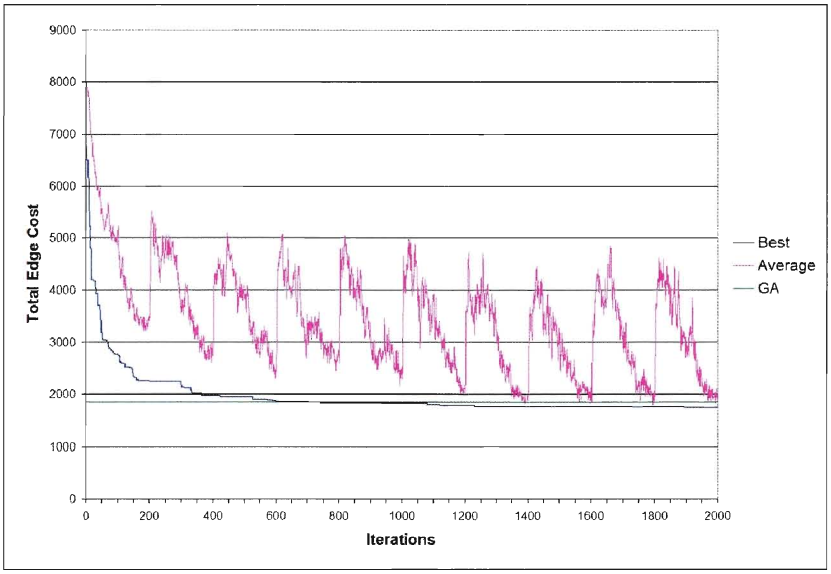
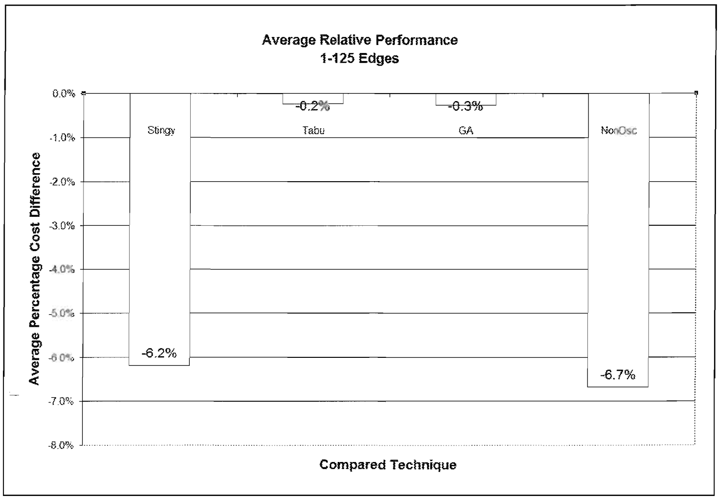
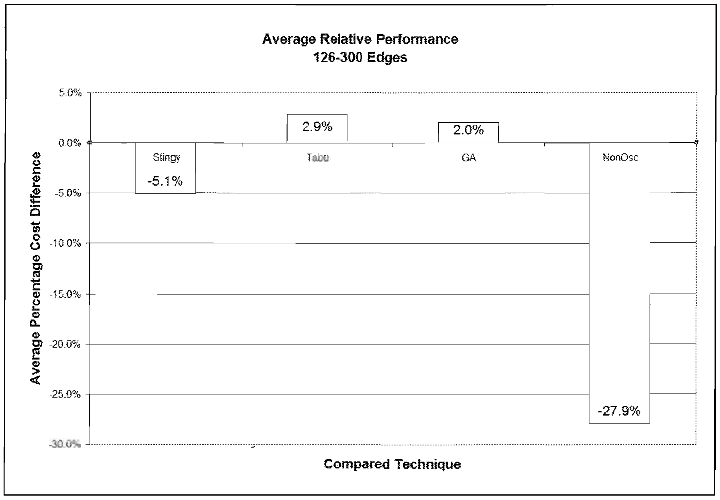


Figure 3.12: Pulsed Oscillation: Training Curve for Best 30-1(200) Solution

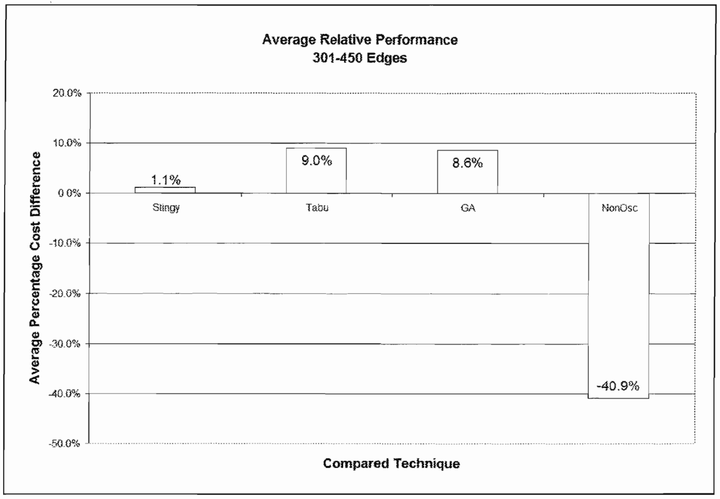


(a) 1-125 Edges

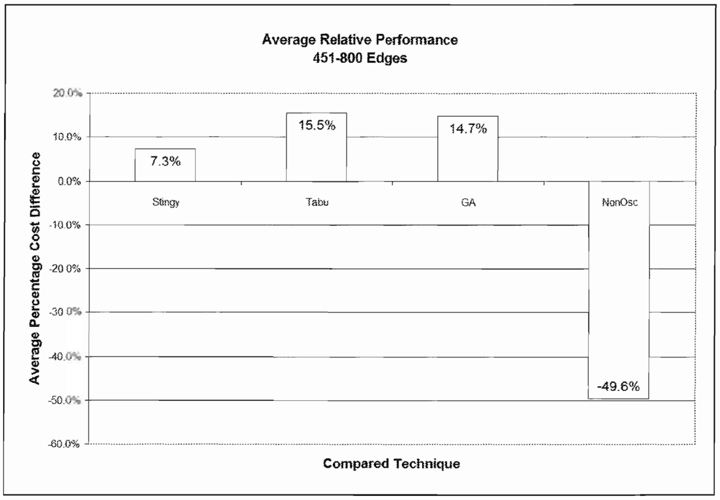


(b) 126-300 Edges

Figure 3.13: Continuous Oscillation Results: Comparison of Averages of Best Costs

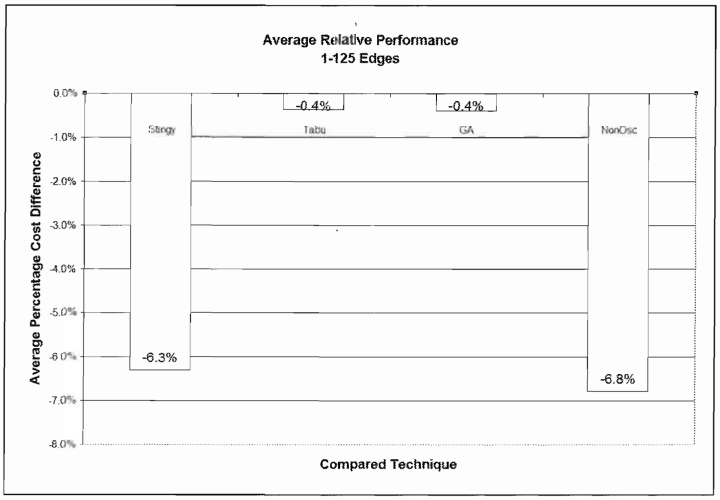


(c) 301-450 Edges

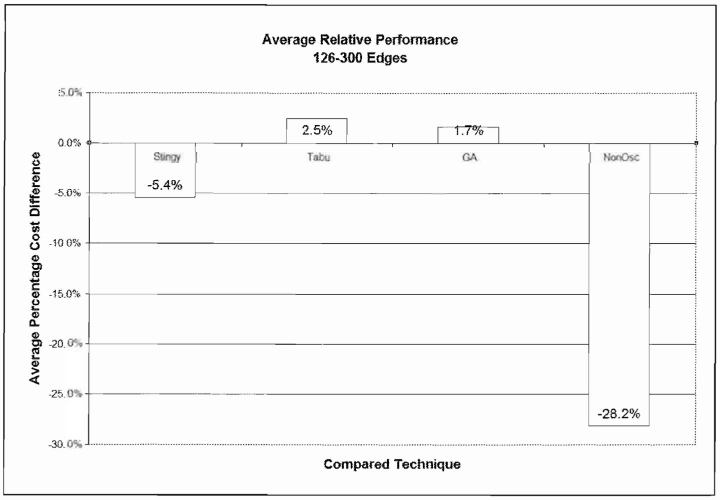


(d) 451-800 Edges

Figure 3.13: Continuous Oscillation Results: Comparison of Averages of Best Costs (continued)

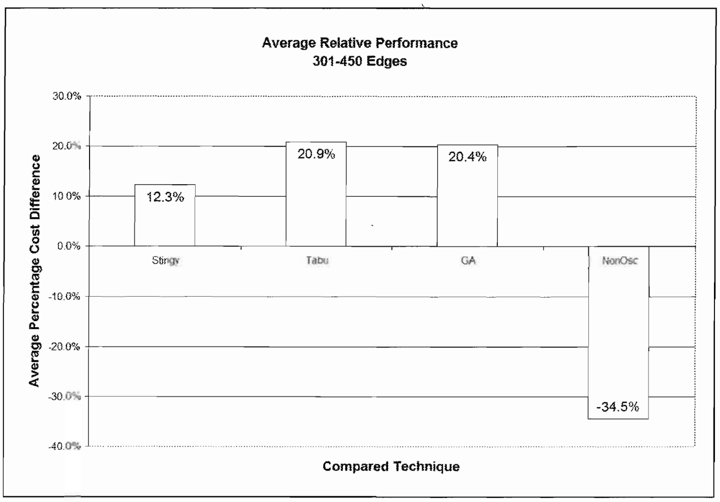


(a) 1-125 Edges

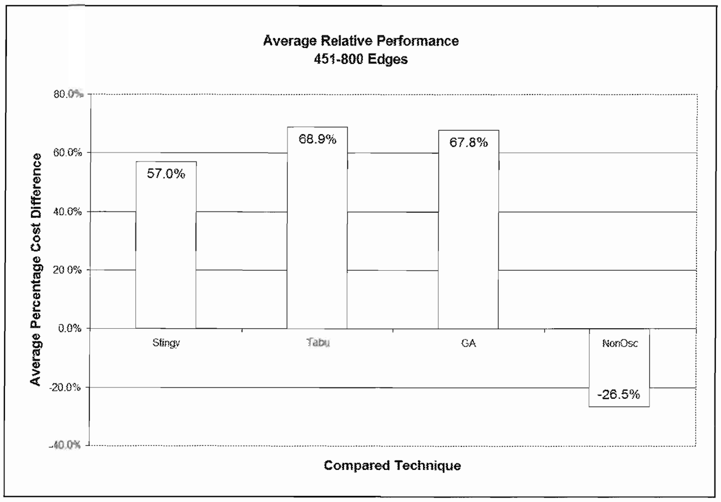


(b) 126-300 Edges

Figure 3.14: Pulsed Oscillation Results: Comparison of Averages of Best Costs



(c) 301-450 Edges



(d) 451-800 Edges

Figure 3.14: Pulsed Oscillation Results: Comparison of Averages of Best Costs (continued)

3.7 Oscillating PSO Discussion

The first thing that should be evident from the results shown in the tables above is that the modifications resulted in a drastic improvement.

Figures 3.11 and 3.12 show the training curves for the global best and swarm averages across their respective runs. Notice that the goal identified in section 3.4.5

Table 3.8: Continuous Oscillation Results: Best and Average Cost Lengths

Data	K	Best	Average	Std. Dev.	Data	K	Best	Average	Std. Dev.
10-1	350	906	907.3	4	20-4	200	1958	1961.5	4.1
	400	898	911.8	11.6		250	1454	1579.7	70.3
	450	896	910.5	15.4		300	1286	1372.4	55.6
	500	854	879.1	17		350	1248	1337.5	63.7
10-2	400	1140	1142.8	4.3		400	1233	1309.6	53.3
	450	1062	1070.4	18.3		450	1164	1262.5	56.5
	500	1031	1049.9	17.3		500	1208	1259.1	41.5
10-3	300	1269	1287.6	29.6	20-5	300	1331	1443.1	94.1
	350	1018	1021.5	15.7		350	1260	1361.4	93.1
	400	954	965.5	17.4		400	1214	1306.8	56.8
	450	952	974	32		450	1189	1269.8	58.6
	500	952	963.8	12.7		500	1088	1212.8	60.2
10-4	300	1391	1391	0	30-1	200	1791	1897.5	77.3
	350	1205	1216.4	18.3		250	1708	1830.8	104
	400	1117	1132.3	19.2		300	1618	1803.9	140.6
	450	980	999.7	35.2		350	1432	1644.2	121.7
	500	980	1007	28.2		400	1405	1586.8	116.2
10-5	350	1383	1386.8	5.6		450	1354	1503.5	74.7
	400	1238	1267.7	30.5		500	1421	1559.5	92.3
	450	1143	1158.8	17.3	30-2	300	1533	1643.1	90.4
	500	1072	1093.2	26		350	1470	1606	102.9
20-1	200	1577	1602.2	40.2		400	1515	1607.7	70.4
	250	1451	1489.3	34.7		450	1450	1570.2	93.5
	300	1416	1478.4	48.1		500	1414	1562.7	114.3
	350	1299	1397.5	68	30-3	250	1573	1735	117.8
	400	1183	1346.7	120.4		300	1554	1684.1	65.9
	450	1192	1266.7	53.2		350	1311	1422.2	77.1
	500	1111	1232.4	82.5		400	1315	1477.2	94.4
20-2	200	1329	1393.6	61.2		450	1218	1429.7	112.7
	250	1108	1145	38.8		500	1195	1366	107.4
	300	987	1024.5	23.4	30-4	200	1568	1663.9	54.1
	350	967	1015.1	38.3		250	1359	1560.8	74.7
	400	958	1017.9	45.4		300	1285	1423.3	114.3
	450	926	1002.7	43.1		350	1246	1328.1	67.7
	500	930	993.2	44.9		400	1159	1281.6	61.6
20-3	200	1474	1605.7	90.4		450	1123	1271.4	91.6
	250	1240	1285	58.1		500	1113	1263.3	79.1
	350	1138	1248.8	89.1	30-5	200	2216	2296.3	57.1
	400	1119	1222.4	64.8		250	1990	2109	103.2
	450	1125	1194.8	48.8		300	1790	1942.7	68.4
	500	1011	1129.2	47.6		350	1663	1834.7	107.7
						400	1578	1763.9	119.5
						450	1564	1723.3	101.1
						500	1509	1692.8	106

Table 3.9: Continuous Oscillation Results: Comparisons of Cost Lengths

Data	K	BB	Stingy	Tabu	GA	PSO
10-1	350	906	906	906	906	***906***
	400	898	976	898	898	***898***
	450	896	955	898	896	***896***
	500	854	880	854	854	***854***
10-2	400	1140	1154	1140	1140	***1140***
	450	1062	1062	1062	1062	***1062***
	500	1031	1062	1031	1031	***1031***
10-3	300	1269	1433	1269	1269	***1269***
	350	1018	1089	1018	1018	***1018***
	400	954	1025	954	954	***954***
	450	952	1025	952	952	***952***
	500	952	1007	952	952	***952***
10-4	300	1391	1445	1391	1391	***1391***
	350	1205	1256	1205	1205	***1205***
	400	1117	1186	1117	1117	***1117***
	450	980	1149	980	980	***980***
	500	980	1142	980	980	***980***
10-5	350	1383	1457	1383	1383	***1383***
	400	1238	1456	1238	1238	***1238***
	450	1143	1292	1143	1143	***1143***
	500	1072	1213	1072	1072	***1072***
20-1	200	1577	1859	1819	1577	***1577***
	250	1445	1501	1455	1445	***1451***
	300	1376	1430	1376	1383	*1416*
	350	1253	1459	1253	1253	*1299*
	400	1183	1416	1183	1183	***1183***
	450	1144	1266	1144	1144	*1192*
	500	1111	1185	1111	1111	***1111***
20-2	200	1325	1360	1329	1390	***1329***
	250	1094	1166	1098	1104	*1108*
	300	984	1065	990	996	***987***
	350	953	974	953	953	*967*
	400	940	974	946	940	*958*
	450	919	959	932	929	***926***
	500	900	925	900	917	930
20-3	200	1449	1512	1449	1610	***1474***
	250	1218	1272	1218	1218	*1240*
	350	1100	1266	1100	1138	***1138***
	400	1100	1239	1100	1103	*1119*
	450	1100	1228	1101	1104	*1125*
	500	1011	1219	1011	1011	***1011***
20-4	200	1958	1962	1958	1958	***1958***
	250	1454	1524	1460	1460	***1454***
	300	1286	1442	1286	1286	***1286***
	350	1235	1334	1235	1235	*1248*
	400	1190	1259	1190	1205	*1233*
	450	1164	1213	1164	1164	***1164***
	500	1149	1213	1174	1164	*1208*
20-5	300	1324	1419	1339	1332	***1331***
	350	1251	1355	1251	1251	*1260*
	400	1211	1300	1213	1239	***1214***
	450	1119	1162	1125	1119	1189
	500	1082	1125	1082	1082	*1088*
30-1	200	1726	1963	1898	1853	***1791***
	250	1477	1689	1612	1655	1708
	300	1413	1657	1474	1557	*1618*
	350	1328	1622	1366	1406	*1432*
	400	1295	1464	1358	1333	*1405*
	450	1240	1382	1240	1248	*1354*
	500	1213	1322	1221	1213	1421
30-2	300	1477	1607	1507	1488	*1533*
	350	1381	1461	1386	1386	1470
	400	1319	1425	1322	1319	1515
	450	1319	1386	1347	1319	1450
	500	1295	1386	1302	1302	1414
30-3	250	1411	1711	1574	1613	***1573***
	300	1325	1526	1329	1325	1554
	350	1239	1250	1239	1239	1311
	400	1198	1349	1198	1198	*1315*
	450	1152	1160	1152	1152	1218
	500	1128	1151	1128	1128	1195
30-4	200	1448	1612	1549	1577	***1568***
	250	1250	1547	1327	1382	***1359***
	300	1164	1484	1316	1391	***1285***
	350	1134	1470	1143	1152	*1246*
	400	1050	1215	1068	1068	*1159*
	450	1044	1206	1044	1097	*1123*
	500	1044	1223	1079	1064	*1113*
30-5	200	2056	2282	2156	2080	*2216*
	250	1759	2195	1915	1915	*1990*
	300	1635	2097	1750	1742	*1790*
	350	1562	1668	1562	1562	*1663*
	400	1493	1578	1497	1497	*1578*
	450	1452	1510	1459	1452	1564
	500	1424	1498	1425	1424	1509

Table 3.10: Additional Continuous Oscillation Results: Best and Average Cost Lengths

Data	K	Best	Average	Std. Dev.	Data	K	Best	Average	Std. Dev.
40-1	200	2492	2592.4	70	40-4	200	2254	2363.4	78.3
	250	2280	2423.4	100.3		250	2140	2299	95.8
	300	2071	2181.1	80.9		300	1872	1984.3	116.6
	350	1911	2058.7	83		350	1866	1973.6	97.8
	400	1749	2007.3	126.5		400	1800	1951.2	120.4
	450	1837	1993.1	94.2		450	1675	1909.9	144.8
	500	1865	2013.6	88.3		500	1811	1950.9	87
40-2	300	1855	2013.3	114.4	40-5	200	1850	1905.6	41.9
	350	1867	2010.5	134.4		250	1826	1943.9	92.8
	400	1686	1848.2	101		300	1651	1955.7	154
	450	1763	1877.7	105.9		350	1733	1895.1	147.4
	500	1763	1878.3	84.8		400	1667	1811	103.2
40-3	200	2528	2596.7	39.7		450	1656	1784.2	110.2
	250	2375	2562.5	148.4		500	1589	1725.7	125.9
	300	1974	2263.6	177.5	50-1	150	2498	2626.7	105.3
	350	1983	2054.1	67.4		200	2247	2409.9	139.7
	400	1864	1993.9	92.2	50-2	250	2375	2583.1	127.7
	450	1771	1985.8	111.1		300	2312	2461.2	135
	500	1747	2043.3	144.7	50-3	200	2386	2566.5	101.8
						250	2344	2569.7	147.1
					50-4	200	2293	2475.8	172.2
						250	2184	2331.6	128.7
					50-5	200	2292	2428.7	134.6
						250	2194	2359.6	148.9

Table 3.11: Additional Continuous Oscillation Results: Comparisons of Cost Lengths

Data	K	BB	Stingy	Tabu	GA	PSO
40-1	200	2067	2549	2232	2320	*2492*
	250	1800	2261	2031	2030	2280
	300	1687	2213	1947	1854	*2071*
	350	1616	1998	1691	1733	*1911*
	400	1558	1699	1609	1627	1749
	450	1533	1767	1571	1697	1837
	500	1520	1751	1537	1589	1865
40-2	300	1558	1737	1621	1617	1855
	350	1496	1592	1514	1527	1867
	400	1459	1544	1477	1535	1686
	450	1434	1540	1462	1462	1763
	500	1416	1505	1422	1416	1763
40-3	200	2031	2421	2317	2355	2528
	250	1821	2146	2077	2077	2375
	300	1688	1897	1815	1815	1974
	350	1620	1747	1654	1694	1983
	400	1582	1649	1611	1649	1864
	450	1561	1622	1575	1561	1771
	500	1539	1622	1576	1539	1747
40-4	200	1894	2159	2069	2163	2254
	250	1718	1877	1791	1795	2140
	300	1610	1768	1706	1688	1872
	350	1551	1664	1616	1616	1866
	400	1503	1603	1552	1572	1800
	450	1476	1590	1524	1536	1675
	500	1458	1511	1492	1475	1811
40-5	200	1626	1747	1720	1703	1850
	250	1455	1727	1657	1676	1826
	300	1393	1712	1607	1544	*1651*
	350	1356	1699	1575	1607	1733
	400	1315	1699	1546	1546	*1667*
	450	1266	1699	1422	1438	*1656*
	500	1246	1699	1291	1246	*1589*
50-1	150	2165	2367	2250	2286	2498
	200	1776	2036	1968	1968	2247
50-2	250	1884	2267	2200	2194	2375
	300	1772	1914	1869	1881	2312
50-3	200	1877	2236	2053	2116	2386
	250	1777	2073	1896	1957	2344
50-4	200	1852	2183	2090	2105	2293
	250	1709	2105	1822	1822	2184
50-5	200	1777	2155	1960	2084	2292
	250	1650	1890	1835	1835	2194

Table 3.12: Pulsed Oscillation Results: Best and Average Cost Lengths

Data	K	Best	Average	Std. Dev.	Data	K	Best	Average	Std. Dev.
10-1	350	906	906.7	2.9	20-4	200	1958	1962.6	5
	400	898	909.8	16.6		250	1454	1542.5	75.6
	450	896	906.8	18.2		300	1286	1342.5	34.4
	500	854	867.2	10.9		350	1248	1299.8	38.6
10-2	400	1140	1142.7	4.7		400	1200	1272	39.9
	450	1062	1069.2	8.7		450	1164	1208.9	26.4
	500	1031	1048.9	20.3		500	1170	1237.1	42.9
10-3	300	1269	1275.1	12.6	20-5	300	1324	1446.6	106.6
	350	1018	1018	0		350	1254	1317.4	58.4
	400	954	974.4	31.7		400	1214	1278.7	31.2
	450	952	954.2	5		450	1190	1226.6	31.8
	500	952	957.7	11.5		500	1128	1189.4	46
10-4	300	1391	1391	0	30-1	200	1751	1844.8	55.9
	350	1205	1211.5	10.7		250	1646	1814.1	100.4
	400	1117	1140.9	26.7		300	1494	1787.9	107.4
	450	980	994.7	25.6		350	1519	1686.5	80
	500	980	1000.6	23.9		400	1474	1673.3	100.4
10-5	350	1383	1384.6	3.2		450	1544	1697.8	77
	400	1238	1264.9	22.9		500	1487	1651.7	118.8
	450	1143	1160.1	19.3	30-2	300	1535	1652.9	66
	500	1072	1101.8	27.8		350	1530	1700.3	84.7
20-1	200	1577	1587.3	14.2		400	1589	1708.6	62.3
	250	1451	1484.6	38.8		450	1579	1765.2	86.5
	300	1389	1448.2	39.6		500	1598	1762.8	69.7
	350	1273	1329	51.6	30-3	250	1667	1792.7	68.1
	400	1183	1244.6	46.3		300	1537	1718.2	114.2
	450	1150	1221.9	41.3		350	1504	1624.3	87.3
	500	1111	1153.5	47		400	1476	1611.8	97.3
20-2	200	1325	1356.9	42.3		450	1358	1655.7	119.1
	250	1107	1140.2	26.5		500	1443	1624.9	88.7
	300	997	1034.7	31.6	30-4	200	1568	1627.5	29.6
	350	953	981.1	21.9		250	1384	1536.2	73.8
	400	949	982.3	24.1		300	1416	1522.1	71.2
	450	933	969.2	29.2		350	1308	1467.9	80.4
	500	910	951.8	27.9		400	1276	1475.1	87.9
20-3	200	1453	1530.4	79.6		450	1277	1473.3	105
	250	1242	1298	77.9		500	1243	1510.1	91.9
	350	1122	1183.4	52.7	30-5	200	2104	2266.4	61.3
	400	1109	1143.8	25.9		250	1967	2109.1	84.6
	450	1107	1146.1	29		300	1827	2003.7	100.4
	500	1057	1102.2	39		350	1762	1923.5	109.9
						400	1677	1889.9	127.7
						450	1756	1925.2	95.9
						500	1694	1930.9	132.4

Table 3.13: Pulsed Oscillation Results: Comparisons of Cost Lengths

Data	K	BB	Stingy	Tabu	GA	PSO
10-1	350	906	906	906	906	***906***
	400	898	976	898	898	***898***
	450	896	955	898	896	***896***
	500	854	880	854	854	***854***
10-2	400	1140	1154	1140	1140	***1140***
	450	1062	1062	1062	1062	***1062***
	500	1031	1062	1031	1031	***1031***
10-3	300	1269	1433	1269	1269	***1269***
	350	1018	1089	1018	1018	***1018***
	400	954	1025	954	954	***954***
	450	952	1025	952	952	***952***
	500	952	1007	952	952	***952***
10-4	300	1391	1445	1391	1391	***1391***
	350	1205	1256	1205	1205	***1205***
	400	1117	1186	1117	1117	***1117***
	450	980	1149	980	980	***980***
	500	980	1142	980	980	***980***
10-5	350	1383	1457	1383	1383	***1383***
	400	1238	1456	1238	1238	***1238***
	450	1143	1292	1143	1143	***1143***
	500	1072	1213	1072	1072	***1072***
20-1	200	1577	1859	1819	1577	***1577***
	250	1445	1501	1455	1445	***1451***
	300	1376	1430	1376	1383	*1389*
	350	1253	1459	1253	1253	*1273*
	400	1183	1416	1183	1183	***1183***
	450	1144	1266	1144	1144	*1150*
	500	1111	1185	1111	1111	***1111***
20-2	200	1325	1360	1329	1390	***1325***
	250	1094	1166	1098	1104	*1107*
	300	984	1065	990	996	*997*
	350	953	974	953	953	***953***
	400	940	974	946	940	*949*
	450	919	959	932	929	*933*
	500	900	925	900	917	***910***
20-3	200	1449	1512	1449	1610	***1453***
	250	1218	1272	1218	1218	*1242*
	350	1100	1266	1100	1138	***1122***
	400	1100	1239	1100	1103	*1109*
	450	1100	1228	1101	1104	*1107*
	500	1011	1219	1011	1011	*1057*
20-4	200	1958	1962	1958	1958	***1958***
	250	1454	1524	1460	1460	***1454***
	300	1286	1442	1286	1286	***1286***
	350	1235	1334	1235	1235	*1248*
	400	1190	1259	1190	1205	***1200***
	450	1164	1213	1164	1164	***1164***
	500	1149	1213	1174	1164	***1170***
20-5	300	1324	1419	1339	1332	***1324***
	350	1251	1355	1251	1251	*1254*
	400	1211	1300	1213	1239	***1214***
	450	1119	1162	1125	1119	1190
	500	1082	1125	1082	1082	1128
30-1	200	1726	1963	1898	1853	***1751***
	250	1477	1689	1612	1655	***1646***
	300	1413	1657	1474	1557	***1494***
	350	1328	1622	1366	1406	*1519*
	400	1295	1464	1358	1333	1474
	450	1240	1382	1240	1248	1544
	500	1213	1322	1221	1213	1487
30-2	300	1477	1607	1507	1488	*1535*
	350	1381	1461	1386	1386	1530
	400	1319	1425	1322	1319	1589
	450	1319	1386	1347	1319	1579
	500	1295	1386	1302	1302	1598
30-3	250	1411	1711	1574	1613	*1667*
	300	1325	1526	1329	1325	1537
	350	1239	1250	1239	1239	1504
	400	1198	1349	1198	1198	1476
	450	1152	1160	1152	1152	1358
	500	1128	1151	1128	1128	1443
30-4	200	1448	1612	1549	1577	***1568***
	250	1250	1547	1327	1382	*1384*
	300	1164	1484	1316	1391	*1416*
	350	1134	1470	1143	1152	*1308*
	400	1050	1215	1068	1068	1276
	450	1044	1206	1044	1097	1277
	500	1044	1223	1079	1064	1243
30-5	200	2056	2282	2156	2080	***2104***
	250	1759	2195	1915	1915	*1967*
	300	1635	2097	1750	1742	*1827*
	350	1562	1668	1562	1562	1762
	400	1493	1578	1497	1497	1677
	450	1452	1510	1459	1452	1756
	500	1424	1498	1425	1424	1694

Table 3.14: Additional Pulsed Oscillation Results: Best and Average Cost Lengths

Data	K	Best	Average	Std. Dev.
40-1	200	2658	2757.8	75.8
	250	2447	2745.9	189.1
	300	2516	2729.5	131.5
	350	2463	2764.6	178.6
	400	2653	3041.1	230.5
	450	2935	3198.4	175
	500	2716	3111.6	195.9
40-2	300	2286	2465.8	143.5
	350	2385	2590.1	97.8
	400	2441	2695	190.1
	450	2809	3032.2	170.3
	500	2636	2988.4	201.1
40-3	200	2569	2727.7	131.6
	250	2515	2731.4	154.4
	300	2299	2612.8	188.4
	350	2611	2880.9	148.7
	400	2712	3109.9	238
	450	2679	3145.9	284.7
	500	2964	3237.5	168.7
40-4	200	2250	2497.4	146.8
	250	2445	2651.1	193
	300	2453	2662.1	143.4
	350	2563	2833	209.7
	400	2579	2809.7	147.5
	450	2818	3107.7	216.6
	500	2775	3070.2	160.7
40-5	200	1870	1961.7	60.6
	250	1977	2238	134.7
	300	2381	2554	115.3
	350	2477	2662.7	148.9
	400	2450	2792.2	166.6
	450	2464	2735.1	196.7
	500	2403	2700.2	144.7
50-1	150	2483	2647.3	116
	200	2713	3075.2	228
50-2	250	3508	3906.5	289.5
	300	3715	4393.8	546.4
50-3	200	3072	3437.5	250.4
	250	3384	4205.4	469.5
50-4	200	2972	3205	142.7
	250	3476	3799.8	190.5
50-5	200	2996	3486.1	242.5
	250	3600	3968.6	284.1

Table 3.15: Additional Pulsed Oscillation Results: Comparisons of Cost Lengths

Data	K	BB	Stingy	Tabu	GA	PSO	Data	K	BB	Stingy	Tabu	GA	PSO
40-1	200	2067	2549	2232	2320	2658	40-4	200	1894	2159	2069	2163	2250
	250	1800	2261	2031	2030	2447		250	1718	1877	1791	1795	2445
	300	1687	2213	1947	1854	2516		300	1610	1768	1706	1688	2453
	350	1616	1998	1691	1733	2463		350	1551	1664	1616	1616	2563
	400	1558	1699	1609	1627	2653		400	1503	1603	1552	1572	2579
	450	1533	1767	1571	1697	2935		450	1476	1590	1524	1536	2818
	500	1520	1751	1537	1589	2716		500	1458	1511	1492	1475	2775
40-2	300	1558	1737	1621	1617	2286	40-5	200	1626	1747	1720	1703	1870
	350	1496	1592	1514	1527	2385		250	1455	1727	1657	1676	1977
	400	1459	1544	1477	1535	2441		300	1393	1712	1607	1544	2381
	450	1434	1540	1462	1462	2809		350	1356	1699	1575	1607	2477
	500	1416	1505	1422	1416	2636		400	1315	1699	1546	1546	2450
40-3	200	2031	2421	2317	2355	2569		450	1266	1699	1422	1438	2464
	250	1821	2146	2077	2077	2515		500	1246	1699	1291	1246	2403
	300	1688	1897	1815	1815	2299	50-1	150	2165	2367	2250	2286	2483
	350	1620	1747	1654	1694	2611		200	1776	2036	1968	1968	2713
	400	1582	1649	1611	1649	2712	50-2	250	1884	2267	2200	2194	3508
	450	1561	1622	1575	1561	2679		300	1772	1914	1869	1881	3715
	500	1539	1622	1576	1539	2964	50-3	200	1877	2236	2053	2116	3072
								250	1777	2073	1896	1957	3384
							50-4	200	1852	2183	2090	2105	2972
								250	1709	2105	1822	1822	3476
							50-5	200	1777	2155	1960	2084	2996
								250	1650	1890	1835	1835	3600

appears to have been met. The average fitness of the swarm starts off very poor, and settles into a better result as the momentum decreases. As it increases, the average cost rises again as the particles build up more velocity and somewhat randomize. However, as the momentum decreases *again*, the swarm tends to find a newer best solution upon which to settle. This trend can be easily identified in both charts as repeating approximately every 200 iterations, which is identical to the period used for the oscillation waves.

As mentioned above, the cumulative results tend to highlight the deficiency in this system. As the k-bound increases, so too does the dimensionality of the problem. This is because the relaxed bounds lead to fewer potential edges being excluded in the preprocessing stage (refer to Section 2.4 for more information). This leads to particles with a larger number of dimensions, which may tend to decrease the chance of a single (potentially valuable) dimension asserting itself. Refer to Table 3.16 for the numbers of allowable edges for each dataset, with each provided k-bound. It still may yet be possible to improve upon these results if some mechanism could be introduced to give ignored edges a chance to be included. Further discussion on this subject is mentioned in Chapter 5.

In spite of having found a few new best metaheuristic results, the system can still not be considered truly competitive with Tabu or Genetic Algorithms. However, the results were still very promising, particularly when the dimensionality was lower. If this issue can be overcome, then this system could have great potential. Additionally, the oscillating momentum can have future application to entirely unrelated problems when there is a similar concern of balancing between fine-tuning and exploration. And, the priority-based transcription mechanism can also be applied to other similarly restricted problems that would otherwise be a challenge for PSO use.

Table 3.16: Number of Allowable Edges for Each Instance and k-Bound

Data	K	*Edges*	K	BB	*Edges*	K	BB	*Edges*
10-1	350	27	20-4	200	59	40-1	200	237
	400	36		250	97		250	348
	450	41		300	124		300	458
	500	45		350	146		350	556
10-2	400	30		400	169		400	628
	450	39		450	177		450	715
	500	43		500	186		500	752
10-3	300	25	20-5	300	109	40-2	300	464
	350	32		350	137		350	549
	400	38		400	164		400	638
	450	39		450	178		450	719
	500	44		500	185		500	752
10-4	300	21	30-1	200	133	40-3	200	220
	350	31		250	207		250	328
	400	34		300	276		300	421
	450	41		350	332		350	521
	500	42		400	373		400	621
10-5	350	25		450	409		450	694
	400	35		500	423		500	738
	450	40	30-2	300	284	40-4	200	268
	500	42		350	342		250	382
20-1	200	52		400	379		300	478
	250	83		450	415		350	569
	300	112		500	427		400	644
	350	144	30-3	250	241		450	714
	400	160		300	306		500	749
	450	183		350	362	40-5	200	307
	500	187		400	394		250	411
20-2	200	82		450	422		300	512
	250	117		500	431		350	613
	300	146	30-4	200	170		400	697
	350	170		250	259		450	757
	400	181		300	318		500	777
	450	189		350	374	50-1	150	243
	500	190		400	410		200	394
20-3	200	70		450	427	50-2	250	553
	250	89		500	434		300	712
	350	151	30-5	200	134	50-3	200	440
	400	177		250	198		250	621
	450	185		300	260	50-4	200	435
	500	188		350	319		250	608
				400	352	50-5	200	500
				450	401		250	703
				500	419			

Chapter 4

Pheromone-Driven PSO

Priority-Based PSO was not the only technique attempted. After two unsuccessful attempts at applying Ant Colony Optimization (ACO)—a metaheuristic created by Marco Dorigo [32]—an alternate approach, primarily founded in PSO but with elements inspired by ACO was also attempted. The technique used was still a particle swarm, but the transcription mechanism was supplemented with characteristics of ACO.

This chapter provides a brief background of ACO, describes initial attempts to apply an ant-style algorithm to 2CNBR, and then goes on to detail the new particle system and lists the results.

4.1 Preliminary ACO Work

An ant-inspired particle swarm was eventually used for the 2CNBR problem. However, in order to fully understand the methodology, and how it came to be, it is first necessary to understand Ant Colony Optimization, and the first modifications to it which eventually lead to the new particle system.

4.1.1 Background on Ant Colony Optimization

Ant Colony Optimization (ACO), first created by Mark Dorigo for his PhD thesis [32], is a metaheuristic inspired by the foraging behaviour of ants. It is typically used for certain graph-based problems, like the traveling salesman problem (TSP) [33] among others [34]. Each edge in the graph contains some level of *pheromone*. Ants find these pheromones attractive, and are more likely to select an edge if it has a higher concentration of pheromones. The ants probabilistically select edges, one at a time, based on the level of pheromones and the perceived cost of the edge[1].

After ants have chosen their complete paths, all pheromone levels are partially evaporated. New pheromones are then added to the remaining levels on those edges that were selected by the ants. Since ants are more likely to follow higher pheromones levels, large pheromone deposits tend to self-reinforce. As such, while although one or two ants may take a 'good' path initially, by the end of an experiment, most will be taking that same path, or a better one if one has been found.

4.1.2 Application of ACO to 2CNBR

Consider the case of applying ACO to the traveling salesman problem. An ant starts at some city, and then progressively selects one city after another, stochastically selecting them based on the pheromone levels and edge costs. To conform with the definition of a TSP, at each city, it may only consider those cities that are still unvisited. After it has visited them all, the algorithm is done.

Typically, a taboo list is maintained for each ant, to guarantee that they will not revisit an old city. However, in the case of 2CNBR, revisitation is *expected*. Indeed, it is even *necessary*. It is true that the biconnectivity constraint can be satisfied with a Hamiltonian cycle (as seen in the TSP), but there is no way to ensure bounded rings without permitting revisitation. This means that the algorithm *must* permit an ant to return to old vertices.

However, this introduces a new question. How can one ensure that the algorithm will end? What is to stop an ant from choosing a ring of particularly high pheromones, and then continuously cycling through that ring? Statistically speaking, the ant will most likely leave *eventually*, but that is a poor guarantee for computational efficiency. Rather, some addition was necessary that would both *allow* the ants to return to old vertices as often as necessary, but also eventually *discourage* that behaviour.

The original plan to allow this was to develop a sort of *antipheromone*. Whenever an ant would traverse an edge, it would lay down some antipheromone, which would partially cancel the strength of the regular pheromone. As such, the next time an ant considered that edge, it would be slightly less attractive.

To get the same effect, but save slightly on both calculations and complexity, a second set of regular pheromones was used instead. For each transcription, the second set of pheromones would be initialized to having the same values as the first set. As the ant traveled, it would evaporate some of those pheromones using the normal algorithm. This still had the effect of making edges less and less attractive

[1]There are different ways to define the 'cost' of an edge for an ant algorithm, but a common choice is to use the reciprocal of the length, with an ant being more likely to edges with higher 'costs', which correspond to shorter lengths.

each time they were selected. The first set of pheromones was not disrupted by the transcription, and was trained as normal.

However, the results were very poor. Even the easiest of problems (only ten vertices, and with the smallest bounds) could not be reliably solved optimally[2]. It was clear that a better solution was necessary, but it was important to first identify any inherent flaws with the design.

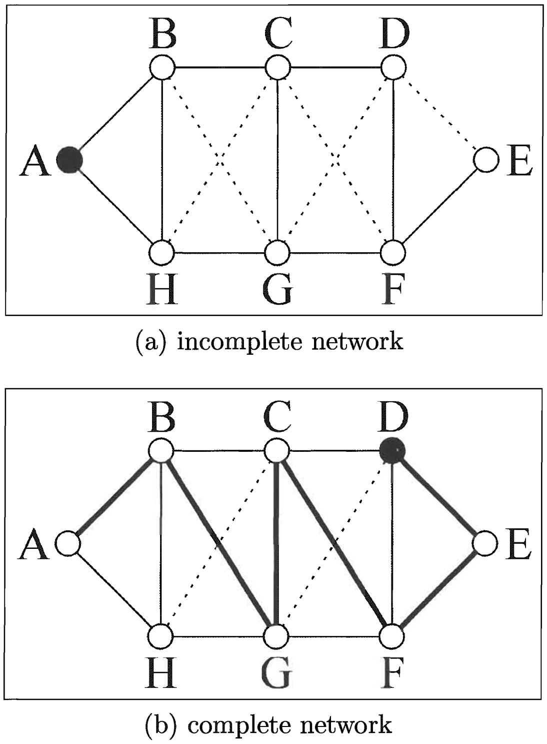


Figure 4.1: Inefficient Path Taken by Ant Through Network

Consider the scenario depicted in Figure 4.1a. In this case, there is only one edge missing (edge DE) from a legal network, but the ant is at the other end of the network, at vertex A. In order to be able to add that final remaining edge, it will need to get to either D or E. In order to get there, it will have to traverse the rest of the network. If it happens to only select previously visited edges, then this is not a problem. However, if it happens to select unvisited edges, then it will result in a network which has a higher cost than is necessary, as shown in 4.1b. The fact that the network can only grow from one point at a time, when trying to find solutions

[2]Finding networks with optimal answers is not normally the purpose of this work. However, for such trivial problems, even finding optimal answers by hand is relatively easy. They are essentially 'toy problems'.

that contain the equivalent of choosing multiple paths from each vertex, appeared to be the biggest problem.

4.1.3 Spill System

If the dilemma with attempting to apply ant colony optimization to the 2CNBR problem was related to the fact that the ants could only grow the network from one vertex at a time, then the logical solution to attempt was to lift this restriction.

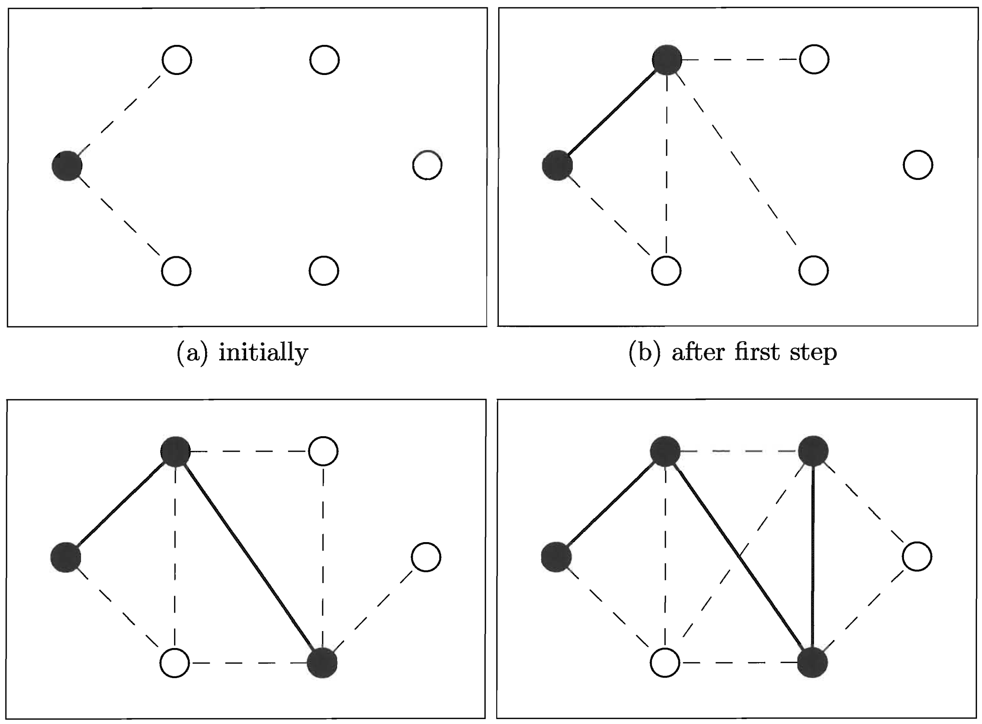
In a new algorithm, called a *spill system*, the spill started at some randomly chosen point, just as with the ant colony. And, the spill would then select the first edge, also identical to how ACO behaves. However, once more than one vertex is selected, the difference is readily apparent. As illustrated in Figure 4.2, the spill is permitted to consider any edges connected to any vertex already included. Thus, once the algorithm has visited all of the vertices, it is allowed to select whatever edges it likes to complete the network. This algorithm has the benefit of being freed of the ACO's limitation of growing only from a single point at a time. The actual edge selection is decided using the same math as the regular ACO, and is detailed in the next section. It continuously alternates between selecting an edge, and then checking if the constraints are satisfied. Once they are both satisfied, the algorithm is finished.

Though the spill system still did not perform to satisfaction, it too needed to be analyzed for flaws and either improved upon or replaced. However, as it performed better than the modified ACO, more trial runs were performed. 10 runs of 5,000 epochs each were conducted for each experiment. The best behaviour was that which most closely resembled a plain Ant System (AS). That is, there was no intra-epoch pheromone evaporation, and all ants (or *spill agents*) were included in the pheromone update rule. The empirically derived parameters which yielded the best results are as follows:

- *# ants*: Equal to the # of vertices (10 ants for 10 vertices, etc.).
- ρ: 0.2.
- α: 2.0.
- β 2.0.
- *Initial pheromone level* (τ_0): 0.001.

An excerpt of the best obtained results is shown in table 4.1.

When attempting to determine the source of the performance problem, it was decided that more information was necessary. As such, the ability to track pheromone



(a) initially

(b) after first step

(c) after second step

(d) after third step

Figure 4.2: Spill System Demonstrated Across First Three Steps

Table 4.1: Preliminary Spill System Results

						Spill System		
Data	K	BB	Stingy	Tabu	GA	Best	*Average*	Std. Dev.
10-1	350	906	906	906	906	***906***	950.0	37.2
20-1	200	1577	1859	1819	1577	1924	2013.9	50.3
30-1	200	1726	1963	1898	1853	3211	3750	233.6

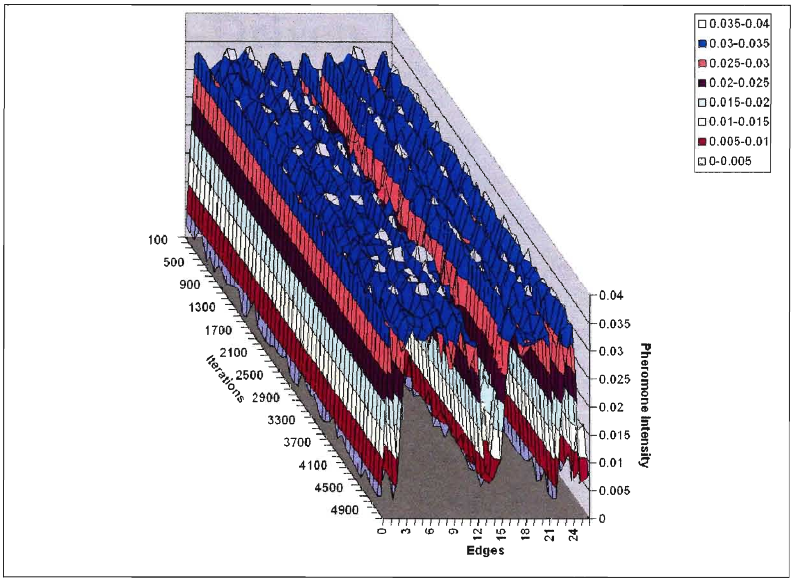


Figure 4.3: Plot of Spill System Pheromone Levels After Each Hundred Iterations

levels across set intervals was added. The change in pheromone levels across all edges was studied and a trend became readily apparent.

As it turns out, the edges *were* receiving pheromone deposits. The problem was with a tendency to deposit pheromones on the same edges evenly and repeatedly. This meant that any given preliminary solution was quickly settled upon, and no edges were able to further distinguish themselves as being more valuable. Stagnation was nearly immediate as evaporation and the introduction of new pheromones quickly found equilibrium. In essence, even though pheromones were being added and evaporated, the system was not actually *training*. The next step was to determine why this was happening, and a likely explanation was soon found.

The original formulas provided by Dorigo were intended for ACO, not a spill system. They assume that pheromones will be applied conservatively to a small subset of the totality of the edges. What ended up happening with the spill system, however, was that each spill was picking a wide assortment of edges. So wide, in fact, that the majority of the network ended up being repeatedly selected, when considered across all spills. As such, the system understandably failed to properly train. This introduced a new question: is there any other way to train the pheromone levels?

Considering the fact that the pheromone levels are continuous floating point values, particle swarms were reexamined.

4.2 Pheromone-Driven PSO

As stated in chapter 2, particle swarms are ideally suited for training vectors of continuous floating-point values. This suggests that it might be very effective for training the pheromone levels of a spill-style system. Some sort of a system combining elements of PSO and ACO was decided upon as being the final solution.

The basic approach is simple. As with the priority-based PSO, each dimension in a particle's position corresponds to a potential edge in the network. However, the actual position within each dimension corresponds to the amount of pheromone on that edge. As such, the PSO side of the system is relatively simple, as shown in algorithm 7.

Algorithm 7 Pheromone-Driven Particle Transcription and Evaluation

```
for each particle p in swarm do
  for each dimension i do
    edge_i.pheromone=p.position_i
  end for
  while Feasible network not yet constructed do
    Select edge based on pheromones
    Check two-connectivity and k-bound constraints
  end while
end for
```

Before actually implementing the system though, there were some fundamental questions that needed to be answered. The first was whether or not any element of the pheromone update rule should be included. Since the *reason* this system was necessary in the first place was the poor performance of the pheromone-training, it was easily decided to not include the pheromone update rule. The second issue to decide was how many ants/spills to use per particle. Eventually, it was decided to not use multiple ants per particle for two reasons. First, computation time was a concern. Second, the particle swarm already introduces its own sense of parallel search. Furthermore, as the swarm will always tend to cluster eventually, very similar networks would be tested anyways, so it seemed as though it would be somewhat redundant to also add multiple ants per particle.

The system devised is another particle swarm, but one with a unique transcription scheme. In contrast to the ant or spill systems, this system is allowed to consider *all* edges, even before spreading out. This was decided mostly for practical reasons. Since there were no longer ants or spills to represent, there was no reason to artificially limit the edges to be considered.

The probability of selecting each edge is as follows:

$$p_i = \begin{cases} \frac{\tau_i^\alpha \nu_i^\beta}{\Sigma_{e_i \in N} \tau_i^\alpha \nu_i^\beta} & : \text{ if } e_i \text{ is already in the network} \\ 0 & : \text{ otherwise} \end{cases}$$

where:

- p_i is the probability of selecting edge e_i.
- τ_i is the pheromone level of edge e_i.
- ν_i is the 'cost' of the edge e_i, defined as the reciprocal of its length.
- α is the user-set parameter that determines the significance of pheromones.
- β is the user-set parameter that determines the significance of 'cost'.

Note that this is identical to a traditional ACO. Also note that α and β are user-defined parameters, and they needed to be experimentally determined for this thesis.

It is important to emphasize the fact that there is *no* pheromone update rule. This is because the PSO becomes the sole entity modifying the pheromone levels. As such, a particle swarm actually represents a collection of multiple pheromone layouts for the problem.

4.3 Experimental Setup

Once again, the final experimental parameters needed to be determined empirically. This included discovering the best parameters for both the pheromone-based edge selection mechanism, as well as the traditional particle swarm parameters. Once again, an iteration-span of 2,000 and a swarm size of 200 were used, to allow comparison with the other PSO results.

The same datasets were used as with the priority-based PSO work. As with that previous work, 20 runs were conducted for all 10, 20, and 30-vertex problems, with extra 10-run experiments conducted on the 40 and 50-vertex problems.

The final setup is shown in Table 4.2.

Note that the X_i^{max} and V_i^{max} are *substantially* different from the previous PSO work. This is not surprising as the it was merely the *relative* values that mattered for the priority-based system; while as the position vector values of the pheromone-based system are used directly in the pheromone-based edge-selection formula. A particularly high pheromone level would render the *cost* (v) of the edge moot.

Also note that, even though they were developed for a priority-based transcription mechanism, the oscillation and supersocial behaviours were still present in the final

Table 4.2: Parameters for Pheromone-Based Experiments

Parameter	Value	Notes
X_i^{max}	100.0	
V_i^{max}	5.0	
Period	200	Continuous Oscillation
c_1	0	No cognitive behaviour
Social Behaviour	Neighbourhood	
Neighbourhood Size	12	
c_2	3.0	
c_3	2.0	
α	4	
β	1	

best setup. This may be due to the fact that they were both applied to problems with identical dimensionality, or the fact that they are still similarly-themed techniques.

4.4 Results

Detailed below are the results of the experiments. Tables 4.5 and 4.7 show the best (lowest) costs obtained by any particle in any run in the experiment, the average of the best cost per run for each experiment, and the sample standard deviation for each experiment.

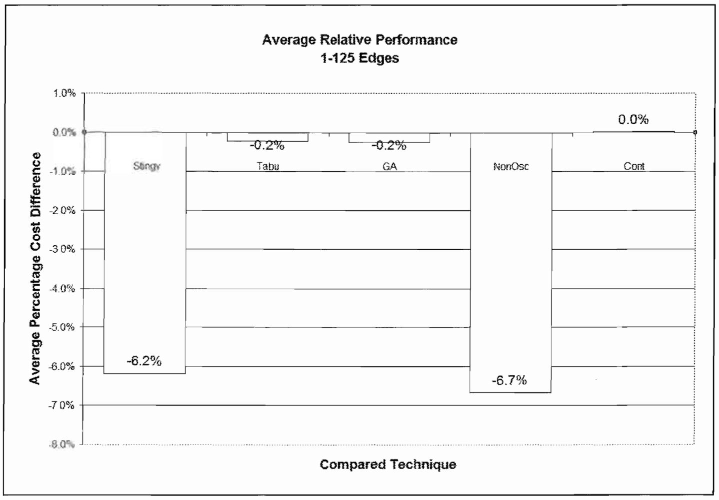
Figure 4.4 depicts the consolidated performance of the pheromone-driven PSO against the other published techniques, as well as against non-oscillating PSO and continuously oscillating PSO from Chapter 3. They were created in the same way as those found in sections 3.3 and 3.6, and once again should be read in the same fashion. Recall that lower values indicate better performance relative the compared techniques. It is easy to see that the pheromone-driven PSO did not perform as well as oscillating priority-driven PSO. For the most difficult of problems, it did not even perform as well, on average, as the non-oscillating priority-based PSO. This implies that, at least in this form, a simple priority list is a better transcription mechanism for this problem than the more elaborate pheromone mechanism.

To gauge the performance of the pheromone-driven PSO with more statistical significance, a nonparametric Mann-Whitney test was again used. As with the previous tests, populations are tested within a 95% confidence level. This time, the pheromone-driven results were compared against the most vanilla system (i.e. the non-oscillating priority-based PSO), and against the continuously oscillating priority-based PSO. The results of the former are shown in Table 4.3. The results of the latter can be found in Table 4.4.

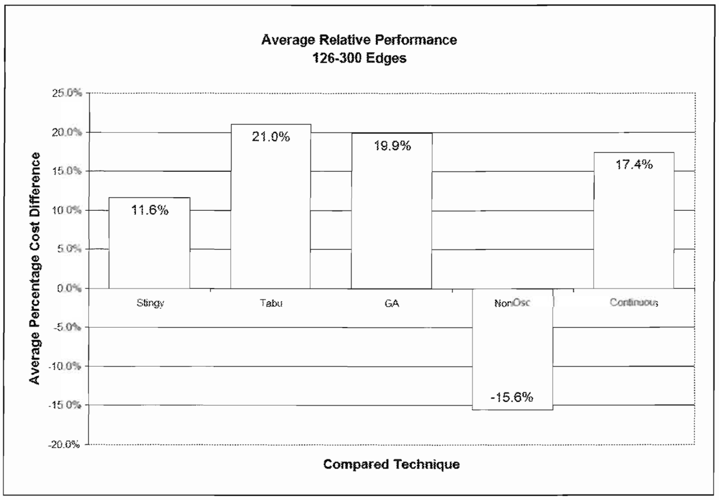
The results of the first test are somewhat mixed. The pheromone-driven PSO wins or ties 123 out of 128 times. However, at the 40-vertex level, it ties more often than it wins. At the 50-vertex level, it loses 5 out of 10 times, and only wins once. As such, it won or tied the vast majority of the time, but it was not terribly competitive with non-oscillating priority-based PSO within the scope of 'very difficult' problems.

The results of the second test are noticeably more definitive. The pheromone-driven PSO only won with statistical significance 8 out of 128 times, and was beaten 92 out of 128 times. Clearly, the pheromone-driven model requires more work before it can match the performance of the priority-based system.

Tables 4.6 and 4.8 show how the best results obtained by the pheromone-driven PSO compare to the best results found by Fortz's Stingy algorithm, his Tabu search, and Ombuki and Ventresca's Genetic Algorithm. Values that are *italicized* indicate that the pheromone-driven PSO matched or beat the results of the Stingy algorithm. Values that are **bold** indicate that the pheromone-driven PSO matched or beat at least one metaheuristic (Tabu, GA, or both).

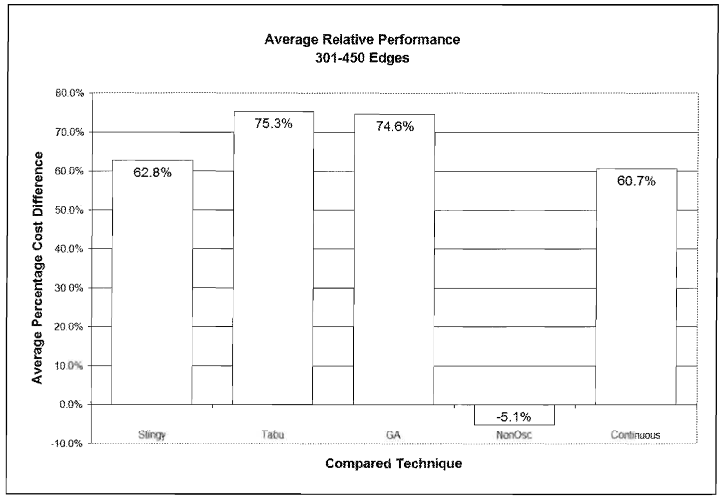


(a) 1-125 Edges

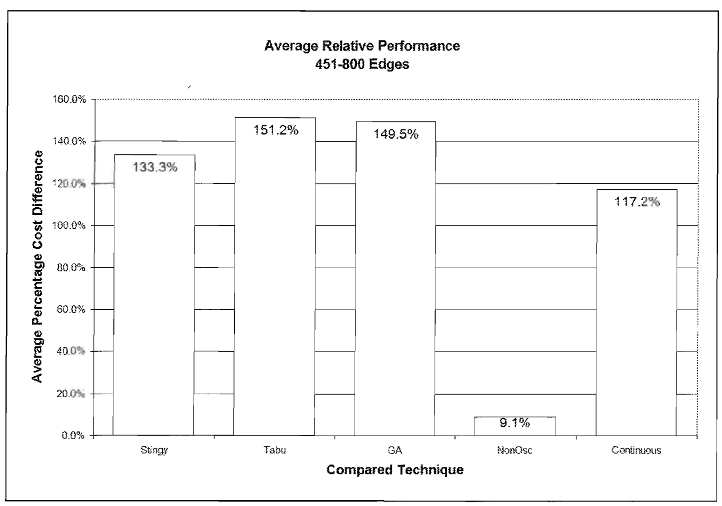


(b) 126-300 Edges

Figure 4.4: Pheromone-Driven Results: Comparison of Averages of Best Costs



(c) 301-450 Edges



(d) 451-800 Edges

Figure 4.4: Pheromone-Driven Results: Comparison of Averages of Best Costs (continued)

Table 4.3: Statistical Comparison of Pheromone-Driven and Non-Oscillating Priority-Based Results

	Tie	N.O. Priority Won	Pheromones Won
10's	0	0	21
20's	0	0	32
30's	0	0	32
40's	20	0	13
50's	4	5	1
Total	24	5	99

Table 4.4: Statistical Comparison of Pheromone-Driven and Continuously Oscillating Priority-Based Results

	Tie	Cont. Priority	Pheromones Won
10's	15	0	6
20's	12	18	2
30's	1	31	0
40's	0	33	0
50's	0	10	0
Total	28	92	8

4.5 Discussion

Some of the results were encouraging, though less-so than with the priority-based PSO.

Figures 4.5 and 4.6 depict the best solutions found for the 30-1 dataset, with k-bounds of 200 and 500, respectively. Notice that the solution quality actually degraded after the k-bound was relaxed. In theory, one should expect solutions with lower costs as the k-bound is raised[3]. However, once again, the particle swarm seemed to have increasing difficulty as the dimensionality of the problems grew.

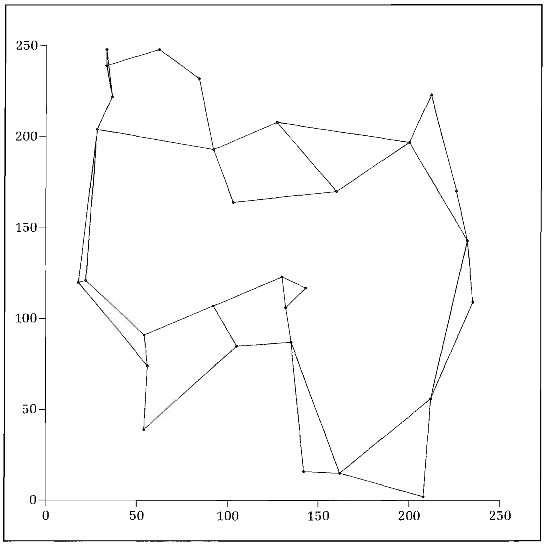


Figure 4.5: Pheromone-Driven Result: Solution for 30-1(200) Plotted

Additionally, the pheromone-driven PSO clearly did not perform as well as the priority-based PSO. While although it still managed to meet or beat at least one previous metaheuristic result on numerous occasions, it hit its ceiling of effectiveness even earlier than the priority-based PSO did.

[3]At the very least, the cost should not *increase*, as any solution possible with a more restrictive k-bound is still possible with a less restrictive k-bound.

Table 4.5: Pheromone-Driven Results: Best and Average Cost Lengths

Data	K	Best	Average	Std. Dev.	Data	K	Best	Average	Std. Dev.
10-1	350	906	906	0	20-4	200	1958	1964.2	7.1
	400	898	898.8	2.5		250	1460	1550.1	43.1
	450	896	898.8	4.7		300	1301	1381.8	61.7
	500	854	862.6	11.7		350	1285	1405.6	75.5
10-2	400	1140	1144.7	5		400	1309	1459.4	69.1
	450	1062	1064.3	4.7		450	1405	1466.2	49.6
	500	1031	1048.9	15.8		500	1330	1464.6	71.5
10-3	300	1269	1279.8	22.9	20-5	300	1346	1394.7	43.8
	350	1018	1018	0		350	1286	1370	50.4
	400	954	960.9	16.7		400	1340	1445.1	52.6
	450	952	952.1	0.4		450	1317	1444.7	54.9
	500	952	952.6	2.5		500	1293	1393	56
10-4	300	1391	1391	0	30-1	200	1800	1943.7	80
	350	1205	1212.8	12.7		250	2195	2340	91
	400	1117	1133.3	22.8		300	2269	2458.3	116.9
	450	980	984.6	14.9		350	2118	2449.9	132.5
	500	980	987.8	18		400	2328	2469.7	99.6
10-5	350	1383	1385.1	3.1		450	2137	2448.2	108.1
	400	1238	1258.5	9.7		500	2135	2343.5	109.1
	450	1143	1165.7	29.2	30-2	300	2220	2431	96
	500	1072	1083	17.9		350	2260	2450.1	91.6
20-1	200	1577	1592.9	23.5		400	2273	2562.3	154.6
	250	1448	1487	29.4		450	2174	2610.9	159
	300	1383	1429	33.4		500	2191	2547.4	149.8
	350	1313	1424.6	68.9	30-3	250	2189	2382.1	76.6
	400	1285	1389.4	64.1		300	2185	2348.7	81.2
	450	1368	1464.7	54.9		350	1900	2296.1	127.1
	500	1285	1383.5	56.6		400	2098	2324.3	117.9
20-2	200	1331	1364.9	18.7		450	2171	2321.1	86.9
	250	1104	1137.9	25.9		500	1981	2266.9	146.1
	300	1027	1111.8	51.4	30-4	200	1683	1876.9	79.1
	350	1063	1174.2	42.1		250	1747	2133.3	130.2
	400	1090	1164	43.5		300	1765	2074.3	118.6
	450	1078	1170.9	42.7		350	1919	2117.9	116.4
	500	1040	1136.7	42.4		400	1834	2099.8	122
20-3	200	1502	1654.6	70.7		450	1966	2069.9	76
	250	1225	1277.2	40.9		500	1828	2050.2	92.7
	350	1269	1360	58.7	30-5	200	2284	2391.6	52.9
	400	1286	1380.8	49.6		250	2339	2593.4	121.5
	450	1264	1344.8	45.3		300	2538	2678	86.2
	500	1232	1307.4	45.4		350	2570	2742.8	110
						400	2482	2706.7	121.6
						450	2630	2885.6	123.8
						500	2482	2822.9	129.1

Table 4.6: Pheromone-Driven Results: Comparisons of Cost Lengths

Data	K	BB	Stingy	Tabu	GA	PSO
10-1	350	906	906	906	906	***906***
	400	898	976	898	898	***898***
	450	896	955	898	896	***896***
	500	854	880	854	854	***854***
10-2	400	1140	1154	1140	1140	***1140***
	450	1062	1062	1062	1062	***1062***
	500	1031	1062	1031	1031	***1031***
10-3	300	1269	1433	1269	1269	***1269***
	350	1018	1089	1018	1018	***1018***
	400	954	1025	954	954	***954***
	450	952	1025	952	952	***952***
	500	952	1007	952	952	***952***
10-4	300	1391	1445	1391	1391	***1391***
	350	1205	1256	1205	1205	***1205***
	400	1117	1186	1117	1117	***1117***
	450	980	1149	980	980	***980***
	500	980	1142	980	980	***980***
10-5	350	1383	1457	1383	1383	***1383***
	400	1238	1456	1238	1238	***1238***
	450	1143	1292	1143	1143	***1143***
	500	1072	1213	1072	1072	***1072***
20-1	200	1577	1859	1819	1577	***1577***
	250	1445	1501	1455	1445	***1448***
	300	1376	1430	1376	1383	***1383***
	350	1253	1459	1253	1253	*1313*
	400	1183	1416	1183	1183	*1285*
	450	1144	1266	1144	1144	1368
	500	1111	1185	1111	1111	1285
20-2	200	1325	1360	1329	1390	***1331***
	250	1094	1166	1098	1104	***1104***
	300	984	1065	990	996	*1027*
	350	953	974	953	953	1063
	400	940	974	946	940	1090
	450	919	959	932	929	1078
	500	900	925	900	917	1040
20-3	200	1449	1512	1449	1610	***1502***
	250	1218	1272	1218	1218	*1225*
	350	1100	1266	1100	1138	1269
	400	1100	1239	1100	1103	1286
	450	1100	1228	1101	1104	1264
	500	1011	1219	1011	1011	1232

Data	K	BB	Stingy	Tabu	GA	PSO
20-4	200	1958	1962	1958	1958	***1958***
	250	1454	1524	1460	1460	***1460***
	300	1286	1442	1286	1286	*1301*
	350	1235	1334	1235	1235	*1285*
	400	1190	1259	1190	1205	1309
	450	1164	1213	1164	1164	1405
	500	1149	1213	1174	1164	1330
20-5	300	1324	1419	1339	1332	*1346*
	350	1251	1355	1251	1251	*1286*
	400	1211	1300	1213	1239	1340
	450	1119	1162	1125	1119	1317
	500	1082	1125	1082	1082	1293
30-1	200	1726	1963	1898	1853	***1800***
	250	1477	1689	1612	1655	2195
	300	1413	1657	1474	1557	2269
	350	1328	1622	1366	1406	2118
	400	1295	1464	1358	1333	2328
	450	1240	1382	1240	1248	2137
	500	1213	1322	1221	1213	2135
30-2	300	1477	1607	1507	1488	2220
	350	1381	1461	1386	1386	2260
	400	1319	1425	1322	1319	2273
	450	1319	1386	1347	1319	2174
	500	1295	1386	1302	1302	2191
30-3	250	1411	1711	1574	1613	2189
	300	1325	1526	1329	1325	2185
	350	1239	1250	1239	1239	1900
	400	1198	1349	1198	1198	2098
	450	1152	1160	1152	1152	2171
	500	1128	1151	1128	1128	1981
30-4	200	1448	1612	1549	1577	1683
	250	1250	1547	1327	1382	1747
	300	1164	1484	1316	1391	1765
	350	1134	1470	1143	1152	1919
	400	1050	1215	1068	1068	1834
	450	1044	1206	1044	1097	1966
	500	1044	1223	1079	1064	1828
30-5	200	2056	2282	2156	2080	2284
	250	1759	2195	1915	1915	2339
	300	1635	2097	1750	1742	2538
	350	1562	1668	1562	1562	2570
	400	1493	1578	1497	1497	2482
	450	1452	1510	1459	1452	2630
	500	1424	1498	1425	1424	2482

Table 4.7: Additional Pheromone-Driven Results: Best and Average Cost Lengths

Data	K	Best	Average	Std. Dev.	Data	K	Best	Average	Std. Dev.
40-1	200	3212	3368	78.6	40-4	200	3168	3393.9	98.2
	250	3720	4048.6	237.8		250	3614	3903.5	217.6
	300	3699	3943.6	165.3		300	3709	4023.2	201.2
	350	3919	4246	213.4		350	3829	4008.6	153
	400	4116	4363.1	142.5		400	3800	4128.2	225
	450	4091	4574.2	213.5		450	3984	4328.3	210
	500	3769	4367.2	234.9		500	3602	4096.7	217.5
40-2	300	3157	3824.3	294.2	40-5	200	2640	2781.4	84.4
	350	3839	4093.4	190.2		250	3415	3622.2	174.5
	400	3881	4228.4	148.1		300	3543	3674.7	129.9
	450	3780	4365.1	338.6		350	3562	3897.8	234.2
	500	3793	4086.7	246.8		400	3550	3923.2	146.9
40-3	200	3187	3310.6	77.4		450	3314	3938.9	325.4
	250	3333	3623.1	178.2		500	3911	3987	152
	300	3423	3780.1	209.9	50-1	150	3103	3242.4	99.5
	350	4106	4333.1	157.6		200	3857	4286.8	203.8
	400	4183	4481.1	233.5	50-2	250	5176	5636.9	323.4
	450	4350	4637.2	156.7		300	5738	6068.2	249.2
	500	4267	4550	148.6	50-3	200	4655	4962.9	184.7
						250	5451	6499.2	619.3
					50-4	200	4308	4460	152.1
						250	5123	5426.9	218.8
					50-5	200	4655	5034.2	289.9
						250	5466	5873.8	260.7

Table 4.8: Additional Pheromone-Driven Results: Comparisons of Cost Lengths

Data	K	BB	Stingy	Tabu	GA	PSO	Data	K	BB	Stingy	Tabu	GA	PSO
40-1	200	2067	2549	2232	2320	3212	40-4	200	1894	2159	2069	2163	3168
	250	1800	2261	2031	2030	3720		250	1718	1877	1791	1795	3614
	300	1687	2213	1947	1854	3699		300	1610	1768	1706	1688	3709
	350	1616	1998	1691	1733	3919		350	1551	1664	1616	1616	3829
	400	1558	1699	1609	1627	4116		400	1503	1603	1552	1572	3800
	450	1533	1767	1571	1697	4091		450	1476	1590	1524	1536	3984
	500	1520	1751	1537	1589	3769		500	1458	1511	1492	1475	3602
40-2	300	1558	1737	1621	1617	3157	40-5	200	1626	1747	1720	1703	2640
	350	1496	1592	1514	1527	3839		250	1455	1727	1657	1676	3415
	400	1459	1544	1477	1535	3881		300	1393	1712	1607	1544	3543
	450	1434	1540	1462	1462	3780		350	1356	1699	1575	1607	3562
	500	1416	1505	1422	1416	3793		400	1315	1699	1546	1546	3550
40-3	200	2031	2421	2317	2355	3187		450	1266	1699	1422	1438	3314
	250	1821	2146	2077	2077	3333		500	1246	1699	1291	1246	3911
	300	1688	1897	1815	1815	3423	50-1	150	2165	2367	2250	2286	3103
	350	1620	1747	1654	1694	4106		200	1776	2036	1968	1968	3857
	400	1582	1649	1611	1649	4183	50-2	250	1884	2267	2200	2194	5176
	450	1561	1622	1575	1561	4350		300	1772	1914	1869	1881	5738
	500	1539	1622	1576	1539	4267	50-3	200	1877	2236	2053	2116	4655
								250	1777	2073	1896	1957	5451
							50-4	200	1852	2183	2090	2105	4308
								250	1709	2105	1822	1822	5123
							50-5	200	1777	2155	1960	2084	4655
								250	1650	1890	1835	1835	5466

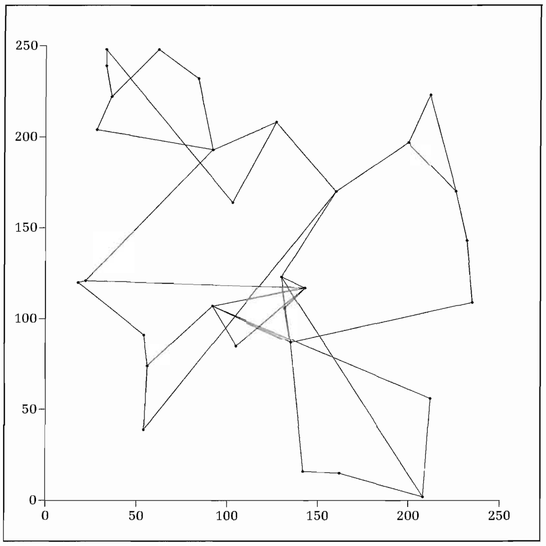


Figure 4.6: Pheromone-Driven Result: Solution for 30-1(500) Plotted

Chapter 5

Conclusions and Future Work

This chapter sums up the final thoughts on using Particle Swarm Optimization for the Two-Connected Networks with Bounded Rings problem, including strengths, weaknesses, and potential room for future improvements.

Overall, the priority-based PSO had encouraging results. Though higher dimensional problems (whether from higher numbers of vertices, or simply more relaxed k-bounds) started to become a more noticeable challenge, the performance for lower-dimensional problems was admirable, and reasonable for moderately difficult problems. Furthermore, it shows great promise for the possible future use of PSO for this style of problem in general.

In spite of the highly-constrained nature of the problem, and the interdependency of edges and associated rings, it was still possible to apply particle swarms (which typically do not permit additional constraints) and obtain reasonable results. This shows the value of using an indirect transcription scheme (in this case, a priority listing) as a means of circumventing a natural limitation of PSO, which represents a significant contribution in and of itself.

The oscillation, as a means of avoiding stagnation, was also a novel method for giving the system sufficient flexibility to both explore and refine solutions. Though other techniques exist for changing the behaviour of metaheuristics over time [22], and for intentionally disrupting a system when it reaches stagnation (as is common for Ant Colony Optimization), this simple function allows a repeating behaviour lacking from simulated annealing or the comparable technique used for PSO by Urfalioglu [22], that also does not require a supervising mechanism as in the case of ACO. Thus, it is effective, but also very efficient.

The pheromone-driven particle swarm did not fare as well, but still represents a unique alternate approach to enabling particle swarm positions to be transcribed into working *legal* networks. Additionally, it represents an interesting technique for combining facets of ant colony and particle swarm behaviours.

Refer to Appendix A for listings of the results of all the techniques used in this thesis. Tables A.1, A.2 and A.3 show the complete final tally of all results, compared against the previous works of Fortz, Ombuki and Ventresca. Again, *italicized* values represent results that matched or beat the Stingy results, and **bolded** values represent results that matched or beat at least one metaheuristic (Tabu, GA, or both).

There still remains a great potential for improving upon these results in the future. First and foremost, it may still be possible to alleviate some of the difficulty associated with the dimensionality issue. Since it is believed that the key impediment is that apparently 'undesirable' edges lack the means to assert themselves[1], it is theorized that some mechanism which tended to punish 'desirable' edges or give 'undesirable' edges a chance might improve results. Since actually modifying the positions of the particles within dimensions corresponding to those undesirable edges would likely be too disruptive, it is logical to suspect that an addition to the velocity update rule may be best. For example, the concept of *entropy* might be a good addition. Entropy is the tendency of organized or clustered items to become disorganized or 'spread out'. Relying upon that general concept, the suggestion is that—either every iteration, or every x iterations—dimensional velocities corresponding to edges could be 'pushed' outwards (i.e. away from the origin) inversely proportional to their dimensional distance from the origin. That is, a very desirable edge would have a force added to its velocity to give it a chance at being shifted past another edge that may otherwise have never had a chance to assert itself in the network.

Additionally, the Spill System may yet still have potential, even if not necessarily for this specific problem. It may, however, require a new function or mechanism for pheromone updates.

In conclusion, being the first use of particle swarms for this largely unexplored problem is not the only contribution of this thesis. The oscillating momentum and indirect transcription schemes introduced in this work have great potential for numerous other combinatorial optimization problems.

[1]In the case of priority-based PSO, this may be because the edges at the least-desirable end of the priority listing actually have no effect at all upon the evaluation of the network. In the case of pheromone-driven PSO, this may similarly be the result of a few edges achieving a very high level of pheromone, stifling the potential of low-pheromone edges to ever compete.

Bibliography

[1] A.R.P. White, J.W. Mann J.W. and G.D. Smith, "Genetic algorithms and network ring design," *Annals of Operations Research*, vol. 86, pp. 347-371, 1999.

[2] B. Fortz, *Design of Survivable Networks With Bounded Rings*, PhD Thesis, Universite Libre de Bruxelles, 1999.

[3] B. Fortz, M. Labbe and F. Maffioli, "Solving the two-connected network with bounded meshes problem," *Operations Research*, 48(6), pp. 866-877, 2000.

[4] B. Fortz and M. Labbe, "Polyhedral results for two-connected networks with bounded rings," *Mathematical Programming*, 93(1), pp. 27-54, 2002.

[5] B. Fortz and M. Labbe, "Two-connected networks with rings of bounded cardinality," *Computational Optimization and Applications*, 27(2), pp. 123-148, 2004.

[6] D. Jungnickel, "Graphs, networks and algorithms," *Algorithms and Computation in Mathematics*, vol. 5, 1999.

[7] M.R. Garey and D.S. Johnson, *Computers and Intractability, A Guide to The Theory of NP-Completeness*, W. H. Freeman and Company, 1979.

[8] K.K. Aggarwal, Y.C. Chopra and J.S. Bajwa, "Topological layout of links for optimizing the overall reliability in a computer communication system," *Microelectronics and Reliability*, vol. 22, no. 3, pp. 347-351, 1982.

[9] M.M. Atiqullah and S.S. Rao, "Reliability optimization of communication networks using simulated annealing," *Micro-electronics and reliability*, vol. 33, no. 9, pp. 1303-1319, 1993.

[10] G. Walters and D.K. Smith, "Evolutionary design algorithm for optimal layout of tree networks," *Engineering Optimization*, vol. 24, pp. 261-281, 1995.

[11] B. Ombuki-Berman and M. Ventresca, "Search difficulty of two-connected ring-based topological network designs," *IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, pp. 267-274, Honolulu, USA, 2007.

[12] A.M. Alvarez, J.L. Gonzalez-Velarde and K. De-Alba, "Graph embedded scatter search for the multicommodity capacitated network design problem," *Journal of Heuristics*, vol. 11, pp. 233-257, 2005.

[13] F. Altiparmak, B. Dengiz and A.E. Smith, "Optimal design of reliable computer networks: a comparison of metaheuristics," *Journal of Heuristics*, 9(6), pp. 471-487, 2003.

[14] T. Magnanti and R. Wong, "Network design and transportation planning: models and algorithms," *Transportation Science*, vol. 19, no. 1, pp. 1-55, 1984.

[15] B. Fortz and M. Labbe, "A tabu search heuristic for the design of two-connected networks with bounded rings", IAG Working Paper 74/02, Universite Catholique de Louvain, 2002. Submitted

[16] B. Ombuki, M. Nakamura, Z. Nakao Z and K. Onaga, "An evolutionary algorithm approach to the design of minimum cost survivable networks with bounded rings," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E84-A, no. 6, pp. 1545-1548, 2001.

[17] M. Ventresca and B. Ombuki, "A genetic algorithm for the design of two connected networks with bounded rings," *Computational Intelligence and Applications, Special Issue on Nature-Inspired Approaches to Networks and Telecommunications*, 5(2), pp. 267-281, November 2005.

[18] L. He, C.P. Botham and C.D. O'Shea, "An evolutionary design algorithm for ring-based SDH optical core networks", *BT Technology Journal*, vol. 22, no. 1, 2004.

[19] M. Armony, J.G. Klinecewicz, H. Luss and M.B. Rosenwein, "Design of stacked self-healing rings using a genetic algorithm", *Journal of Heuristics*, vol. 6, no. 1, pp. 85-105, April 2000.

[20] W. Chen and J. Zheng, "Capacity design of ATM rings with genetic algorithms", *2000 IEEE Asia-Pacific Conference on Circuits and Systems, IEEE Electronic Communication Systems*, Proceedings, pp. 883-886, December 2000.

[21] C.P. Botham, N. Hayman, A. Tsiaparas and P. Gaynord, "Advanced modelling techniques for designing survivable telecommunications networks", *BT Technology Journal*, vol. 21, no. 2, April 2003.

[22] Urfalioglu, O., "Robust estimation of camera rotation, translation and focal length at high outlier rates", 2004.

[23] D.Y. Sha, Cheng-Yu Hsu, "A hybrid particle swarm optimization for job shop scheduling problem", October 2006.

[24] Jianbo Yu, Lifeng Xi, Shijin Wang, "An improved particle swarm optimization for evolving feedforward artificial neural networks", October 2007.

[25] E. Foxwell, B. Ombuki-Berman, "Particle swarm optimisation for the design of two-connected networks with bounded rings," *International Journal of High Performance Systems Architecture*, vol. 1, no. 4, pp. 220-230, 2008.

[26] A. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley Publishing Company, 1974.

[27] F. Kishino, Y. Takamatsu, S. Watanabe, "Optical submarine cable systems. STM-16 optical ring system", *Toshiba Review*, vol. 55, no. 4, pp. 33-36, 2000.

[28] X. Yijun, L. G. Mason, "Restoration strategies and spare capacity requirements in self-healing ATM networks", *IEEE/ACM Transactions on Networking*, vol. 7, no. 1, pp. 98-110, February 1999.

[29] Xiaohui Hu, *Particle Swarm Optimization: Tutorial*, `http://www.swarmintelligence.org/tutorials.php`

[30] V.G. Gudise, G.K. Venayagamoorthy, "Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks", *Swarm Intelligence Symposium, 2003*, Proceedings, pp. 110-117, April 2003.

[31] M.A. Abido, "Optimal power flow using particle swarm optimization", *International Journal of Electrical Power & Energy Systems*, vol. 24, issue 7, pp. 563-571, 2002.

[32] M. Dorigo, *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, Italie, 1992.

[33] M. Dorigo, "Ant colonies for the travelling salesman problem", *Bio Systems*, vol. 43, no. 2, pg 73, 1997.

[34] J.E. Bell, "Ant colony optimization techniques for the vehicle routing problem", *Advanced Engineering Informatics*, vol. 18, no. 1, pg 41, 2004.

[35] M. Grotschel, C.L. Monma, M. Stoer, "Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints", *Operations Research*, vol. 4, no. 2, pp 309-330, 1992.

[36] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press and McGraw-Hill, 2nd edition, 2001.

[37] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Optimization by simulated annealing", *Science*, New Series, vol. 220, no. 4598, pp 671-680, May 1983.

[38] C.H. Brase, C.P. Brase, *Understandable Statistics*, Houghten Mifflin Company, 2003.

Appendix A

Comparison of all Results

Below are tables representing the best results of all the techniques used in this thesis, compared against the best results obtained by Fortz, Ombuki, and Ventresca.

Table A.1: Final Results: Comparison for 10 and 20 Vertices

Data	K	BB	Stingy	Tabu	GA	NonOsc	ContOsc	PulOsc	Pher
10-1	350	906	906	906	906	906	906	906	906
	400	898	976	898	898	934	898	898	898
	450	896	955	898	896	896	896	896	896
	500	854	880	854	854	919	854	854	854
10-2	400	1140	1154	1140	1140	1146	1140	1140	1140
	450	1062	1062	1062	1062	1086	1062	1062	1062
	500	1031	1062	1031	1031	1093	1031	1031	1031
10-3	300	1269	1433	1269	1269	1269	1269	1269	1269
	350	1018	1089	1018	1018	1018	1018	1018	1018
	400	954	1025	954	954	1058	954	954	954
	450	952	1025	952	952	952	952	952	952
	500	952	1007	952	952	988	952	952	952
10-4	300	1391	1445	1391	1391	1391	1391	1391	1391
	350	1205	1256	1205	1205	1205	1205	1205	1205
	400	1117	1186	1117	1117	1135	1117	1117	1117
	450	980	1149	980	980	980	980	980	980
	500	980	1142	980	980	1031	980	980	980
10-5	350	1383	1457	1383	1383	1383	1383	1383	1383
	400	1238	1456	1238	1238	1260	1238	1238	1238
	450	1143	1292	1143	1143	1158	1143	1143	1143
	500	1072	1213	1072	1072	1106	1072	1072	1072
20-1	200	1577	1859	1819	1577	1645	1577	1577	1577
	250	1445	1501	1455	1445	1631	1451	1451	1448
	300	1376	1430	1376	1383	1786	1416	1389	1383
	350	1253	1459	1253	1253	1629	1299	1273	1313
	400	1183	1416	1183	1183	1714	1183	1183	1285
	450	1144	1266	1144	1144	1686	1192	1150	1368
	500	1111	1185	1111	1111	1541	1111	1111	1285
20-2	200	1325	1360	1329	1390	1646	1329	1325	1331
	250	1094	1166	1098	1104	1383	1108	1107	1104
	300	984	1065	990	996	1422	987	997	1027
	350	953	974	953	953	1347	967	953	1063
	400	940	974	946	940	1351	958	949	1090
	450	919	959	932	929	1304	926	933	1078
	500	900	925	900	917	1375	930	910	1040
20-3	200	1449	1512	1449	1610	1716	1474	1453	1502
	250	1218	1272	1218	1218	1402	1240	1242	1225
	350	1100	1266	1100	1138	1414	1138	1122	1269
	400	1100	1239	1100	1103	1553	1119	1109	1286
	450	1100	1228	1101	1104	1413	1125	1107	1264
	500	1011	1219	1011	1011	1456	1011	1057	1232
20-4	200	1958	1962	1958	1958	1971	1958	1958	1958
	250	1454	1524	1460	1460	1822	1454	1454	1460
	300	1286	1442	1286	1286	1822	1286	1286	1301
	350	1235	1334	1235	1235	1653	1248	1248	1285
	400	1190	1259	1190	1205	1682	1233	1200	1309
	450	1164	1213	1164	1164	1749	1164	1164	1405
	500	1149	1213	1174	1164	1737	1208	1170	1330
20-5	300	1324	1419	1339	1332	1612	1331	1324	1346
	350	1251	1355	1251	1251	1631	1260	1254	1286
	400	1211	1300	1213	1239	1673	1214	1214	1340
	450	1119	1162	1125	1119	1640	1189	1190	1317
	500	1082	1125	1082	1082	1676	1088	1128	1293

Table A.2: Final Results: Comparison for 30 Vertices

Data	K	BB	Stingy	Tabu	GA	NonOsc	ContOsc	PulOsc	Pher
30-1	200	1726	1963	1898	1853	2502	***1791***	***1751***	***1800***
	250	1477	1689	1612	1655	2350	1708	***1646***	2195
	300	1413	1657	1474	1557	2357	*1618*	***1494***	2269
	350	1328	1622	1366	1406	2400	*1432*	*1519*	2118
	400	1295	1464	1358	1333	2500	*1405*	1474	2328
	450	1240	1382	1240	1248	2803	*1354*	1544	2137
	500	1213	1322	1221	1213	2416	1421	1487	2135
30-2	300	1477	1607	1507	1488	2047	*1533*	*1535*	2220
	350	1381	1461	1386	1386	2225	1470	1530	2260
	400	1319	1425	1322	1319	2601	1515	1589	2273
	450	1319	1386	1347	1319	2523	1450	1579	2174
	500	1295	1386	1302	1302	2671	1414	1598	2191
30-3	250	1411	1711	1574	1613	2153	***1573***	*1667*	2189
	300	1325	1526	1329	1325	2266	1554	1537	2185
	350	1239	1250	1239	1239	2225	1311	1504	1900
	400	1198	1349	1198	1198	2173	*1315*	1476	2098
	450	1152	1160	1152	1152	2457	1218	1358	2171
	500	1128	1151	1128	1128	2292	1195	1443	1981
30-4	200	1448	1612	1549	1577	1962	***1568***	***1568***	1683
	250	1250	1547	1327	1382	2232	***1359***	*1384*	1747
	300	1164	1484	1316	1391	2082	***1285***	*1416*	1765
	350	1134	1470	1143	1152	2113	*1246*	*1308*	1919
	400	1050	1215	1068	1068	2225	*1159*	1276	1834
	450	1044	1206	1044	1097	1825	*1123*	1277	1966
	500	1044	1223	1079	1064	2017	*1113*	1243	1828
30-5	200	2056	2282	2156	2080	2677	*2216*	***2104***	2284
	250	1759	2195	1915	1915	2869	*1990*	*1967*	2339
	300	1635	2097	1750	1742	2671	*1790*	*1827*	2538
	350	1562	1668	1562	1562	2534	*1663*	1762	2570
	400	1493	1578	1497	1497	3000	*1578*	1677	2482
	450	1452	1510	1459	1452	2716	1564	1756	2630
	500	1424	1498	1425	1424	2853	1509	1694	2482

Table A.3: Final Results: Comparison for 40 and 50 Vertices

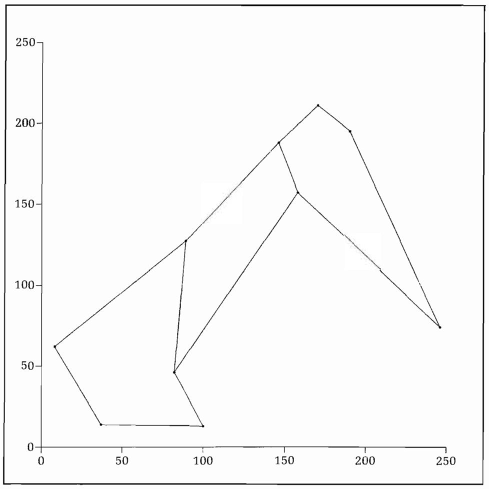
Data	K	BB	Stingy	Tabu	GA	NonOsc	ContOsc	PulOsc	Pher
40-1	200	2067	2549	2232	2320	3339	*2492*	2658	3212
	250	1800	2261	2031	2030	3763	2280	2447	3720
	300	1687	2213	1947	1854	3860	*2071*	2516	3699
	350	1616	1998	1691	1733	3678	*1911*	2463	3919
	400	1558	1699	1609	1627	4135	1749	2653	4116
	450	1533	1767	1571	1697	3602	1837	2935	4091
	500	1520	1751	1537	1589	3628	1865	2716	3769
40-2	300	1558	1737	1621	1617	3530	1855	2286	3157
	350	1496	1592	1514	1527	3592	1867	2385	3839
	400	1459	1544	1477	1535	3681	1686	2441	3881
	450	1434	1540	1462	1462	3315	1763	2809	3780
	500	1416	1505	1422	1416	3318	1763	2636	3793
40-3	200	2031	2421	2317	2355	3477	2528	2569	3187
	250	1821	2146	2077	2077	3597	2375	2515	3333
	300	1688	1897	1815	1815	3617	1974	2299	3423
	350	1620	1747	1654	1694	4407	1983	2611	4106
	400	1582	1649	1611	1649	4046	1864	2712	4183
	450	1561	1622	1575	1561	4048	1771	2679	4350
	500	1539	1622	1576	1539	4245	1747	2964	4267
40-4	200	1894	2159	2069	2163	3240	2254	2250	3168
	250	1718	1877	1791	1795	3268	2140	2445	3614
	300	1610	1768	1706	1688	3360	1872	2453	3709
	350	1551	1664	1616	1616	3655	1866	2563	3829
	400	1503	1603	1552	1572	4113	1800	2579	3800
	450	1476	1590	1524	1536	3514	1675	2818	3984
	500	1458	1511	1492	1475	3431	1811	2775	3602
40-5	200	1626	1747	1720	1703	2847	1850	1870	2640
	250	1455	1727	1657	1676	2850	1826	1977	3415
	300	1393	1712	1607	1544	3572	*1651*	2381	3543
	350	1356	1699	1575	1607	3101	1733	2477	3562
	400	1315	1699	1546	1546	3227	*1667*	2450	3550
	450	1266	1699	1422	1438	3177	*1656*	2464	3314
	500	1246	1699	1291	1246	3551	*1589*	2403	3911
50-1	150	2165	2367	2250	2286	3584	2498	2483	3103
	200	1776	2036	1968	1968	3261	2247	2713	3857
50-2	250	1884	2267	2200	2194	4434	2375	3508	5176
	300	1772	1914	1869	1881	4324	2312	3715	5738
50-3	200	1877	2236	2053	2116	3898	2386	3072	4655
	250	1777	2073	1896	1957	4351	2344	3384	5451
50-4	200	1852	2183	2090	2105	3743	2293	2972	4308
	250	1709	2105	1822	1822	4564	2184	3476	5123
50-5	200	1777	2155	1960	2084	3667	2292	2996	4655
	250	1650	1890	1835	1835	3925	2194	3600	5466

Appendix B

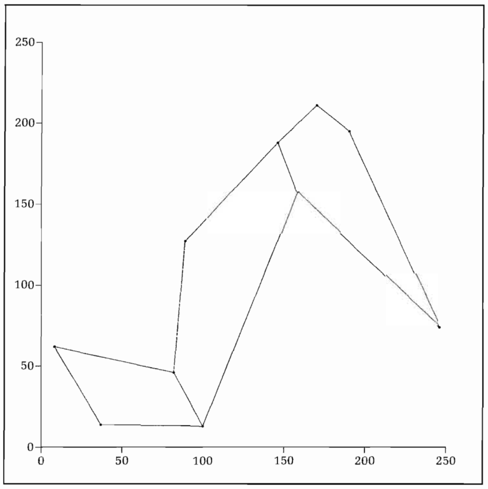
Priority-Based PSO Solution Plots

This appendix contains plots of the best solutions found by priority-based particle swarms for each k-bound of each problem instance.

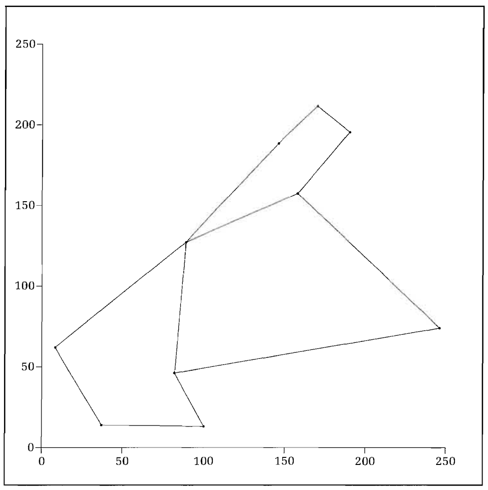
B.1 10 Vertices



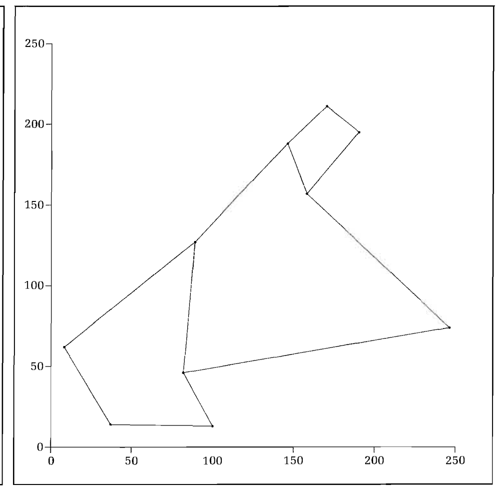
(a) 10-1(350)



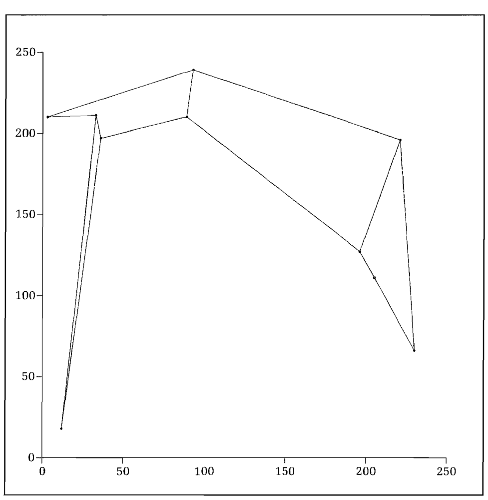
(b) 10-1(400)



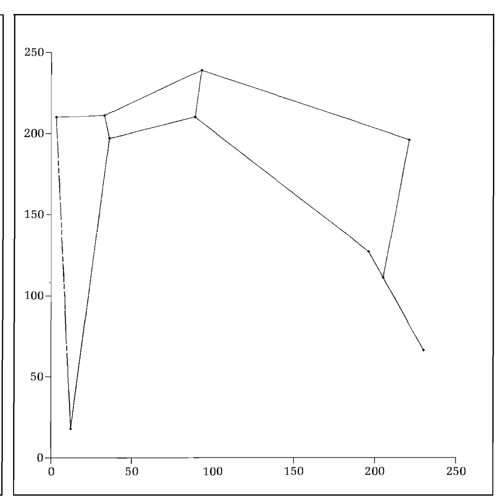
(c) 10-1(450)



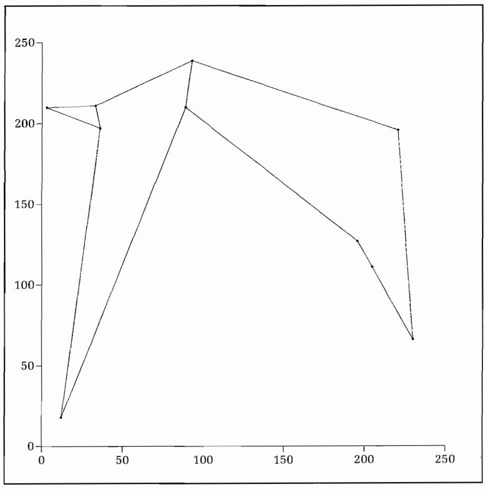
(d) 10-1(500)



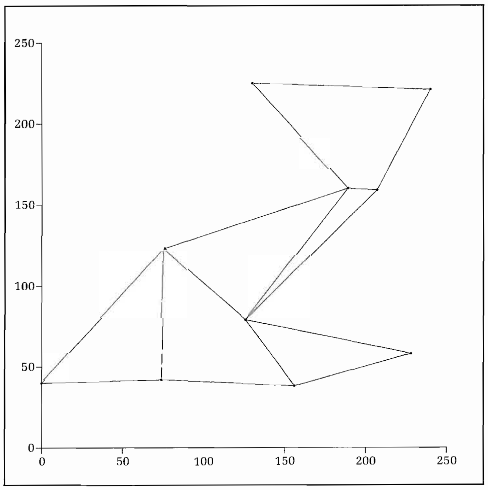
(a) 10-2(400)



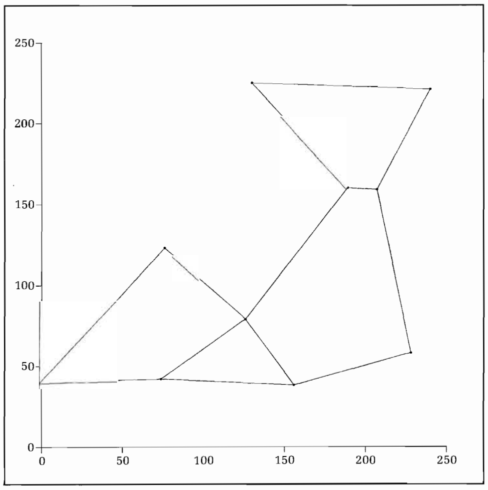
(b) 10-2(450)



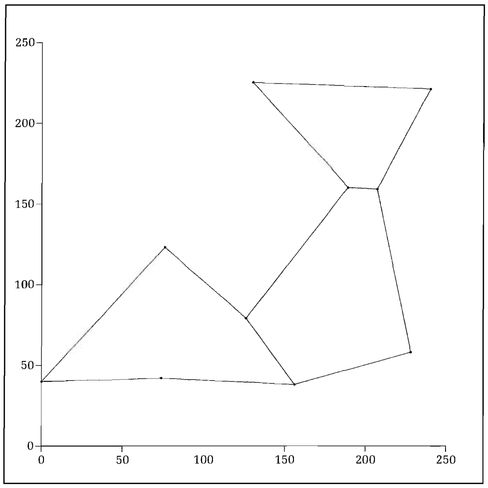
(c) 10-2(500)



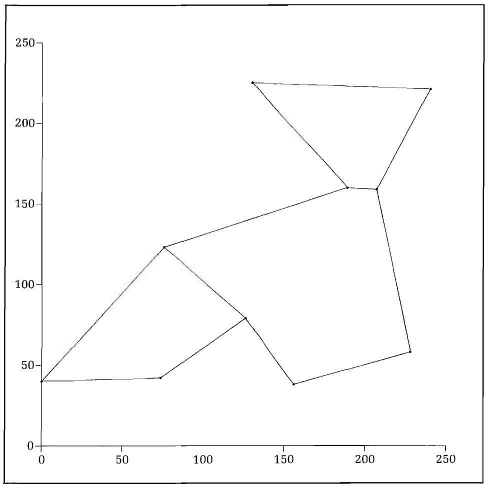
(a) 10-3(300)



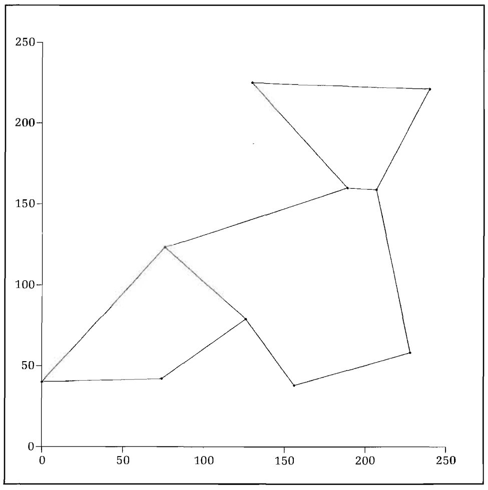
(b) 10-3(350)



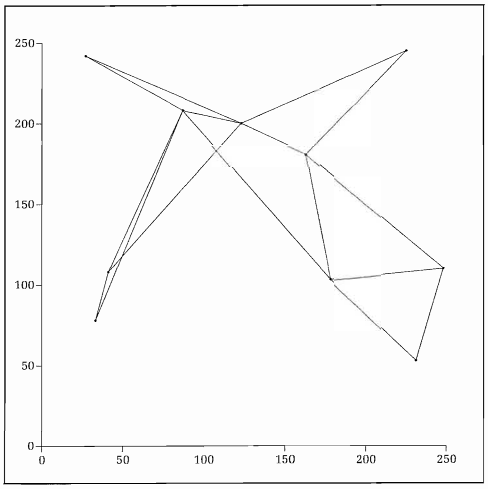
(c) 10-3(400)



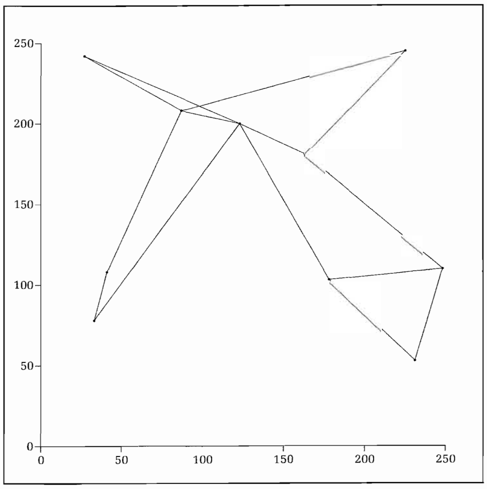
(d) 10-3(450)



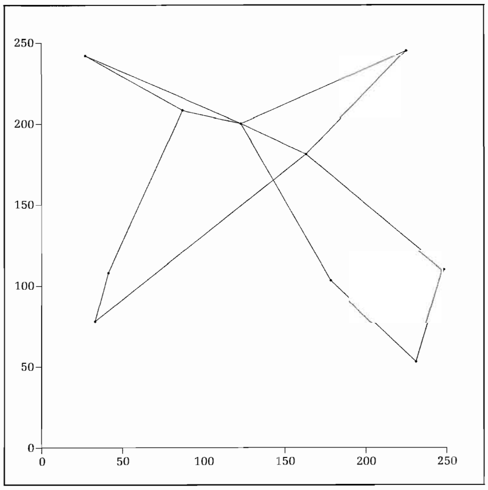
(e) 10-3(500)



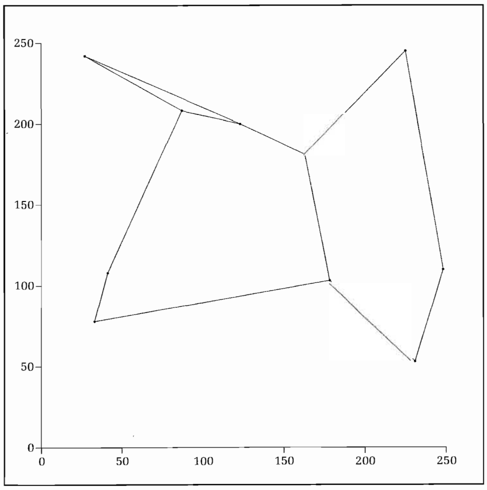
(a) 10-4(300)



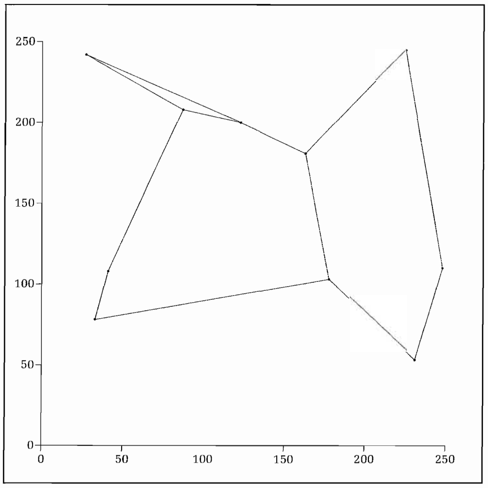
(b) 10-4(350)



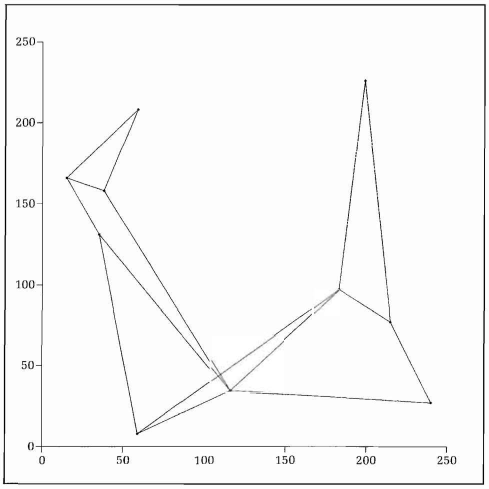
(c) 10-4(400)



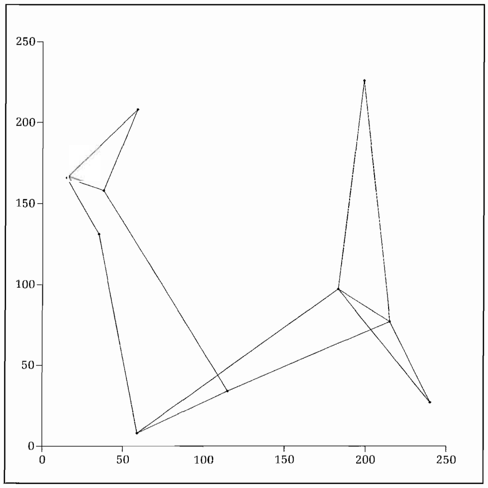
(d) 10-4(450)



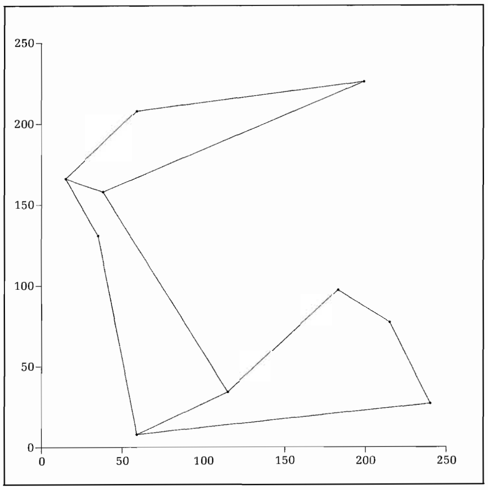
(e) 10-4(500)



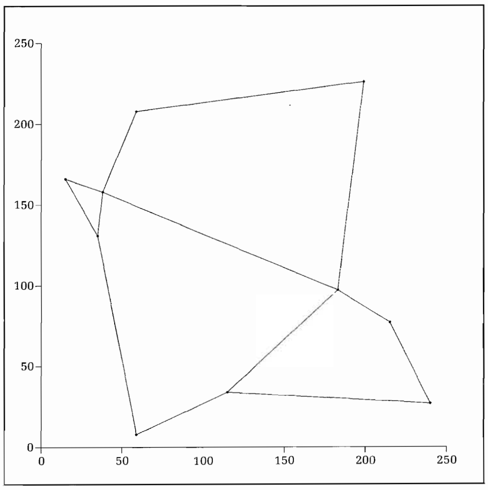
(a) 10-5(350)



(b) 10-5(400)

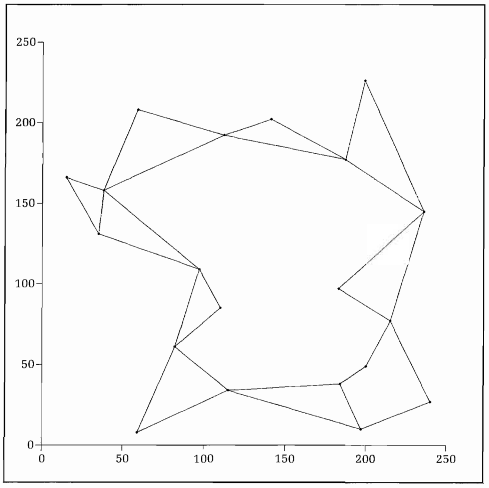


(c) 10-5(450)

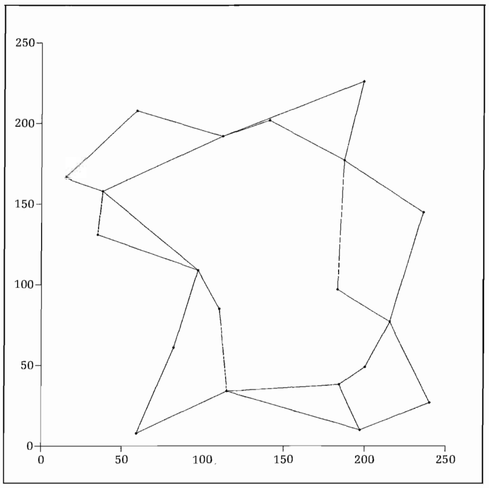


(d) 10-5(500)

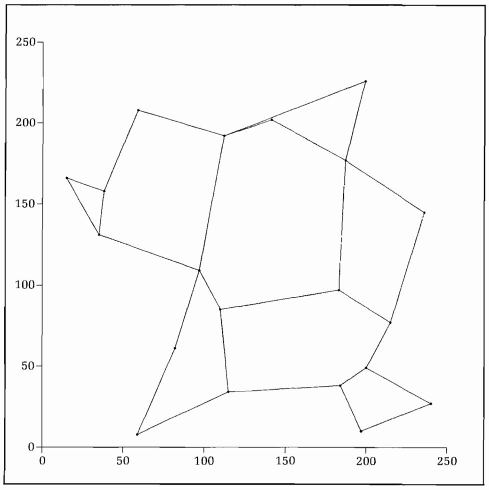
B.2 20 Vertices



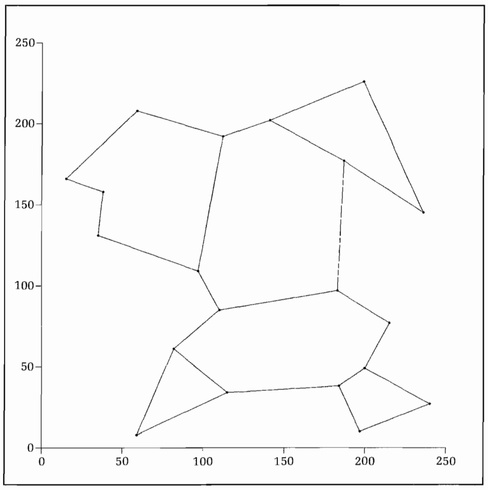
(a) 20-1(200)



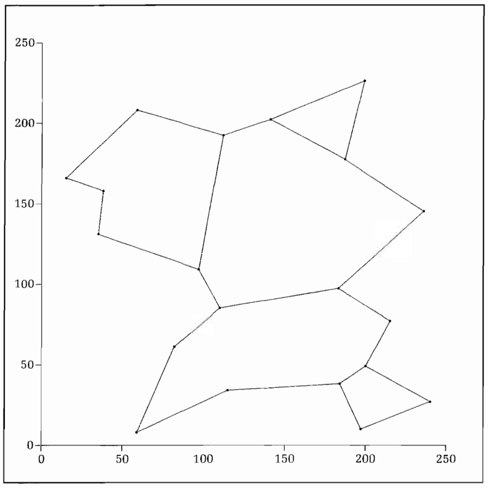
(b) 20-1(250)



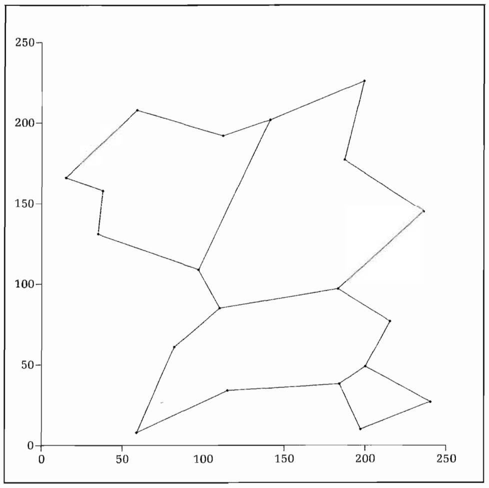
(c) 20-1(300)



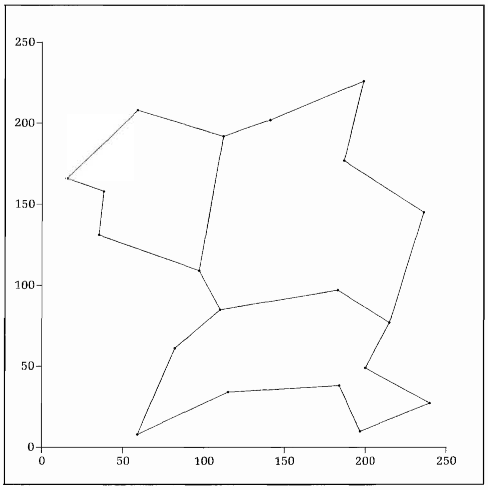
(d) 20-1(350)



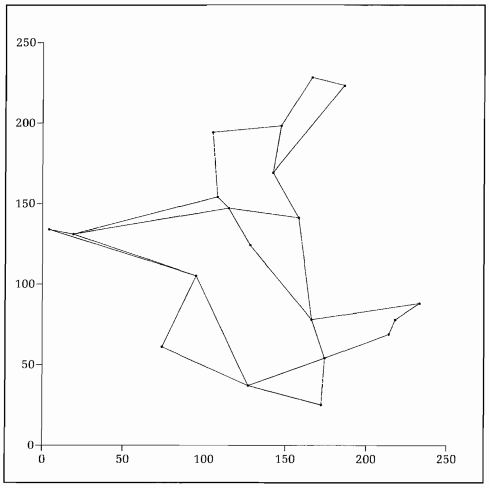
(e) 20-1(400)



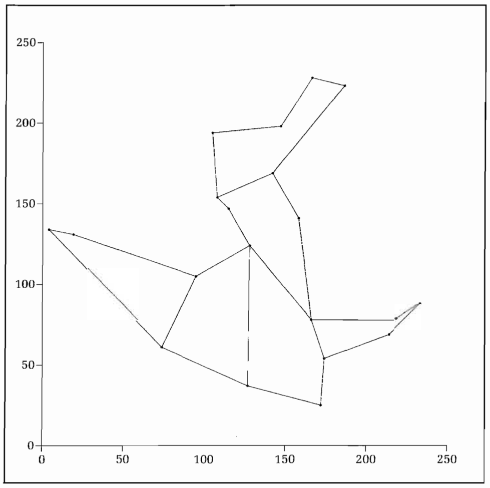
(f) 20-1(450)



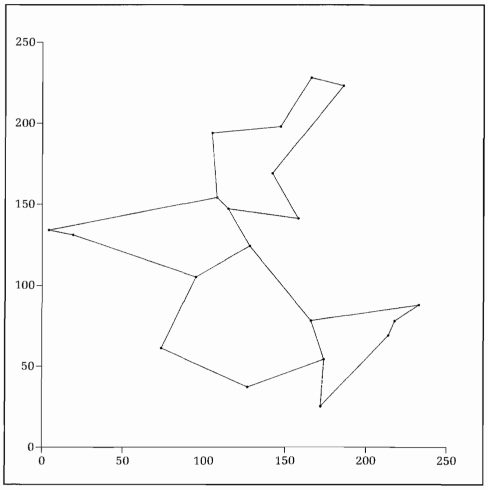
(g) 20-1(500)



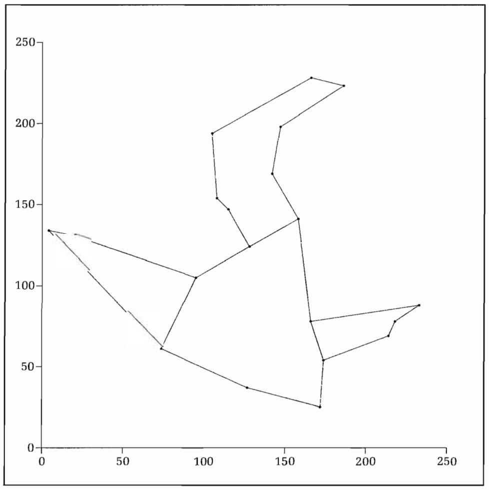
(a) 20-2(200)



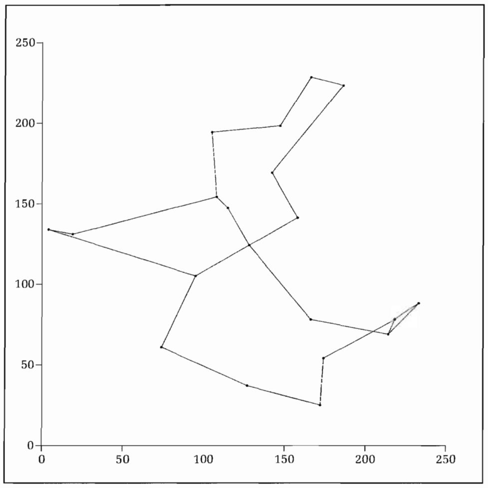
(b) 20-2(250)



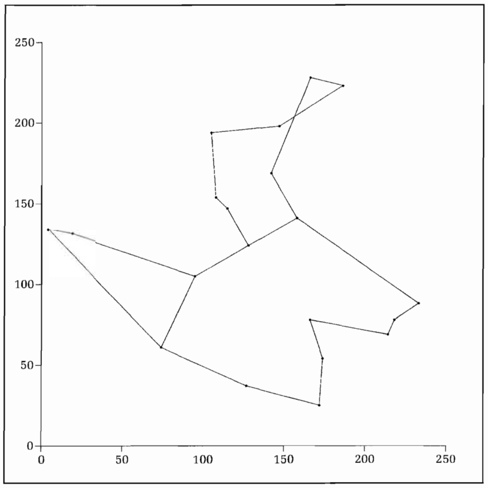
(c) 20-2(300)



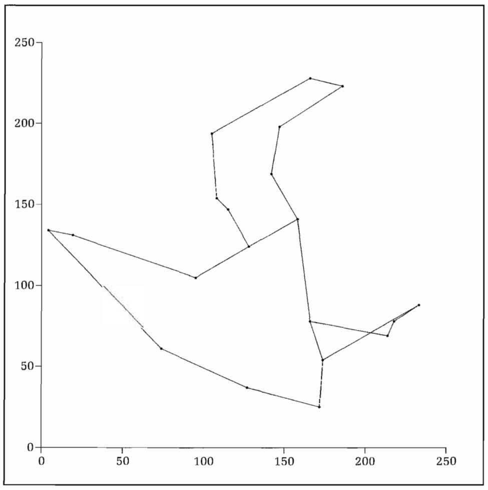
(d) 20-2(350)



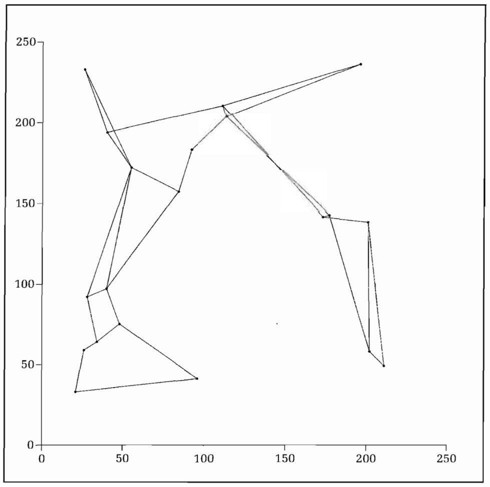
(e) 20-2(400)



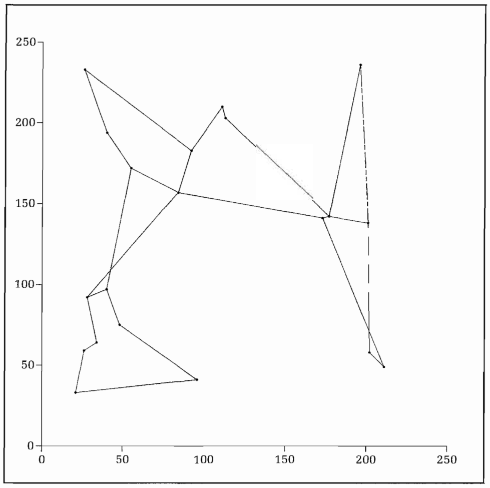
(f) 20-2(450)



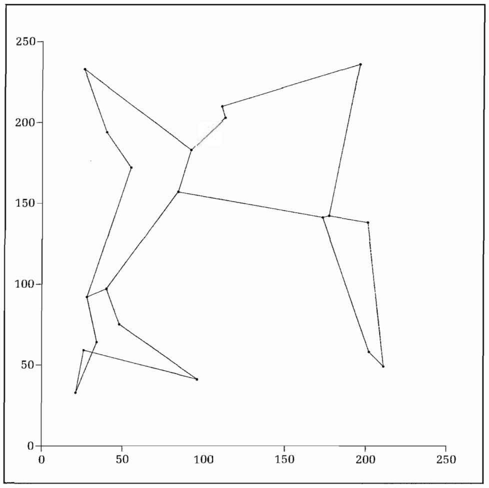
(g) 20-2(500)



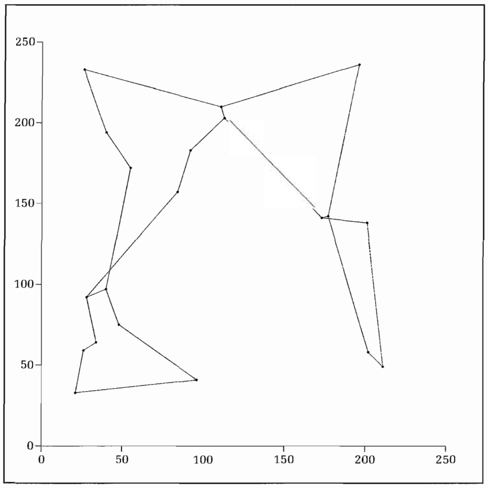
(a) 20-3(200)



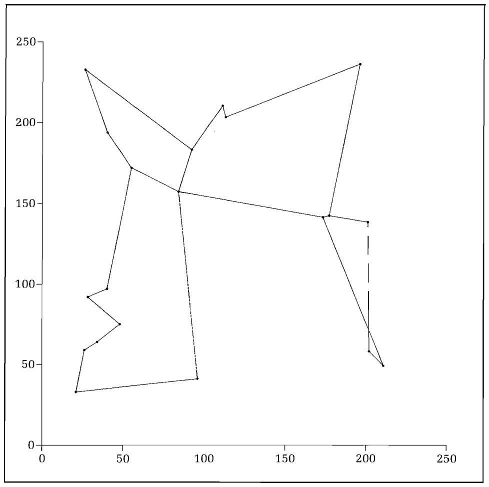
(b) 20-3(250)



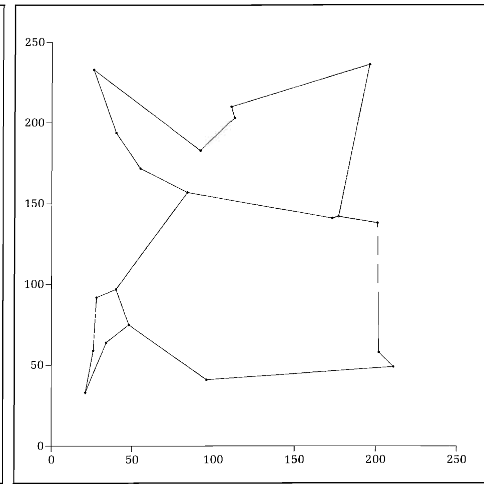
(c) 20-3(350)



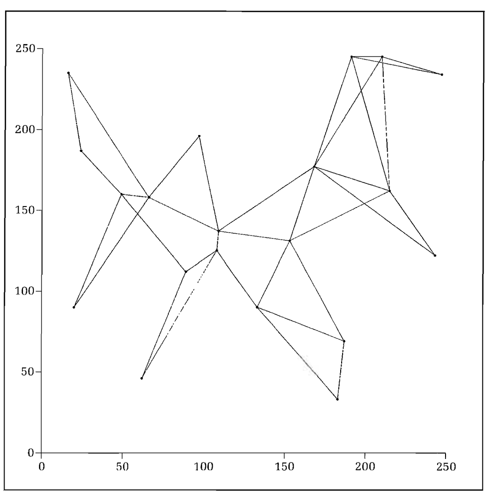
(d) 20-3(400)



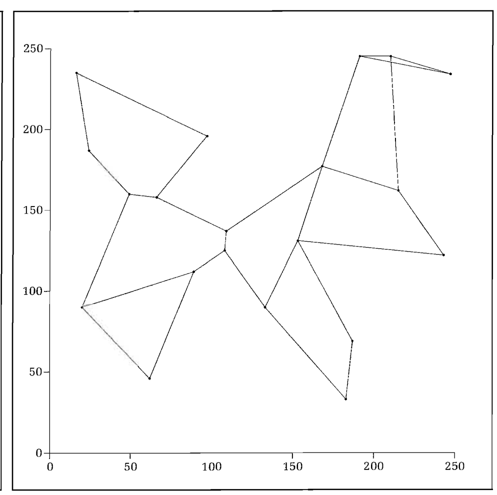
(e) 20-3(450)



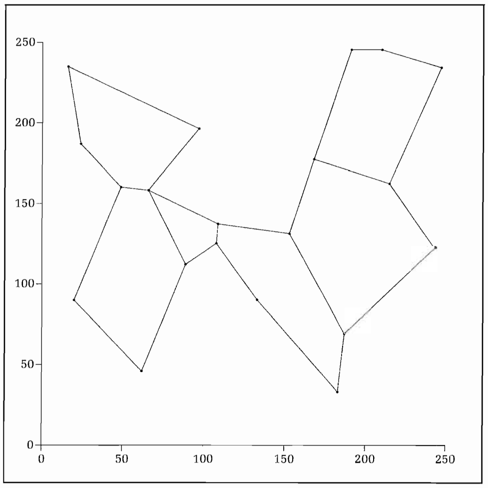
(f) 20-3(500)



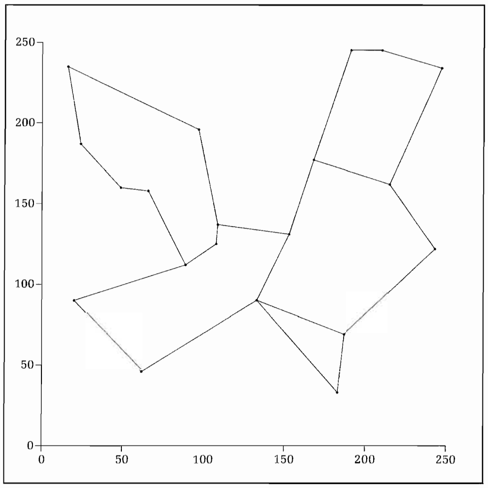
(a) 20-4(200)



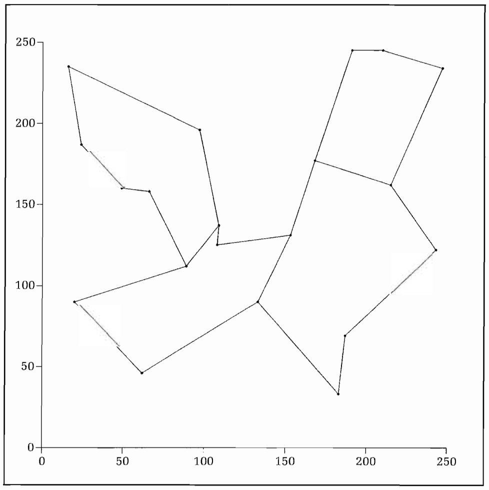
(b) 20-4(250)



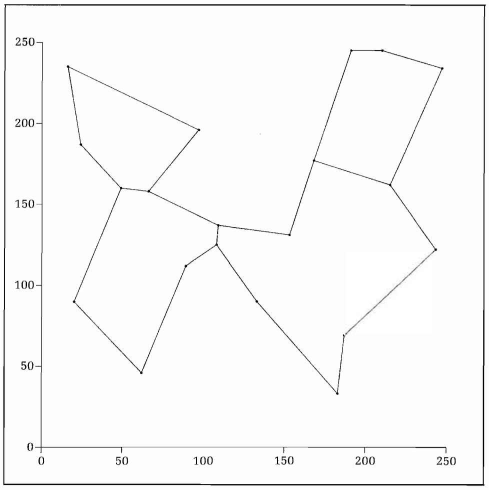
(c) 20-4(300)



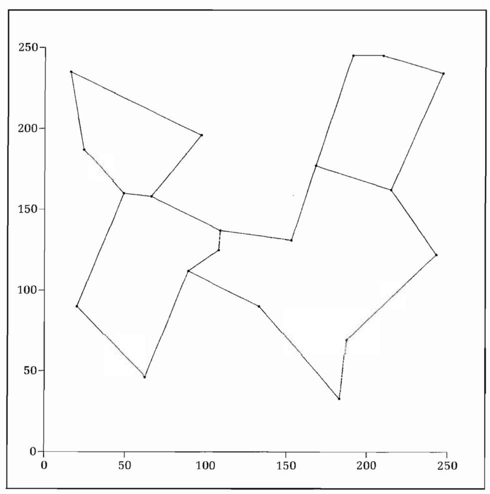
(d) 20-4(350)



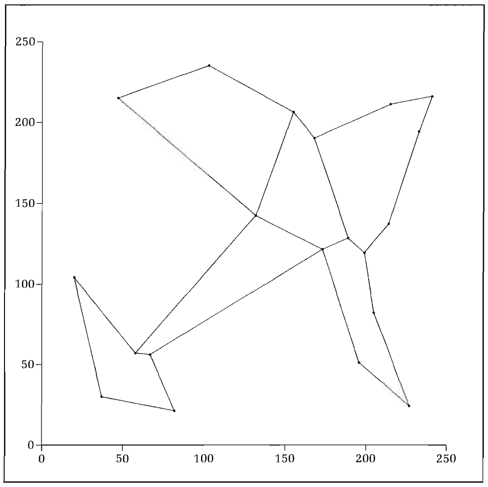
(e) 20-4(400)



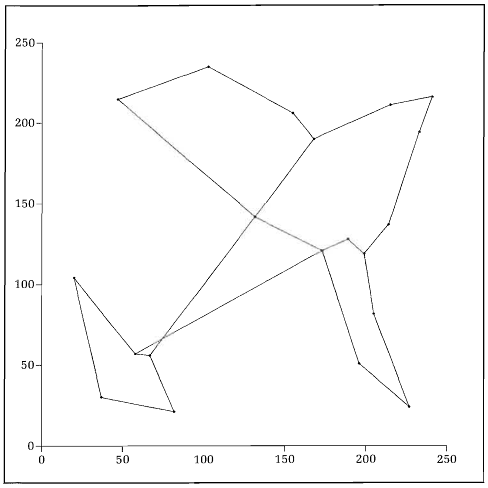
(f) 20-4(450)



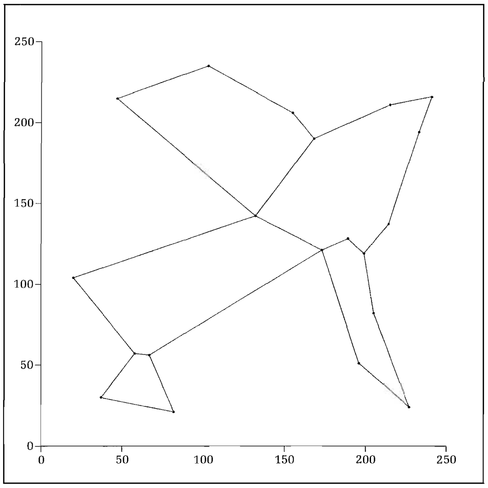
(g) 20-4(500)



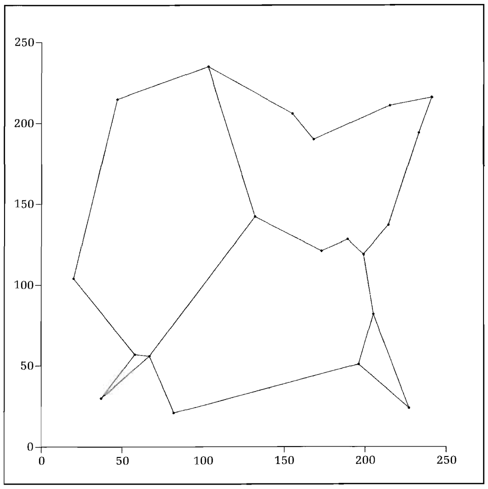
(a) 20-5(300)



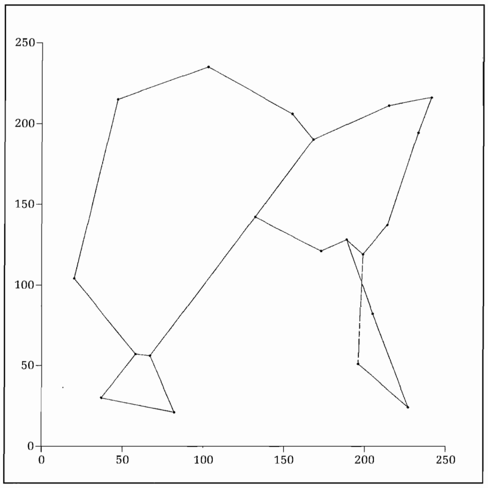
(b) 20-5(350)



(c) 20-5(400)

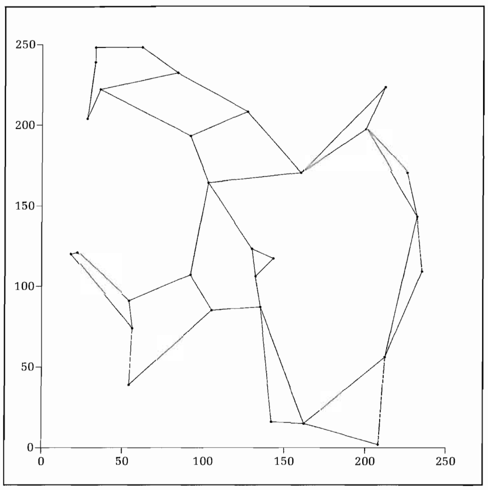


(d) 20-5(450)

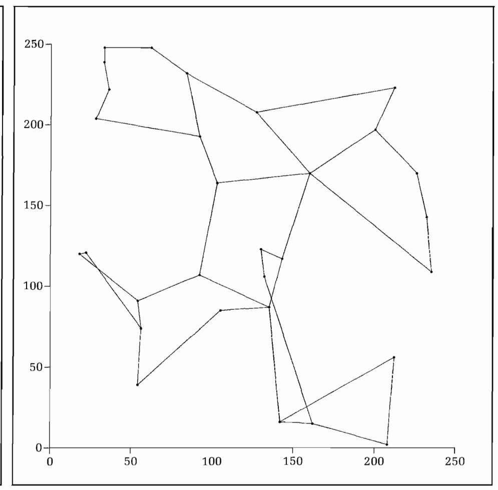


(e) 20-5(500)

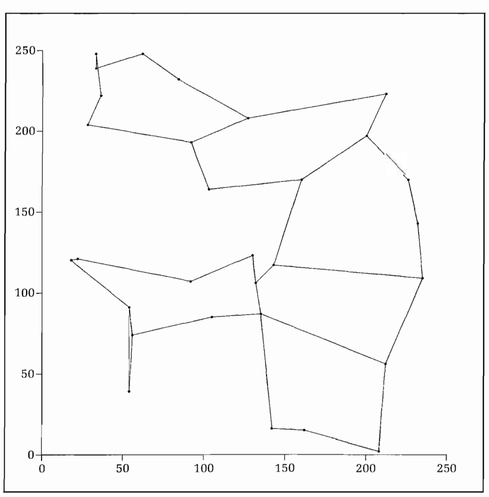
B.3 30 Vertices



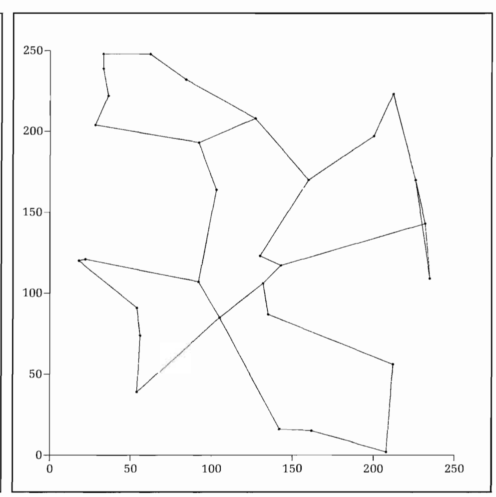
(a) 30-1(200)



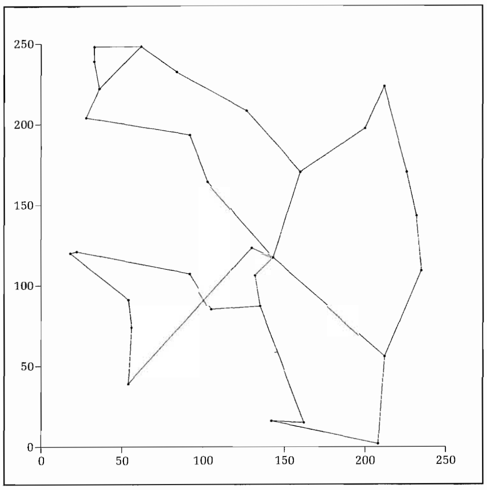
(b) 30-1(250)



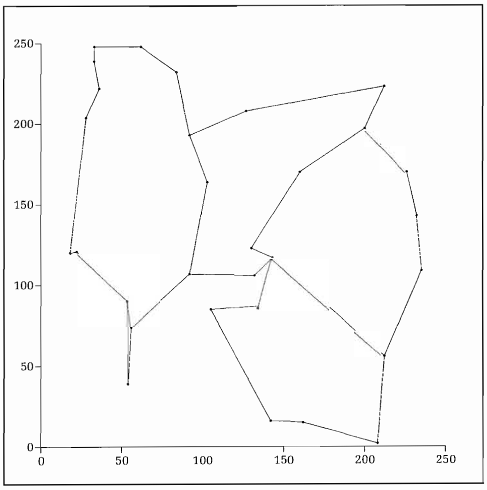
(c) 30-1(300)



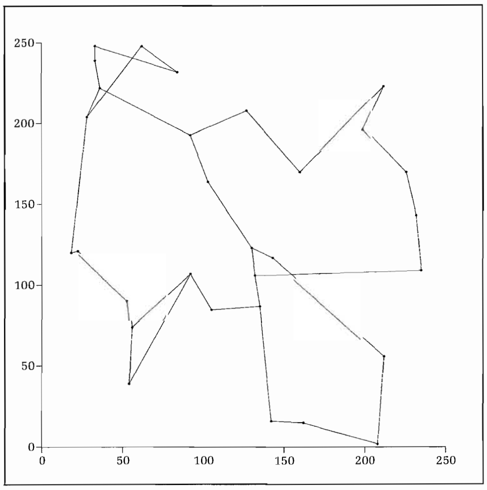
(d) 30-1(350)



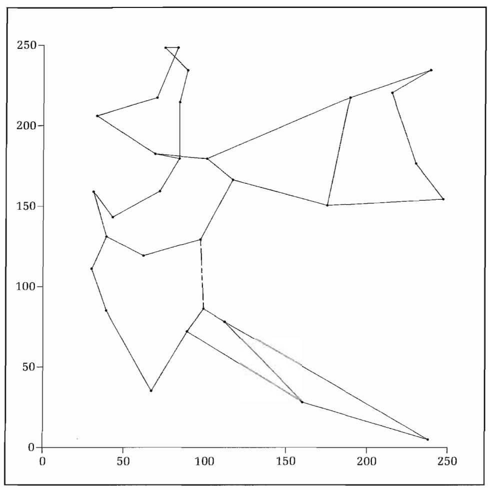
(e) 30-1(400)



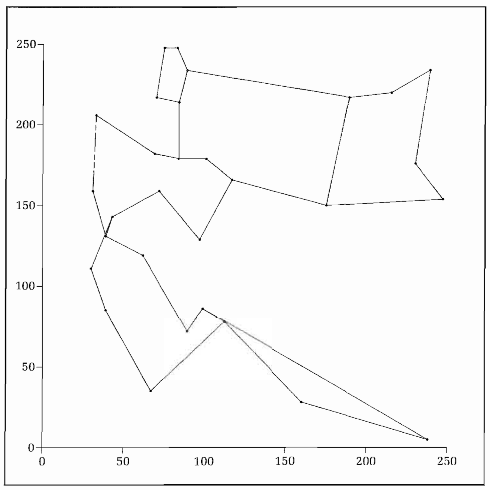
(f) 30-1(450)



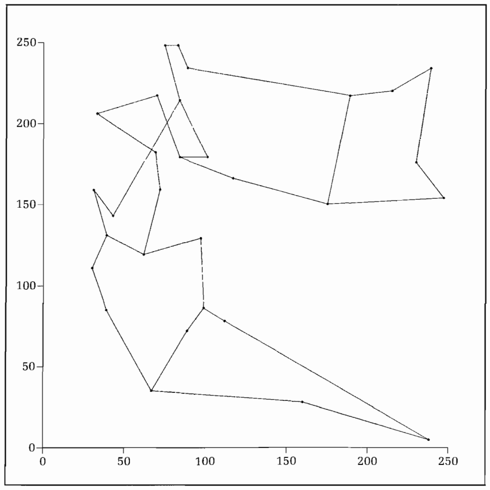
(g) 30-1(500)



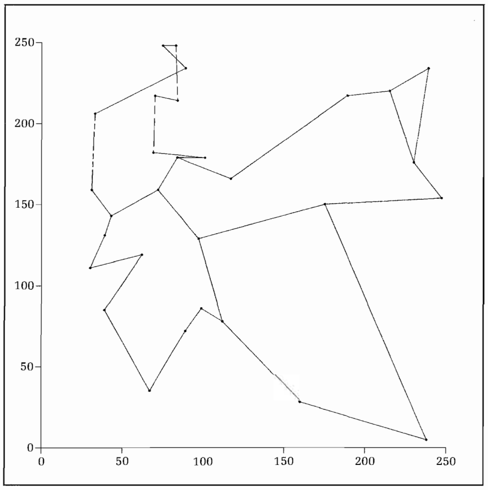
(a) 30-2(300)



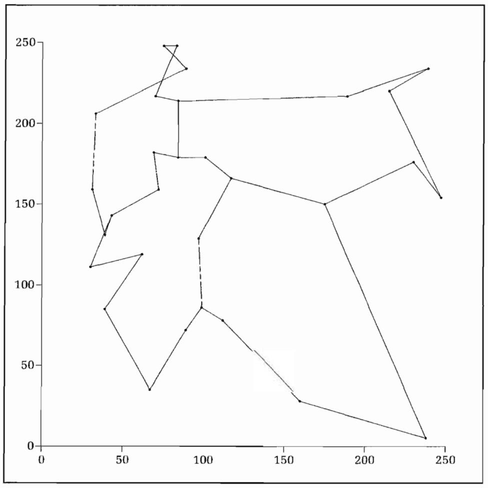
(b) 30-2(350)



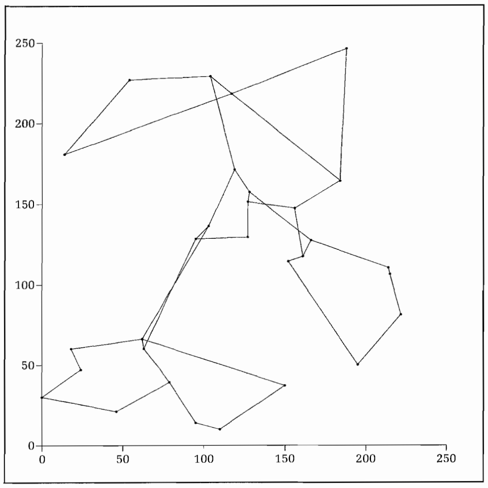
(c) 30-2(400)



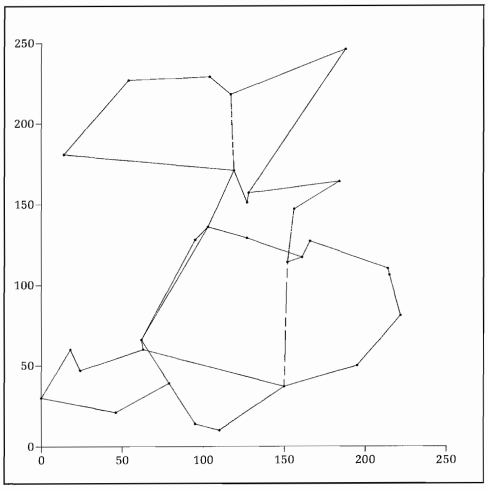
(d) 30-2(450)



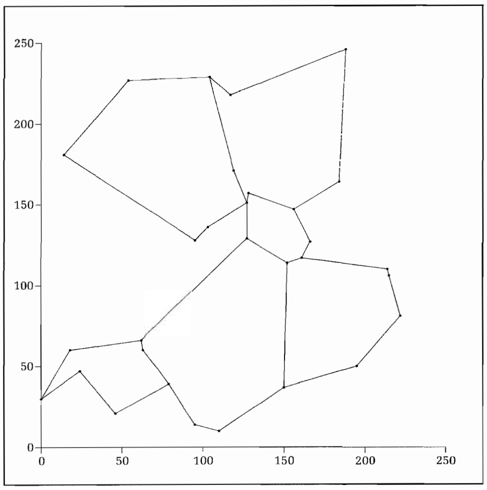
(e) 30-2(500)



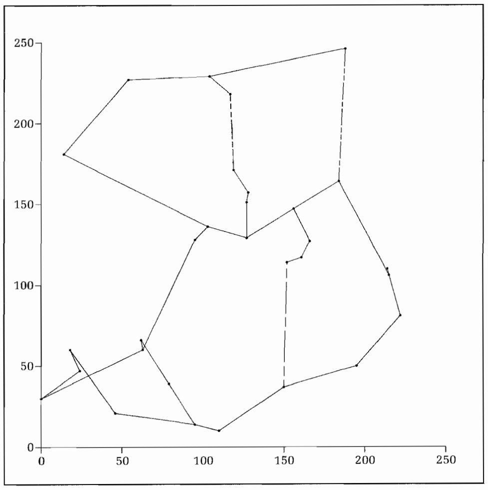
(a) 30-3(250)



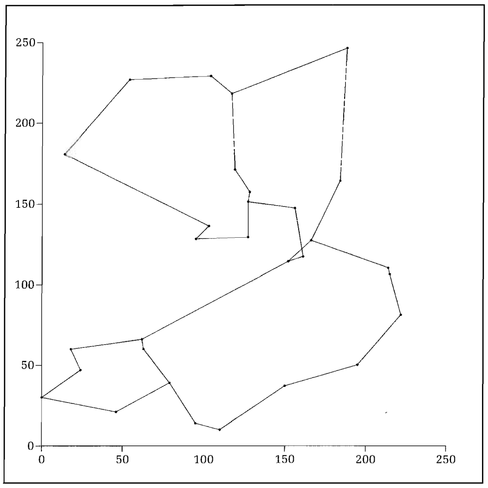
(b) 30-3(300)



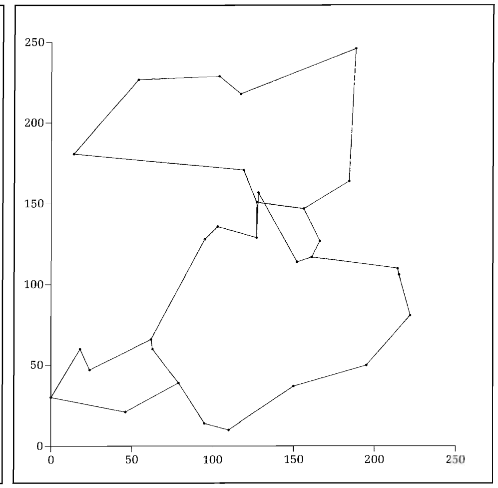
(c) 30-3(350)



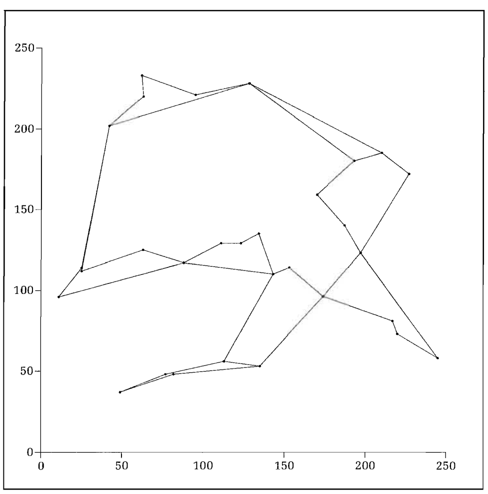
(d) 30-3(400)



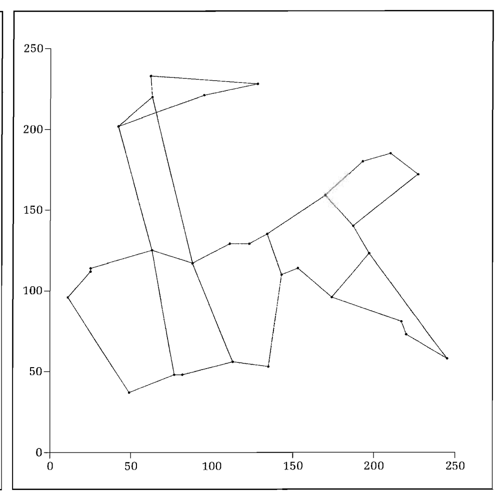
(e) 30-3(450)



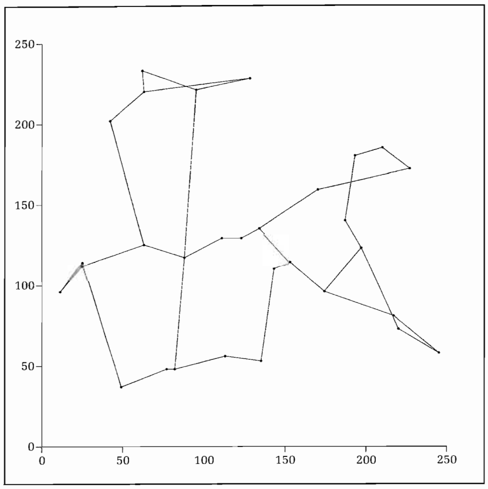
(f) 30-3(500)



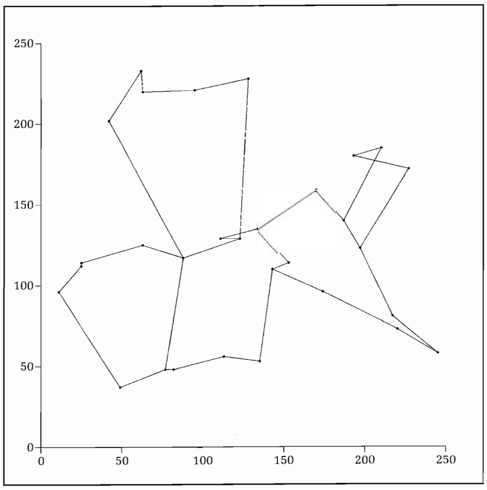
(a) 30-4(200)



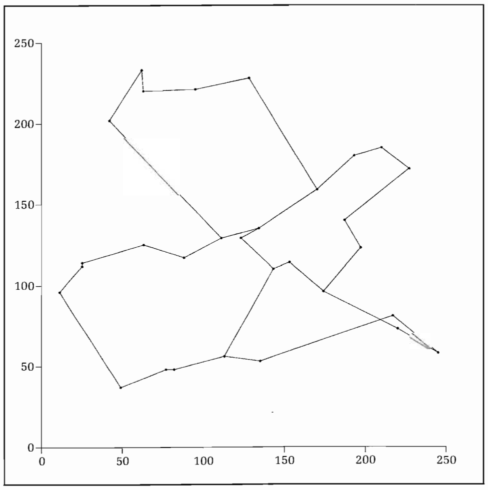
(b) 30-4(250)



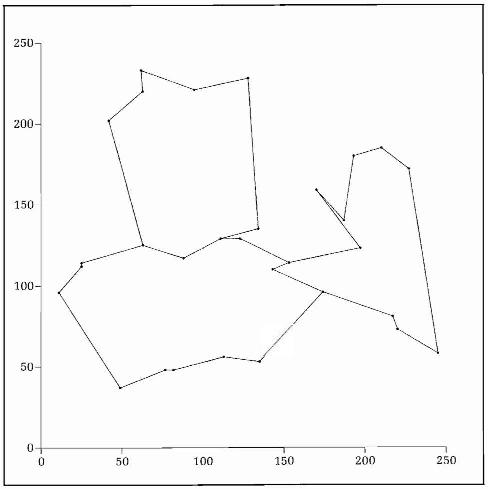
(c) 30-4(300)



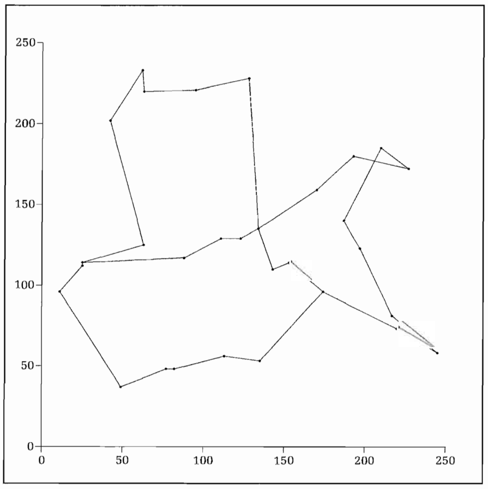
(d) 30-4(350)



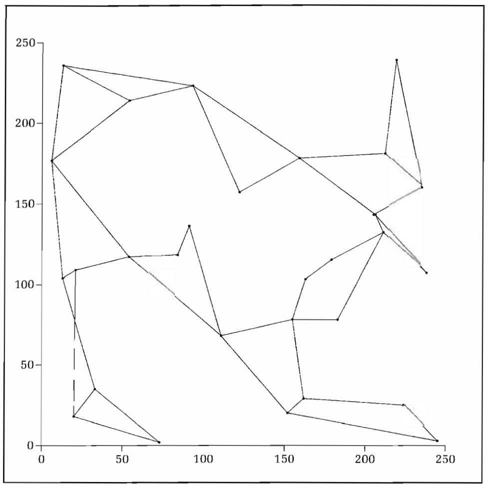
(e) 30-4(400)



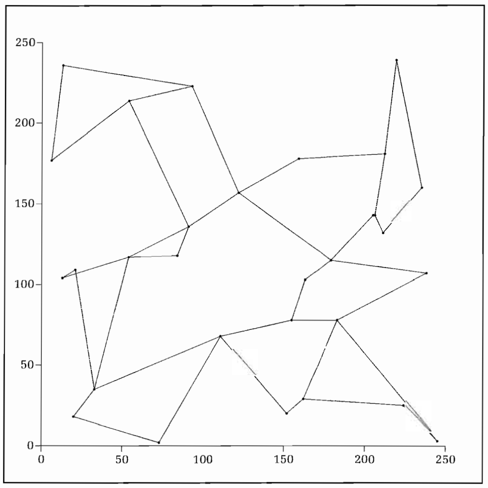
(f) 30-4(450)



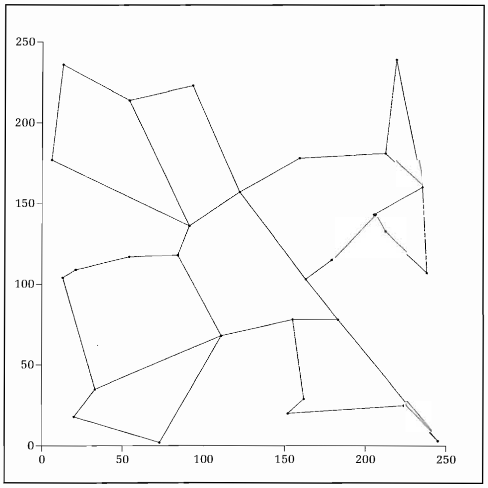
(g) 30-4(500)



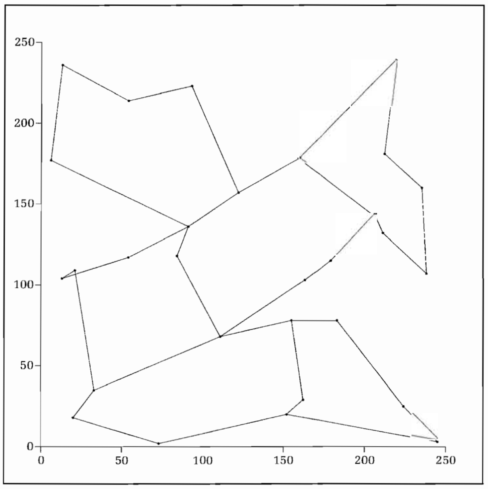
(a) 30-5(200)



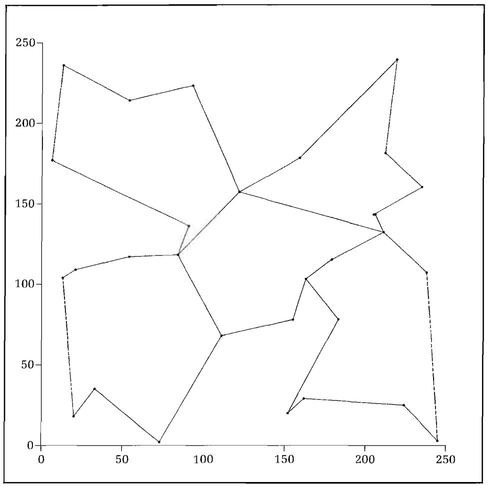
(b) 30-5(250)



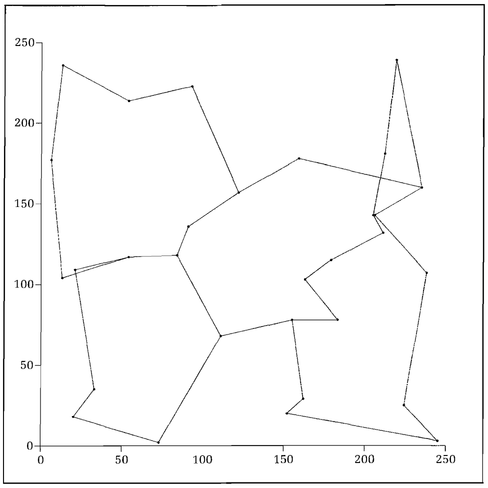
(c) 30-5(300)



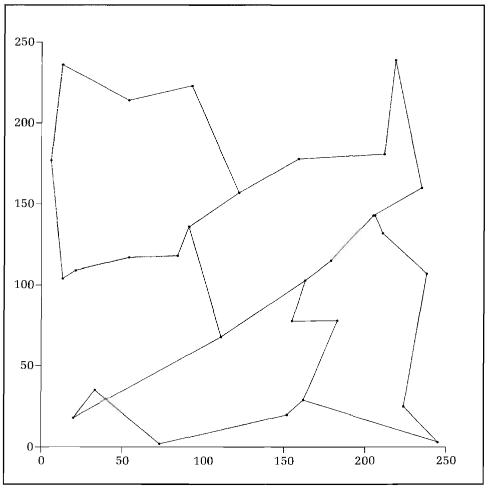
(d) 30-5(350)



(e) 30-5(400)

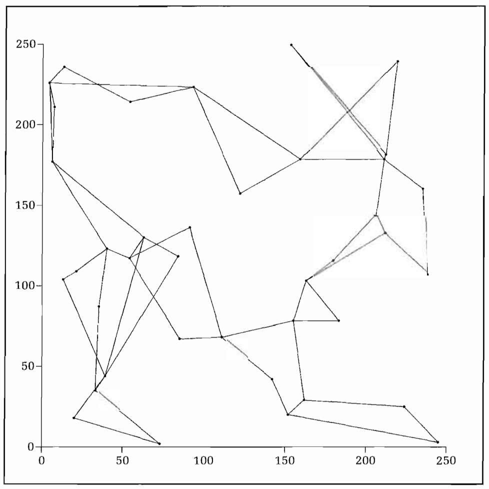


(f) 30-5(450)

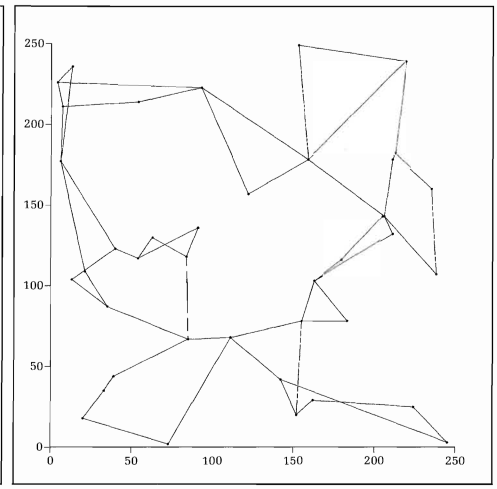


(g) 30-5(500)

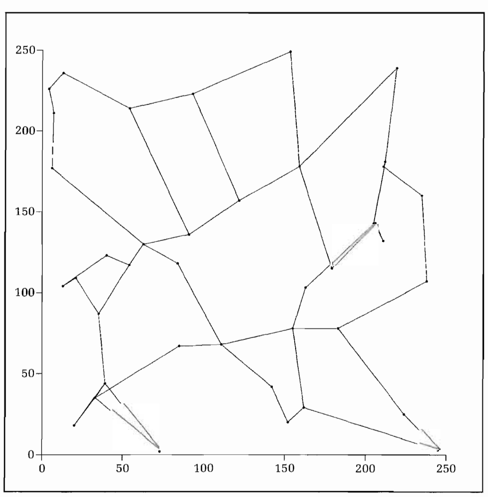
B.4 40 Vertices



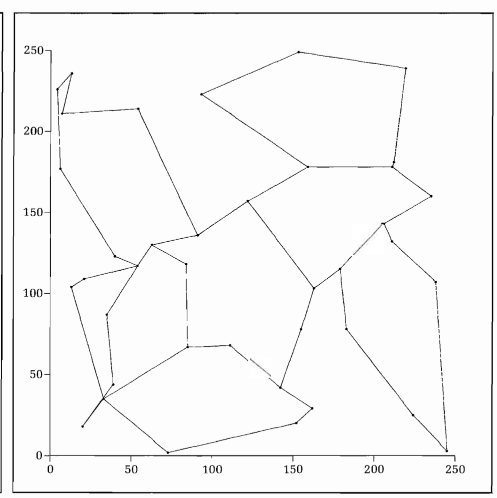
(a) 40-1(200)



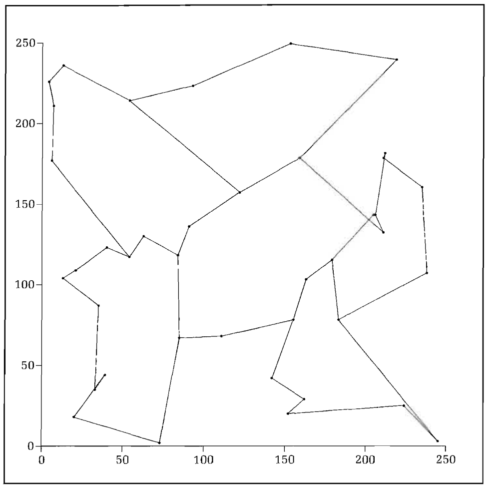
(b) 40-1(250)



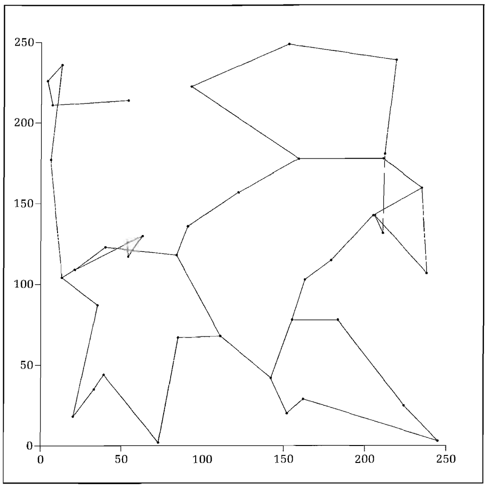
(c) 40-1(300)



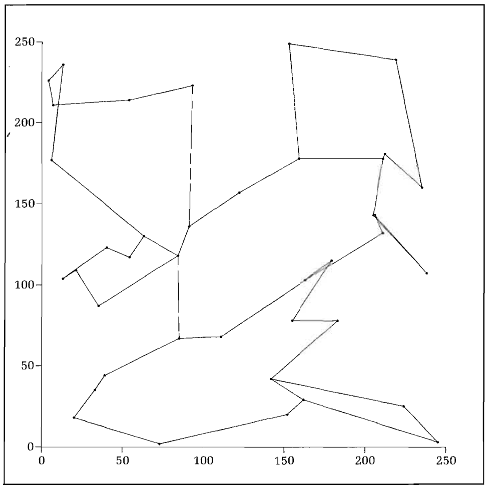
(d) 40-1(350)



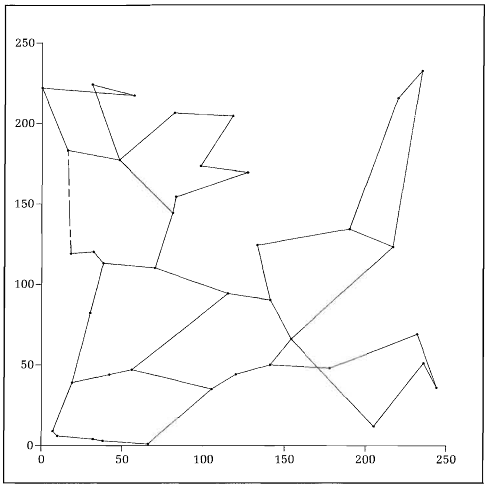
(e) 40-1(400)



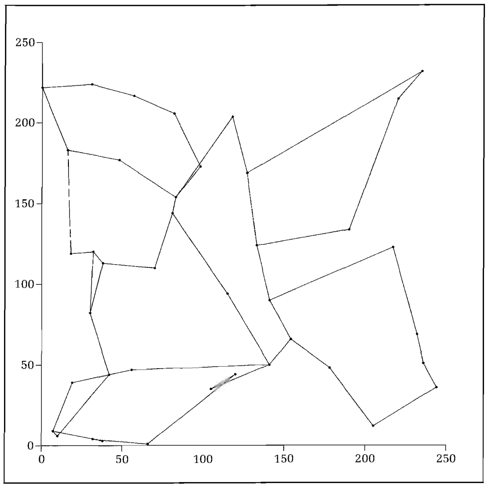
(f) 40-1(450)



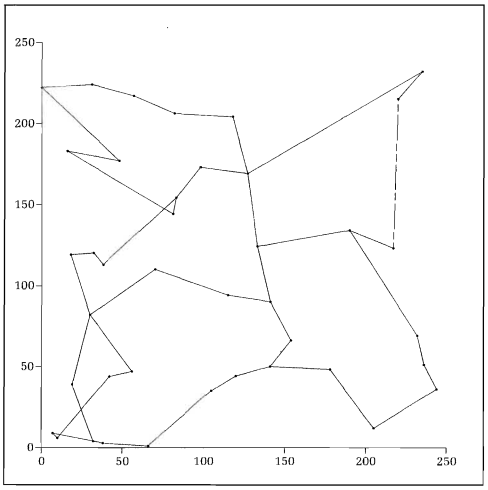
(g) 40-1(500)



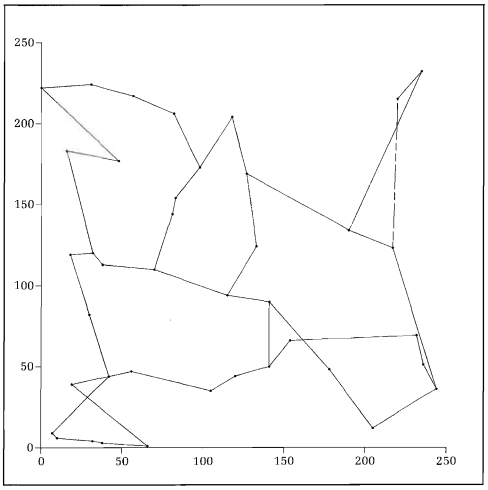
(a) 40-2(300)



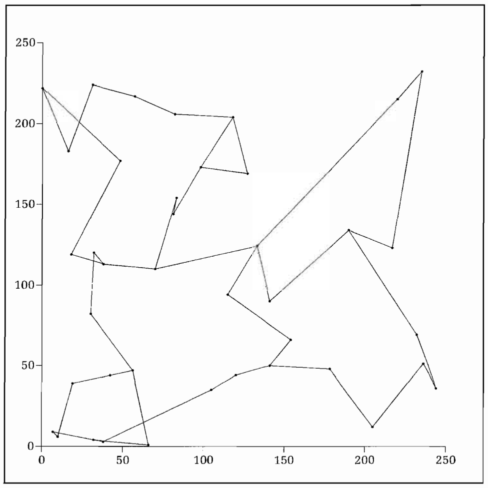
(b) 40-2(350)



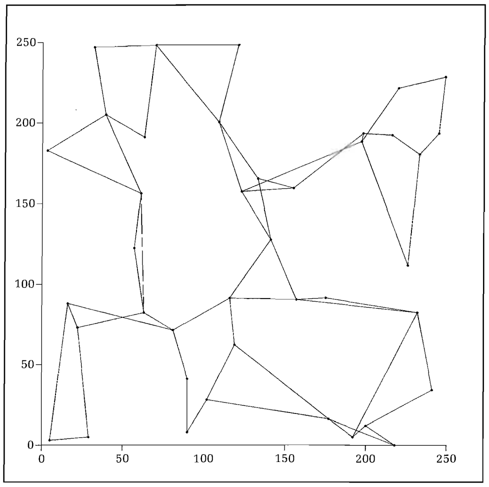
(c) 40-2(400)



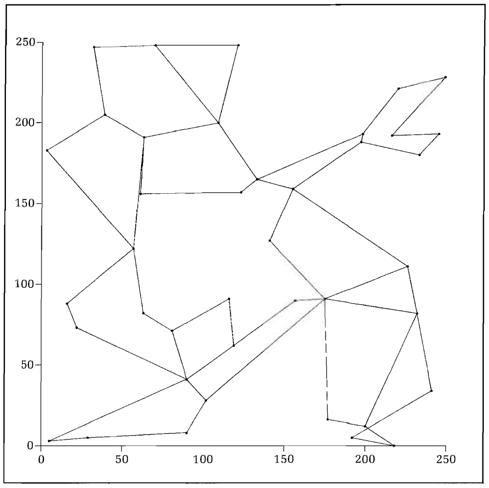
(d) 40-2(450)



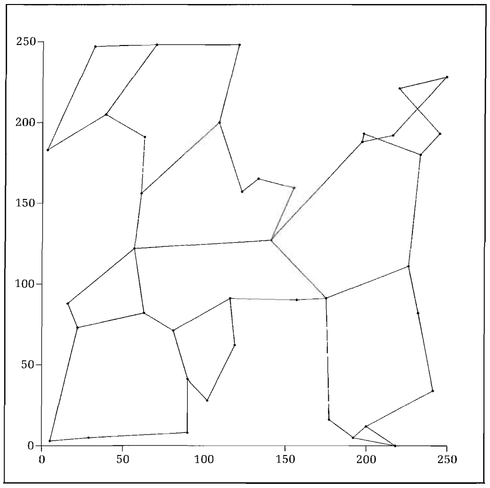
(e) 40-2(500)



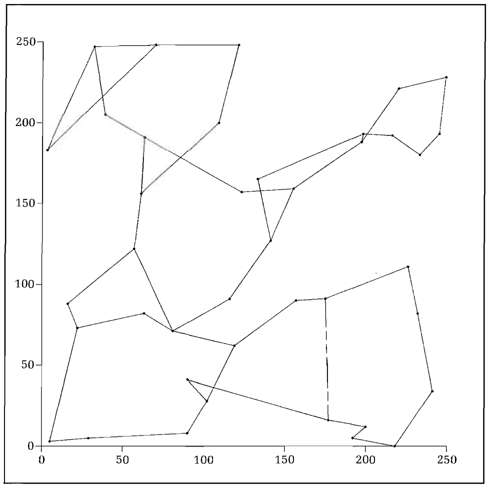
(a) 40-3(200)



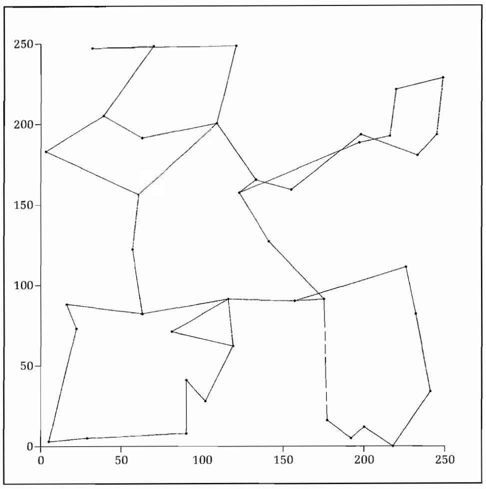
(b) 40-3(250)



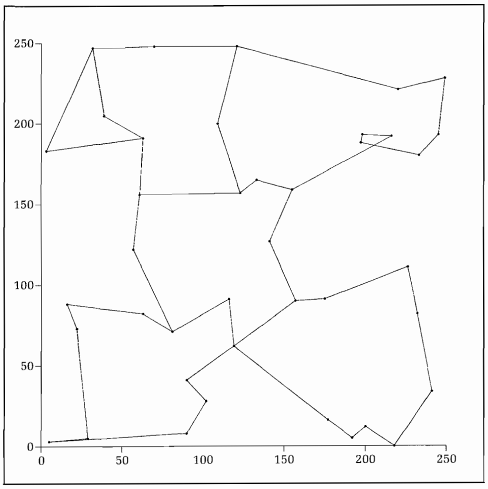
(c) 40-3(300)



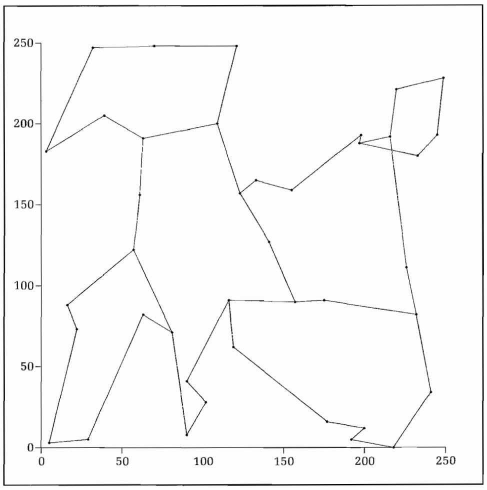
(d) 40-3(350)



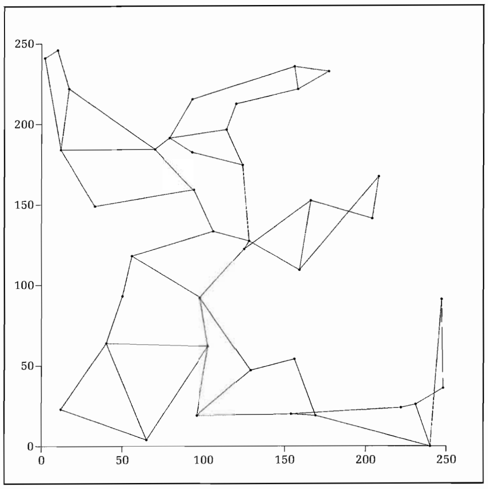
(e) 40-3(400)



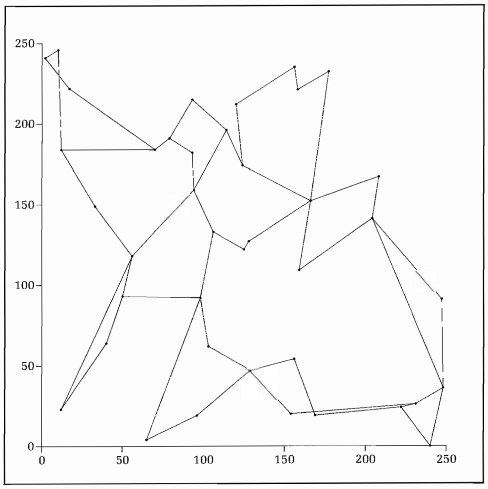
(f) 40-3(450)



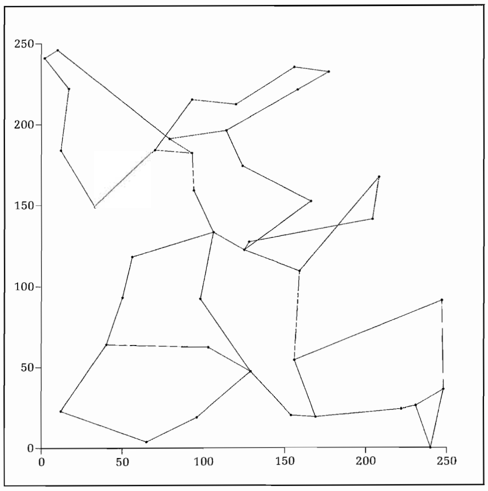
(g) 40-3(500)



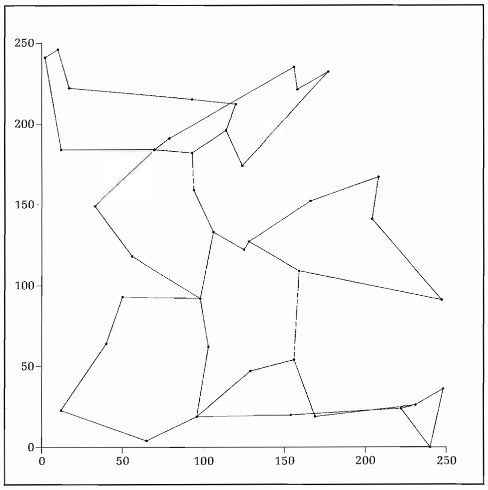
(a) 40-4(200)



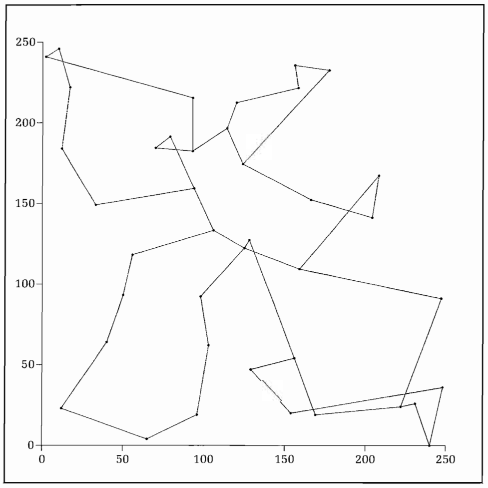
(b) 40-4(250)



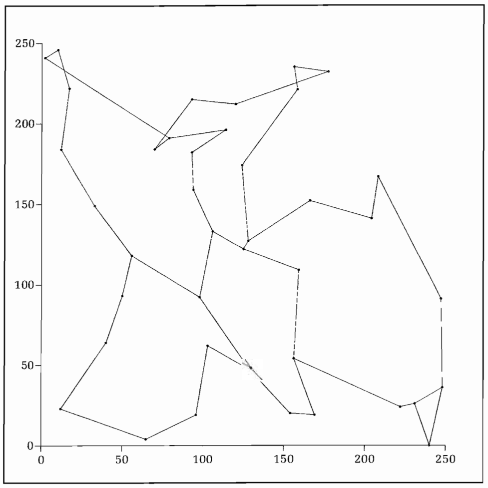
(c) 40-4(300)



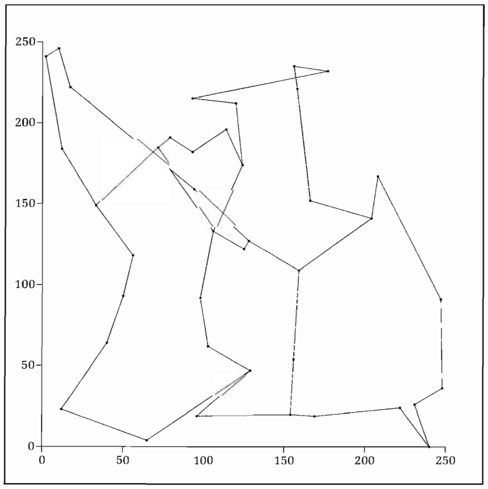
(d) 40-4(350)



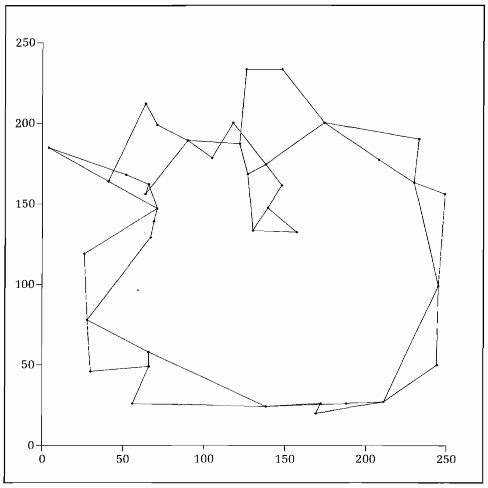
(e) 40-4(400)



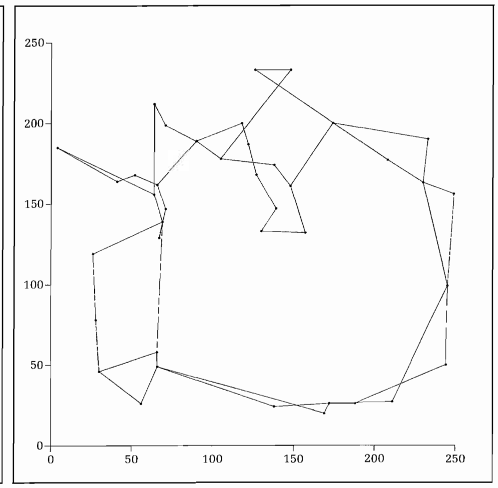
(f) 40-4(450)



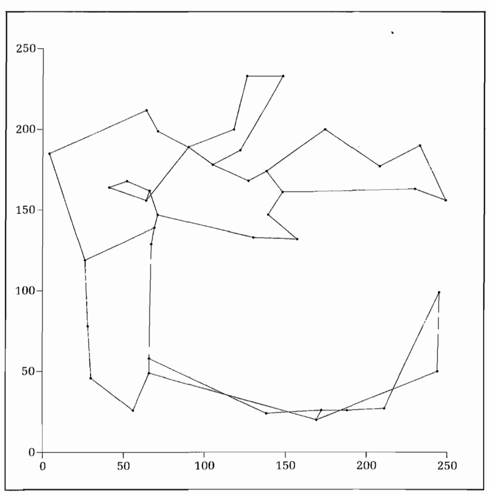
(g) 40-4(500)



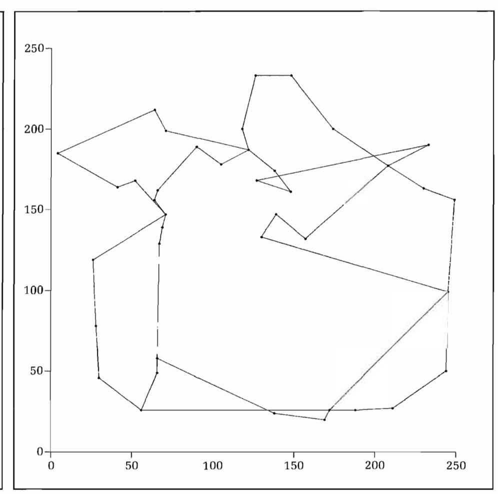
(a) 40-5(200)



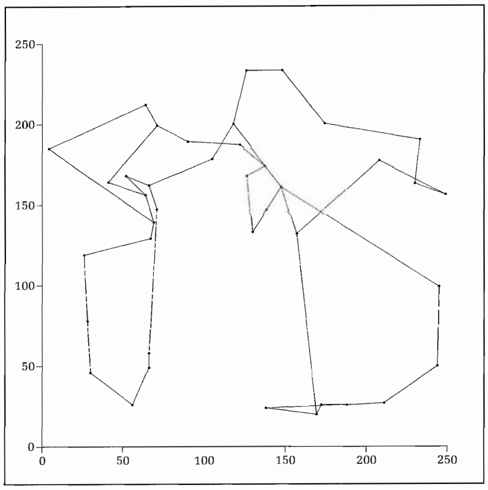
(b) 40-5(250)



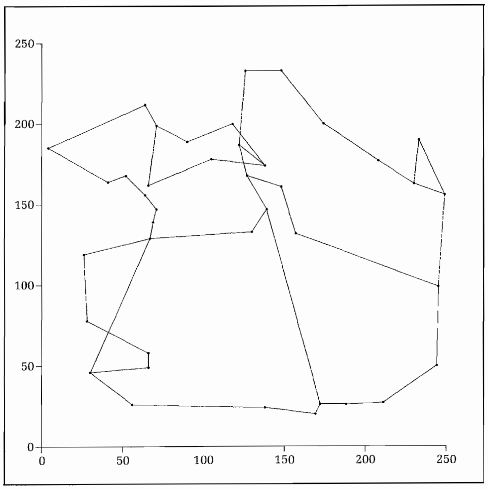
(c) 40-5(300)



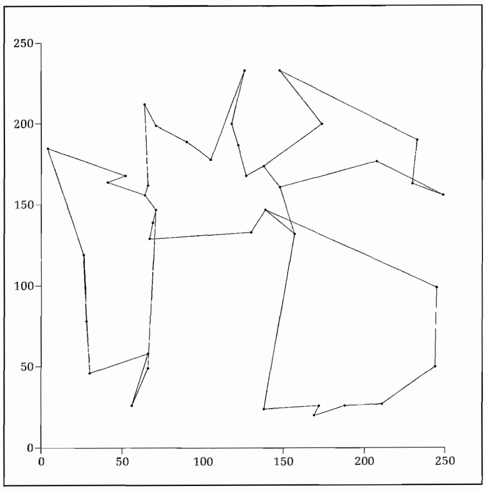
(d) 40-5(350)



(e) 40-5(400)

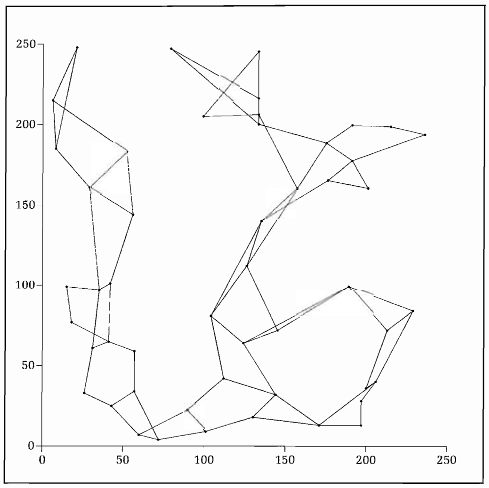


(f) 40-5(450)

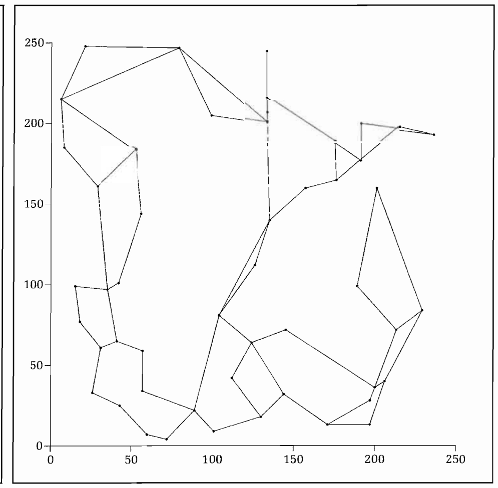


(g) 40-5(500)

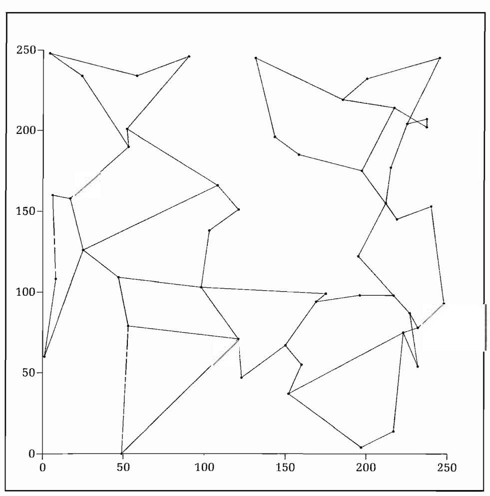
B.5 50 Vertices



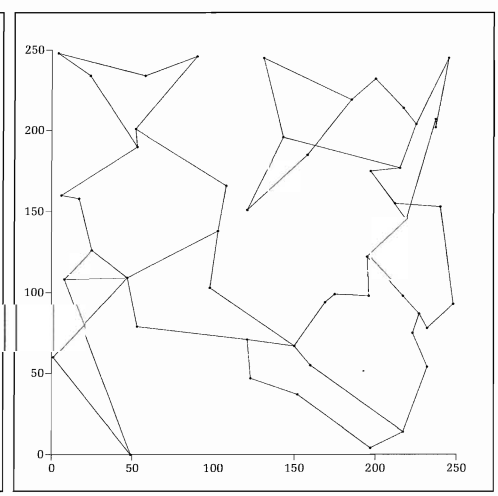
(a) 50-1(150)



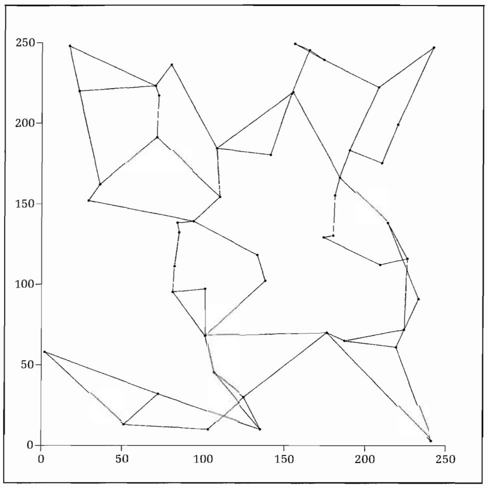
(b) 50-1(200)



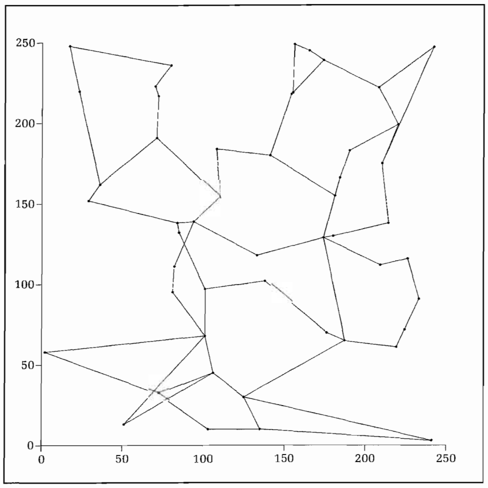
(a) 50-2(250)



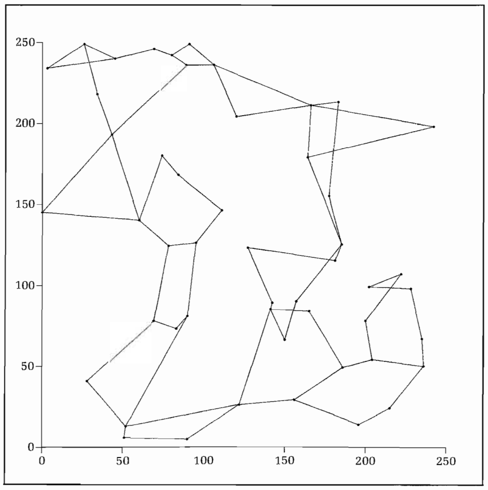
(b) 50-2(300)



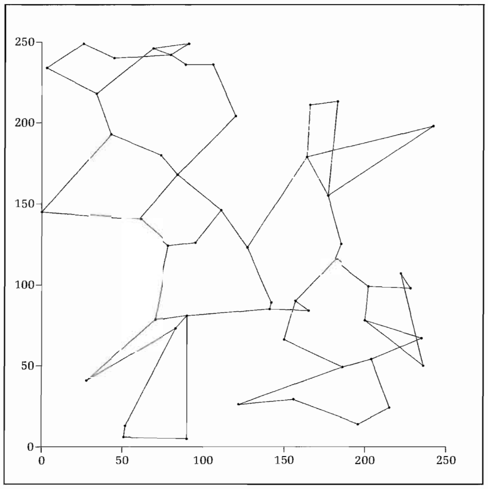
(a) 50-3(200)



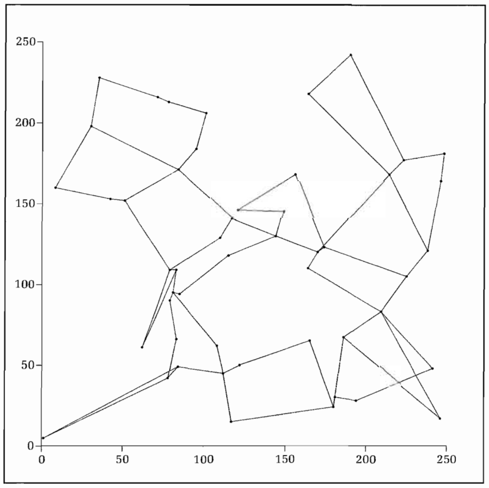
(b) 50-3(250)



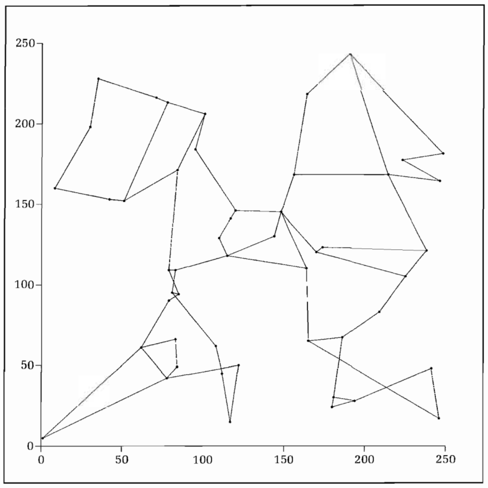
(a) 50-4(200)



(b) 50-4(250)



(a) 50-5(200)



(b) 50-5(250)

Appendix C

Graphical Comparisons of All Techniques

This appendix contains additional graphs comparing the cumulative effectiveness of the techniques previously used by Fortz, Ombuki, and Ventresca against the non-oscillating, continuously oscillating, and pulse oscillating priority-based particle swarms and the pheromone-driven particle swarm explored in this thesis. Each chart shows the cumulative network costs of the solutions found across all provided k-bounds for a problem instance.

C.1 10 Vertices

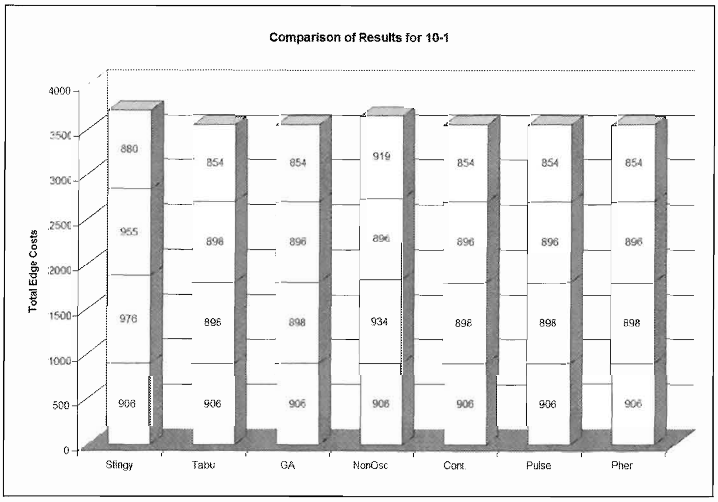


Figure C.1: Comparison of cumulative performance for problem 10-1

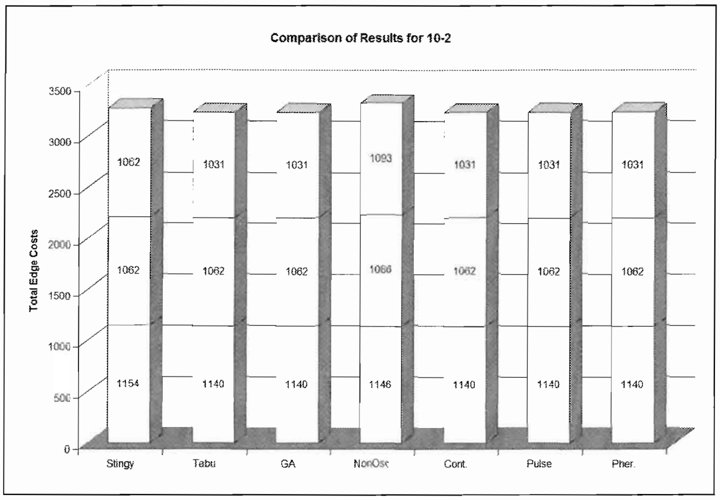


Figure C.2: Comparison of cumulative performance for problem 10-2

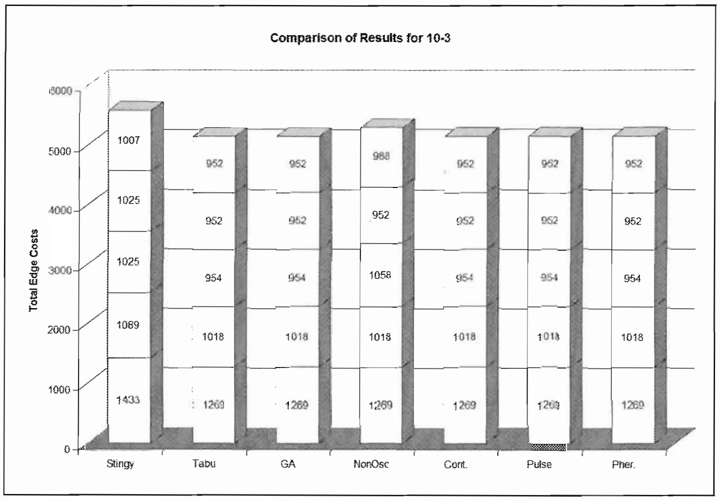


Figure C.3: Comparison of cumulative performance for problem 10-3

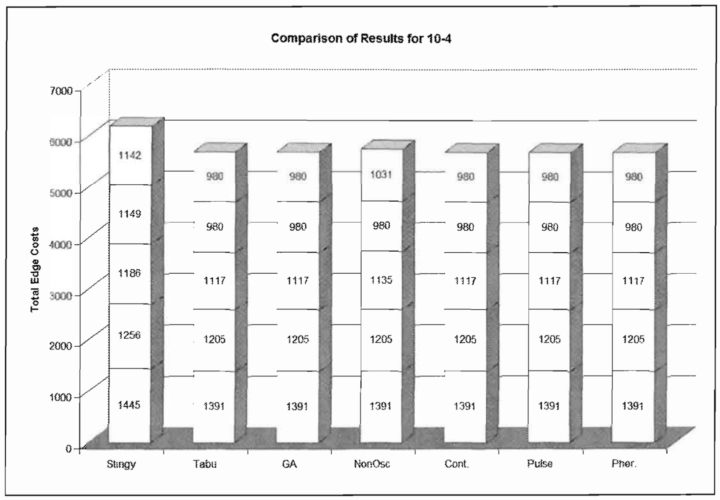


Figure C.4: Comparison of cumulative performance for problem 10-4

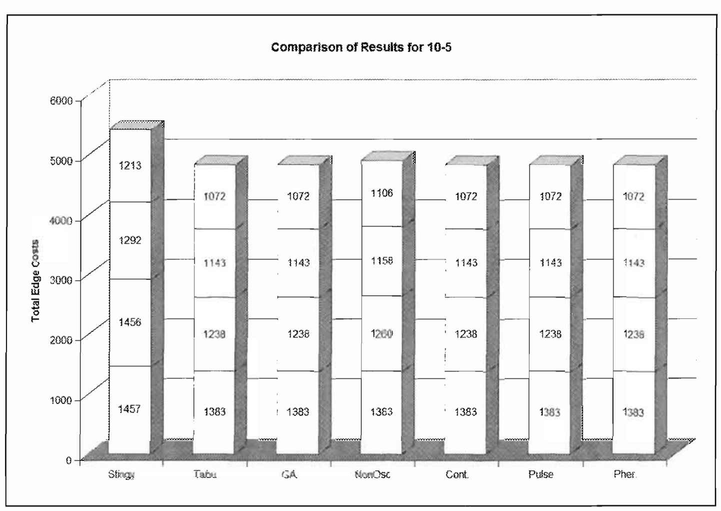


Figure C.5: Comparison of cumulative performance for problem 10-5

C.2 20 Vertices

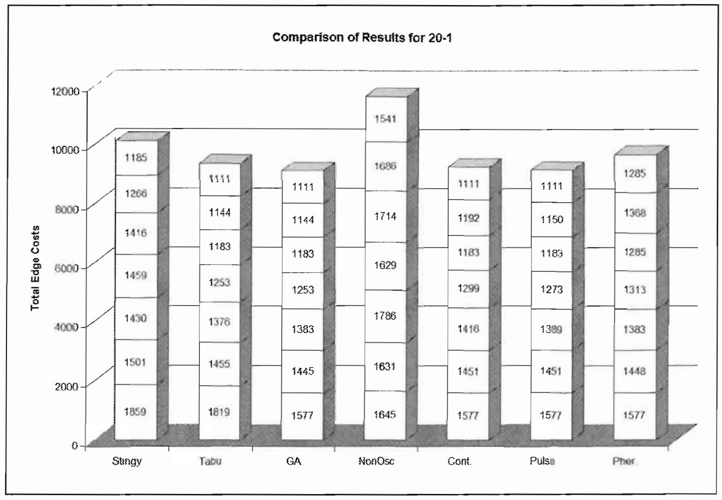


Figure C.6: Comparison of cumulative performance for problem 20-1

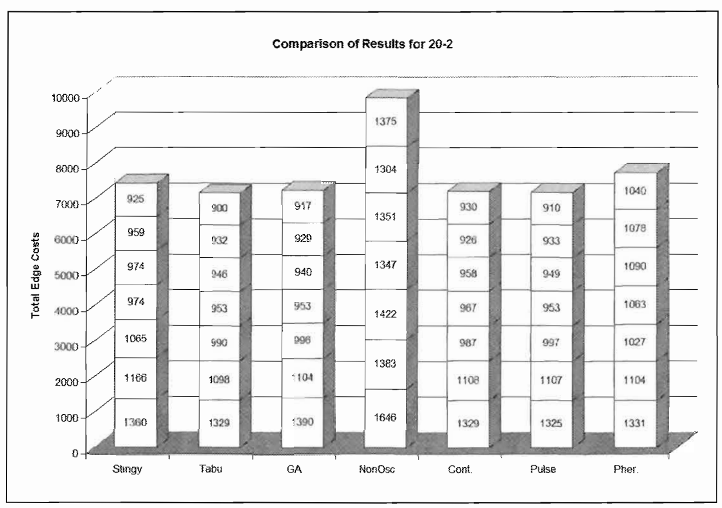


Figure C.7: Comparison of cumulative performance for problem 20-2

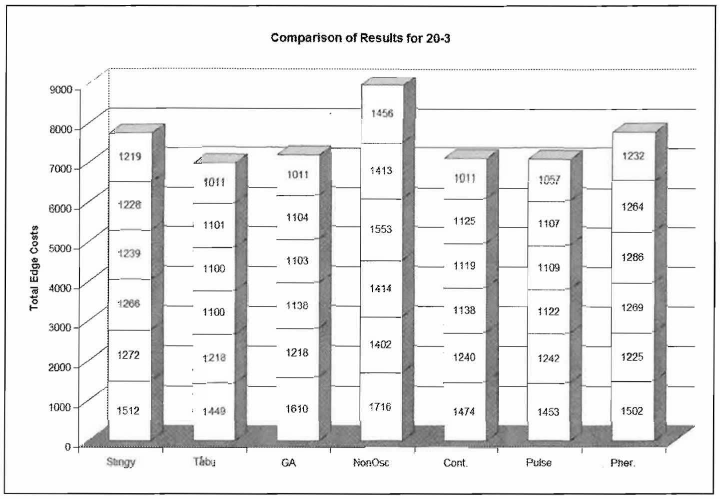


Figure C.8: Comparison of cumulative performance for problem 20-3

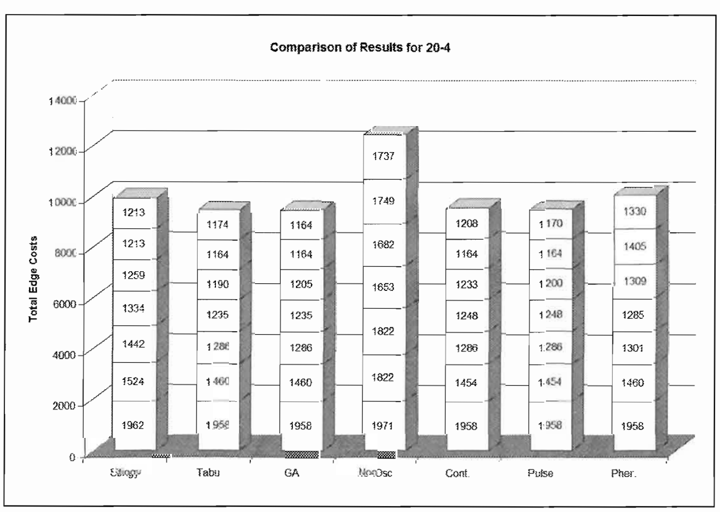


Figure C.9: Comparison of cumulative performance for problem 20-4

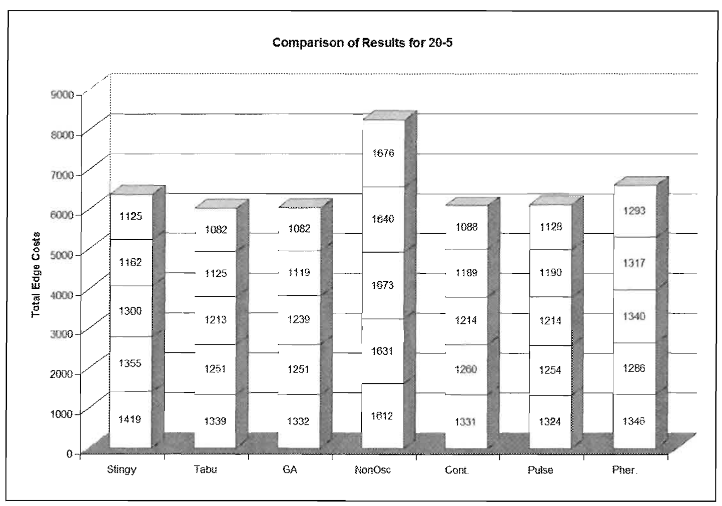


Figure C.10: Comparison of cumulative performance for problem 20-5

C.3 30 Vertices

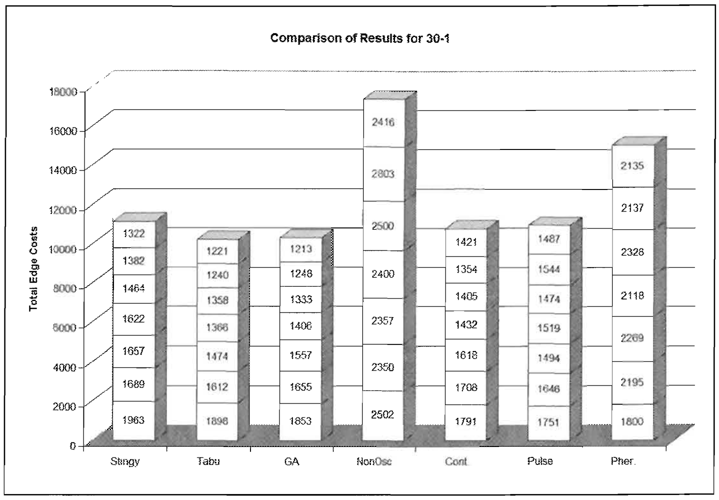


Figure C.11: Comparison of cumulative performance for problem 30-1

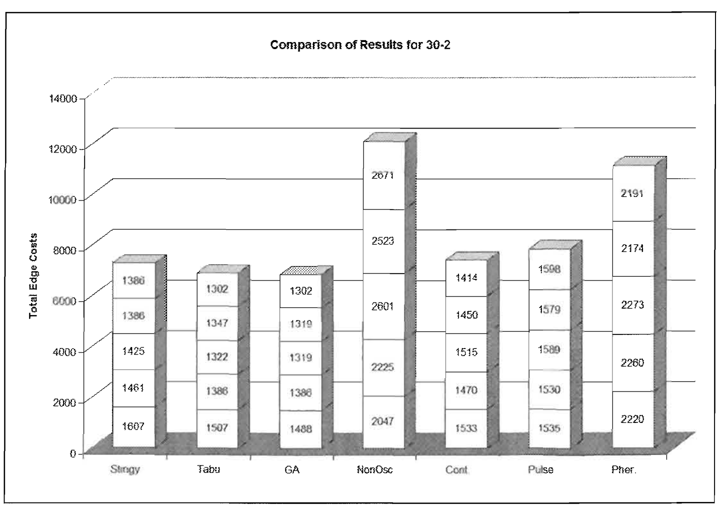


Figure C.12: Comparison of cumulative performance for problem 30-2

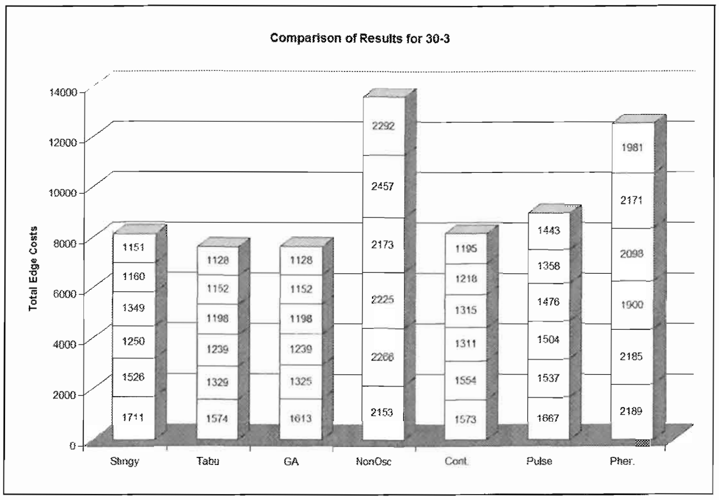


Figure C.13: Comparison of cumulative performance for problem 30-3

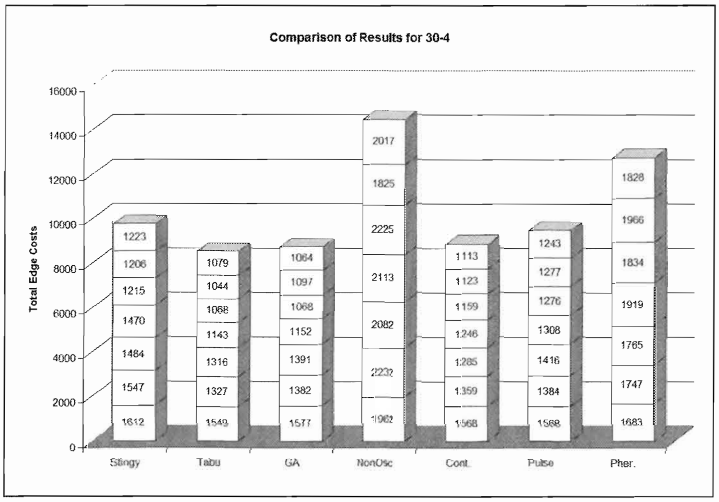


Figure C.14: Comparison of cumulative performance for problem 30-4

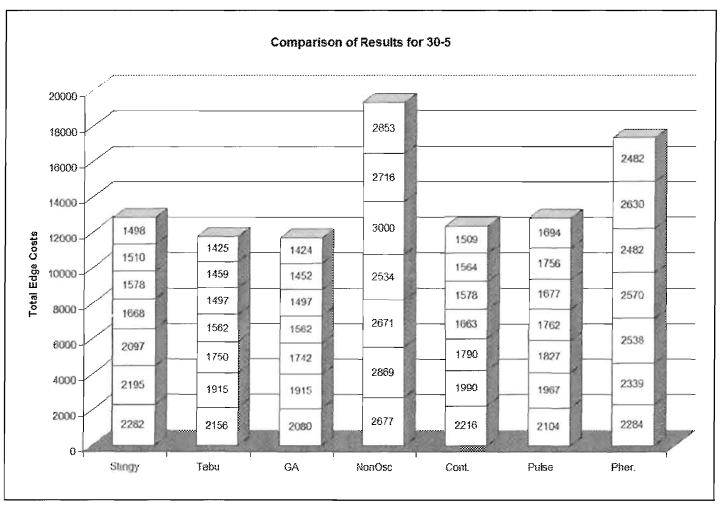


Figure C.15: Comparison of cumulative performance for problem 30-5

C.4 40 Vertices

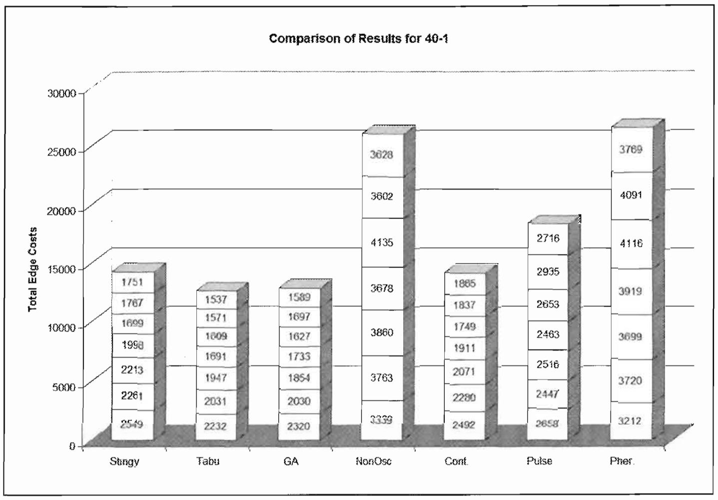


Figure C.16: Comparison of cumulative performance for problem 40-1

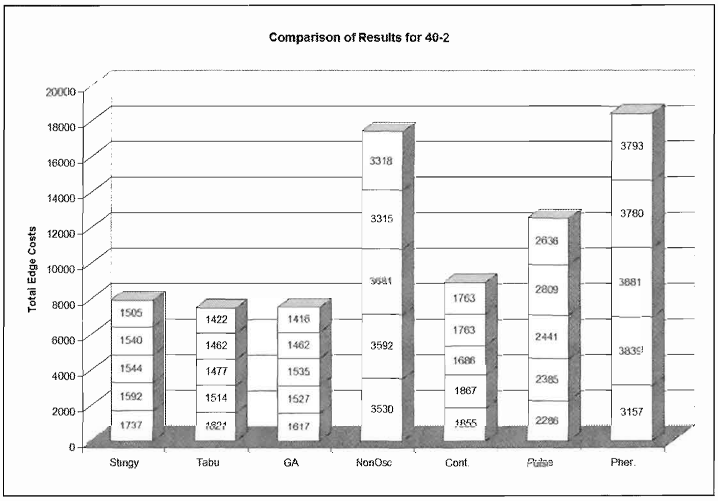


Figure C.17: Comparison of cumulative performance for problem 40-2

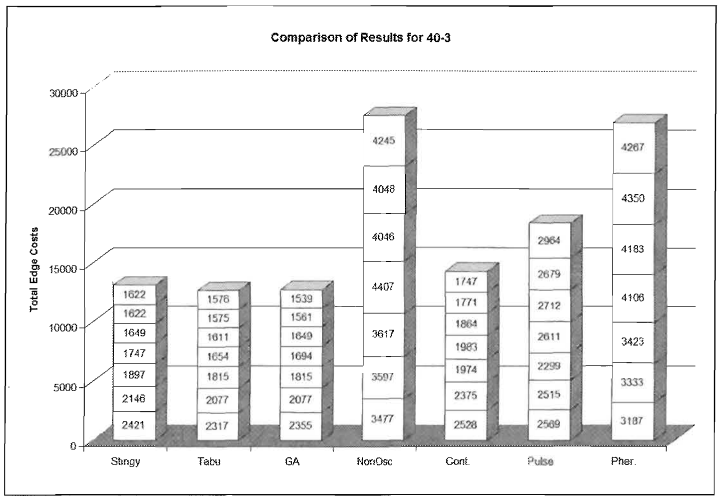


Figure C.18: Comparison of cumulative performance for problem 40-3

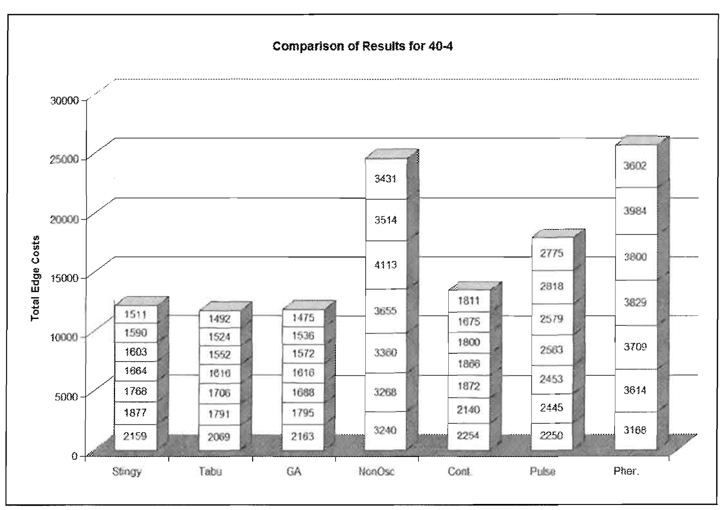


Figure C.19: Comparison of cumulative performance for problem 40-4

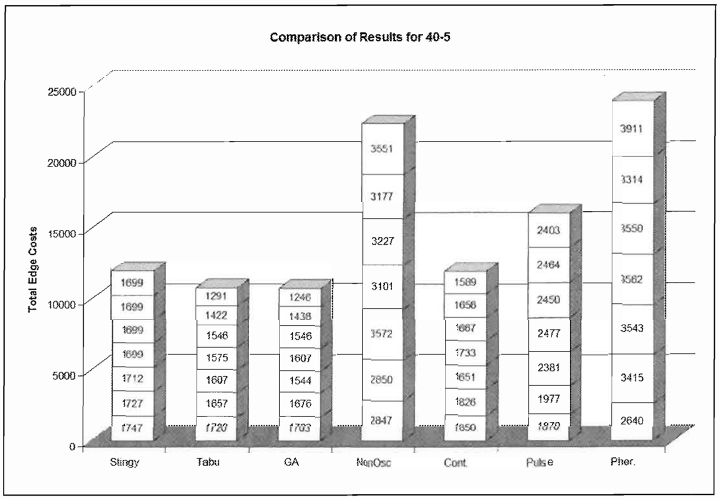


Figure C.20: Comparison of cumulative performance for problem 40-5

C.5 50 Vertices

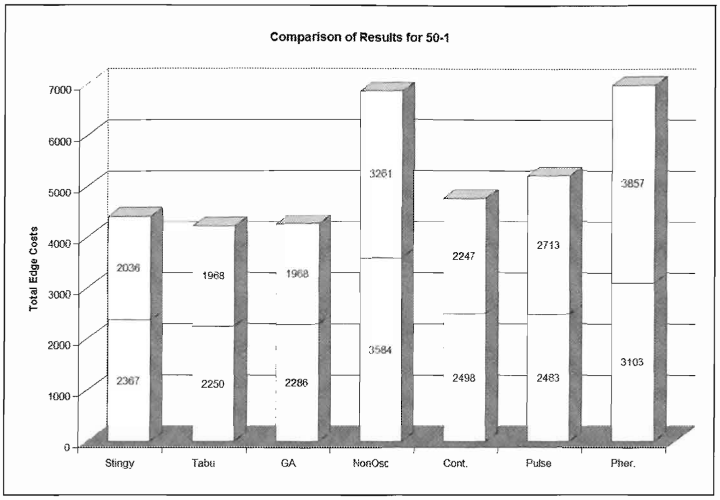


Figure C.21: Comparison of cumulative performance for problem 50-1

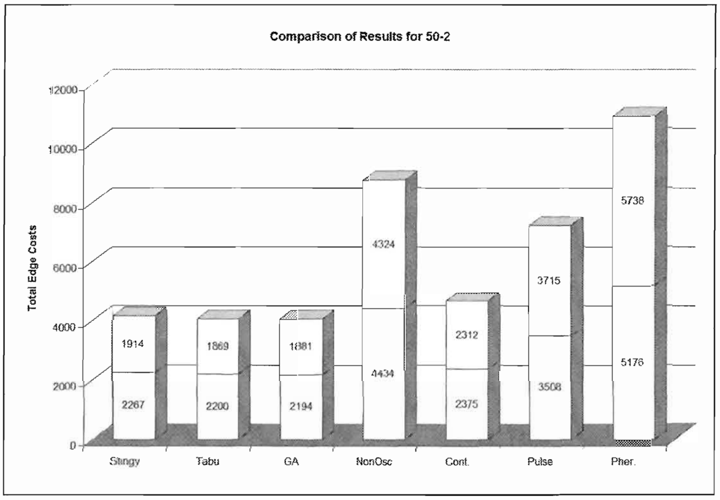


Figure C.22: Comparison of cumulative performance for problem 50-2

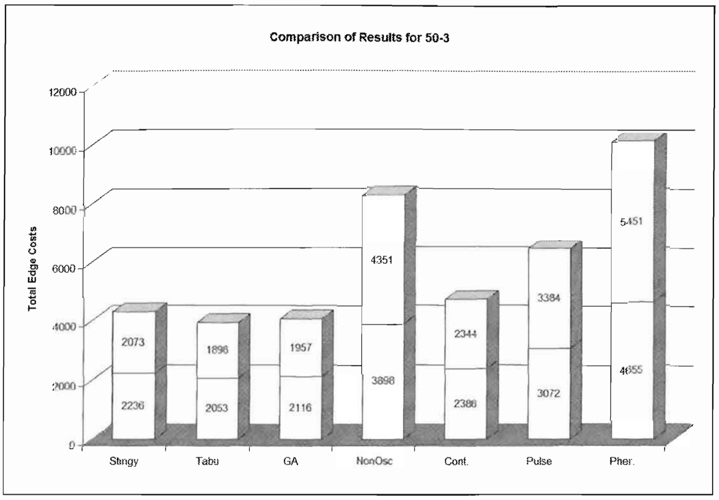


Figure C.23: Comparison of cumulative performance for problem 50-3

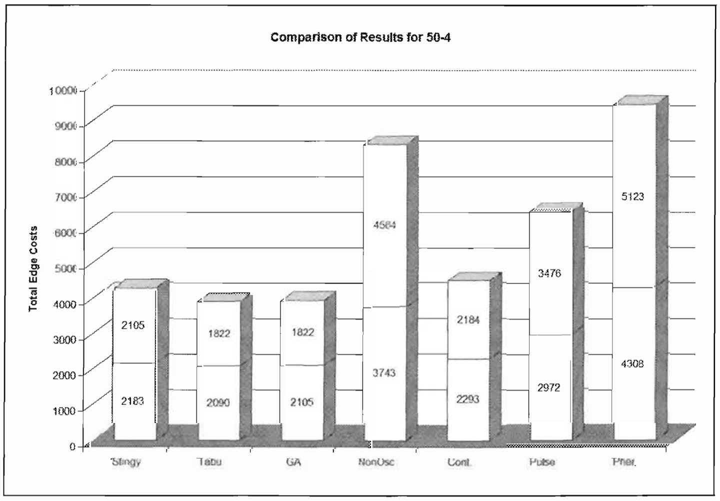


Figure C.24: Comparison of cumulative performance for problem 50-4

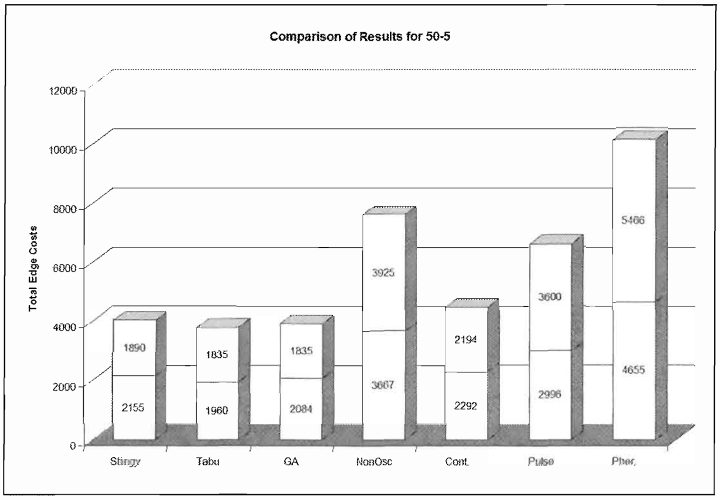


Figure C.25: Comparison of cumulative performance for problem 50-5