Heuristic Approaches to Solve the Frequency Assignment Problem

PhD Thesis
Goldsmiths College, University of London.
1998

Abstract

The frequency assignment problem is a computationally hard problem with many applications including the mobile telephone industry and tactical communications. The problem may be modelled mathematically as a T-colouring problem for an undirected weighted graph; it is required to assign to each vertex a value from a given set such that for each edge the difference in absolute value between the values at the corresponding vertices is greater than or equal to the weight of the edge. This problem was solved using novel and existing metaheuristic algorithms and their relative successes were compared. Early work of this thesis used greedy, steepest descent and backtracking algorithms as a means of investigating the factors which influence the performance of an algorithm (selection of frequency, ordering of variables, provision of an incremental objective function). Later simulated annealing, tabu search and divide and conquer techniques were used and the results compared. A novel divide and conquer technique incorporating metaheuristics is described and results using test data based on real problems is presented. The divide and conquer technique (with either tabu search or simulated annealing) was found to improve significantly upon the corresponding metaheuristic when implemented alone and acting on non-trivial scenarios. The results were significant and consistent. The divide and conquer (with simulated annealing) algorithm in particular was shown to be robust and efficient in its solution of the frequency assignment problems presented. The results presented in this thesis consistently out-perform those obtained by the Defence, Evaluation and Research Agency, Malvern. In addition this method lends itself to parallelisation since the problem is broken into smaller independent parts. The divide and conquer algorithm does not exploit knowledge of the constraint network and should be applicable to a number of different problem domains. Algorithms capable of solving the frequency assignment problem most effectively will become valuable as demand for the electromagnetic spectrum continues to grow.

Acknowledgements

I would like to thank my supervisor Prof. Nelson Stephens for his support and guidance throughout my studies. His comments on earlier drafts of this thesis have been invaluable.

My thanks also to the EPSRC for funding, Mr Roger Edwards at D.E.R.A. Malvern for a CASE studentship and for supplying some of the test data used in this thesis.

Thank you to the staff at Goldsmiths and Cardiff Universities who have helped in many practical ways and also encouraged me. In particular thanks to Steve Hurley at Cardiff University for supplying the MATRIX test data and also the random number generator used during this investigation.

My thanks to my colleagues at Coventry University who have encouraged me during my write-up year, especially Lisa, Mike and Saad. Special thanks to Colin Reeves for taking an interest in my research and also for reading earlier drafts of this thesis.

My love to Adeline, Paul and Doug who helped to keep me sane and provided fun-filled weekends breaks, putting the world to rights over a bottle of wine and generally discussing the philosophy of love, life and the universe. Oh, and getting very muddy!

Finally a big thank you to all my family and friends who believed in me and without whose support life would have been far more difficult.

Copyright Notice

Dedication

To my husband, James, I love you so much. The past four years have been a bit of a whirlwind: 7 addresses, 6 offices, 3 universities and 5 counties! I truly appreciate your support, patience and understanding. I dedicate this thesis to you, with love, your being there has made it all worthwhile.

In loving memory to Kathleen and Jack, you remain a guiding force in my life.

Contents

List of Tables

List of Figures

Notation

A	An Assignment
C	The total number of constraints in a scenario
C_{ij}	Required minimum frequency separation between links i and j
$cycle\_list$	Tabu list data structure
F	The set of available frequencies
f_i	The frequency assigned to link i
$frequency$	Tabu list data structure
N	The total number of nodes in constraint graph
P	The length of the tabu list $cycle\_list$
Q	Tabu search parameter, Frequent (see Chapter 7 section 7.3.2)
R	Tabu search parameter, Recent (see Chapter 7 section 7.3.2)
$recency$	Tabu list data structure
S	Size of the Search Space

Acronyms

CALMA	Combinatorial Algorithms for Military Applications
CASE	Co-operative Awards in Science and Engineering
CELAR	Centre d'Electronique d'Armement
COP	Combinatorial Optimisation Problem
DC	Divide and Conquer (with either SA or TA)
DCSA	Divide and Conquer with Simulated Annealing
DCTS	Divide and Conquer with Tabu Search
D.E.R.A.	Defence Evaluation and Research Agency
DERA	Program in use at D.E.R.A.
EPSRC	Engineering and Physical Sciences Research Council
EMC	ElectroMagnetic Compatibility
EUCLID	European Co-operation for the Long Term in Defence
FAP	Frequency Assignment Problem
GA	Genetic Algorithm
GCP	Graph Colouring Problem
ICANNGA	International Conference on Neural Networks and Genetic Algorithms
ITU	International Telecommunications Union
LS	Local Search
NATO	North Atlantic Treaty Organisation
NT	Non-Tabu
QAP	Quadratic Assignment Problem
RLFAP	Radio Link Frequency Assignment Problem
SA	Simulated Annealing
T	Tabu
TS	Tabu Search
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem

Chapter 1

Introduction

The frequency assignment problem is a combinatorial optimisation problem which seeks to assign frequencies to communication links in such a way that the interference suffered is minimised and demand is satisfied.

In recent years the demand on the electromagnetic spectrum due to the wider use of communications has increased dramatically. However the frequency spectrum available is limited and this has led to extensive research into techniques that are able to use the available frequencies in the most economical way. Frequencies must be reused many times. However, some pairs of allocated frequencies must be separated by a predetermined amount, depending on distance between transmitters and geographical terrain, in order to minimise interference. Ideally we wish to obtain a solution with zero interference using a small set of frequencies of minimal span (the difference between the largest and smallest frequencies used by the assignment). However the density of the transmitters and restricted frequency set means that minimal interference is a more reasonable aim. Since the communications industry looks set to increase further in the future, the management of the available frequencies is becoming an increasing concern.

A great deal of research has already been done in this field which investigates the effectiveness of various heuristic techniques in the solution of the frequency assignment problem. The primary aim of this research project was to investigate how to split the frequency assignment problem into subproblems for more rapid solution and to establish a software library of techniques based on existing and novel algorithms (comparing the performance of the alternative techniques). Further aims were to investigate factors

which influenced performance, ways of dividing the problem into subproblems and the development of metrics for measuring the goodness of an assignment.

In order to solve the practical problem we need to represent it as a model which inevitably becomes an abstraction of the problem to be solved. Some loss of information is acceptable since the simpler model can be solved efficiently on a computer and enables us to reach reasonably good solutions in a reasonable time. It is convenient to represent the frequency assignment problem as a graph where the nodes represent the communication links and the edges represent possible interference between those links. It can be shown that if all the constraints are of the co-channel type (frequencies assigned to interfering links must be different) then the problem is equivalent to the classical graph colouring problem. Obtaining a feasible colouring of the graph leads to a zero interference frequency assignment.

Since the frequency assignment problem is a special case of the general graph colouring problem it is classified computationally as NP-Hard. Hence, there is no known algorithm that can generate a guaranteed optimal solution in an execution time that can be expressed as a polynomial of the problem size.

This study of the frequency assignment problem was initiated by the tactical communications division of the Defence Evaluation and Research Agency (D.E.R.A) in response to a need for techniques which would provide better solutions in less time than those currently available. Current algorithms, implemented in ADA and running on a VAX 4000 workstation, were unable to produce solutions of sufficient quality in the limited time available. Preliminary studies, resulting in a North Atlantic Treaty Organisation (NATO) report by Lanfear [Lan89], indicated that further investigation into hybrid heuristic techniques would prove fruitful.

The divide and conquer, simulated annealing and tabu search algorithms described in this thesis were tested using 46 tactical communications scenarios supplied by D.E.R.A. at Malvern. These scenarios were generated to represent real-life scenarios.

The divide and conquer strategy described in this thesis subdivides the original problem into a number of smaller frequency assignment problems, which are solved using a neighbourhood search algorithm. The subproblems are significantly smaller and can be solved more effectively. The solutions of the subproblems are then combined to give a solution of the initial problem.

Experimental results have shown that final solutions obtained when using pre-ordered links are of better quality than solutions obtained using unordered links. Several methods for dividing the search space have been investigated and the outcomes compared. The provision of an incremental objective function has been shown to improve the final solution quality in a fixed amount of time. In the implementations described here a move from one solution to another involves the assignment of a new frequency to a chosen link. The method of selecting a possible frequency has been shown to affect the amount of spectral resources required.

The divide and conquer algorithm was compared with classic implementations of simulated annealing and tabu search algorithms acting on the same data. Solutions were compared using several metrics: speed of execution, interference measures of solutions, span (difference between the largest and smallest frequencies used by the assignment) and order (number of distinct frequencies used by the assignment).

The results showed the divide and conquer algorithm to be superior to each of the single metaheuristic algorithms (simulated annealing and tabu search) applied with respect to solution quality, use of resources and computational time when acting on non-trivial scenarios. Some further improvements have been suggested to capitalise on the results presented.

The D.E.R.A. data sets have been solved by the divide and conquer algorithm and have obtained solutions far superior to those obtained using existing methods.

The use of problem-specific knowledge has been shown to improve significantly the final solution quality and execution time of algorithms used to solve the frequency assignment problem [THL95]. The divide and conquer technique uses information about the co-site constraints when dividing the problem into subproblems.

Some of the divide and conquer results from earlier work presented in this thesis have been published in 'Artificial Neural Nets and Genetic Algorithms', Springer Computer Science [PS98], a summary of results from chapter 10 can be found in [Whi98].

The divide and conquer technique has been shown to obtain superior solutions when compared with classic implementations of TS and SA, and when compared with results obtained by the program currently in use at D.E.R.A. The results have shown the divide and conquer technique to be effective and robust for the D.E.R.A. data. Since it does not exploit any problem-specific knowledge it should prove to be robust enough

to deal with a variety of different test data sets. In addition the divide and conquer technique lends itself to parallelisation.

It is hoped that this research may be useful for operation researchers, computer scientists and practitioners investigating into heuristic solution methods for the frequency assignment problem since it has given an insight into a new technique that works well for the FAP as well as factors that may influence their performance. This research is also informative for those interested in heuristic techniques in general. Knowledge of how well these techniques can perform is encouraging and provides some valuable ideas and information.

1.1 Thesis Outline

The second chapter begins by establishing why the frequency assignment problem (FAP) is important at the current time. The problem is then defined and its practical applications outlined. The third section discusses the various types of interference which can occur and how the interference is defined in terms of constraints for the FAP. In the fourth section the mathematical model of the problem is given. Section five describes the objective function used in the main investigation discussed in this thesis, and also outlines other functions from the literature. Finally the computational complexity of the FAP is shown to be NP-Hard.

The third chapter outlines the recent developments in the field of metaheuristics with particular attention being paid to techniques used to solve the frequency assignment problem (FAP). An earlier report [Lan89] covered this research area up to 1989 and this chapter aims to bring the study up to date. Variations of the FAP, are also mentioned. The research objectives for this thesis are given in the context of recently published work. Finally, the research contributions are given.

In chapter four the algorithms used in the investigations described in this thesis are given in their general form. The term combinatorial optimisation is explained; this is followed by a description of exact techniques which guarantee that a globally optimal solution will be found. The need for heuristic techniques is then discussed and metaheuristics are briefly introduced. General heuristic algorithms are then described along with the relevant terminology, before a more detailed discussion on several meta-

heuristic techniques is given. Simulated annealing and tabu search are described and definitions of the terms used when discussing these algorithms are given. Finally the divide and conquer technique is described.

The fifth chapter explores the influence of three factors on the performance of a backtracking and a greedy algorithm (frequency selection, vertex ordering, provision of an incremental objective function). A summary of results is given.

In the sixth chapter the test data are described. Three different formats of test data were used in this research: EUCLID, MATRIX and TNET. The EUCLID data set is a small example problem, the MATRIX data set describes 3 moderate-size problems, and finally, the TNET scenarios represent real-life problems. There are 46 TNET scenarios. The EUCLID and MATRIX data sets were used only during preliminary experimentation as described in Chapter 5. The TNET data were used for the heuristic algorithm investigation.

In the seventh chapter some benchmark results are established for the simulated annealing and tabu search heuristics. Common decisions regarding problem representation are discussed followed by descriptions of the simulated annealing and tabu search algorithms as implemented for the FAP. Computational results are given and discussed before concluding remarks and suggestions for further work.

In the eighth chapter the divide and conquer implementation is described in detail. Possible ways of dividing a problem into subproblems and ways of overcoming non-independent subproblems are discussed. The various stages of the technique are described in detail including the metaheuristic improvement stages which use either simulated annealing or tabu search. Results are provided for both implementations of the divide and conquer technique. The chapter ends with conclusions and suggestions for further work.

In chapter nine the results obtained in chapters seven and eight are compared. The divide and conquer algorithm was found to significantly improve upon the results obtained by the respective improvement heuristic on its own. Some conclusions and ideas for further work are given.

In chapter ten the results obtained in chapter eight (using the divide and conquer algorithm) are compared with results provided by D.E.R.A. for the same scenarios. Some comparison difficulties are discussed.

Finally, chapter eleven discusses the relevance and importance of the results obtained, the research contributions of this thesis and some ideas for further work.

Chapter 2

Frequency Assignment Problem

This chapter first establishes why the frequency assignment problem (FAP) is important at the current time. The problem is then defined and its practical applications outlined. The third section discusses the various types of interference which can occur and how the interference is defined in terms of constraints for the FAP. In the fourth section the mathematical model of the problem is given. Section five describes the objective function used in the main investigation discussed in this thesis, and also outlines other functions from the literature. Finally the computational complexity of the FAP is shown to be NP-Hard.

2.1 Available Spectrum is Limited

During the last two decades significant developments have been made in communications technology. This has placed great demand on the electromagnetic spectrum resulting in extensive research into techniques that use the available frequencies in the most economical way. Owing to these rapid developments the spectrum has become a significant but limited resource, as the range of frequencies available for radio communication is only a fraction of the electromagnetic spectrum. In order for the available spectrum to be fully utilised there must be cooperation among the users. The International Telecommunications Union (ITU), in Geneva, is responsible for overseeing this cooperation. The spectrum has been compartmentalised and made available to various users such as the military, broadcasting, amateur radio and aeronautical radio

navigation. The resource must be utilised to the maximum economic benefit whilst safeguarding access for defence, emergency services and social use.

It seems likely that demand on the spectrum will continue to increase and so effective solution of the frequency assignment problem will become even more important in the future. Frequency assignment techniques that are able to reduce the spectral or computational resources required will continue to be of practical value both in civil and military communications systems.

2.2 The Frequency Assignment Problem

In order to effect radio communication it is necessary to emit a signal from a transmitter to be picked up by a receiver. Interference occurs when signals combine to produce unwanted frequencies or when a receiver is unable to distinguish between similar frequency signals. A transmitter-receiver pair makes up a communication link. In order to reduce interference, constraints are imposed on the assignment. For each pair of links a separation value for the frequencies can be computed from path loss prediction data which will ensure no interference. Essentially path loss is dependent on the physical separation of the radios together with terrain information.

The primary objective of the frequency assignment problem is to assign frequencies, from F, a discrete set within the available band-width, to the communication links, subject to the set of constraints, so that minimal interference is suffered. Depending on the problem, the discrete set of frequencies may have different forms, for example,

1. Integers / real numbers.
2. Equally distributed across the possible range, fixed differences between consecutive frequencies.
3. Randomly generated within the possible range, variable differences between consecutive frequencies.

Constraints are of the form:

$$|f_i - f_j| \geq C_{ij}$$

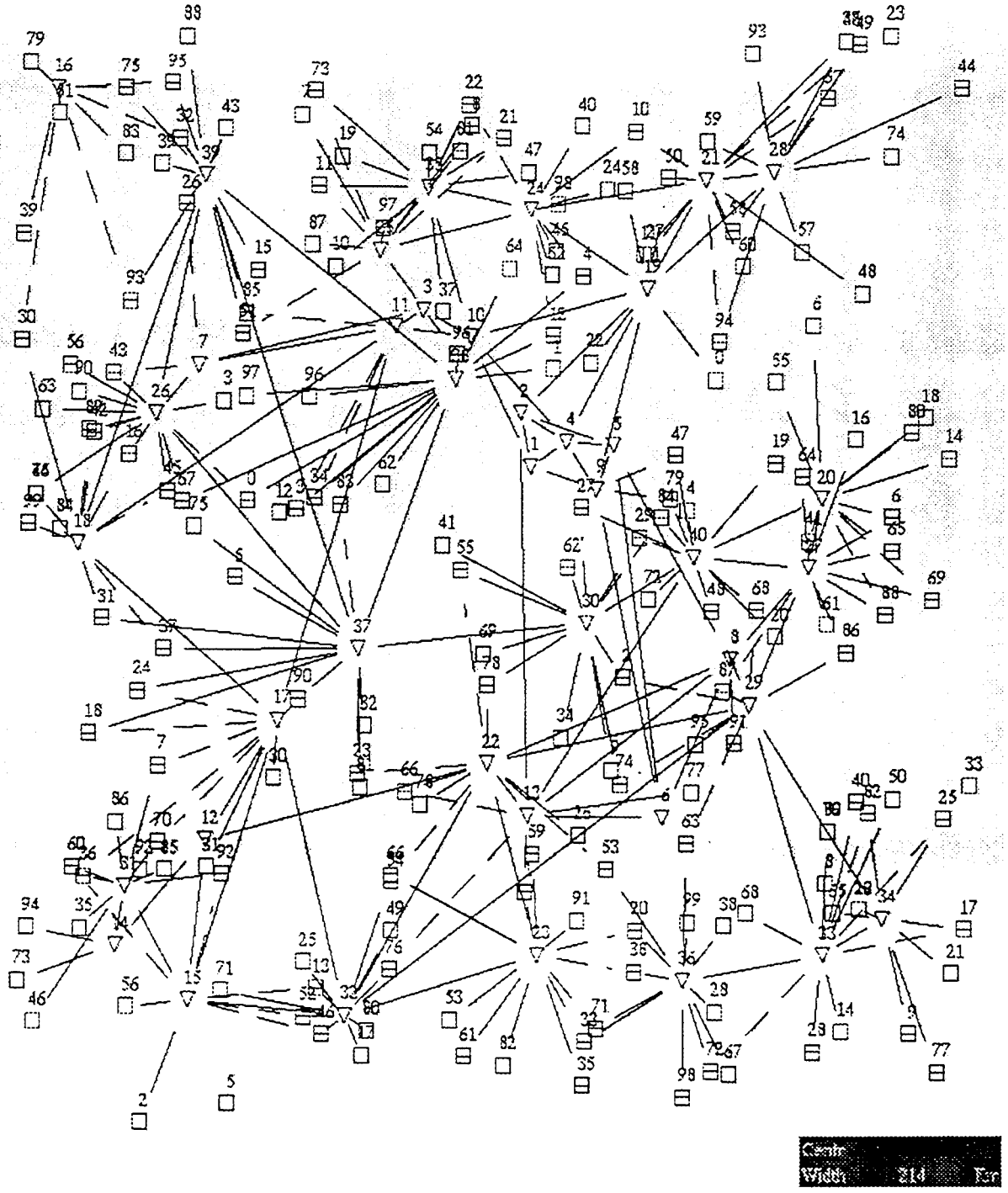
where f_i and f_j are the frequencies assigned to links i and j respectively and C_{ij} is the separation required measured in MHz. If there is no constraint, C_{ij} is zero. There are two types of constraints, co-site and far-site (see section 2.3).

Ideally we wish to obtain a solution with zero interference using a small set of frequencies of minimal span (difference between the largest and smallest frequencies used by the assignment). However the density of the transmitters and restricted frequency set means that minimal interference is a more reasonable aim. Typically there are a large number of transmitters and a relatively low number of frequencies available so solving frequency assignment problems of a practical size continues to present a difficult problem. For example, a problem containing 100 frequencies and 500 nodes would have a search space of 100^{500}. Fig 1 shows the physical model of an example communication network. The lines represent required communication links (not constraints as in the mathematical model). The nodes represent different types of tactical communications bases with varying communication requirements (the numbers). Unfortunately further information cannot be given due to confidentiality imposed by D.E.R.A.

There are two factors contributing to the difficulty of the frequency assignment problem. The first is the enormous number of candidate solutions among which a feasible solution must be found. Secondly the number and variety of constraints which must be satisfied to produce satisfactory assignments tend to destroy nice mathematical structures preventing the use of efficient exact algorithms used to solve simpler problems. As optimal solutions can be extremely difficult to determine, much research has concentrated on heuristic procedures capable of generating near-optimal solutions.

There are two distinct frequency management problems; the bulk assignment problem and the updating assignment problem. In the bulk assignment problem all transmitters from the band are known in advance and the task is to assign a frequency to each of them. The manager has complete knowledge of all the assignments to be made. In the updating assignment problem the transmitters to be assigned are presented sequentially, either individually or in small groups, and frequencies must be assigned without knowledge of future demands on the spectrum. A practical management problem may well be a combination of the two with an initial assignment of many requests followed by a sequential assignment against this background. The work presented in this study refers to the bulk assignment problem where all values of C_{ij} are known constants.

Figure 1. Example Communication Network



TNET 7

2.3 Interference

In order to reduce interference, constraints are imposed on the assignment. Interference occurs when certain pairs of links are assigned frequencies which are the same or close together. This can happen when transmitters or receivers of links are at the same site, within a few tens of metres of each other (co-site interference) or when equipment is at a distance of several kilometres or more (far-site interference). For details of how the required frequency separation is calculated refer to [Lan89]. In essence the required frequency separation is calculated using path loss prediction [SH88] and interference predictions. In the literature constraints formulated in this way are often referred to as frequency-separation constraints, they are thought to provide a more accurate model than frequency-distance [1] constraints [Bat et al. 98].

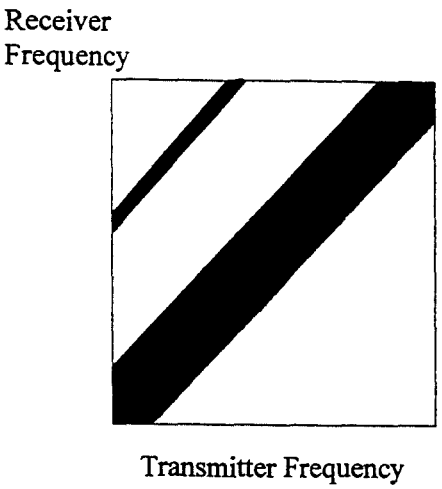
2.3.1 Co-site

Co-site interference occurs when transmitters or receivers of two links are closer than two kilometers. Co-site interference is due to the technical limitations of the equipment used; the receivers cannot differentiate between frequencies which are very close together. A mutual interference chart (Fig 2), dominated by a diagonal band, shows that the transmitter and receiver frequencies should differ by more than a certain frequency separation. The width of the band depends on the power of the transmitter.

Co-site constraints are of the form $|f_i - f_j| \geq C_{ij}$, where C_{ij} is large. Typically co-site frequency separations may have values between $15 - 49\%$ of the total range available. For example, if the range of frequencies available is 400MHz then co-site constraint values C_{ij} may be between 60 and 196MHz.

[1] frequency-distance constraints are based on minimum euclidean transmitter separations on a simplified geographic model

Figure 2. Mutual Interference Chart



2.3.2 Far-site

Far-site interference occurs between equipment separated by some distance. The separation required depends on the design of the transmitter and receiver, the alignment of the antennae and path loss calculations which depend on terrain. The most important factor is *co-channel interference* which states that a pair of communication links cannot be assigned the same frequency unless they are sufficiently geographically separated. Constraints are of the form $|f_i - f_j| > 0$.

Similarly, *adjacent-channel constraints* prevent certain link pairs from interfering if the equipment is using similar frequencies. Constraints are of the form $|f_i - f_j| \geq C_{ij}$, where C_{ij} is small. Typically adjacent-channel constraint separations may have values up to 5% of the total range available. For example if the range of frequencies available is 400MHz then constraint values C_{ij} may be up to 20MHz.

For each pair of links a separation value for the frequencies can be computed from path loss prediction data which will ensure no interference. The path loss prediction calculation is based on the two communication frequencies, f_i and f_j, the distance between the communicating links and terrain information. Interference may also depend on the prevailing weather conditions and the power of the transmitters. When processing these data to formulate the constraints a worst-case scenario is used. The resulting constraints, in the form $|f_i - f_j| \geq C_{ij}$, therefore represent the minimum frequency separation required to guarantee zero interference between the links under consideration. Of course, the required minimal separation may be zero if no possible interference can

occur between two links whatever the assignment.

2.3.3 Intermodulation Constraints

Intermodulation constraints are a form of co-site constraint. When two or more signals enter a non-linear system further signals may be generated which are related to the frequencies of the initial signals. The new signals are called intermodulation products. If an intermodulation product has a frequency close to the frequency of one of the required links the receiver may not be able to distinguish between them and so interference will be suffered. A typical intermodulation constraint has the form

$$|\lambda f_i + \mu f_j - (\lambda + \mu) f_k| \geq C_{ijk}$$

where λ and μ are small positive integers and f_i, f_j, f_k are the frequencies assigned to the three links and C_{ijk} is the required frequency separation. A typical intermodulation product gives the constraint

$$|2f_i - f_j - f_k| \geq 0 \quad \text{(two signal, third order)}$$

Intermodulation constraints involve three or more transmitters and therefore cannot be represented by the chosen model (see section 2.4) which restricts the problem to constraints between transmitter pairs.

After discussions with Roger Edwards at D.E.R.A. Malvern it became clear that such constraints accounted for only a small percentage of the total number of constraints. In addition the value of the required frequency separation C_{ijk} is small, usually 1 or 2 MHz. Intermodulation constraints do not have a significant impact on the interference suffered by a network. The systems at D.E.R.A. do not consider intermodulation constraints, nor have they been included in the investigations described in this thesis. It is common for such constraints to be considered after a good assignment has been found.

2.4 Mathematical Model

For any practical optimisation problem we must first model the physical situation so as to derive the mathematical function to be minimised. Since we are optimising a model of the real-world problem there is no guarantee that an optimal solution to the model is also an optimal solution to the real problem. However, the objective functions and constraints of real problems are very difficult to express as an exact model since they are rarely linear. Approximate models enable more complicated problems to be modelled since they are more flexible. In recent years, the development of heuristic search techniques has enabled very good solutions to be obtained to approximate models of challenging real-world problems. Examples of application areas (see Chapter 3 section 3.1.6) include the travelling salesman problem, bin packing and vehicle routing. An excellent bibliography of heuristic applications is given in Osman and Laporte [OL96]. The pursuit of finding good solutions to an approximate model using heuristics is the corner stone of the research described here.

Graph theory provides a useful analytical tool and appropriate model for the frequency assignment problem. In particular if only co-channel constraints are considered then the problem is equivalent to the classical graph colouring problem and as such is classified computationally as NP-hard (see section 2.7). The classification NP-hard indicates that there is no known algorithm that can generate a guaranteed optimal solution in a maximum execution time that may be expressed as a polynomial of the problem size.

The problem of frequency assignment in tactical communications can neatly be modelled as a T-colouring problem as outlined by Hale [Hal80]. A graph-theoretic approach has been widely used in the literature [Lan89] [ZB77] [SHT98].

In the general graph colouring problem (GCP) a graph $G = (V, E)$ has colours assigned from a finite colour set to each vertex such that no adjacent vertices have the same colour. That is, the integer values representing the colours differ by at least 1. Expressing this constraint in graph notation: the edge joining two vertices has a weight of 1. In T-colouring the edges of the graph may take weights other than 1. The edge T_{xy} that joins vertices x and y has a non-zero weight, for example $T_{12} = 3$. This constraint means that the integers assigned to vertices 1 and 2 must differ by at least 3.

A function, $f(x)$ returns the integer assigned to x, the constraint is satisfied if

$$|f(x) - f(y)| \geq T_{xy}$$

If it is possible to find a solution such that all the constraints are satisfied it is called a '*feasible colouring*'.

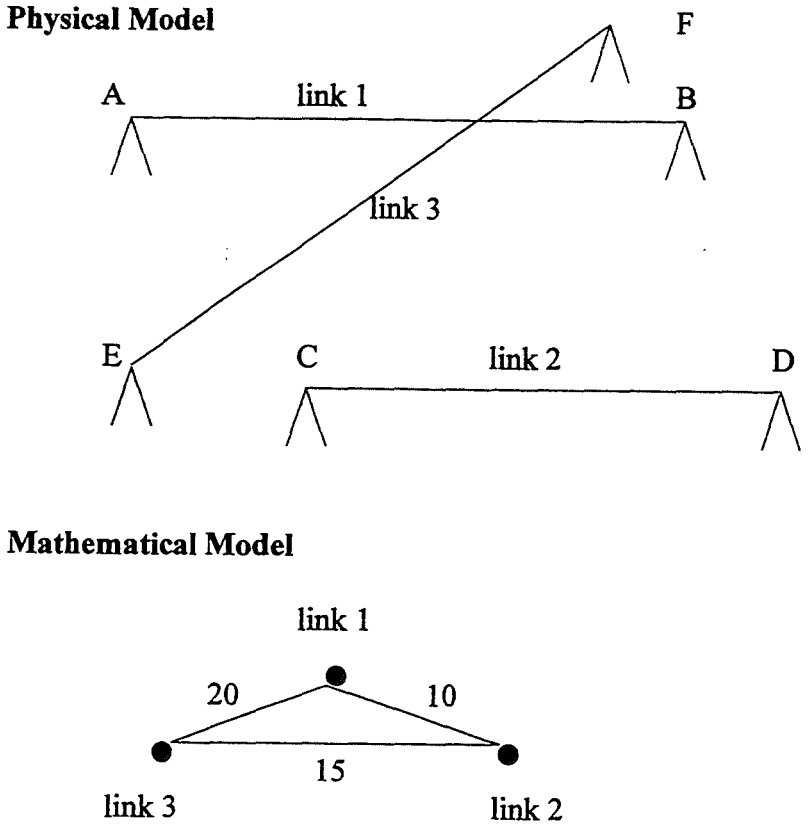
The N communication links of the FAP can be considered as the vertices of an undirected weighted graph. Two vertices are adjacent if the corresponding links may not use the same frequency. The weight of an edge between two vertices i and j is the constraint value C_{ij}. The problem is to assign to each vertex i a frequency f_i from a frequency set F such that $\forall\ i, j,\ |f_i - f_j| \geq C_{ij}$. A solution to this model can be interpreted back into terms of the physical context of the problem, that is, the interference of the network can be calculated given knowledge of the edges violated. A feasible colouring is synonymous with a network which suffers no interference.

For example the following constraints (Table 1) are shown in the physical and mathematical models in Fig 3.

Table 1. Example Constraint Values for Physical and Mathematical Models

Link i	Link j	Frequency Separation
1	2	10
1	3	20
2	3	15

Figure 3. Physical and Mathematical Models



2.5 Objective Function

An objective function considers the current solution and uses knowledge of the problem to arrive at some measure of desirability of the solution, usually expressed as a real number. Which parameters of the problem are considered and their relative importance are defined in such a way that the objective function is able to estimate as well as possible whether the solution under consideration will lead to an optimal solution. Clearly what constitutes a good solution depends on the problem being solved.

A number of possible objective functions have been suggested for measuring the quality of a proposed assignment. One such objective function measures interference by calculating the sum of the positive discrepancies [Cas97],

$$\sum_{i,j=1}^{i,j=N} \max(0, C_{ij} - |f_i - f_j|)$$

The numerical value resulting from the objective function calculation is often referred to as the *cost* of a solution. The term *cost* has been used extensively in the literature and is also used in this thesis. The terms *objective function* and *cost function* are used interchangeably.

In evaluating an assignment attention is also paid to the span (the difference between the largest and smallest frequencies used by the assignment) and the order (the number of distinct frequencies used by the assignment). Ideally we seek a solution that firstly satisfies all the constraints (no interference) and secondly minimises span and thirdly minimises order (releasing frequencies for other applications).

Other measures for evaluating infeasible assignments include

1. Calculating the number of violated constraints [Lan89] [DKR 93] [MM93] [CHS93] [CHS94b] [CS95] [HTS96].

2. The difference between the largest and smallest frequencies used — the span of the assignment. This measure is typically used after zero-interference solutions have been obtained [Met70] [ZB77] [Gam86] [Lan89] [Cos93] [dWG94].

3. Weighting the constraints to reflect their importance and calculating the weighted sum of constraint violations [CAL95] [THL95].

4. The distinct number of frequencies used; this is the order of the assignment. This has been used by [Bou et al 95b].

5. Calculating the sum of the percentage deviations from the required separation.

6. Calculating a weighted sum of the above interference measures.

2.6 Objective Function and Representation Used

Most of the decisions regarding constraint data structures and ordering of links has necessarily depended on the data set being used. A description of these elements has been given later, where the implementation is described. However, some definitions

were common and these are described here to orient the reader already familiar with the FAP.

2.6.1 Objective Function

Preliminary experimentation described in chapter 5 used the number of violated constraints for the objective function.

$$\sum_{i,j=0,j\neq i}^{i,j=N-1} X_{ij} \quad where X_{ij} = \begin{cases} 1 \text{ if } |f_i - f_j| < C_{ij} \\ 0 \text{ otherwise} \end{cases}$$

For all of the heuristic algorithms (chapter 7 onwards) the sum of the positive discrepancies was used.

$$\sum_{i,j=1}^{i,j=N} \max(0, C_{ij} - |f_i - f_j|)$$

2.6.2 Representing an Assignment

Throughout this thesis a frequency assignment, $A = (f_1, f_2, ...f_N)$, is represented using an array of frequencies. The indices $1..N$ refer to the link to which the frequency is assigned.

2.6.3 Definition of a Move and a Neighbourhood

For all implementations a move was defined as the assignment of a new frequency to a chosen link. If an assignment A is $(f_1, f_2, ...f_N)$ then a neighbour of A can be described as $A' = (f'_1, f'_2, ...f'_N)$, where for precisely one i, $f'_i \neq f_i$. A neighbourhood of A is the set of all possible A' and has size $(|F| - 1)N$.

2.7 Computational Complexity

Computational complexity is used to express the 'degree of difficulty' of a problem to be solved using a computer. It is useful to know the complexity of a problem for two reasons: first, if a problem belongs to a very difficult class then the methods used to find

a solution can concentrate on obtaining near-optimal rather than optimal solutions. Secondly, when two problems belong to the same class then often one problem can be transformed into the other problem and the same methods used to solve each of them. It is not always straight-forward to identify which problem belongs to which class. The classification of a problem indicates how the algorithm will behave under the worst possible case; of course it may behave better on average. The complexity of an algorithm does not change depending on the encoding of the data or the computer used to solve it. A formal approach of the theory of NP-complete problems can be found in Garey and Johnson [GJ79].

2.7.1 P

P stands for Polynomial. This class contains all decision problems that have polynomial-time deterministic algorithms. For example $O(n), O(nlogn), O(n^2)$ and so on. Here n represents the size of the problem; for example, the number of bits required to represent the values of the inputs to the algorithm. Polynomial algorithms are better than exponential algorithms when the problem is large. A function is considered an exponential function if n appears in the exponent, thus $O(2^n)$ is considered an exponential function.

2.7.2 NP

NP stands for Nondeterministic Polynomial. This class contains all decision problems that have polynomial-time non-deterministic algorithms. Such problems have exponential deterministic algorithms but it has not yet been proven that they cannot have polynomial-time algorithms.

For example, the travelling salesman problem (finding the shortest circuit in a graph such that all vertices are used) a (deterministic) backtracking algorithm that takes exponential time will implicitly try all of the tours and find the shortest tour. If the same problem is solved using a nondeterministic algorithm then the algorithm can *correctly guess* which arc should be travelled next and so takes polynomial time. Thus, this problem is in the class NP.

P is a subclass of NP and it is still unknown whether $P = NP$. To prove that $P = NP$, it needs to be proven that all problems in NP can be solved in polynomial time by

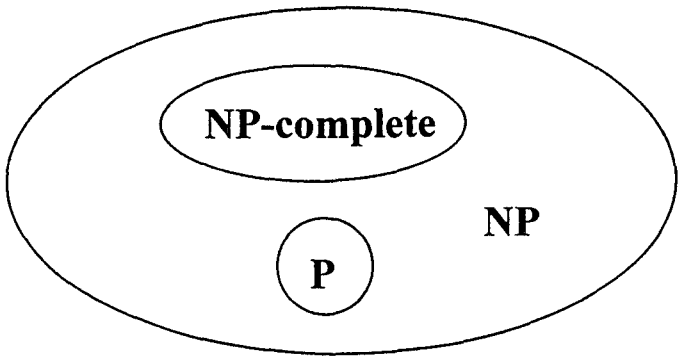
deterministic algorithms. To prove $P \neq NP$ it must be shown that there is a problem in NP which cannot be solved deterministically in polynomial time. This remains a leading question in theoretical computer science today. It is generally thought that $P \neq NP$.

2.7.3 NP-Complete Problems

The Hamiltonian circuit problem asks if there is a tour which visits every vertex exactly once. The travelling salesman problem (TSP) asks if there is a tour of sufficiently short distance which visits every vertex of the graph. The Hamiltonian circuit problem can be transformed to the TSP, this transformation takes a polynomial amount of time. If a polynomial time determinisitic algorithm is found for the TSP, there is also a polynomial algorithm for the Hamiltonian circuit problem.

There is a subset of problems in NP (see Fig 4) which are the hardest in the following sense: Any problem in NP can be polynomially transformed to any problem in the subset. So if there exists a polynomial algorithm for a problem in the subset then all of the problems in NP can be solved in polynomial time. The algorithms in this subset are called NP-complete problems and they are the hardest problems within NP. If we have a new algorithm L then to prove that it is NP-complete requires two things to be proven: first that the problem L belongs to NP, and second that there is a known NP-complete problem which reduces to L. A list of NP-complete problems was given by Garey and Johnson [GJ79] and is updated periodically in the *Journal of Algorithms*.

Figure 4. Diagrammatic Representation of NP



2.7.4 NP-hard

If a problem L is such that every problem in NP is polynomially transformable to L, then L is NP-hard. If in addition problem L belongs to NP, then L is NP-complete. Problems in NP are either 'decision' problems or 'optimisation' ones. Using the TSP as an example: the decision version might be '*is there a tour of length less than X?*' whereas the optimisation version would ask '*what is the optimal tour length for this TSP?*' The two versions are similar and clearly an algorithm for the decision version can be used a number of times to solve the optimisation version (although there is often a more efficient way to solve the optimised version). The terms NP-hard and NP-complete are frequently misused in the literature on combinatorial optimisation problems. When a problem is referred to as being NP-complete, what is often meant is that '*the decision version of this optimisation problem is NP-complete*'. NP-hard is preferred for describing optimisation problems.

2.7.5 Complexity of the FAP

This section shows that the frequency assignment problem is NP-hard. To find an F-colouring of a graph $G = (V, E)$, where $V = \{1, ..., N\}$, is known to be NP-hard. This can be transformed to the FAP with N links. A colouring of G is a set $\{f_1, ..., f_N\}$ of integer colours and the corresponding frequency assignment is the set $\{f_1, ..., f_N\}$ of frequencies satisfying the constraints $|f_i - f_j| \geq C_{ij}$, where $C_{ij} = 1$ if the edge connecting vertices i and j is in E and 0 otherwise. This shows that graph colouring is polynomially reducible to the FAP and hence that FAP is also NP-hard. Since the FAP is NP-hard exact algorithms will be computationally unrealistic for large problems (for approximately $N > 50$ [HST97]) and so this thesis concentrates on the development of heuristic algorithms.

Chapter 3

Literature Overview

In the last decade there have been major developments in communications technology. Unfortunately technological advances have not been able to expand the usable spectrum at the same pace. The electromagnetic spectrum is limited and has become a scarce resource. To accommodate the high demand the frequencies must be assigned in an optimal way. The frequency assignment problem and heuristic search have provided a highly active research area.

This chapter outlines the recent developments in the field of metaheuristics with particular attention being paid to techniques used to solve the frequency assignment problem (FAP). An earlier report [Lan89] covered this research area up to 1989 and this chapter aims to bring the study up to date. Variations of the FAP are also mentioned. In the second section the research objectives for this thesis are given in the context of recently published work. Finally the research contributions are given.

3.1 Research Motivations

3.1.1 Radio Communications

The first transatlantic radio signals were transmitted by Guglielmo Marconi at the beginning of the 20th Century. Many advances in radio communications have been made in the last century. Further developments will continue to be required to keep

pace with the expanding vision of communications researchers.

Some everyday examples of radio communications technology are: television, pagers, cellular phones, cordless phones, garage door openers, radio controlled model airplanes and security systems. The existence of radio communication is often taken for granted in today's living environment.

3.1.2 Frequency Use

There are many varied uses for radio communications. The three examples below give an indication of the frequencies used by familiar technology.

(Very) Low Frequencies (VLF and LF) in the range 3-300KHz create signals that can span thousands of miles and are typically used by submarine and maritime communications.

Frequencies in the range 300KHz-30MHz are not usually used for communications. Their lower bandwidth means that they support fewer channels. An example of this is AM broadcast stations.

Signals above 30MHz are typically used when the transmitter and receiver are within 25-30 miles of each other. An example of this is the FM broadcast band.

3.1.3 The Radio Frequency Spectrum

The usable frequency spectrum is limited both physically and in addition by constraints imposed by the ITU. In order that frequency availability can keep pace with increasing user demand it is essential that frequency use (and reuse) is maximised whilst still maintaining low interference levels. The recent developments that have made mobile communications widely available have acted as a catalyst for research in this area. Many papers on the solution of frequency assignment problems (FAPs) have been published in the last decade. This chapter aims to provide an overview of the developments and indications of where to look for further information.

3.1.4 Frequency Assignment Problems

In its most general form the frequency assignment problem simply requires frequencies to be assigned to communication links in such a way that demand is satisfied and interference is minimised. However, even a brief investigation into the literature will show that the FAP area encompasses a range of related problems which all differ in subtle, but important, ways. A theoretical treatment of many FAPs is given in [Hal80], where they are classified according to their computational complexity. It is beyond the scope of this chapter to survey each of the problems. A brief overview is given for each in the next section. The FAP is often formulated as a generalised graph colouring problem: [ZB77] [Hal80]. A review of FAPs and their relation to graph theory is given in [Lan89].

In addition to the various types of FAP there are also several other assignment descriptors. For example, *bulk assignment problems* assume knowledge of the positions of all transmitters and frequencies must be assigned to all communication links. The natural extension of this problem is the *update assignment problem* whereby transmitters are presented sequentially (with new requirements) and no knowledge of future demands is known. Variations on the general problem are also given in the aims of *minimum span frequency assignment* and *fixed spectrum frequency assignment.* For minimum span frequency assignment the aims are to satisfy demand, minimise interference and also to minimise the span (difference between the smallest and largest frequencies used) of frequencies used. In contrast, fixed spectrum frequency assignment requires frequencies to be assigned from a fixed set of frequencies. Typically this set is too small and not all constraints can be satisfied, so instead an objective function is minimised.

3.1.5 Types of FAP

Broadcast Frequency Assignment Problem (BFAP)

In the BFAP, found in terrestrial broadcasting, applications are made for transmission frequencies and the objective is to find an assignment which does not violate either the co-channel or adjacent-channel constraints. In addition the order (number of distinct frequencies used) of frequencies used in the final assignment should be minimised. Existing users are treated as new applicants with the additional constraint that they

should be assigned frequencies that they are currently licensed to use [Bay82].

Air Ground Air problem

This problem is based on the need for air bases to communicate with aircraft. The region of space in which it is possible for a base to communicate with an aircraft is known as the service volume. For convenience these are usually described by polygons or circles. Service volumes may overlap. The objective of the AGA-FAP is to enable communication between aircraft and ground station without interference from another ground station or any aircraft flying in the same or another service volume [Lan89]. Adjacent-channel, co-channel and intermodulation constraints are considered. Additional constraints may be imposed due to the tuning range of equipment, pre-assignment of frequencies to the transmitters and the existence of pools (collections of transmitters assigned the same frequency). Further details of the AGA-FAP can be found in Chadwick et al.[CMB92].

Frequency Assignment for Cellular Radio

Cellular networks (found in mobile telephony) typically exhibit regular geometries, the most commonly studied geometry is a mesh of regular hexagons. In its simplest form the area covered by the hexagon is served by a single transmitter at its center and all transmitters are identical.

The geometry of these problems is exploited when solving the FAP. In addition assignments are made to *channels* rather than *frequencies*. In fact the channels are simply a convenient way of labelling a narrow band of frequencies by using the central frequency. The required channel separation which will guarantee no interference between a given pair of links is referred to as the spectral unit. Work on cellular networks can be found in [And73] [Lee94] [Lee97] [CS98] [SHT98] and [Lee99-to appear].

Radio Link Frequency Assignment Problem

The bulk assignment, fixed spectrum Radio Link Frequency Assignment Problem (RLFAP) has been investigated in the research presented in this thesis (see Chapter 2).

The RLFAP [Ray92] [Bat et al. 95] is also referred to as the Radio Relay Network Frequency Assignment Problem (RRNFAP)[Lan89]. In the RLFAP the principal goal is to minimise the interference suffered whilst satisfying demand for communication links. Co-channel, adjacent-channel and far-site interference may be considered. Both minimum span assignments and fixed spectrum assignments have been applied to this problem. In addition bulk and update assignments have also been considered. The transmitters vary in power (and range) and are irregularly placed. A more detailed description is given in Chapter 2.

3.1.6 Heuristics

The FAP is NP-hard (see Chapter 2 section 2.6.5). This means that there is no known algorithm that can generate a guaranteed optimal solution in an execution time that may be expressed as a polynomial of the problem size. Practical problems involve large numbers of communication links and constraints and so solution by exact algorithms (see Chapter 4 section 4.2) are computationally infeasible. Heuristics (see Chapter 4 section 4.3) provide near optimal solutions within a reasonable amount of time. Many earlier heuristics mimicked the method used when solving problems manually — a form of sequential assignment. Whilst heuristic algorithms may improve on the solutions obtained by exact methods in a fixed time period the solutions are still often far from optimal for large problems. Since heuristics do not guarantee to find an optimal solution their success is often measured in terms of distance from a lower-bound (where one is available). The family of heuristics known as local search (LS) heuristics are moderately successful for large problems. They are often incorporated in the so-called metaheuristics. An introduction to local search is given in [Sch98], this is followed by local search and genetic algorithm implementations for scheduling.

Guided Local Search (GLS) has been applied to the RLFAP and has been shown to provide good results for the CELAR problems [VT96]. However, in the same study, it was found that GLS was out-performed by GAs. Descent algorithms have been applied to the Vehicle Routing Problem (VRP) [Osm93]. A greedy algorithm for T-colouring was investigated by [Ray94]. A dynamic locking heuristic is developed in [Hof94] and applied to the design of VLSI chips (graph bipartitioning). A problem-specific heuristic was developed in the implementation of an intelligent timetable information system for an integrated public transport system [Leh93].

Colin Reeves has written several introductions to heuristics and metaheuristics [RB95][Ree96a], these concentrate on SA, TS and GAs [Ree95c] in the most part but some also outline the distinction between heuristics and metaheuristics and briefly cover other areas of interest to researchers new to this field (No Free Lunch Theorem, 'hard' and 'easy' problems) [Ree96b].

3.1.7 Metaheuristics

> 'A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions'[OL96].

Some examples of metaheuristics are tabu search, simulated annealing, genetic algorithms, neural networks and their hybrids. Metaheuristics are discussed in more detail in Chapter 4 section 4.4. Implementations of tabu search and simulated annealing for the FAP are described in Chapter 7. Anyone considering investigating metaheuristics will find the paper 'Metaheuristics : A bibliography'[OL96] invaluable, it lists and categorises over 1400 papers on the theory and application of metaheuristics. Also included are sections on hybrid combinations of metaheuristics and comparative results. Metaheuristics have provided good quality solutions to a range of large-scale commercial optimisation problems. Applications which have been solved using metaheuristics are wide and varied, an indication of these areas follows.

Tabu Search Applications

An introduction to tabu search (TS) is given by Glover [Glo89] and refinements to the original algorithm are given in[Glo90][GTdW93] [DV93] [GTdW93] [WZ93]. A more recent publication by Glover and Laguna can be found in [GL95], this is an excellent review paper providing an extensive list of TS applications. Some examples of applications are error correcting codes [BB95], quadratic assignment problem [CSK93], maximum clique problem [GSS93], bandwidth packing [LG93], employee scheduling [MG86], scheduling [MR93], vehicle routing [Osm93], graph partitioning [RPG96], and convoy problem, course scheduling and flow shop problems are all discussed in [HdW90]. Herts

and deWerra have also applied tabu search to university course scheduling [HdW90]. Strategic oscillation and stochastic tabu search have been applied to a class of scheduling problems [MR93]. A reactive tabu search has been investigated by Battiti [Bat96], in this implementation feedback parameter tuning uses the past history of the search to increase its efficiency in solving the QAP. Finally, tabu search with strategic oscillation is used by Kelly et al [KGA93] to investigate large scale rounding of census data.

Simulated Annealing Applications

Simulated annealing (SA) was introduced by Kirkpatrick et al [KGV83]. This technique has been applied to a large number of applications and a number of variants have been implemented. A good introduction to simulated annealing is given by Dowsland [Dow95], this paper also gives an overview of recent application areas. Some applications are multi-objective scheduling and timetabling [TD96][Dow96], graph partitioning [Joh et al. 89], graph colouring [Joh et al 91], vehicle routing with time windows [CR96], and the travelling salesman problem (TSP) [MO96]. A parallel application of SA for the TSP was investigated by [Soh96]. The convergence of simulated annealing with feedback temperature schedules applied to job-shop scheduling problems was investigated by [KT97]. A hybrid application combining SA with a SFC (space-filling curve) heuristic has been applied to the TSP by [LH94]. A review of many applications is given by Johnson and McCeoch [JM96].

Genetic Algorithms Applications

Genetic algorithms (GAs) were first introduced by Holland [Hol75] and have since been applied to a variety of applications, many variants have also been developed. A good overview of genetic algorithms is given in [BBM93a] with applications and research topics being covered in [BBM93b]. A variant known as expansive coding is discussed in [BBM93c] and [BBM94]. Expansive coding is a representation methodology which makes complicated Combinatorial Optimisation Problems (COPs) easier to solve for a GA. The representation, operators and fitness function used by the GA are more complicated but the search space becomes less epistatic and is therefore easier for the GA to solve. Reeves [Ree95c] also provides a good introduction to genetic algorithms and brings the review of applications up to date. Two examples of genetic algorithm

applications are graph colouring [FF96], and the travelling salesman problem [FM96].

Hybrids and Their Applications

Yamada and Reeves [YR98] have investigated a SA/TS/GA hybrid and applied this to the flow-shop scheduling problem. Their algorithm obtained very good results compared to published benchmarks. Variants of SA/TS have been applied to the Vehicle Routing Problem (VRP) with success [Osm93]. A theoretical paper proposing an algorithm which combines features of TS/SA/GA with potential applications in optimisation is given in [Fox92]. Another GA hybrid has been used to solve a crossword problem with an expensive evaluation function [Ran et al.93].

Comparisons of Heuristics and Metaheuristics

Several authors have carried out experiments to compare the performance of different (meta)heuristic approaches:

A comparison of GA, hill-climbing, random search, SA and TS in solving the Vehicle Routing Problem (VRP) where the time is limited to ten minutes has been carried out by [BMP95]. Simulated annealing proved to be the best technique overall, and was also robust when acting on a variety of problem instances. When the algorithms were run for four hours SA still obtained the best (and most reliable) solutions. SA and TS are compared by [AC96] for 0-1 linear programs. TS, SA and GAs have been applied to a special case of the Quadratic Assignment Problem (QAP) for a real-world application [Sin93]. TS and SA were applied to T-colourings and compared to a branch-and-bound algorithm by Costa [Cos93]. A comparison of SA and GAs applied to the ring-loading and ring-sizing problem is given in [MS98], Mann and Smith have also applied SA and GAs to traffic routing for telecommunications with the aim of load-balancing [MS96].

Further Comments

Problem-specific implementations should out-perform generic ones but, in my opinion, it is also important to carry out experiments with general purpose algorithms which are able to deal with problems without any special structure. Generic versions of TS and SA have been implemented in the research presented in this thesis.

A modified SA has been applied to workforce scheduling by Lesiant [Les98]. In this paper he stresses the importance of having generic problem modelling and solving capabilities which still allow for the customisation and fine-tuning of core resolution mechanisms. Radcliffe and Surry [RS95] advocate the exploitation of domain-specific knowledge in order to provide techniques acceptable to practitioners (requiring solutions to complex real-world search problems).

3.1.8 Evaluating Heuristic Performance

Guidelines for reporting on heuristic and metaheuristic investigations are given in [Bar et al. 95] [Ree 95b]. Both references highlight three main ways in which the effectiveness of (meta)heuristics can be compared.

1. Analytical.

   Analysis of worst-case and average performance analysis. As yet developments in this area for heuristic (rather than exact) algorithms is sparse. Some work has been done on finding lowerbounds for the FAP [SH97], there is some dispute as to whether the conditions under which these bounds are constructed are effective for practical assignment problems [Bat et al 98] [JDB98].

2. Empirical Testing

   Comparison of performance against existing techniques on a set of benchmark problems. Unfortunately real data are scarce and so benchmarks are often randomly generated to reflect assumed characteristics of real problems. A large number of instances for a variety of problems can be obtained electronically as described in [Bea90].

3. Statistical Inference

   Again this is an area where there is much scope for further research. It has been suggested that the statistical theory of extreme values (which applies to continuous distributions) can be applied to COPs (with discrete objective function values).

The method of empirical testing is widely used to compare the results of heuristic algorithms. There are no set guidelines for empirical testing although several performance

measures have been proposed [Bar et al 95] [Ree 95b]. The three main performance measures are quality of solution, computational effort and robustness. All of these performance measures have been considered when evaluating each of the metaheuristics developed in this thesis.

In addition to performance measures Bar et al. give detailed guidelines on the design of heuristic experiments [Bar et al. 95]. The abstract of this paper reads:

> "This report discusses the design of computational experiments to test heuristic methods and provides reporting guidelines for such experimentation. The goal is to promote thoughtful, well-planned, and extensive testing of heuristics, full disclosure of experimental conditions, and integrity in and reproducibility of the reported results."

Key points taken from this paper have lead to the following guidelines being adopted throughout the research presented in this thesis.

- The quality of a solution should be defined by a stated evaluation metric or criterion.
- The experimenter should fully specify the steps and stopping rules of new methods.
- A research experiment should have a purpose, stated clearly and defined prior to the actual testing.
- Algorithms should be tested with their best competition. Rather than making comparisons with published results on different problems and machines, it is preferable to obtain (or create if necessary) software for the competing methods and make comparisons within the same computing environment. All of the algorithms developed were written by the same programmer and run on the same platform.
- It is important to "stress-test" the codes by running as large instances as possible. Many factors do not show up on small instances, and running on smaller sizes may not yield accurate predictions for larger, more realistic problems. Real-world problems reflect the ultimate purpose of heuristic methods, and are important for assessing the effectiveness of a given approach.

- Values of any parameters employed by the heuristic should be defined and, where problem-dependent, the rules for establishing appropriate values should be specified.
- Some attempt to identify factors or factor combinations that seem to contribute to (are correlated to) performance.
- Time taken should clearly state whether creation of data structures, preprocessing and other preliminary stages are included in the timing. The times given in this thesis include all stages.
- Reproducibility. All experiments should be thoroughly documented including detail of the algorithm, parameters and non-trivial data structures.

3.1.9 (Meta)Heuristics for the FAP

An investigation of the suitability of metaheuristics for the FAP is given in [Lan89]. A great deal of work has already been done for the frequency assignment problem which investigates the use of heuristic neighbourhood search and other techniques. Much of this work is referred to below. A number of technical documents resulted from the CALMA project and these can be obtained from [CAL95], and also include [Aar et al. 95] [BB95b] [THL95].

Heuristic Approaches

FAP (with multiple frequency domains) has been formulated as an integer linear programming problem and solved using a branch-and-cut algorithm which employs problem reduction methods [Aar et al.95]. A generalised greedy algorithm for FAP was presented by deWerra and Gay [dWG94]. Kim and Kim [KK94] proposed an efficient two-phase optimisation procedure for the channel assignment problem based on frequency reuse patterns.

Tabu Search + Variants

Tabu search seems to be a promising method for the RLFAP, as shown in the literature [Cos93] [CSH93]. Castelino [CS95][CHS96] investigated the application of genetic al-

gorithms, tabu search and tabu thresholding. Castelino [Cas97] also investigated tabu search with surrogate constraints and gives a good overview of the literature for the FAP until 1997. Boyce and Boujou [BB95b] have applied TS to the RLFAP, they show that arc-consistency is an effective pre-processing operation for the CELAR problems which could be dramatically simplified by its use. In addition they discovered that by satisfying constraints with higher required frequency separation values first the run-time of the algorithm was reduced.

Simulated Annealing

Two independent approaches to the use of SA to solve channel assignment problems each using different models and different neighbourhood structures were proposed by [DKR93] and [MM93].

Genetic Algorithms + Variants

A comparison of several chromosome representations used within a parallel genetic algorithm has been carried out by [CHS94a]. Crompton et al have investigated the application of GAs [CHS94a] and parallel GAs to the FAP [CHS93][CHS94b]. A hybrid GA/greedy algorithm has been investigated [VJH98] on small minimum span frequency assignment problems and has been found to obtain better results than sequential algorithms.

Problem Modelling and Bounds

Lower-bounds for the FAP have been investigated by Gamst [Gam86], Geomans and Kodialam [GK93] and Smith and Hurley [SH97]. Lower-bounds for a simplified (using problem reduction) cellular constraint network has also been investigated [SH97]. Smith, Hurley and Thiel [SHT98] exploited their lower-bound findings by assigning a maximal clique to generate an initial solution, fixing this assignment and then solving the remaining graph. By using this technique they have obtained the best results published so far for some of the Philadelphia problems. For a regular hexagonal lattice it is suggested by [JDB98] that binary constraint satisfaction problems do not sufficiently represent the FAP and that higher-order constraints are able to provide a more

adequate model which potentially uses fewer spectral resources. In addition to this argument Bater et al. [Bat et al. 98] cast doubt on the validity of lower-bounds calculated using maximal cliques [SH97]. They suggest that 'holes' in the coverage which occur as a result of using binary constraints make the use of binary constraints inadequate and consequently that lower-bounds calculated using maximal cliques on a binary constraint graph lead to optimistic estimations of the required spectral resources. Work on modelling the FAP has also been carried out by [GL97] [Bat et al.98] [BJC98] [Dun et al 98] [JDB98].

Cellular Assignments

Cellular radio assignment has unsurprisingly enjoyed the most industrial funding. This is due to the direct application of this problem to mobile telephony which has seen a dramatic increase in popularity in recent years. Publications in this area have been prolific in the last couple of years, research has been carried out by [MM93] [You], applications using simulated annealing [DKR93] have also been considered. A comprehensive survey of assignment schemes for channel assignment is given in [KN96]. A two-phase approach is considered by [KK94] and graph-theoretic developments are described in [Sen et al. 94]. Leese [Lee99] has presented a unified approach to the problem of using regular tilings to cover a lattice of hexagonal cells, this approach combines co-channel and adjacent-channel lattices. A two-phase adaptive local search algorithm has been applied to cellular radio networks. This approach combines a deterministic-probabilistic neighbour generation with a two-phase local search [WR96]. Further work on cellular frequency assignment can be found in [And73] [Lee94] [HST97] [GL97] [Lee97] [SH97] [SHT98] [JDB98] [Dun et al. 98] [CS98]

Graph Theory

FAPs are commonly solved using techniques borrowed from the theory of abstract graphs, more specifically, graph colouring. Roberts [Rob91] concisely reports on the graph-theoretic proofs applicable to T-colouring and highlights some open problems concerned with T-colourings and their variations and generalisations. A variation on SA, called probabilistic iterative improvement [Mor89], has been applied to the graph colouring problem (GCP). This paper also investigates other definitions of a 'move'

applicable to restricted graphs. Chromatic scheduling and frequency assignment are discussed in [dWG94] where upper-bounds are given for a generalised chromatic number. Graph colouring algorithms have been applied to the FAP [Pee91] [PL96] [Smi88]. Chromatic scheduling and frequency assignment is the topic of a paper by de Werra and Gay [dWG94]. Raychaudhuri investigates maximal clique frequency assignment and traffic phasing in [Ray92]. Graph colouring by cliques has been investigated by [WZ93b] [Hal93]. Finally, Tuza presents a paper on graph colouring [Tuz92].

Hybrid Approaches and Metaheuristic Comparisons

Hybrids using SA, TS and GA and local search techniques have been applied to both the FAP [SHT98] and to a wide variety of other problems [Fox93]. Comparisons of several metaheuristics applied to the FAP are given in [HST97] [HTS96].

Divide and Conquer

An extensive literature search has revealed very few papers which consider divide and conquer techniques applied to the FAP. Some divide and conquer papers for general applications are [FK92] [KMD95] and [WF96]. A divide and conquer algorithm for the minimal cut bisectioning of graphs is given in [LSM96]. Some early results from the methods described in this thesis were given in [PS98], results from chapter 10 can be found in [Whi98] and a further paper is soon to be submitted. Despite extensive literature searches and discussion with experts in the field, there are no papers considering the combination of divide and conquer with heuristics that the author is aware of. Most of the FAP publications describe techniques that have been applied to the whole problem throughout computation, although early processing may concentrate on the vertices with large vertex degree.

3.2 Benchmarks

3.2.1 CELAR

In 1994/5 an 18 month project was undertaken by the French, Dutch and British industries and academia. The CALMA[1] project [CAL95] was funded by EUCLID[2]. The main objective of the project was to gain a better understanding of what makes a combinatorial problem-solving approach adequate on some problems, and inadequate on others, in order to improve the cost effectiveness of the many military systems relying on the proper treatment of difficult combinatorial problems. This investigation was carried out using a characteristic example: RLFAP. A realistic data set was obtained from CELAR[3], France and each member of the consortium applied its own expert knowledge to solve the radio link frequency assignment problem. Methods tried included: genetic algorithms; interior point methods; constraint satisfaction methods; simulated annealing; local search and tabu search. The resulting performances of these techniques were compared and their success assessed. It was found that standard local search was able to give reasonable solutions in moderate running times. However, there was a strong positive correlation between the amount of problem-dependent information used, the extent to which mathematical insight is exploited, the development and implementation effort required and the quality of the results obtained [THL95]. Many of the techniques developed to solve the CELAR problems relied heavily on the special property of the data, in particular the existence of an equality constraint between consecutive odd and even links [Aar et al. 95] [Bou et al.95b].

The CELAR data set was fundamentally different from the TNET data used in this thesis, a brief discussion of these differences is given in Chapter 6 section 6.4.

3.2.2 Philadelphia

A large number of papers relating to mobile communications have been published recently, for example [SHT98] [Lee94] [Lee97]. For the most part these are based on the 'Philadelphia' problems, originally presented by Anderson [And73] where the un-

[1]CALMA = Combinatorial Algorithms for Military Applications.

[2]EUCLID = European Cooperation for the Long Term in Defence.

[3]CELAR = Centre d'Electronique d'Armement.

derlying graph is based on a regular hexagonal cell structure. Unfortunately these do not generalise to random graphs and so the principles cannot be exploited in solving tactical communications-based problems.

3.3 Software

Three software toolkits have been developed at the University of East Anglia (UEA): GAmeter, SAmson and TABasco. Each of the toolkits is based on a different meta-heuristic: genetic algorithms, simulated annealing and tabu search respectively. All of the toolkits have been built on the Common Toolkit Framework (CTF). This allows the problem-specific parts of the program to be ported to each of the toolkits developed thus enabling a direct comparison of the techniques. The CTF also defines many of the user interface routines providing consistency across the toolkits developed.

GAmeter [MKS95a] is a software toolkit which enables the user to specify the problem dependent parts to be acted on by a standard genetic algorithm (GA) with optional add-ons. GAmeter provides an environment for developing and experimenting with optimisation problems. This toolkit was used during the CALMA project. Several representation options, crossover operators and mutation operators are available. Extensive statistics are used to provide performance measurement of the toolkit. Empirical studies implied that the developed GA was less sensitive to parameter tuning than simulated annealing; this was seen to be an advantage. Typically the success of GA approaches are dependent on 'good' representations of the problems, particularly for problems where the search space is epistatic.

SAmson is a general software package which enables the user to specify the problem dependent parts to be acted on by a simulated annealing algorithm. Like GAmeter, SAmson can be used with several representation options. A number of neighbourhood functions are available for use on those representations (for example, for binary representations, a neighbour is obtained by replacing one bit by its complement and, for integer representations, one integer is replaced by another chosen at random). However, studies have shown that for most problems, a problem-specific neighbourhood function will outperform many of these standard functions.

TABasco, the third software toolkit from UEA, is a general purpose tabu search toolkit.

As with the previous toolkits, many of the representations and neighbourhood functions are provided. In addition the tabu memory data structures are linked to the chosen representation. The parameters can be fine-tuned after the problem-specific aspects have been specified. This toolkit is currently a beta release.

FASoft is a system for discrete channel frequency assignment problems. It incorporates simulated annealing, tabu search and genetic algorithms for solving the FAP as well as backtracking and hill climbing. Numerous vertex ordering, frequency selection and move definition options are available and a weighted multi-objective objective function is used. The system is able to act on data which can be described using a constraint matrix (see Chapter 5 section 5.4.4). More details of FASoft are given in [HST97] with extensive references to the lower-bounding techniques described in [SH97]. In [HST97] results are primarily presented for minimum span cellular radio applications including the Philadelphia problems.

3.4 Research Objectives

Having described the general problem and discussed previous work, it is now possible to state the objectives of this research. The main concern of this thesis is to conduct an investigation of tabu search, simulated annealing and divide and conquer methodologies for the solution of a class of simulated but realistic FAPs.

The research objectives for this thesis are:

1. To establish a software library of techniques based on existing and novel algorithms and to compare the performance of the alternative techniques.

2. To develop metrics for measuring the goodness of assignments.

3. To investigate how to split the problem into subproblems for more rapid solution.

4. To investigate the benefits of the divide and conquer technique compared with the application of a single heuristic.

5. To study the significant factors which influence performance, e.g. vertex ordering, frequency selection and the provision of an incremental objective function.

6. To devise suitable experiments to assess the performance of each algorithm in isolation, and with other available techniques for solving a class of simulated but realistic problems.

7. To compare the difference between problems considered in this thesis and others in the literature; for example the CELAR data.

3.5 Research Contributions

1 *To establish a software library of techniques based on existing and novel algorithms and to compare the performance of the alternative techniques.*

A suite of programs have been developed. The following algorithms are available for solving the MATRIX test data: hill descending, steepest descent, greedy, backtracking. The following algorithms are available for solving the TNET test data: hill descending, steepest descent, greedy, SA, TS, DCSA and DCTS.

2 *To develop metrics for measuring the goodness of assignments.*

The initial suggestion of counting the number of constraint violations has been superceded by a new objective function which calculates the sum of the positive discrepancies of the violated constraints.

3 *To investigate how to split the problem into subproblems for more rapid solution.*

The divide and conquer program divides the problem by dividing the frequencies available into separate bands and then solves the smaller versions of the problem before recombining them to obtain a solution to the whole problem. Several division criteria were investigated and division of the frequency set yielded the higher quality results. This division criterion has the benefit of being independent of the constraint network and should therefore be applicable to a range of FAPs.

4 *To investigate the benefits of hybrid techniques compared with the application of a single type of algorithm.*

The divide and conquer (DC) program, using either simulated annealing or tabu search as the improving heuristic, outperforms the corresponding heuristic when used alone. The DC algorithm obtains solutions with less interference, in less time and using fewer resources than the single heuristic program for realistic non-trivial data sets.

5 *To study the significant factors which influence performance, e.g. vertex ordering, frequency selection and the provision of an incremental objective function.*

Early experimentation showed that the best results were obtained by pre-ordering the links in descending order of co-site vertex degree. For algorithms evaluating all possible neighbourhood moves at each iteration the selection of an equivalent best-cost frequency at random provided the best results. The implementation of an incremental objective function was found to have significant speed advantages enabling better assignments to be found in a fixed time period. Data structures are described which enable a very fast incremental objective function to be employed.

6 *To devise suitable experiments to assess the performance of each algorithm in isolation, and with other available techniques for solving a class of simulated but realistic problems.*

Extensive computational results are provided to compare the effectiveness of each of the algorithms developed (simulated annealing, tabu search and divide-and-conquer). In conclusion simulated annealing was found to yield good quality results consistently and these were further improved upon when simulated annealing was used within the divide and conquer framework. Tabu Search proved effective for the easier problems but was less predictable. Once again results were improved when tabu search was incorporated into the divide and conquer framework. The low-cost divide and conquer results were found in less time, using comparable of fewer resources than either simulated annealing or tabu search used alone when acting on non-trivial data sets.

7 *To compare the difference between problems considered in this thesis and others in the literature; for example the CELAR data.*

Chapter 6 section 6.4 describes the differences between the TNET and CELAR data and explains why the techniques employed here are not appropriate for solving the CELAR scenarios.

Chapter 4

Basic Techniques

Perhaps the simplest of all search algorithms is one of generate-and-test. A solution is generated randomly and this is compared with some acceptance criteria. If it meets these then it is accepted; otherwise a new random solution is tried. Such primitive methods rarely give satisfactory results. Fortunately research over the last couple of decades has provided a variety of more advanced search techniques from which to choose. These make use of the approximate smoothness of the objective function — the objective function evaluated at two solutions 'close' in the search space have similar values. The techniques are described with particular reference to the frequency assignment problem. However, they are general techniques which can be applied to most combinatorial optimisation problems. Algorithms can be classified into two distinct types; exact and heuristic.

In this chapter the term 'combinatorial optimisation' is explained. This is followed by a description of exact techniques which guarantee that a globally optimal solution will be found. The need for heuristic techniques is then discussed and metaheuristics are briefly introduced. General heuristic algorithms are described, along with the relevant terminology, before a more detailed discussion on several metaheuristic techniques is given. Simulated annealing and tabu search are described and definitions of the terms used when discussing these algorithms are given. Finally the divide and conquer technique is described. All of the algorithms described in this chapter have been used in the investigations described in this thesis. Chapters 7 and 8 will discuss the implementation details of simulated annealing, divide and conquer and tabu search for the

Frequency Assignment Problem (FAP).

4.1 Combinatorial Optimisation

In its simplest form the frequency assignment problem can be described as a minimisation problem taking the general form: minimise $f(x)$ subject to $g_i(x) \geq b_i \quad \forall i$. Here x is a vector of variables, for example, an ordered list of frequencies describing the assignment, $f(\cdot)$ and $g_i(\cdot)$ are general functions. In the particular case $f(\cdot)$ would be the objective function used to evaluate the interference of a proposed assignment. If we restrict the values that the decision variables may take to a discrete set then the problem assumes a combinatorial nature. The problem of finding an optimal solution to a problem like this is therefore known as combinatorial optimisation.

Over the past few years there has been considerable academic interest in the development of combinatorial optimisation techniques; it remains a fruitful area of research largely owing to the vast number of practical applications. Well known combinatorial optimisation problems include the travelling salesman, knapsack and the frequency assignment problems. Applications such as vehicle routing and bin packing are readily described as combinatorial optimisation problems.

Typically combinatorial optimisation problems have very large numbers of possible bad solutions and considerably fewer good solutions. There are generally several good solutions which are a long distance from one another and which would require a number of transition moves to arrive at one from the other. Typically the series of moves required are not all improving steps making it difficult to recognise paths leading to alternative good solutions. This is essentially what makes solving combinatorial optimisation problems so difficult.

4.2 Exact Techniques

For small combinatorial problems it is possible to guarantee that certain algorithms will find the (globally) optimal solution. Such methods are known as exact techniques and rely on some form of implicit enumeration. Techniques such as backtracking and branch-and-bound [HS78] examine the possible solutions systematically and generally

find optimal solutions in less time than would be required for complete enumeration.

4.2.1 Exhaustive Search / Complete Enumeration

By definition finding a solution to a combinatorial optimisation problem requires that some vector of values from a set of discrete elements is found. If we were to generate all the possible solutions to a problem we could evaluate each in turn and would then be able to find the best solution. This method would guarantee to find the global optimum solution if one exists. The simplicity of complete enumeration is appealing. However it is far from efficient and its computing time grows exponentially with problem size. This phenomenon is commonly known as combinatorial explosion.

Consider the frequency assignment problem. If there are $N = 10$ links, and $|F| = 10$ frequencies then the search space has size $|F|^N = 10^{10}$. If we have a machine capable of enumerating all solutions to this problem in one hour then for an 11 link, 10 frequency problem it would take 10 hours. Similarly the following table (Table 2) indicates how long it would take to consider all solutions for other values of N with $|F| = 10$.

Table 2. Combinatorial Explosion

N	$\|F\|$	Time
10	10	1 hour
11	10	10 hours
12	10	4 days
13	10	1.5 months
14	10	over 1 year
15	10	over 1 decade
16	10	over 1 century

An average tactical communications network could have 250 links and 50 frequencies giving a search space of 50^{250} which is more than 10^{400}. Clearly this method is suitable only for very small problem instances. More advanced techniques are needed to solve more realistic problems.

4.2.2 Backtracking

Backtracking is in its worst case equivalent to complete enumeration but on average it performs far better and is a reliable technique for small problems.

When the solution space consists of ordered configurations of elements then each prefix[1] to a solution represents a partial solution. If it can be shown that the prefix does not lead to any desired solutions then there is no need to expand the prefix and continue searching that branch. For frequency assignment an optimal solution is ideally a complete assignment with zero interference; alternatively a non-zero lower-bound of the problem may be known. The backtracking algorithm begins with the smallest possible configuration and continues to add elements until an optimal solution is reached or until it can be proven that no optimal solution exists with that prefix. If the latter is true the algorithm backtracks by removing the last element from the configuration and replacing it with the next possibility. The pseudocode is given in Fig 5.

A variation on backtracking uses a threshold cost value and any solutions which would incur a cost greater than this threshold are not investigated. In this way the algorithm can prune the remaining solutions to evaluate and so reduce the time which would be required to evaluate all the solutions. Conventional backtracking has a threshold value of zero, that is a perfect solution is sought and the lowerbound is zero. However it is sometimes the case that a good but imperfect solution will suffice or that the lowerbound is known (and non-zero) and in this case the threshold value can be set higher. Using a threshold value greater than lowerbound prevents the algorithm from guaranteeing optimality but enables good but imperfect solutions to be found more quickly. A number of backtracking variants have been investigated by Prosser [Pro93].

[1] A possible solution to the frequency assignment problem is represented as an ordered list of frequencies which are assigned to the links in the corresponding position. For this problem a prefix to a solution might be an assignment of frequencies to only the first x links, the rest being unassigned

Figure 5. Backtracking Pseudocode

```
C := SENTINEL
While (links unassigned) AND (iteration < max-iters) AND (still solns to check) Do
    For all non-forbidden frequencies (forbidden if previously backtracked)
        C' := number of violations if assign this frequency to this link
        note frequencies yielding minimum cost
        calculate ΔC := C − C'
    EndFor
    If    C' < threshold
    Then  assign frequency according to selection criteria
    Else  undo last assignment and make frequency forbidden
EndWhile
```

4.3 Heuristic Techniques

The word 'heuristic' means 'guiding in investigation'[2]. Heuristic has become a term in common usage in the field of combinatorial optimisation to describe techniques which do not guarantee to find the global optimal solution. The following definition is well-established.

> 'A heuristic is a technique which seeks good (i.e. near-optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is.'[RB95]

[2]From Collins Dictionary and Thesaurus 1987

In the last twenty years several thousands of papers have been published that use heuristics to solve combinatorial optimisation problems. Whilst many mathematicians consider the approach to lack rigour and despair at the lack of proofs it is clear that empirically these methods have been very successful.

If we take the naive approach of complete enumeration it is not difficult to see that whilst it is possible in principle to solve any problem, in practice it is not. This is because of the vast number of possible solutions to any problem of a reasonable size. Generally the search space increases exponentially with problem size thereby limiting the size of problems which can realistically be solved using exact techniques.

Heuristic techniques search the problem space 'intelligently' using knowledge of previously tried solutions to guide the search into fruitful areas of the search space. Local search is an example of a heuristic technique. In recent years a number of so-called *metaheuristic* techniques have been developed; typically these use other heuristics to guide the local search procedure. Examples of such metaheuristics are simulated annealing, tabu search and genetic algorithms. Briefly these can be outlined as follows.

Genetic algorithms are based on population evolution. At each iteration a population (set) of possible solutions is considered. By selecting pairs of solutions and combining them to produce new solutions the set is increased. Lower quality solutions are then removed from the set and the process is repeated. Combinations of good characteristics from the solutions are propagated into the next population and so the average quality of the solutions in the population gradually improves. The algorithm is generally terminated after a set number of iterations.

Tabu search always performs a move to the best solution in a certain subset of a neighbourhood. The quality of solutions visited during the search is not necessarily always improving. To prevent this method from cycling, usually several recently visited solutions are removed from any neighbourhood. A stopping criterion has to be defined, for example the number of iterations without improvement.

Simulated annealing modifies a solution to one chosen at random from its neighbourhood. Improvements are always accepted. Deteriorations are accepted with a certain probability, decreasing during the run. Simulated annealing stops when this probability becomes smaller than a specified parameter or a solution below a specified threshold is found.

Several papers [HTS96] [THL95] have indicated that simulated annealing and tabu search are more effective for solving the FAP than genetic algorithms and so these two metaheuristics have been investigated in the research presented in this thesis.

An in-depth discussion of genetic algorithms is outside the scope of this thesis, the interested reader may refer to a [BBM93a] [BBM93b] [Ree95c]. Tabu search and simulated annealing are described in more detail in sections 4.4.2 and 4.4.1 respectively.

4.3.1 Local Search

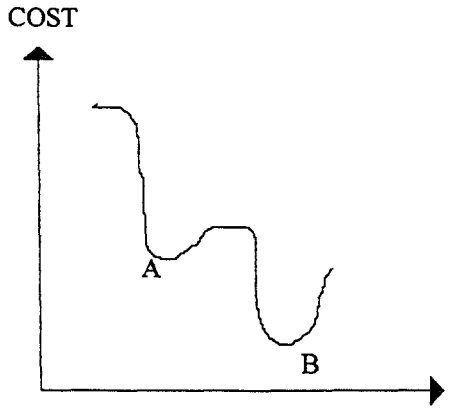
Local search is a general heuristic algorithm which can be applied to find approximate solutions to hard combinatorial optimisation problems in a reasonable time. The set of all solutions is known as the search space. Starting with an initial solution we define its neighbourhood as the set of solutions which differ from the initial solution in some small way. The technique used to change the solution to a neighbour is referred to as a 'move'. At each iteration of the search we apply a move to the current solution with the aim to reach, after more iterations, a better solution in the search space. Various search strategies use the neighbourhood mechanism, the concepts of 'move', 'neighbourhood' and 'local' and 'global' optima will be helpful in understanding the strategies.

Local and Global Optima

A feature of many combinatorial optimisation problems is that they have only a small number of 'global' optima, and considerably more 'local' optima. This causes problems for a number of search paradigms - consider the graph (Fig 6).

If an algorithm accepts improving moves only it will become trapped at A. In order for the search to continue it must also accept disimproving moves, in which case it can escape the local optimum at A and may then find the global optimum at B. In order fully to understand what is meant by local and global the concept of a neighbourhood must be explained.

Figure 6. Graph Showing Local and Global Optimum



Neighbourhoods

If we have a solution x then its neighbours are those solutions which can be reached from x by some simple operation. This simple operation is often referred to as a 'move'. A typical move for the frequency assignment problem might be the assignment of a different frequency, from the set F, to one of the N links, in which case any given solution has $N(|F| - 1)$ neighbours. If a solution x is better than any solution in its neighbourhood then x is the local optimum with respect to this neighbourhood. A global optimum is defined as a solution whose cost is less than or equal to any solution in the whole search space.

Defining a Move

A 'move' is a usually a small change of one of the variables defining the current solution. In frequency assignment a typical move involves assigning a different frequency to a given link. The frequency may be chosen at random from those available or it may be the best frequency available after evaluating all possible alternatives. For a more basic technique such as generate-and-test a 'move' may mean generating an entirely new solution by randomly assigning frequencies to links, a rather larger change to the solution. The definition of a move is problem and algorithm dependent.

4.3.2 Selective Algorithms

Selective algorithms are so called because they begin with a complete solution and move from one solution to another in search of a global optimum. In moving from one solution to the next the characteristics of the original solution are changed. The acceptance (or selection) of the generated move will depend on the search algorithm being used.

Hill Climbing

Hill climbing is the term used to describe iterative search techniques which accept only improving moves. 'Climbing' implies maximising some function describing the solution quality, but the technique could similarly be used for minimising functions. In hill climbing (descending) the heuristic is simply 'accept a solution if it is better (in some sense) than the current one'. If no disimproving moves are accepted the technique is susceptible to becoming trapped in the first local optimum it finds, which may or may not be the global optimum. Typically a single move is generated at random from the available neighbourhood at each iteration.

Steepest Descent

Steepest descent is a variation on hill climbing (descending) whereby all possible moves from the current solution are considered and the best one is selected to provide the next solution. The generation of the initial solution and the definition of a 'move' are the problem specific decisions which affect the quality of the final solution. Pseudocode for the steepest descent algorithm is given in Fig 7.

The steepest descent algorithm is easy to code and results can be obtained quickly. The major disadvantage of this technique is its tendency to become stuck in a local optimum (minimum) early in the search. This local minimum may not necessarily be the global optimum and once a local optimum is reached the algorithm terminates. It is common to use steepest descent to obtain a reasonably good initial solution to be acted on by more advanced algorithms. It may also be used as a fast final step, to improve further the best solution obtained by a different heuristic — if possible.

Figure 7. Steepest Descent Pseudocode

```
LocalOptimum := FALSE
C := cost(S_initial)
S_current := S_initial
While LocalOptimum = FALSE AND C > 0 Do
        BestCost := SENTINEL
        For all S_i ∈ N(S_current), i = 1...N(|F| − 1)
                C' := cost(S_i)
                If C' < BestCost Then
                        BestCost := C'
                        BestSoln := S_i
                EndIf
        EndFor
        IF BestCost < C Then
                C := BestCost
                S_current = BestSoln
        Else    LocalOptimum := TRUE
        EndIf
EndWhile
```

4.3.3 Constructive Algorithms

Constructive algorithms differ from selective algorithms in that they gradually build up a solution to the problem. Backtracking is an example of an 'exact' constructive algorithm, and the greedy algorithm is a 'heuristic' constructive algorithm. Initially the 'solution' is empty, elements are added one by one after some selection criteria have been satisfied. Arriving at a complete solution to the problem usually terminates the algorithm. These methods use a form of sequential assignment which mimic the way a

frequency assignment problem may be solved manually.

Greedy Algorithm

Greedy heuristics attempt to solve the problem in a single pass. Consider the frequency assignment problem. The general principle is to place the links in some order and then consider them in sequence, assigning a frequency to each in turn. The assignment of frequencies is based only on knowledge of previously assigned links and lacks any look-ahead facility which causes it to incur large costs towards the end of the algorithm. The cost of a solution is some measure of the level of interference caused by the assignment. With this method the initial ordering of the links is essential to the success of the algorithm, while the method of selection of a frequency is also important. Several methods have been suggested for the initial ordering of the links: random ordering, numerical ordering and largest vertex degree ordering. The vertex degree is the number of other links with which the given link has a constraint. Often the frequency selection method is to accept the frequency which would yield least cost although frequency reuse and less exhaustive methods have also been used. Some algorithms allow the ordering to be changed during the search incorporating knowledge of previously assigned transmitters. Another possible extension of the basic greedy algorithm is to repeat the process after finding an initial assignment. This involves making another pass of the links, but whereas the first pass had only knowledge of the assignments made to previous links, the second pass has knowledge of the whole first pass assignment. The algorithm is still greedy in the sense that the best re-assignment possible is made each time a link is considered. The pseudocode for the greedy algorithm is given in Fig 8.

The greedy heuristic generally only makes one pass of the links and so (assuming that the time taken to select a frequency is constant) takes time $O(N)$. Whilst this method is simple and fast it does not produce good results for complex problems. It is often used to generate the initial solution that is acted on by algorithmically superior search methods which make many iterations and explore the search space systematically.

Figure 8. Greedy Algorithm Pseudocode

```
Initialise assignment to nil
Order the links
For each link (in turn) Do
    For all frequencies (in ascending order) Do
        cost := number of violations if assign this frequency to this link
        note frequencies yielding minimum cost
    EndFor
    assign frequency according to selection criteria
EndFor
```

4.4 Metaheuristics

4.4.1 Simulated Annealing

Simulated annealing is an iterative heuristic search procedure which begins with a possible solution and proceeds to explore the neighbourhood space via a sequence of moves in search of optimal solutions. In this algorithm one possible random move is considered at a time. The neighbourhood solution is compared to the current one according to an appropriate cost function, and improving moves are always accepted. Simulated annealing overcomes the problem of solutions becoming trapped in a local optimum by accepting inferior moves occasionally, according to some probability. The probability decreases exponentially to zero as the number of moves increases. This allows a large area of the search space to be reached initially but constrains the search

towards the end of the algorithm causing it to become trapped in a local optimum, which is hoped to be the global optimum.

Probability is calculated using a parameter T — called 'temperature' because of the origins of the idea in thermodynamics. The algorithm executes a number of iterations, k, with this temperature and then reduces the temperature by a constant factor every k iterations. The algorithm terminates when T reaches some small predefined minimum value. The probability also depends on ΔC, the difference between the objective function for the disimproving move and the current move. Probability is calculated using the equation $p = e^{\frac{-\Delta C}{T}}$. The success of the algorithm is largely dependent on the cooling schedule which will vary from problem to problem.

The classical algorithm [KGV83] is given in Fig 9. For a detailed description of the method and its refinements the reader is referred to Dowsland [Dow95][Dow96]. An indication of some of the applications solved using simulated annealing is given in Chapter 3 section 3.1.7.

Selection of $\mathbf{T}_{start}$

The initial value of T_{start} is determined in such a way that virtually all new assignments are accepted i.e. T_{start} is such that $e^{\frac{-\Delta C}{T}} \approx 0.8$ for almost all ΔC.

This is implemented as follows: Select a moderate value for T_{start} and perform a number of moves. If the proportion of accepted moves is less than 0.8 then double the value of T_{start}. Continue this procedure until the observed acceptance rate is > 0.8 i.e. $\frac{no\_accepted}{no\_tried} > 0.8$

Figure 9. Simulated Annealing Pseudocode

initialise parameters, $T := T_{start}$

generate initial random assignment

evaluate initial assignment, C

While $T > MinTemp$ and $C > 0$

    $iteration := 0$

    While $iteration < MaxIterations$ and $C > 0$

        suggest a move

        evaluate the suggested move, C'

        calculate $\triangle C := C' - C$

        If $\triangle C > 0$ (inferior solutions)

            $p := e^{\frac{-\triangle C}{T}}$

            guess := random(0,1)

            If $guess < p$

                accept move

        Else accept move

        $CurrentCost := C$

        $iteration := iteration + 1$

    EndWhile

    $T = T * cool$

EndWhile

4.4.2 Tabu Search

Tabu search is an iterative heuristic search procedure which begins with a possible solution, s, and proceeds to search the solution space for optimal solutions, via a sequence of moves. It was first developed by Glover [Glo89] and has since been applied to a wide variety of problems with success (see Chapter 3 section 3.1.7). Two fundamental elements of tabu search are to constrain the search by classifying certain of its moves as tabu, and to free the search by a short term memory function that provides strategic forgetting. Tabu restrictions allow a move to be admissible if they do not apply, while aspiration criteria allow a move to be admissible if they do apply. Associated with each move is a move value which represents the change to the objective function value as a result of the proposed move. Since the algorithm aims to minimise the interference suffered, a positive move value indicates a disimproving move whilst a negative move value indicates an improvement over the current objective function value. At each iteration a set of possible moves defines the neighbourhood and the best possible move is selected subject to its tabu status.

In order to diversify the search and prevent cycling (returning to solutions recently visited) the algorithm makes a list of the prime attributes of recently and frequently made moves. *Recency* and *frequency* refer to the short-term and long-term history of moves. Any moves whose attributes appear on the list are said to be tabu and are therefore passed over in favour of other non-tabu moves. In some circumstances it may be beneficial to over-ride a tabu status. When a tabu move would result in a solution better than any visited so far, its tabu classification may be overridden. This is one example of an aspiration criterion.

The dual relationship between tabu restrictions and the aspiration criteria is a means for constraining and guiding the search process. The motivation underlying this form of integration of aspiration criteria and tabu restrictions is the hypothesis that different attributes of moves can have relatively different influences on the quality of solutions generated, and thus should be subject to different tenures of tabu status. Initially the search acts like a *steepest descent* algorithm (see section 4.3.2) because none, or very few, of the moves are tabu. In order that the search is not constrained by the tabu status of moves that were made many iterations ago, the algorithm incorporates strategic forgetting — after a certain time a move may have its status changed from tabu (T) to non-tabu (NT).

Tabu Status

Tabu restrictions prevent cycling and induce vigour into the search. In some instances, a good search path will result in revisiting a solution encountered before. The procedure is given a memory to prevent reversal of moves but allows the memory to decay so choices are not influenced by decisions that should be regarded as ancient history. The values given to the parameters R (*recent*, short-term) and Q (*frequent*, long-term) prevent cycling. This prevents the search becoming trapped in a local optimum and has proven to be an effective strategy.

For memory conservation and ease of processing it is often desirable to record less than the full range of attributes required to characterise a move, so partial range attributes are often stored instead.

Short-term memory : Recency

If notes were made on all of the solutions previously visited it would be extremely space consuming and inefficient. Instead a more practical and effective memory structure is employed, where a list of *moves* performed in the last R iterations is maintained. This list is used as a lookup table to see if a considered move is the reverse of a move on the list. If the reverse move is present in the list the candidate move is given the classification tabu.

Long-term memory : Frequency

The use of frequency is a simple diversification approach which penalises frequently occurring solution attributes. Frequency based memory provides a type of information that complements the information provided by recency based memory. Frequency of a move is usually expressed as a percentage of the iterations done so far.

Objective Function

A perfect assignment (as distinguished from an optimal assignment) occurs when all the constraints are satisfied. To this end we need some way of establishing the 'goodness' of a proposed solution. The objective function fulfils this task and provides a measure

of a solutions 'goodness' as a numerical value. Various objective functions have been suggested for the frequency assignment problem and these are given in Chapter 2 section 2.5.

A Move

A move is a transition from one solution to another. The aim is to move from the current solution to a solution that yields greatest improvement — or, lacking the possibility of improvement, the least disimprovement — in the objective function subject to the restriction that only non-tabu moves are allowed. In fact tabu moves are often also evaluated and their selection depends on the aspiration criteria.

Associated with each reassignment is a move value, which represents the change in the objective function value as a result of the proposed reassignment. Move values generally provide a fundamental basis for evaluating the quality of a move.

Neighbourhood

Tabu search methods operate under the assumption that a neighbourhood can be constructed, $N(s) \subseteq S$, to identify an *adjacent solution*, s', that can be reached from any current solution, s. An *adjacent solution* is one which can be reached by executing a single move from the current solution. Tabu search generates a candidate list $N(s)$ of neighbours of s and moves to the best solution $s' \in N(S)$.

Aspiration Criteria

The appropriate use of aspiration criteria can be very important for enabling a tabu search method to achieve its best performance levels. In some instances the conditions which make a move tabu are too restrictive in that they also forbid moves which may lead to previously unvisited solutions — more specifically, to attractive unvisited solutions. We therefore design aspiration criteria that define conditions under which a tabu move may be accepted. The role of aspiration criteria is to provide added flexibility to choose good moves whilst retaining the ability to avoid cycling.

A simple aspiration criterion might be: if the tabu move under consideration is better

than any solution found so far then accept it.

Termination Criteria

A number of possible termination criteria can be suggested: the reaching of some predefined maximum number of iterations; the solution reaches a predefined level of optimality deemed 'good enough'; or the number of iterations done since the solution value last changed is greater than a specified maximum number.

Tabu Search Algorithm

The classical algorithm is given in Fig 10 [CHS96]. For a detailed description of the method the reader is referred to Glover [Glo89]. Refinements to the original algorithm are described in Glover [Glo90]. Tabu thresholding and surrogate constraint tabu search have been applied to the frequency assignment problem with success, [CS95].

Figure 10. Tabu Search Pseudocode

step 1: Generate an initial solution s in S and set best solution so far $s_b := s$
step 2: Determine the neighbourhood of the solutions s, $N(s)$
step 3: Select s' from $N(s)$ such that s' is the best tabu move also satisfying the aspiration criterion or else the best nontabu move
step 4: Set $s := s'$ and if $Cost(s) < Cost(s_b)$ then $s_b := s$
step 5: Update memory structures
step 6: If termination criteria not satisfied then go to Step 2

4.4.3 Divide and Conquer

The divide and conquer strategy involves dividing a problem into smaller similar subproblems, solving these subproblems either individually or recursively and then combining the results to obtain a solution to the whole problem. The subproblems can be solved more quickly than the problem can be solved taken in its entirety.

Ideally the subproblems should be independent problems — that is, the solution of one has no effect on the solution of another part and they can be recombined without further computations to obtain a solution to the whole. Not all problems can be divided into independent subproblems; the frequency assignment problem is one such problem. In order to use the essence of the strategy a problem can be divided as best possible and an approximate solution obtained. A minimal amount of further computation is then required to obtain a valid solution after combining the solution parts. The pseudocode is given in Fig 11.

When using a divide and conquer strategy there are two main components to consider: how to divide the original problem and how to solve the subproblems. Neither of these decisions is trivial; both will affect the computation time and quality of the solution obtained.

Figure 11. Divide and Conquer Pseudocode

Initialise data structures
Generate frequencies (given number and range)
Sort links into descending co-site vertex degree order
Divide the problem into k bands according to division criteria
Generate Initial Solution
Solve the k bands individually
Recombine solutions from separate bands
If non-independent subproblems then apply improvement heuristic
Report results for whole problem

Dividing the Problem

Identifying ways in which a problem may be divided into smaller, independent subproblems requires knowledge of the problem and how various elements interact. Often there is more than one way to divide the problem and these alternatives should be considered carefully.

Conquering the Subproblems.

Solution of the subproblems individually ideally provides a final solution more quickly than trying to solve the problem in its entirety. Often the application of the same algorithm to the subproblems as the whole problem exhibits this characteristic due to the efficiency of the algorithm in solving simple instances of the problem. Another approach is to solve the subproblems using naive algorithms if the subproblems are of significantly reduced size.

These two goals are in conflict giving rise to a trade-off situation. If the problem is divided into many small subproblems then their individual solution may be more readily found. However, the overhead involved in the division and subsequent recombining of the subproblems may mean that the total time-saving is reduced.

Using the divide and conquer method in a recursive framework can be particularly effective if the problem lends itself well to this approach, recursively dividing the problem down to a trivial case. Alternatively, parallel computing can massively reduce computation time if the problem can be divided into independent problems.

The implementation details for the application of tabu search, simulated annealing and divide and conquer to the FAP are given in Chapters 7 and 8.

Chapter 5

Factors Influencing Algorithm Performance

The algorithm chosen to solve a particular problem will affect the quality of the solutions obtained and is therefore the primary decision when attempting to solve a problem. However, having selected an algorithm there are often other factors able to influence the quality of the final solution. For example, if the algorithm evaluates all possible moves and several of those moves yield solutions of equal improvement which should be chosen? The selection strategy will influence the search and ultimately the quality of the final solution.

This chapter explores the influence of three factors on the performance of a given algorithm:

1. The order in which the data is presented.
2. The selection of a move from those yielding equivalent cost solutions.
3. The provision of an incremental objective function.

The MATRIX and EUCLID test data (described in Chapter 6) was used for this investigation. The problems are relatively small and were solved using greedy and backtracking algorithms (described in Chapter 4). The results are summarised and some

conclusions are drawn. These conclusions were used when developing the simulated annealing and tabu search algorithms described in Chapter 7 and the divide and conquer algorithm described in Chapter 8.

5.1 Ordering the Links

Intuitively, if the vertices of the graph (links) are ordered in some sensible way then it would not be unreasonable to anticipate better results than if no ordering was imposed. If the links are placed in descending order of difficulty then the most difficult links will be assigned frequencies first and this could avoid problems later in the algorithm when trying to find a suitable frequency for a difficult link. The ordering of the links has a significant impact on the quality of the solutions obtained by sequential assignment algorithms, particularly greedy algorithms where only one pass of the links is made. Firstly the concept of 'difficult' needs to be defined, various definitions have been used in the literature. For example the difficulty of a given link may be calculated by:

1. The number of other links with which it has a non-zero constraint (vertex degree)[ZB77] [Met70].

2. The sum of the frequency separation values (weights) on the incident edges [Hal80].

3. The number of other links with which it has a co-site constraint [PS98].

4. The number of distinct frequencies used by assigned adjacent links plus the sum of the weights on any edges whose links are as yet unassigned [Lan89].

The advantage of the first three definitions is that they do not change during the algorithm and so need only be computed once. Definition 4 is clearly dynamic and would need to be recomputed during the running of the algorithm. Other measures of 'difficulty' use arbitrary weightings. The above definitions describe the criteria by which the links are assigned a 'difficulty' but do not specify the ordering. Typically a list of links

is based on *largest first*[1] [ZB77] or *smallest last*[2] [Met70] ordering - both of these orderings have proven successful for networks with only co-channel constraints. Generalised vertex degree orderings were introduced by Hale [Hal80]; these are more appropriate to networks with co-channel and adjacent-channel constraints. In the *generalised largest-first* and *generalised smallest last* orderings the weighted degree (definition 2 above) of the vertex was used.

Another method has been suggested by Smith and Hurley [SH97] whereby a maximum clique[3] is assigned first. This method has been used with success for their data sets although they recognise that constraint graphs exist with no large cliques, or even triangles. The EUCLID and MATRIX data sets were small and solvable using exact techniques and so analysis of the graphs was unnecessary. The TNET data used for the majority of the work described in this thesis has no large cliques and so the maximum clique method was inappropriate in this case. In addition the algorithm used to search the graph for cliques takes a few minutes on a 133MHz PC. This run-time increases rapidly if a less than effective initial vertex ordering is used. The maximum run-time for the solution of the TNET data is 15 minutes and so this type of preprocessing was infeasible.

5.2 Selecting a Frequency

Assume that at each iteration the frequency assigned to a given link is changed. If all available frequencies are evaluated at each iteration and several frequencies yield equivalent best-cost solutions a criterion for selecting one of these frequencies is required. One possibility is to select the first frequency which yielded a best-cost solution, this would generate different frequencies depending on whether the frequencies were evaluated at random or in some predefined order. Another criterion is to select a frequency if it is the *smallest*, *largest* or *least frequently* assigned so far. Alternatively a frequency could be selected at *random* from those which yielded equivalently best-cost solutions.

[1] Largest-first order sorts the nodes of a graph according to their vertex degree with the node of largest degree at the beginning of the list.

[2] Smallest-last order is found by repeatedly deleting the node of minimum vertex degree from a graph and finally reversing the order so that the last node to be deleted is the first in the list.

[3] A clique is a maximal set of mutually interfering transmitters

When randomly selecting a frequency it is typical to maintain a list of the frequencies yielding the best-cost solutions and to generate an index at random which in turn indicates the selected frequency. This method is satisfactory although there may be a predetermined upper limit on the list size in order to reserve memory, otherwise dynamically allocated memory must be used.

A more efficient method (with respect to storage space and probably with respect to time) for selecting a frequency at random (from those yielding solutions of equivalent best-cost) is described in Chapter 7 section 7.3.4. This method does not rely on lists to preserve the best frequencies found. Rather than maintain a list of all the equivalently best-cost frequencies it is possible to just store one frequency whilst still maintaining the random selection strategy. This method is preferred for data sets with large numbers of frequencies or where many equivalent-cost solutions are anticipated.

5.3 Incremental Objective Function

If all of the constraints are inspected each time the objective function is evaluated then the complexity of computing the objective function is $O(C)$. Typically a move only makes a small change to the assignment during a single iteration; therefore only links involved in that move need re-evaluating, the evaluation of the rest of the assignment remains unchanged. If there are C constraints and N links, then a given link is involved in approximately $\frac{C}{N}$ constraints and so it should be possible to reduce the complexity of the objective function to $O(\frac{C}{N})$. Given that the objective function is evaluated after every move, of which there may be tens of thousands, it is clear that substantial time savings are possible by using an incremental objective function. Ideally the constraints should be presented in such a way that the constraints involving any given link are easily identifiable.

5.4 Implementation

During this early experimentation the 'quality' of a solution, given by the evaluation of the objective function, was determined by the number of violated constraints. All the results were obtained on a 486 Desktop PC, the code was written in C using Turbo C

version 2.01 software.

5.4.1 Algorithms

The EUCLID and MATRIX data sets were small and were solved using three algorithms: greedy algorithm, repeated greedy algorithm and a backtracking algorithm. The greedy and backtracking algorithms are given in Chapter 4. The repeated greedy algorithm simply repeated the greedy algorithm until no further improvement was obtained (described in chapter 4 section 4.3.3).

5.4.2 Ordering the Links.

For this investigation three presentations of the vertices were considered. In the first two the links (vertices) were placed in descending order of difficulty using definitions 1 and 2 as described in section 5.1. To recall:

- Def 1 The number of other links with which it has a non-zero constraint (vertex degree).

$$vertex\_degree_i = \sum_{j=0, j\neq i}^{j=N-1} \delta_{ij}, where \delta_{ij} = \begin{cases} 1 \text{ if } C_{ij} > 0 \\ 0 \text{ otherwise} \end{cases}$$

- Def 2 The sum of the frequency separation values (weights) on the incident edges

$$vertex\_sum_i = \sum_{j=0, j\neq i}^{j=N-1} C_{ij}$$

The third presentation was to leave the vertices as they were originally presented, in random order.

5.4.3 Selecting a Frequency

If several frequencies exist that yield equivalent best-cost solutions then a criterion for selecting one of these frequencies is required. Two criterion were used during this early investigation :

- Def 1. Selection of the *smallest* frequency from those yielding equivalent best-cost.

- Def 2. Selection of a frequency at *random* from those yielding equivalent best-cost.

Another selection criteria used in minimum span frequency assignment problems is to try and eliminate the largest frequency used. Therefore, if the largest currently-assigned frequency is in the set of frequencies yielding equivalent best-cost solutions then a different frequency would be chosen in preference. The frequencies were assessed in ascending order and so the first frequency to obtain the best-cost solution was also the smallest frequency. The data sets used for this investigation were small making it feasible for the equivalent best-cost frequencies to be held in a list and a randomly generated index used to select a frequency.

5.4.4 Incremental Objective Function

The way in which the frequency assignment constraints are represented determines the method used to exploit that representation to achieve the most efficient evaluation of the objective function. In many cases the constraints are simply provided in a list and the implementor is able to translate this into the most appropriate data structure (depending on problem size and any currently existing software).

Representing the Constraints

Co-site and far-site constraints describe a relationship between two links, and so a matrix is a convenient way to describe the constraints. The row and column indices represent the links and the matrix values represent the required frequency separation. This method is suitable for smaller problems where storage of the (relatively sparse) matrix does not present a difficulty with the memory available. For larger problems other data structures can be used which are less memory intensive; an indexed linear linked list representation is described in Chapter 7, section 7.1.4.

The EUCLID data set was very small (85 constraints), and also contained constraints between three links, and so the presented list was used during the evaluation of the objective function, the total run-time for the greedy algorithm was less than 2 seconds and so optimising the objective function was not pursued.

The MATRIX data set has constraints of the form $|f_i - f_j| \geq C_{ij}$ with C_{ij} integer and positive, for this data set the value 0 represents 'no constraint'. An example is given

in Fig 12. The matrix is upper diagonal with valid constraint values of 1, 2, 3, 4 and 9 which are defined as : $|f_i - f_j| \geq x$ where $x \in \{1, 2, 3, 4, 9\}$. Recall that f_i and f_j are the frequencies assigned to links i and j respectively. In the example given the value representing the constraint $|f_1 - f_2| \geq 3$ is emboldened. The metrics describing the MATRIX data set are given in Chapter 6.

Figure 12. Matrix Showing FAP Constraints

	0	1	2	3	4
0	0	1	2	9	0
1	0	0	**3**	4	1
2	0	0	0	3	2
3	0	0	0	0	1
4	0	0	0	0	0

Objective Function

The purpose of the objective function is to evaluate the current solution and provide a measure of its desirability (cost), usually expressed as a number. This information is used by the search technique in its pursuit of optimal solutions. Since the current solution (assignment) is being altered during each iteration of the algorithm then the cost of the assignment also needs to be re-evaluated at each iteration.

In the naive method the objective function considers every constraint in turn, decides whether the most recent iteration of the algorithm has affected that constraint, and re-evaluates its contribution to the total cost as necessary. This method has computational complexity of $O(C)$ where C is the number of constraints. By exploiting data structures used to represent the constraints it is often possible to develop an incremental objective function which updates the current cost depending on the most recent move. This is explained further in the next section.

For this investigation the objective function used was the number of constraints violated.

$$\sum_{i,j=0,j\neq i}^{i,j=N-1} X_{ij} \quad where X_{ij} = \begin{cases} 1 \text{ if } |f_i - f_j| < C_{ij} \\ 0 \text{ otherwise} \end{cases}$$

Matrix Objective Function

Using a naive evaluation of the solution cost using the matrix data structure would require inspection of each value of the matrix, this has complexity $O(N^2)$ where N is the number of links. Clearly this method can easily be improved by inspecting only the values in the upper diagonal $O(\frac{N^2}{2})$. These naive methods are satisfactory but they would become impracticable as N increases, for realistic problems N can have values over 1000 and so any improvement in the time taken to evaluate the objective function would be an advantage.

An incremental objective function updates the current cost according to the changes made (or proposed) during the current iteration. Assume that in the current iteration a single link, x, is assigned a new frequency. To calculate the effect of this change on the current solution cost only the constraints involving that link need to be considered. For the matrix representation all constraints involving a link x are found in row x (columns $x+1$ to $N-1$) and in column x (rows 0 to $x-1$). It is not necessary to check entry x, x since this is 0 ('no constraint') in each case. The complexity of the incremental objective function is $O(N)$. Partial assignments (incomplete solutions) can also be evaluated using this method — any unassigned links involved in a constraint are assumed to satisfy the constraint. The matrix entries considered in the incremental objective function for $x = 1$ are shown in Figure 13.

Figure 13. Incremental Matrix Evaluation

	0	1	2	3	4
0	0	**1**	2	9	0
1	0	0	**3**	**4**	**1**
2	0	0	0	3	2
3	0	0	0	0	1
4	0	0	0	0	0

The incremental objective function previously described can be optimised further for the greedy algorithm (not the repeated greedy algorithm) and the backtracking algorithm when no ordering is imposed on the links. For these two algorithms the links are assigned frequencies in ascending numerical order, therefore any links with higher index numbers (appearing later in the list), do not need to be checked since they are unassigned and therefore unable to contribute to constraint violations. In relation to the matrix this means that only constraints in column x (rows 0 to $x-1$) need to be re-evaluated. This is shown in Fig 14 for $x = 2$. The complexity of this incremental objective function is $O(\frac{N}{2})$.

Figure 14. Improved Incremental Matrix Evaluation

	0	1	2	3	4
0	0	1	**2**	9	0
1	0	0	**3**	4	1
2	0	0	0	3	2
3	0	0	0	0	1
4	0	0	0	0	0

The above discussion demonstrates the implications of using an efficient incremental objective function when solving realistic frequency assignment problems. The examples show how exploiting the matrix representation can reduce the complexity of the objective function from $O(N^2)$ to $O(N)$ (the final improvement was for a special case

and so is not included).

5.5 Summary of Results

The algorithms applied to the EUCLID and MATRIX data sets were: greedy, repeated greedy and backtracking algorithms. The links were ordered using the definitions in section 5.4.2. A frequency yielding equivalent best-cost solutions was chosen according to the criteria given in section 5.4.3. Every combination of ordering and frequency selection was investigated for each of the algorithms. The best solutions possible for each of the data sets was known (1 constraint violation for EUCLID and $MATRIX_{32}$, 0 constraint violations for $MATRIX_{38}$ and $MATRIX_{50}$).

For ease of reference the following notation has been used for the ordering criteria : definition 1 = *vertex-degree*, definition 2 = *edge-sum*, definition 3 = *no-order*. Similarly for selection of frequencies: definition 1 = *smallest*, definition 2 = *random*

- The greedy algorithm consistently obtained solutions with a relatively high cost — however these solutions were obtained very quickly (approximately 1 second) and provided lower cost solutions than a randomly generated solution especially when the links were considered in *vertex-degree* ordering.
- The backtracking algorithm consistently obtained the lowest-cost solutions, obtaining optimal solutions for two of the data sets within the time allocated.
- The two greedy algorithms generally obtained lower cost solutions using *smallest* frequency selection than *random* frequency selection. The reverse was true for the backtracking algorithm.
- The time taken to obtain solutions when *vertex-degree* or *edge-sum* ordering or *random* frequency selection were used was longer due to overhead involved. This time overhead seemed excessive for the greedy algorithm, considering the total run time.
- Solutions obtained with *random* frequency selection generally used a smaller range of the available frequencies.
- Solutions with the lowest cost were generally obtained using *vertex degree* ordering.

- For the EUCLID data set the span used in the final solution was less for links which were ordered using *vertex-degree*, for the other data sets there was little discernible difference.
- The speed advantages due to the use of an incremental objective function were only noticeable for the longer runs of the backtracking algorithm.
- The backtracking algorithm had the longest run-times of the three algorithms investigated (maximum of 40 minutes). Even for these small problems the time to evaluate all solutions in the search space would have been 10^{10}years for MATRIX_{38}.

5.6 Conclusions

All of the algorithms were naive. However they provided a means for judging the effect of the various ordering, selection and objective function factors that were investigated.

The greedy algorithm yielded relatively poor cost solutions compared to the backtracking algorithm. However, reasonably low cost solutions (using comparable spectral resources) were found quickly. The solutions had a lower cost than randomly generated solutions. The use of a greedy algorithm to obtain an initial solution has been used in the investigations carried out in Chapters 7 and 8.

The backtracking algorithm consistently obtained the lowest cost solutions, the number of solutions investigated (in a fixed time period) increased when the incremental objective function was used. This algorithm had the longest run-times, it was also the only algorithm to obtain optimal solutions. The lowest cost solutions were found with *random* frequency selection, they also used a smaller range of frequencies. Any algorithm that executes a high number of iterations, or acts on a realistic data set will benefit from the implementation of an incremental objective function. All of the investigations described in later parts of this thesis have used an appropriate incremental objective function. The selection of a *random* frequency generally enabled lower cost solutions to be found which used a smaller range of frequencies — this selection criteria has been used in all further work described in the thesis.

The time overhead incurred when using *vertex-degree* ordering, *edge-sum* ordering or *random* frequency selection seemed excessive. However, the extra time for ordering is only incurred once (at the beginning of the algorithm) and the selection of a frequency

at random from those yielding equivalent best-cost solutions has benefits in terms of cost and reduction of frequencies used. Whilst the time taken for the greedy algorithms was doubled it should be considered in perspective of the extremely short run-times. For algorithms running for a longer time period it is suggested that the additional time overhead will be small in comparison to the total run time and that using these ordering and selection measures will continue to improve solution quality.

When links were ordered using *vertex-degree* solutions with the lowest cost were generally found. In addition the EUCLID data set used a smaller range when *vertex-degree* ordering was used, for the other data sets the range was comparable. The objective function used in this investigation calculated the total number of constraint violations. The lower cost solutions were found when using the *vertex-degree* link ordering. A further investigation considered an alternative objective function which found the sum of the positive discrepancies (see Chapter 2 section 2.5). When this objective function was used the lower cost solutions were found when using the *edge-sum* link ordering. It would appear that the ordering criteria is directly linked to the definition of the objective function and so the two should be chosen to complement each other.

When using the *vertex-degree* ordering the connectivity of the graph is used to describe the difficulty of a vertex. However the frequency separation requirements of the adjacent edges are not included in the measure of difficulty (co-site constraints have larger frequency separation requirements than far-site and are thus harder to satisfy). When using the *edge-sum* ordering the difficulty of a link gives an indication of the number of co-site constraints with which that link is involved. With *edge-sum* ordering links with a high vertex degree and many far-site constraints will still be considered to be difficult. However the connectedness of the link is not reflected in the difficulty of a link.

When the objective function sums violations the *vertex-degree* ordering is most appropriate, when the cost is calculated using the sum of the positive discrepancies the *edge-sum* ordering is most appropriate. Ideally the difficulty of an edge should reflect both its connectedness and the values of the weights on adjacent edges.

Chapter 6

Test Data

Three different formats of test data have been used during this research: EUCLID, MATRIX and TNET. The EUCLID data set is a small example problem, the MATRIX data set describes 3 moderate-size problems, and finally, the TNET scenarios represent real-life problems. There are 46 TNET scenarios. The EUCLID and MATRIX data sets were used only during preliminary experimentation as described in Chapter 5. The TNET data were used for the heuristic algorithm investigation.

6.1 EUCLID

The EUCLID[1] test data set is a description of one simple scenario intended as an example only. It was provided by the EUCLID group in appendix 3 of the CALMA[2] project specification [EUC]. There are constraints between pairs of links and also between triples of links. There are four different types of constraints. If f_i, f_j and f_k are the frequencies assigned to links i, j and k respectively, the objective is to find an assignment satisfying all of the constraints, there are four types of constraint:

- C1 $|f_i - f_j| >= 6$
- C2 $|f_i - f_j| >= 3$

[1] EUropean Co-operation for the Long-term In Defence

[2] Combinatorial ALgorithms for Military Applications

- C3 $|f_i - f_j| = 10$
- C4 $|f_i + f_j - 2f_k| >= 5$

Table 3 gives an indication of the size and complexity of the data set. The values given are: the number of frequencies $|F|$, the number of links N, the number of non-zero constraints C, the number of constraints of each type, and S the size of the search space.

Table 3. Euclid Data Metrics

$\|F\|$	N	C	$\|C1\|$	$\|C2\|$	$\|C3\|$	$\|C4\|$	S
14	18	85	34	34	6	11	10^{20}

6.1.1 Input Format

The data are presented as a single text file containing the pairs (and triples) of links involved in each type of constraint.

6.1.2 Objective Function

The objective function gives the number of constraints that are violated.

6.2 MATRIX

The MATRIX data set was provided by Steve Hurley at Cardiff University. The generation of the realistic scenarios was made possible by discussions with several people at the Defence, Evaluation and Research Agency at Malvern and the Army EMC [3] Agency. The data sets are based on combat net scenarios. There are three computer generated scenarios having 32, 38 and 50 links respectively.

The constraints are between pairs of links and take the form $|f_i - f_j| >= C_{ij}$ where f_i and f_j are the frequencies assigned to links i and j respectively and C_{ij} is the

[3] ElectroMagnetic Compatibility

required frequency separation. The objective is to find an assignment satisfying all of the constraints. For the 32 link data set the optimal solution has one constraint violation, while for the other two sets the optimal solution has no constraint violations.

There are five different types of constraint:

- C1 $|f_i - f_j| >= 1$
- C2 $|f_i - f_j| >= 2$
- C3 $|f_i - f_j| >= 3$
- C4 $|f_i - f_j| >= 4$
- C5 $|f_i - f_j| >= 9$

Table 4 gives the metrics for the data sets. The values given are: the number of frequencies $|F|$, the number of links N, the number of non-zero constraints C, the number of constraints of each type, and S the size of the search space.

Table 4. MATRIX Data Metrics

$\|F\|$	N	C	$\|C1\|$	$\|C2\|$	$\|C3\|$	$\|C4\|$	$\|C5\|$	S
25	32	199	60	58	48	33	0	10^{44}
30	38	248	73	62	18	33	62	10^{56}
40	50	465	143	136	141	45	0	10^{80}

6.2.1 Input Format

The data are provided in a single data*.txt file containing the constraint matrix (see Chapter 5 section 5.4.4). The frequencies generated are numbered consecutively from 1 to some user-defined percentage of the number of links.

6.2.2 Objective Function

The objective function gives the number of constraints that are violated.

6.3 TNET

The TNET data was supplied by Mr. Roger Edwards at the Defence, Evaluation and Research Agency in Malvern. The data sets represent 46 different scenarios, which are constraint-based descriptions of real life scenarios used in tactical communications. The networks were generated randomly. However, they exhibit features atypical of real networks and are sufficiently close to reality to demonstrate the validity of the algorithms. Tables 5, 6 and 7 give an indication of their size and complexity. Tables 5 and 6 describe scenarios for which there are known zero interference solutions, henceforth referred to as Group A data. The starred entries in tables 5 and 6 show scenarios for which D.E.R.A. had obtained zero cost solutions prior to supplying the data, the other entries have subsequently been shown to have zero-cost solutions by the algorithms implemented in the research described in this thesis. Table 7 describes scenarios for which there are *no* known zero interference solutions, henceforth referred to as Group B data. The metrics for each scenario are: the number of frequencies $|F|$, the number of links N, the number of non-zero constraints C, the percentage of co-site constraints, the span of the frequency set, the largest vertex degree, the largest co-site and far-site channel separation values ($L.co-site$ and $L.far-site$) and S the size of the search space.

Table 5. Group A Data

Scen ario	|F|	N	C	%co-site	span(F) MHz	Largest V.degree	L.co-site C_{ij}	L.far-site C_{ij}	Search Space
1	40	158	3099	5.3	169	123	27.1	9.1	10^{253}
2*	40	60	244	16.4	169	25	27.1	6.1	10^{96}
3*	80	98	193	35.8	209	18	40.1	8.1	10^{187}
4	40	124	215	63.3	399	8	195.9	9.5	10^{199}
5	40	158	4058	3.3	169	93	27.1	9.1	10^{253}
6*	40	12	20	0.3	169	6	27.1	0.6	10^{19}
7	80	164	846	17.0	209	24	40.1	8.1	10^{312}
8	40	92	1298	5.70	169	59	27.1	9.1	10^{147}
9*	40	16	30	26.67	169	5	27.1	0.6	10^{26}
10	80	224	1363	15.55	209	29	40.1	8.1	10^{426}
11	40	312	1105	43.80	399	18	195.9	9.5	10^{500}
12	40	170	3565	4.57	169	88	27.1	9.1	10^{272}
13*	40	4	4	50.00	169	2	27.1	6.1	10^{6}
14	80	312	2129	14.19	209	41	40.1	8.1	10^{594}
15	40	174	3177	4.82	169	85	27.1	9.1	10^{279}
16*	40	4	2	100.0	169	1	27.1	0.0	10^{6}
17	80	364	2655	14.16	209	34	40.1	8.1	10^{693}
18	40	470	1666	43.64	399	15	195.9	9.5	10^{753}
19	40	164	3956	3.6	169	94	27.1	0.1	10^{263}
20*	40	12	27	22.2	169	7	27.1	0.3	10^{19}

Table 6. Group A Data contd.

Scen ario	|F|	N	C	%co-site	span(F) MHz	Largest V. Degree	L.co-site C_{ij}	L.far-site C_{ij}	Search Space
21	80	344	2570	13.1	209	43	40.1	0.1	10^{655}
22	40	484	2016	35.5	399	23	195.9	0.1	10^{775}
23	40	164	257	55.3	399	7	195.9	1.1	10^{263}
24	40	154	3495	3.5	169	103	27.1	9.1	10^{247}
25*	40	12	16	37.5	169	6	27.1	6.1	10^{19}
26	80	358	2230	29.6	209	38	40.1	2.3	10^{681}
27	40	500	2392	88.5	399	22	45.1	3.5	10^{801}
28*	40	328	670	0.0	399	14	—	5.5	10^{525}
29	40	1040	8554	93.7	399	45	28.7	3.5	10^{1666}
30*	40	16	50	16.0	169	10	27.1	6.1	10^{26}
31	80	346	3423	24.3	209	58	40.1	2.3	10^{658}
32	40	490	2834	87.3	399	27	45.1	3.5	10^{785}
33*	40	16	19	52.6	169	4	27.1	0.6	10^{26}
34	80	356	1880	29.0	209	32	40.1	2.3	10^{678}
35	40	472	1358	89.7	399	14	45.1	3.5	10^{756}
36*	40	422	1191	0.0	399	31	—	5.5	10^{676}
37*	40	448	1782	0.0	399	33	—	5.5	10^{717}

Table 7. Group B Data

Scenario	|F|	N	C	%co-site	span(F) MHz	Largest V.Degree	L.co-site C_{ij}	L.far-site C_{ij}	Search Space
1	40	240	779	36.5	399	16	195.9	9.5	10^{384}
2	40	410	1578	39.6	399	24	195.9	9.5	10^{657}
3	80	740	11964	12.8	209	82	40.1	0.1	10^{1408}
4	40	1100	8563	91.0	399	39	28.7	0.1	10^{1762}
5	80	512	9916	8.7	209	97	40.1	8.1	10^{974}
6	40	170	5584	2.6	169	136	27.1	9.1	10^{272}
7	40	1090	9325	93.3	399	47	28.7	3.5	10^{1746}
8	80	504	15025	5.7	209	163	40.1	8.1	10^{959}
9	80	436	11552	6.7	209	130	40.1	8.1	10^{830}

6.3.1 Input Format

The test data are provided in 2 separate files

- *.frq gives the range of the frequency band and the number of discrete frequencies to generate.
- *.ctr describes the interference network; pairs of links (i, j) which have the potential to interfere are given along with the required frequency separation C_{ij} which would guarantee zero interference.

6.3.2 Objective Function

The objective is to find an assignment which minimises the interference suffered by the whole network. The interference is measured by the sum of the positive discrepancies:

$$cost(A) = \sum_{i,j=1}^{i,j=N} \max(0, C_{ij} - |f_i - f_j|)$$

6.3.3 Choice of Objective Function

For the EUCLID and MATRIX data sets the sum of the constraint violations was an appropriate objective function. All of the constraints had relatively small C_{ij} values compared to the TNET data and so any constraint violated would have had a similar effect on the interference suffered. However, for the realistic TNET scenarios the sum of constraint violations would not accurately represent the interference suffered. This is due to the significant difference between the largest and smallest frequency separation requirements and the larger frequency span available. An objective function using the sum of constraint violations would not indicate whether it was a far-site (small C_{ij}) or co-site (large C_{ij}) constraint which had been violated; these each have a different impact on the interference suffered. The sum of the positive discrepancies therefore gives a more accurate measure of the interference suffered when solving the TNET scenarios.

6.3.4 Availability

The TNET scenarios provided by D.E.R.A. are unclassified. Persons interested in obtaining these data sets are asked to contact Mr. Roger Edwards directly as distribution needs to be documented. The address is: Mr. Roger Edwards, Room PC309, St. Andrews Road, Malvern, Worcs, WR14 3PS.

6.4 CELAR

The CELAR data [CEL] was provided for the CALMA project which ran for 18 months and investigated the effectiveness of several metaheuristic techniques: tabu search, genetic algorithms, simulated annealing and local search. The results of the CALMA project [CAL95] would conceivably provide good benchmark results with which to compare the divide and conquer approach described here.

The CELAR data were not used to obtain results for the divide and conquer approach because there were significant differences in the *nature* of the two data sets some of which would require an essentially new implementation. These fundamental differences are outlined below.

1. In the TNET data there is one *frequency set*, in the CELAR data there are several overlapping *frequency domains* : division of the search space by dividing the frequency set would be complex for the CELAR data as different links may belong to different, overlapping frequency domains.

2. The TNET objective function measures the sum of the positive discrepancies whereas the CELAR objective function calculates a cost depending on a weighted function.

3. There are two forms of constraint for the CELAR data $|f_i - f_j| > C_{ij}$ or $|f_i - f_j| = C_{ij}$ whereas all the TNET constraints are of the form $|f_i - f_j| \geq C_{ij}$. After lengthy discussions with D.E.R.A. it was decided that there was no reason to have a fixed frequency separation for some constraints under the circumstances that were used to generate the TNET scenarios; this further implied that the scenarios were not of a similar nature.

4. The TNET scenarios had no variables pre-assigned whereas the CELAR data required some variables to be pre-assigned and gave varying mobility penalties should they be assigned different frequencies.

5. All of the TNET constraints were mandatory ('hard') whereas the CELAR data had some 'soft' and some 'hard' constraints.

One of the findings of the CALMA project was that preprocessing of the data was vital to the success of all the heuristics investigated [THL95]. By using arc consistency, and exploiting the equality constraints, it was possible to reduce the size of the scenario considerably. Similar investigative preprocessing measures were applied to the TNET scenarios and it was found that the scenarios were not reducible in the same way.

It would be possible to alter the existing divide and conquer program to accept the CELAR data. However, the difference in the type of frequency sets available means that a new investigation would be required to decide how to divide the problem into subproblems as the current method could not be employed.

Chapter 7

Benchmarks

The TNET test data were used for this investigation; these are non-trivial problems which require solution by heuristic, rather than exact, algorithms. A general description of the two heuristic algorithms, simulated annealing and tabu search, was given in Chapter 4. This chapter describes how the two algorithms have been implemented for the frequency assignment problem. In Chapter 8, a divide and conquer algorithm is described which incorporates these heuristics; results are also given. In order to compare fairly the divide and conquer implementation with the implementation of a single metaheuristic, program modules have been developed which enable the same code to be used in either implementation.

All of the code was written in ANSI C using Borland C++ version 4.5 software. The results were obtained on a Pentium desktop computer (100MHz).

This chapter is divided into six parts: part one describes the common decisions regarding problem representation; part two describes the implementation of the simulated annealing method; part three describes the tabu search implementation; part four gives computational results; part five discusses the results; and finally part six gives some concluding remarks for the chapter.

7.1 Common Decisions

When implementing the heuristics some decisions about data structures, the definition of a move and the choice of objective function were common to both implementations. These are described here. Once the algorithms had been implemented the various parameters were adjusted until a combination of parameters yielding good results was found. These parameters then remained constant when applied to new scenarios.

7.1.1 Representing an Assignment

A frequency assignment, $A = (f_1, f_2, ...f_N)$, is represented using an array of frequencies. The indices $1..N$ refer to the link to which the frequency is assigned. The initial assignment was obtained using stages 1-3 described in Chapter 8 so that the results from the heuristics on their own and within the divide and conquer algorithm could be compared.

7.1.2 Representing the Frequency Set

A given number of distinct frequencies were randomly generated within a given range and placed in an array. The frequencies were then sorted into ascending order; the minimum separation of any two adjacent frequencies was 0.5MHz.

7.1.3 Definition of a Move and a Neighbourhood

For this implementation a move was defined as the assignment of a new frequency to a chosen link. If an assignment A is $(f_1, f_2, ...f_N)$ then a neighbour of A can be described as $A' = (f'_1, f'_2, ...f'_N)$, where for precisely one i, $f'_i \neq f_i$. A neighbourhood of A is the set of all possible A' and has size $(|F| - 1)N$.

7.1.4 Representing the Constraints

For the real-life scenarios the total number of links N was much greater than the average vertex degree, so a matrix representation (described in Chapter 5 section 5.4.4) would have been very sparse. Instead an indexed linear linked list representation was used.

The index was an array 1..N of pointers to nodes, initialised to NULL, each element of the array represented one of the links, numbered consecutively and accessed using the array subscript. Each node contained the *other_link* with which the given link has a constraint, the required frequency separation C_{ij} and a *pointer* to the next node in the list.

Method One

One way to represent the constraints using an indexed linear linked list is to create *one* node per constraint, the lower number link is used for indexing. In this definition the final array index is unused since a link cannot have a constraint with itself. All the nodes for a given index (link) are sorted into ascending order of *other link.*

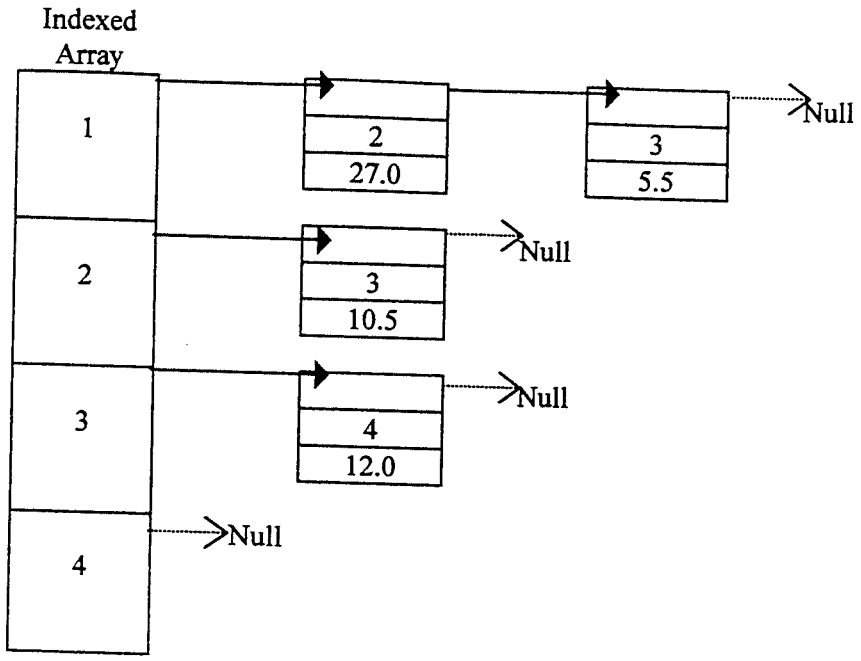
Example : For the constraints given in table 8 the data structure shown in Fig 15 is created.

Table 8. Constraints for data structure example

Link i	Link j	C_{ij}
1	2	27.0
2	3	10.5
3	1	5.5
3	4	12.0

To evaluate the contribution to the solution cost of a given move, (link, frequency) pair, it is necessary to examine all constraints involving that link. Using this data structure all of the indexed lists from 1..*link* need to be examined. The nodes in a list are in ascending order of *other link* and so it might not be necessary to traverse the entire linked list. The complexity of this objective function is $O(C)$ where C is the total number of constraints.

Figure 15. Indexed Linear Linked List Data Structure - method 1



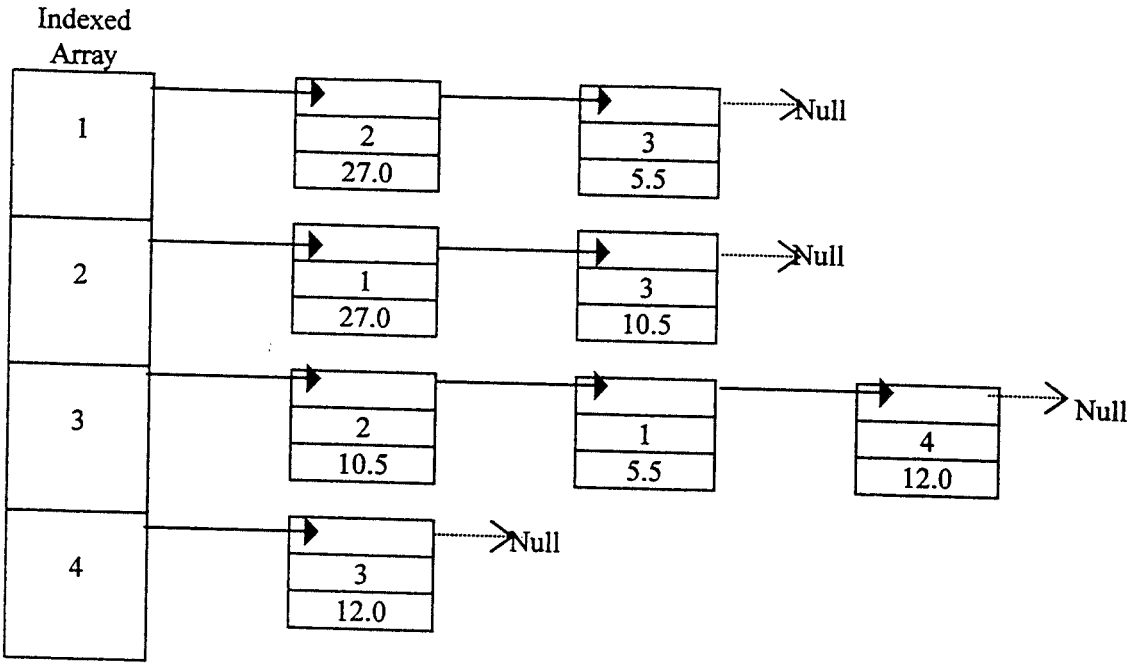
Method Two

An alternative way to represent the constraints using an indexed linear linked list is to create *two* nodes per constraint — one for each link. This doubles the amount of memory required to create the data structure, but enables a faster incremental objective function to be used. The nodes in a list do not need to be sorted.

To evaluate the contribution to the solution cost of a given (link, frequency) pair the linear linked list accessed via the index (subscript *link*) is traversed. All constraints involving this link are present in this list and so no further lists need to be examined. The complexity of this objective function is $O(\frac{C}{N})$.

Example : For the constraints given in table 8 the data structure shown in Fig 16 is created.

Figure 16. Indexed Linear Linked List Data Structure - method 2



7.1.5 Ordering the Links

The conclusions of preliminary experimentation described in chapter 5 suggested that the ordering of links and the selection of an objective function should be chosen to complement one another. For all algorithms presented in the remainder of this thesis the links were pre-ordered in descending *co-site vertex degree* order.

$$vertex\_degree_i = \sum_{j=0, j\neq i}^{j=N-1} \delta_{ij}, \text{ where } \delta_{ij} = \begin{cases} 1 \text{ if } C_{ij} > 10 \\ 0 \text{ otherwise} \end{cases}$$

7.1.6 Objective Function

The objective function used for this implementation was the sum of the positive discrepancies described in Chapter 2 section 2.5. Recall that f_i and f_j are the frequencies assigned to links i and j respectively and that C_{ij} is the required frequency separation to guarantee no interference. Then the cost of an assignment A is given by

$$Cost(A) = \sum_{i,j=1}^{i,j=N} \max(0, C_{ij} - |f_i - f_j|)$$

The objective function must be evaluated every time an adjustment to the assignment is made, i.e. every iteration. Since there might be hundreds of thousands of iterations it is essential that the evaluation of the objective function is as efficient as possible.

The incremental objective function does not need to examine each link pair, (which has complexity $O(N^2)$), in order to evaluate the change in the objective function as a result of the proposed move — it is sufficient to examine only constraints containing the link in the proposed move. The linked list data structure described as method two in the previous section was used for this implementation and has average time complexity $O(\frac{C}{N})$.

7.2 Simulated Annealing

A general introduction to simulated annealing was given in Chapter 4. The aim of this section was to explore the effectiveness of the simulated annealing heuristic when applied to the frequency assignment problem.

7.2.1 Implementation

The pseudocode algorithm for the frequency assignment problem implementation of simulated annealing is given in Fig 17.

7.2.2 Parameter settings

The initial cooling schedule parameters were taken from an example in the literature [Dow96]; these were then altered to investigate their effect on the quality of the final solution. The maximum number of iterations was also adjusted until a combination of parameters yielding good quality assignments was found. Once suitable parameters had been found they were fixed whilst obtaining results for new scenarios.

T_{start} was found by experimentation as described in chapter 4 section 4.4.1. The parameters used throughout for the simulated annealing algorithm were: $T_{start} = 1.25$, MinTemp = 0.11, MaxIters = 25N, Cool = 0.99, LocalOpt = 5000.

Figure 17. Simulated Annealing for the FAP

initialise parameters, $T = T_{start}$

generate initial assignment

evaluate initial assignment, C

$C_{best} := C$

While $T > MinTemp$ and $C > 0$ and LocalMin = FALSE and NOT(OutOfTime)

$iteration := 0$

While $iteration < MaxIters$ and $C > 0$ and LocalMin = FALSE and NOT(OutOfTime)

suggest a move

evaluate the suggested move, C'

calculate $\triangle C := C' - C$

If $\triangle C > 0$ (inferior solutions)

$p := e^{\frac{-\triangle C}{T}}$

guess := random(0,1)

If $guess < p$ Then AcceptMove = TRUE

Else AcceptMove = TRUE

If AcceptMove = TRUE

update assignment

$C := C'$

If $C < C_{best}$ Then $C_{best} := C$

Else If $C = C_{best}$

update NoEquivMoves

If NoEquivMoves > LocalOpt Then LocalMin = TRUE

EndIf

EndIf

$CurrentCost := C$

$iteration := iteration + 1$

EndWhile

$T = T * cool$

EndWhile

7.3 Tabu Search

7.3.1 Implementation

The pseudocode algorithm for the frequency assignment problem implementation of tabu search is given in Fig 18.

7.3.2 Tabu Lists

Additional data structures were required to maintain the memory features of the tabu search algorithm. These features were described in Chapter 4 and consisted of : *recency* (short-term)and *frequency* (long-term). To reduce the amount of memory required and for ease of processing it is often desirable to record less than the full range of attributes required to characterise a move; partial range attributes are often stored instead. In this implementation the short and long-term memory data structures (*recency* and *frequency*) stored only the link involved in a move.

The *recency* and *frequency* tabu lists were represented by two arrays, initialised to zero, each having N elements. The *recency* array contained the iteration number during which the given link (index of array) was last involved in a move. The *frequency* array values were incremented by one every time a given link (index of array) was involved in a move. Both of the data structures were updated after every iteration.

Figure 18. Tabu Search for the FAP

Initialise parameters and tabu data structures

Generate an initial solution s in S and set best solution so far $s_b := s$

While Iteration < MaxIters and $C > 0$ and NOT(OutOfTime)

and LocalOpt = FALSE and NOT (AllTabu)

For all possible neighbourhood moves (Candidate list has size N(F-1)) Do

evaluate move and determine its tabu status

If move = TABU

If move better than current BestTabu

Then update BestTabu

Else if move = current BestTabu

accept move as new BestTabu according to probability PT

Else

If move better than current BestNontabu

Then update BestNontabu

Else if move = current BestNontabu

accept move as new BestNontabu according to probability PNT

EndIf

EndFor

IF BestTabu < BestNontabu and aspiration(BestTabu)

Then ChosenMove = BestTabu

Else ChosenMove = BestNontabu

If all moves are tabu

Then AllTabu = TRUE

Else

update assignment according to chosen move

update tabu memory structures

update current value

update BestSoFar if necessary

EndIf

EndWhile

PT = $probability = \frac{1}{NoEquivTabuMoves}$

PNT = $probability = \frac{1}{NoEquivNontabuMoves}$

The tabu lists were used to determine the tabu status of a given move. A move was tabu if one or more of the following criteria were true.

1. The link had been involved in a move in the last R moves.
   $recency[link] > iteration - R$

2. The link had been involved in $Q\%$ of all moves done so far (Q is used in preference to F to avoid confusion with the frequency set F.)
   $\frac{frequency[link] * 100}{iteration} > Q$

7.3.3 Parameter settings

The following relationship exists between the number of links (N) and the parameters R (recency) and Q (frequency). This relationship must hold to avoid the algorithm terminating after N non-tabu moves. A more detailed explanation of this relationship is given in [Cas97].

$$\frac{1}{N} \leq Q \leq \frac{1}{R}$$

The values of R and $MaxIters$ which yielded high quality assignments, were found by experimentation. The value of Q is calculated using

$$Q = \frac{1}{2}\left(\frac{1}{R} - \frac{1}{N}\right)$$

After this initial experimentation the parameters remained constant whilst obtaining results for new scenarios. The values of R and $MaxIters$ were 4 and 8000 respectively.

7.3.4 Selection of a Move

A move was selected according to the following rule: If a move is tabu and it is better than the best non-tabu move and it satisfies the aspiration criterion then accept it, else execute the best non-tabu move.

At each iteration all the possible moves are evaluated and assigned a tabu status. The best tabu and non-tabu moves are noted.

Efficient Selection.

When all the moves are evaluated, there might be more than one move that yields the same best-cost value. In this implementation one of the equivalent best-cost moves is selected at random. Rather than maintain a list of all the equivalently best-cost moves of either status it is possible to just store one move of each status whilst still maintaining the random selection strategy. To do this each move is considered against several acceptance criteria as it is generated.

The following abbreviations have been used: Cost(A) provides a measure of the quality of the solution that would result if this move were chosen. T = Tabu, NT = Non-Tabu $Move_{TorNT}$ = The suggested move (tabu or non-tabu) $BestCost_{TorNT}$ = The value of the best cost move evalutated. $NoEquivMoves_{TorNT}$ = The number of equivalent best cost solutions that have been generated. $random(a, b)$ is the name of the random number generator which returns a value between a and b.

- if (move is the first tabu move to be generated) OR $(cost(A) < BestCost_T)$
  then $Move_T$:= move, $BestCost_T := cost(A)$ and $NoEquivMoves_T := 1$
- elseif (move is the first non-tabu move to be generated) OR $(cost(A) < BestCost_{NT})$
  then $Move_{NT}$:= move, $BestCost_{NT} := cost(A)$ and $NoEquivMoves_{NT} := 1$
- elseif (move is tabu) AND $(cost(A) = BestCost_T)$
  then $NoEquivMoves_T := NoEquivMoves_T + 1$,
  if $(random(0,1) < \frac{1}{NoEquivMoves_T})$
  then $Move_T$:= move
- elseif (move is non-tabu) AND $(cost(A) = BestCost_{NT})$
  then $NoEquivMoves_{NT} := NoEquivMoves_{NT} + 1$
  if $(random(0,1) < \frac{1}{NoEquivMoves_{NT}})$
  then $Move_{NT}$:= move

7.3.5 Aspiration Criterion

An aspiration criterion defines the conditions under which a tabu move may be accepted. For this implementation the aspiration criterion stated that if the move under consideration was better than any assignment found so far then it should be accepted.

7.3.6 Termination Criteria

The algorithm stopped when any of the following termination criteria were true.

- A zero interference assignment had been found.
- The maximum number of iterations had been executed.
- A given number of consecutive iterations had resulted in an equivalent cost solution (suggesting that the algorithm was trapped in a local optimum).
- All moves were tabu.
- The algorithm had been running for a predefined maximum time.

7.4 Results

For the TNET scenarios provided by D.E.R.A. it is not known whether zero interference assignments are possible for all the data sets. The results provided in this thesis show that zero interference assignments have been found for 80% of the scenarios (Chapter 8 results). For ease of comparison between the benchmark results (provided here) and the divide and conquer results (in Chapter 8) the graphs are divided into scenarios with/without known optimal solutions.

The two classic implementations described above were tested using the 46 TNET scenarios. Both algorithms were given an upper time limit of approximately 780 seconds (13 minutes) but each had several termination criteria which meant that the full time allocation was not always required. Each scenario was solved using ten different seeds for the random number generator. For each scenario the frequency set and initial solution were constant over all runs. Since the standard deviation was small in most cases bar charts have been chosen to graph the results.

Graphs are given to show the average cost of solutions obtained for the simulated annealing and tabu search algorithms (Figs 19 and 24). Graphs showing the average time taken are also given (Figs 20 and 25).

Graphs comparing the cost of the best solution found by each of the algorithms are also given (Figs 21 and 26) along with the time taken to obtain that best solution (Figs 22 and 27).

Finally, graphs showing the average percentage of frequencies used in the final solution is given (Figs 23 and 28).

Figure 19. Graph showing SA vs. TS Average Cost (Group A)

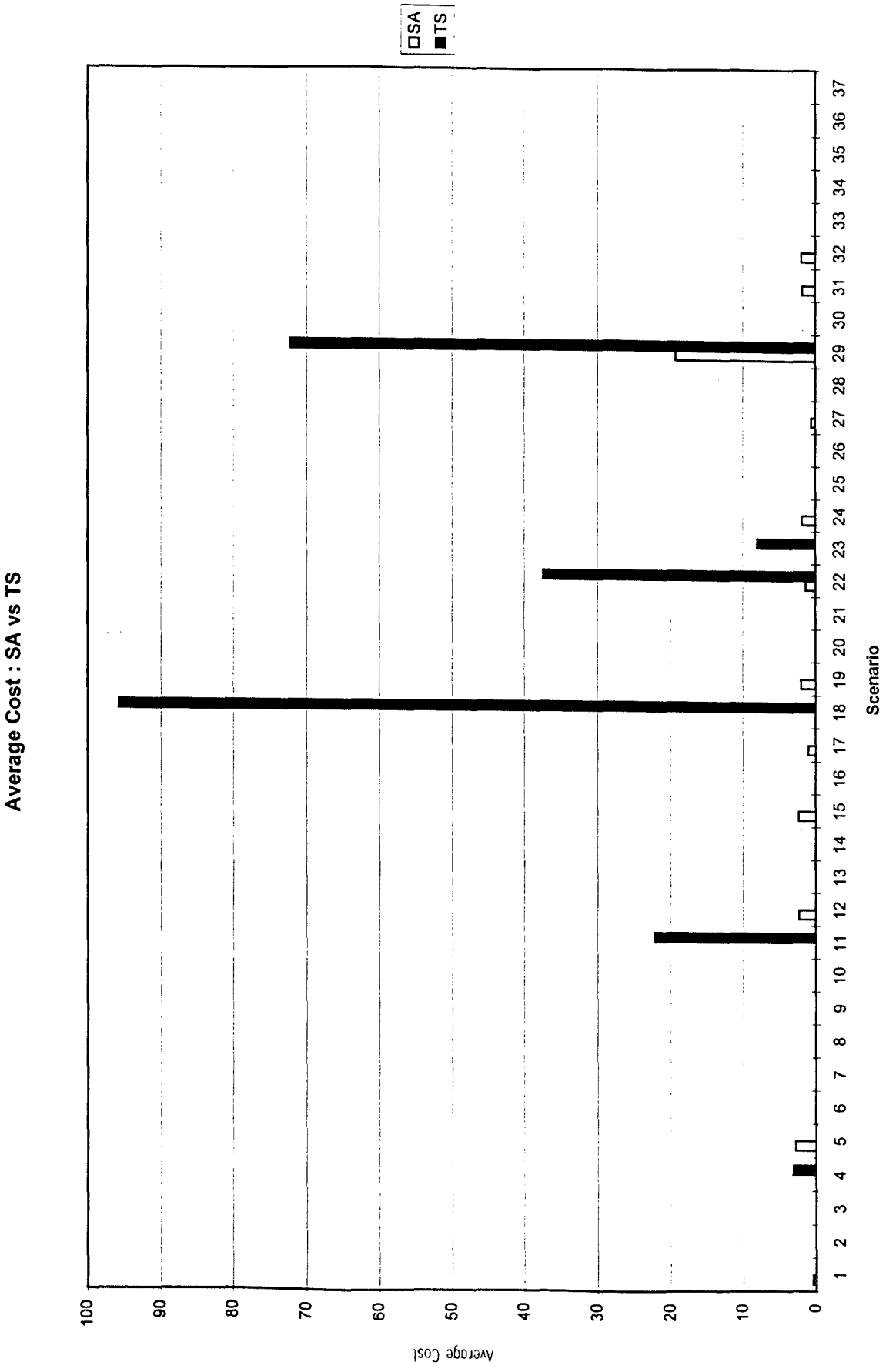


Figure 20. Graph showing SA vs. TS Average Time (Group A)

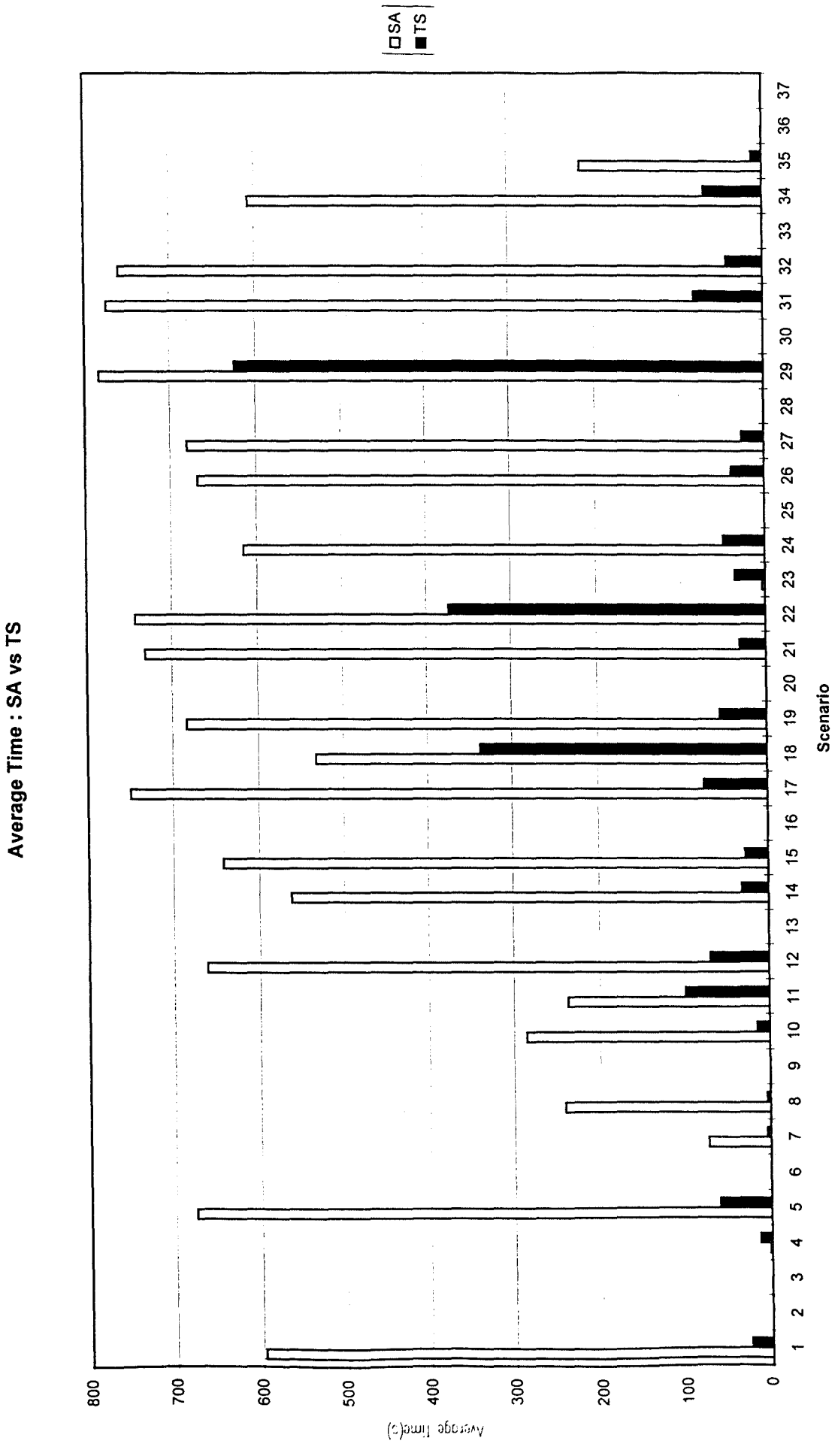


Figure 21. Graph showing SA vs. TS Best Cost (Group A)

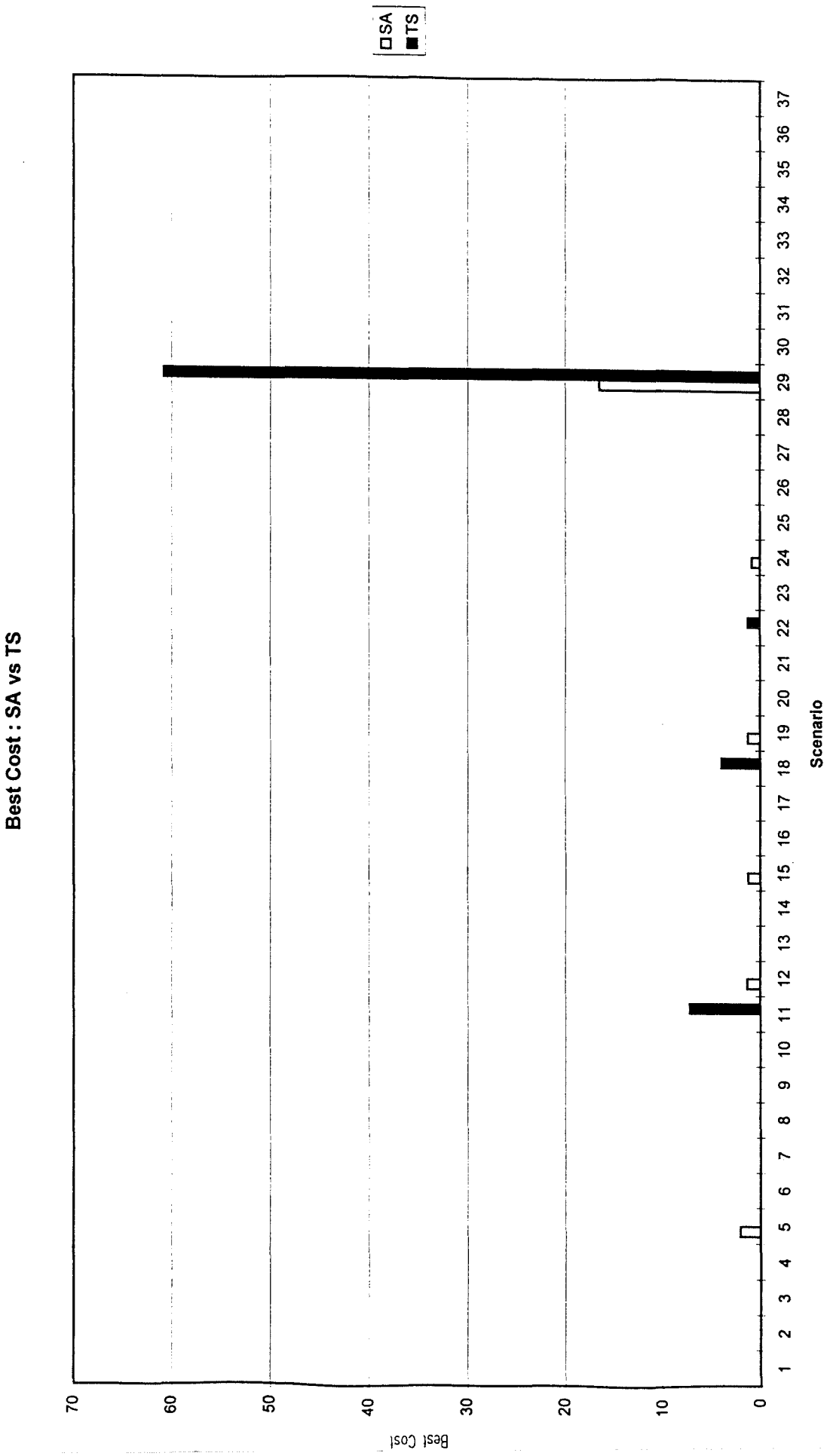


Figure 22. Graph showing SA vs. TS Time to Best Cost (Group A)

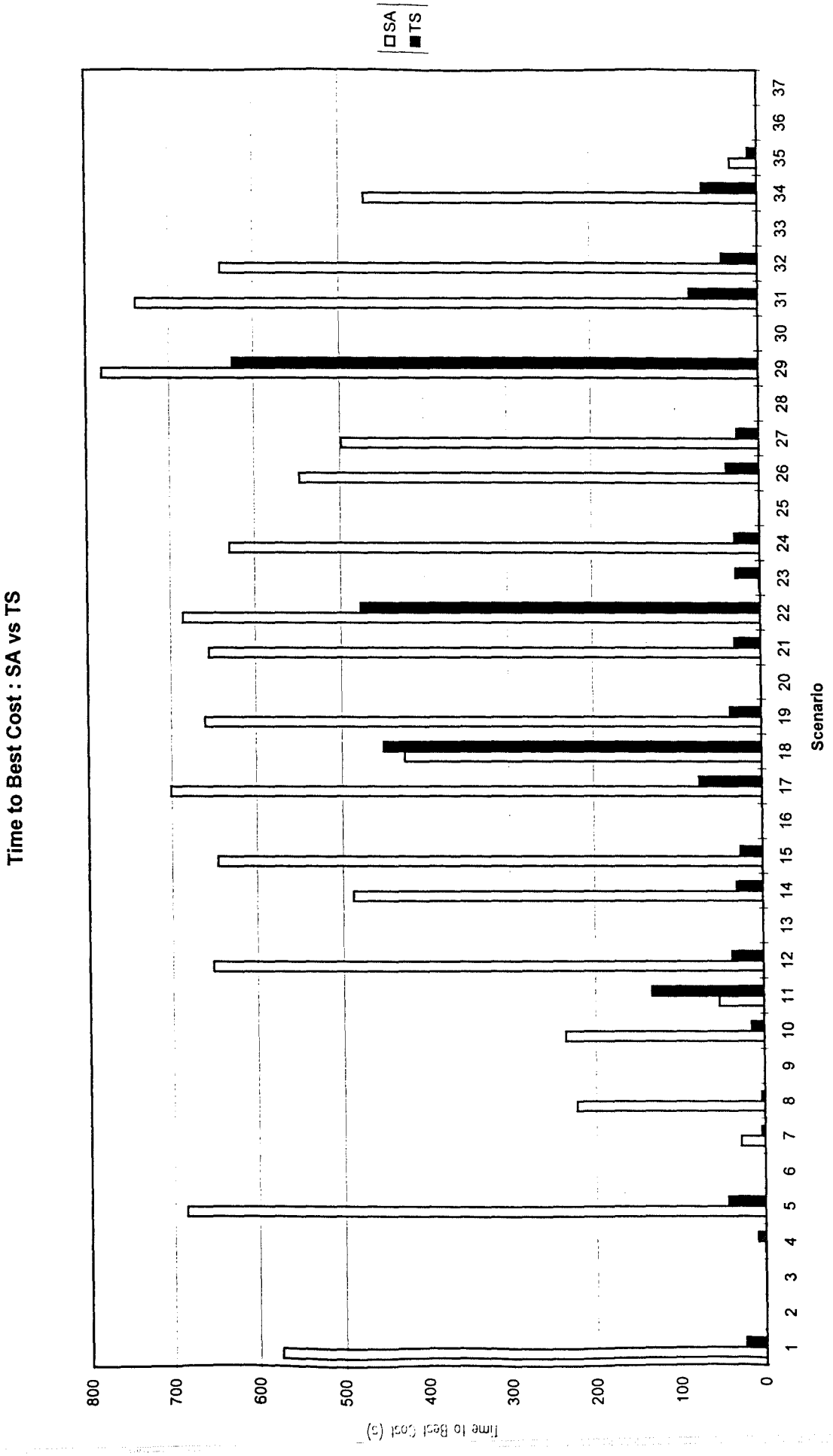


Figure 23. Graph showing SA vs. TS Average % Frequencies Used (Group A)

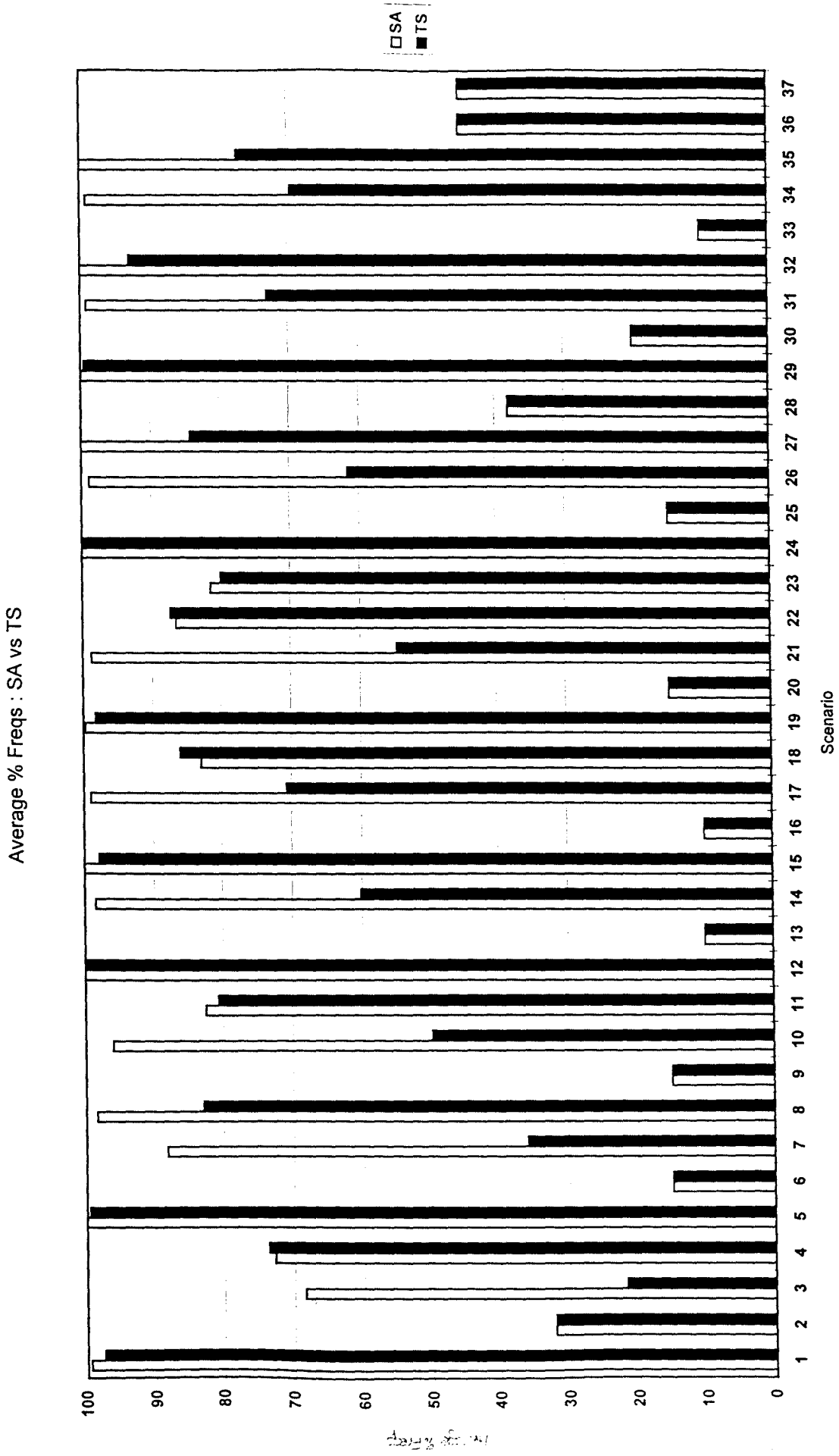


Figure 24. Graph showing SA vs. TS Average Cost (Group B)

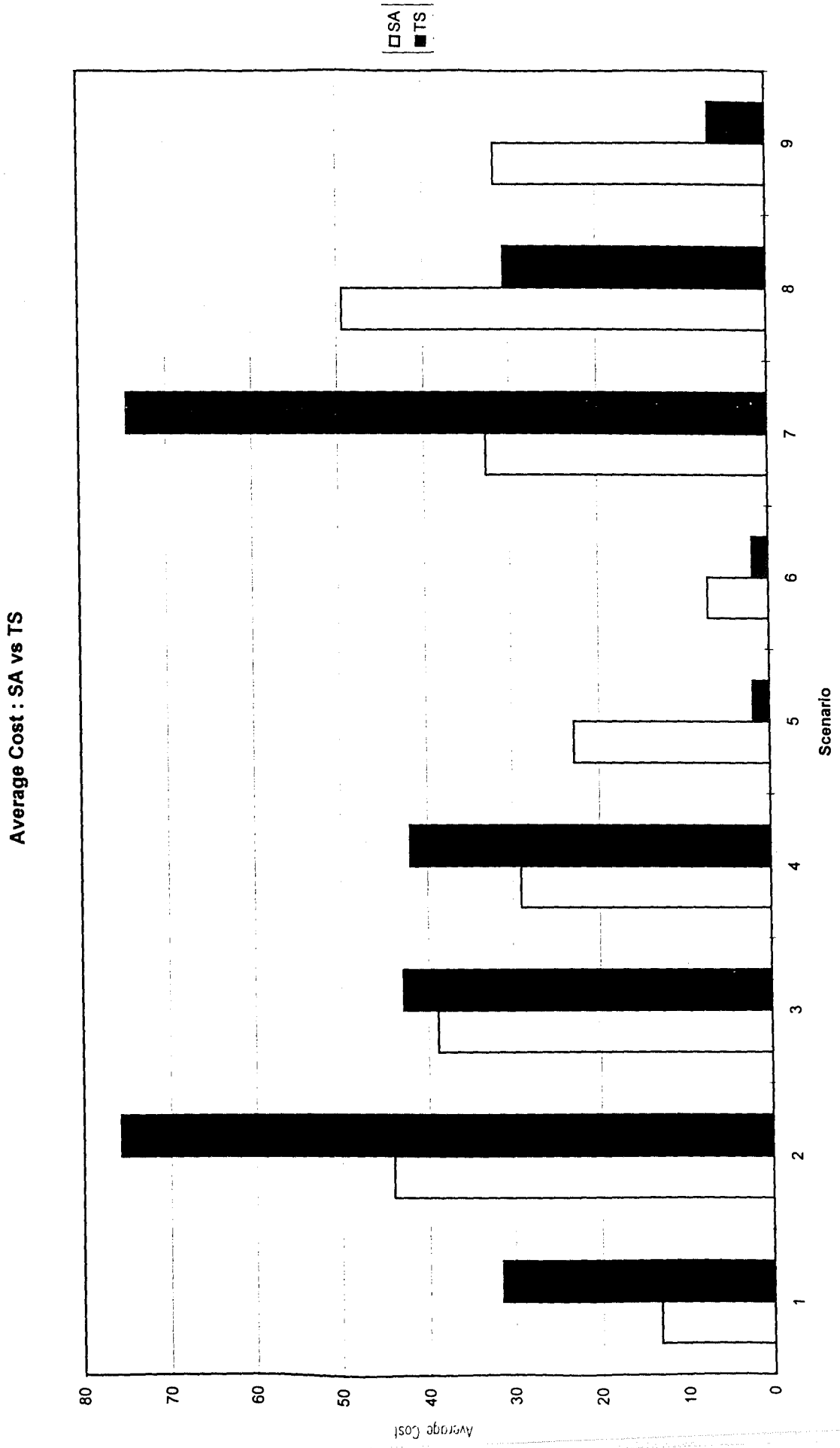


Figure 25. Graph showing SA vs. TS Average Time (Group B)

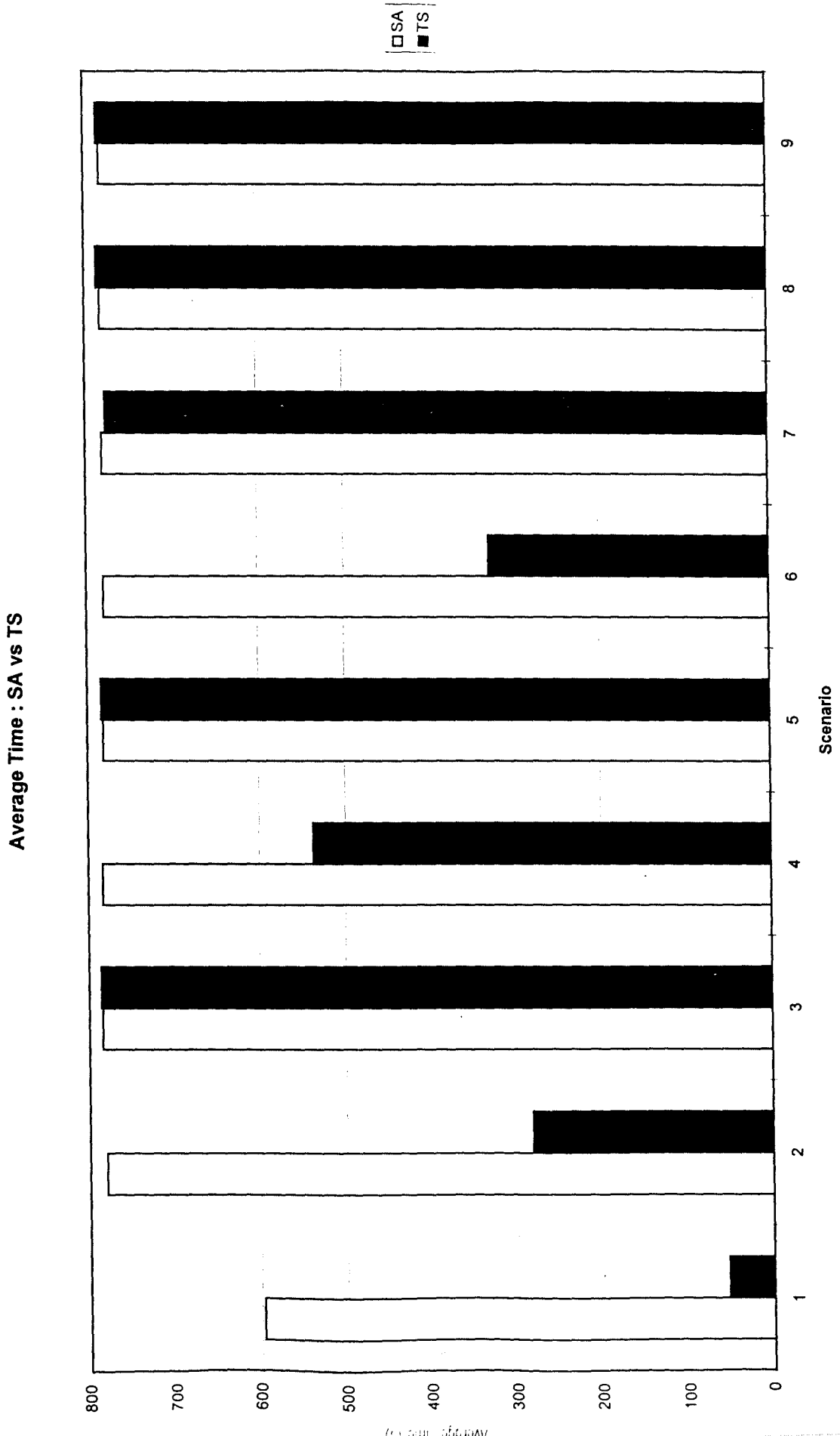


Figure 26. Graph showing SA vs. TS Best Cost (Group B)

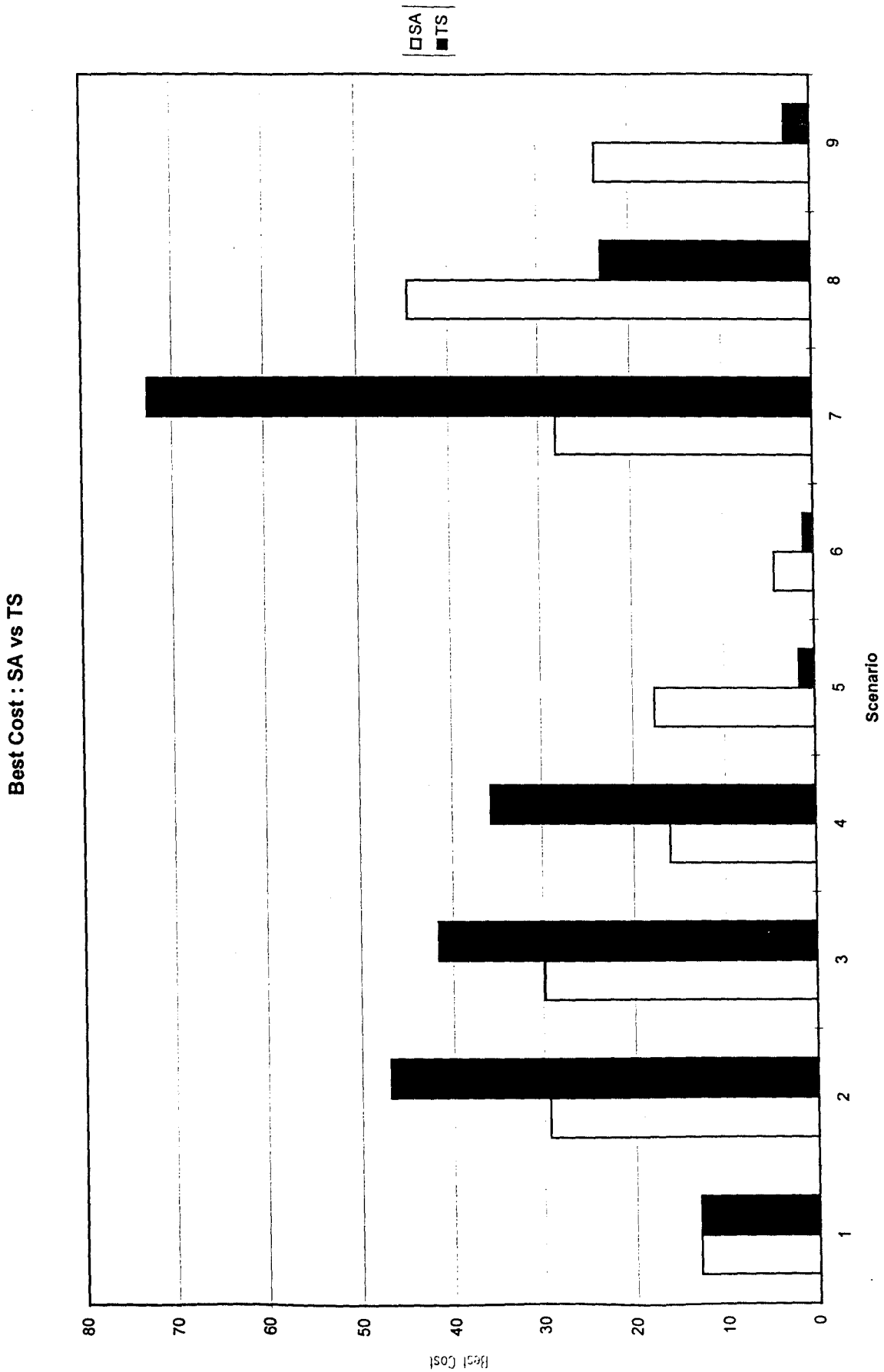


Figure 27. Graph showing SA vs. TS Time to Best Cost (Group B)

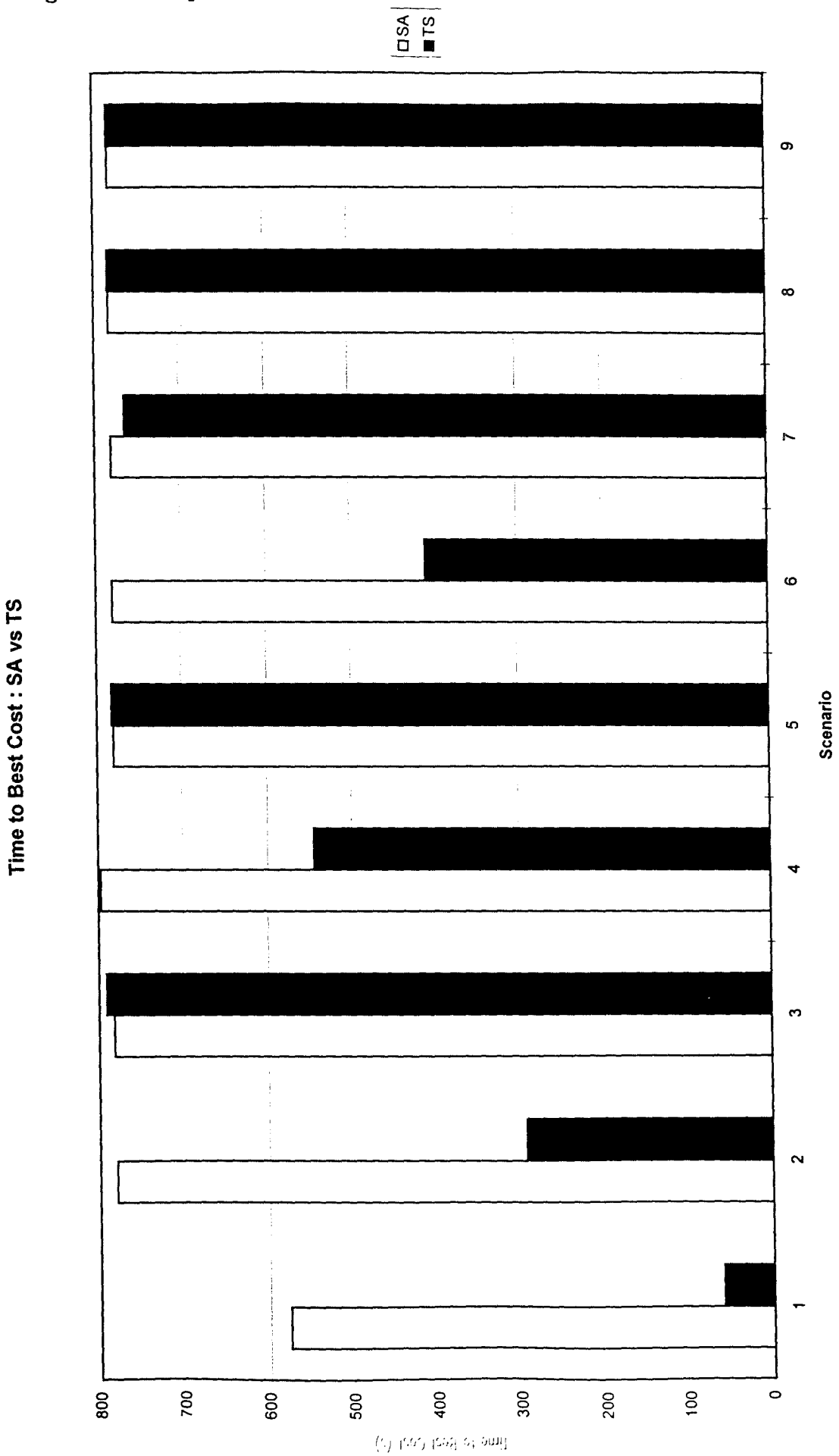
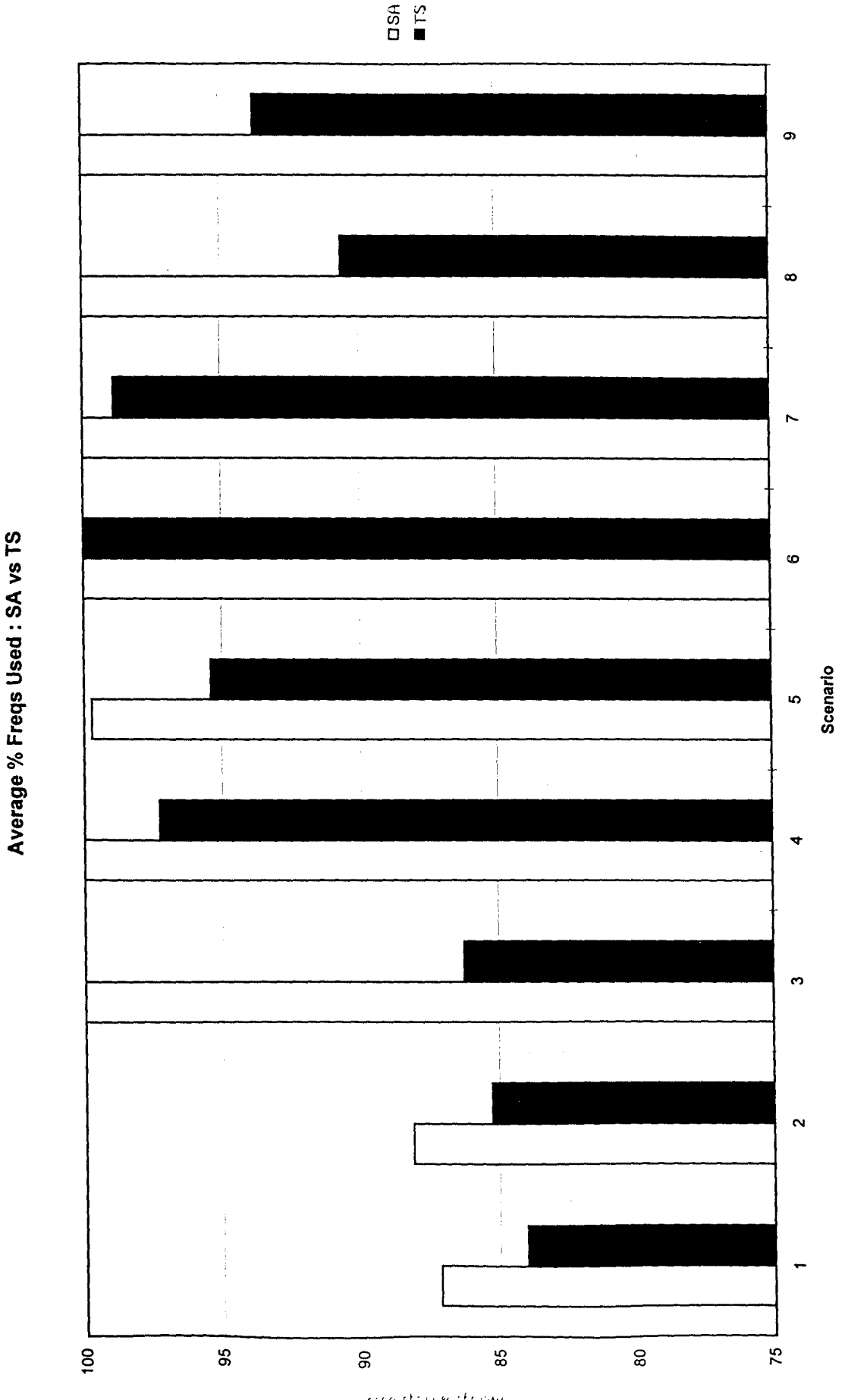


Figure 28. Graph showing SA vs. TS Average % Frequencies Used (Group B)



7.4.1 Comparison Difficulties

Run-Time

Both simulated annealing and tabu search have various criteria which might cause the algorithm to terminate before the maximum run-time has elapsed. In addition the check for maximum run time is outside a loop and so the algorithm may complete another execution of the loop (beyond a time close to the termination time) before stopping and therefore take slightly longer than the maximum run time. Owing to the differing run times it is difficult to compare fairly the cost of the final solutions found by the two algorithms. The maximum time allowance (15 minutes) was given by D.E.R.A. when the data was provided. Owing to the nature of the data (soldiers in combat) the updating is so extensive that the bulk assignment method is often used. The ground covered by soldiers in 15 minutes dictates the frequency of re-running the assignment program.

Multiple Performance Measures

In addition to cost and time, other performance measures are used to evaluate the overall quality of a solution : span and order of frequencies used. It is advantageous if frequencies not used in a solution are able to be released for other uses thereby ensuring maximum utilisation of this scarce resource. A comparison of the two algorithms considers all four performance measures; as there is not a constant standard for any of these measures comparisons are difficult. The span and order of the assignment are used in the evaluation of a solution even though they do not appear in the objective function.

7.5 Discussion

This section is divided into two parts: part one compares the effectiveness of the simulated annealing and tabu search algorithms on the scenarios for which there are known zero interference solutions (Group A data), while part two compares their effectiveness on the scenarios for which there are no known zero interference solutions (Group B data). The results spreadsheets used to create these graphs are given in Appendix A.

The following abbreviations have been used in this section :
SA = simulated annealing, TS = tabu search.

7.5.1 Group A Data

Average Case

On average SA obtained optimal solutions for 68% of scenarios, whereas TS obtained optimal solutions for 76% of scenarios. There were 21 scenarios solved optimally by both algorithms, and for the remaining 16 scenarios TS obtained marginally lower cost solutions than SA in 63% of cases, some of these scenarios were solved optimally. However, the graph of average cost (Fig 19) clearly shows that there are 6 scenarios (4, 11, 18, 22, 23 and 29) for which SA obtained significantly lower cost solutions than TS on average.

By looking at the graph of average time (Fig 20) it can clearly be seen that TS generally finds its solutions in considerably less time than SA. For the 6 scenarios for which TS obtained comparatively high cost solutions the time taken by SA was 3.5 times longer on average. For the 21 scenarios for which both algorithms obtained optimal solutions 13 solutions were found by both algorithms in less than one second, for the 8 remaining scenarios TS found its solutions using on average just 5.6% of the time taken by SA. Tabu search obtained lower cost solutions than SA for 10 of the 16 non-optimally solved scenarios, for these scenarios the time taken by TS was approximately 7% of the time taken for SA to obtain its inferior solutions.

Best Case

In the best case SA obtained optimal solutions for 86% of scenarios, whereas TS obtained optimal solutions for 89% of scenarios. Both algorithms obtained optimal solutions for 28 of the scenarios. Of the remaining 9 scenarios TS obtained optimal solutions to 5 scenarios which were not solved optimally by SA, while for the other 4 scenarios SA obtained lower cost solutions than TS of which 3 were solved optimally. By looking at the graph (Fig 21) it can be clearly seen that for scenario 29 SA obtained a significantly lower cost solution than TS, but the other cost values were more comparable.

Looking at the graph showing the time taken to reach the best cost for each scenario (Fig 22) it can again be seen that TS generally finds its solutions in considerably less time than SA. For the 28 scenarios for which both algorithms obtained optimal solutions, 13 solutions were found by both algorithms in less than one second (as for the average case), which for the 15 remaining scenarios TS found 13 of its solutions using on average just 7.8% of the time taken by SA. For the other 2 scenarios (4 and 23) SA found its solutions significantly more quickly than TS although the times for both algorithms was small. Tabu search obtained lower cost solutions than SA for 5 of the 9 non-optimally solved scenarios, for which the time taken by TS was approximately 5% of the time taken for SA to obtain its (inferior) solutions. Simulated annealing obtained lower cost solutions than TS for 4 scenarios, for 2 of which SA found its lower cost solution using 25% more time than taken by TS, and for the other 2 TS took approximately 80% longer. The time taken by SA for scenario 29, for which TS obtained a comparatively high cost solution, was 20% longer than TS.

Group A Summary

- All of these scenarios have known zero cost solutions.
- The general pattern of the average and best case graphs was the same.
- For the average (best) cases TS solved 76% (89%) of the scenarios optimally, SA solved 68% (86%) optimally.
- 13 scenarios were solved in less than one second by both algorithms, for these scenarios only 68% of the available range and less than 50% of the available frequencies were used (Fig 23). For the remaining scenarios 80-90% of the available range was used; the values were comparable for TS and SA.
- There were 10 scenarios for which TS obtained a lower cost than SA. These solutions were obtained in approximately 7% of the time taken by SA to obtain its inferior solutions.
- There were 6 scenarios for which SA obtained significantly lower cost solutions than TS. However, SA ran for approximately 3.5 times longer than TS.
- For the majority (59%) of scenarios TS used a much smaller percentage of the available frequencies in its solutions (and also obtained lower cost solutions) than

SA. For the remaining scenarios the results were comparable.

- Whilst TS outperformed SA for the majority of scenarios it is clear that for those scenarios for which TS did not perform well on average the SA algorithm was far superior.
- TS generally took less time than SA.

7.5.2 Group B Data

There are 9 scenarios in Group B, and it is not known whether zero-cost solutions exist for these scenarios.

Average Case

By looking at the graphs of average cost and time (Figs 24 and 25) it can be seen that TS obtained lower cost solutions than SA for 4 scenarios, while for the remaining 5 scenarios SA obtained lower cost solutions than TS.

For the 4 scenarios for which TS obtained lower cost solutions: in each case SA used the full time allowance, TS used the full time allowance for 3 scenarios, and for scenario 6 TS found its lower cost solution in approximately 40% of the available time.

For the 5 scenarios for which SA obtained lower cost solutions: for 2 of the scenarios both algorithms used the full time available, and for the 3 remaining scenarios SA ran for considerably longer than TS on average.

Best Case

The graphs showing the best cost and time to best cost (Figs 26 and 27) exhibit the same pattern as those for the average case. The most improvement in cost from the average to best cases for TS was for scenarios 1 and 2. The most improvement in cost from the average to best cases for SA was for scenarios 2 and 4. For all other scenarios the best cost values were only slightly less than the average cost values. For scenario 1 both algorithms obtained the same best cost, TS algorithm stopped sooner. For all the other scenarios it can be seen that the best cost for SA was arrived at using the

full time available and similarly for TS in 3/4 cases. This implies that the group B scenarios are considerably more difficult than those in Group A.

Group B Summary

- These scenarios are typically solved well when the full time allowance is used
- For scenario 6 (for which low cost solutions were obtained by both algorithms). TS used approximately half of the available time to obtain a very good solution.
- SA generally used the full time available whereas TS stopped early on several occasions. When TS stopped prematurely the solutions were of higher cost than the corresponding SA solution.
- For Group B the entire range was used for all scenarios.
- For scenario 6 both algorithms used 100% of the available frequencies.
- TS obtained lower cost solutions using a smaller percentage of the available frequencies than SA for 3 of the 9 scenarios (Fig 28). For the remaining 5 scenarios SA obtained lower cost solutions and used a higher percentage of frequencies.
- Both algorithms obtained best solutions with the same cost for scenario 1.

7.5.3 What makes a Scenario 'Hard'?

The TNET data has been split into two groups, A and B. The group A scenarios have known optimal (zero-interference) solutions, many of the scenarios were solved optimally by both TS and SA. The solutions for the group A scenarios frequently used less than the full resources available with respect to time, span and order. In contrast the solutions for the group B scenarios regularly used all of the available resources and all of the solutions had some measure of interference. This empirical evidence suggests that the group B scenarios are in some way 'harder' than those in group A. What makes a scenario 'hard'? The results also show that 13 of the group A scenarios were solved optimally in less than 1 second, using few resources, by both algorithms. What makes a scenario 'easy'?

Determining in advance how difficult a scenario (particular instance of a problem) will be is non-trivial, and often not possible. Test data metrics can provide an indication

of the size and complexity of the search space. However, a scenario with a large search space is not necessarily a more difficult one. The interplay of the various factors which could influence the difficulty of a scenario mean that categorising scenarios using a single metric is unlikely to prove successful.

For the frequency assignment problem some factors which could influence the difficulty of a problem instance are given below:

- **The percentage of co-site constraints**

  The C_{ij} values for co-site constraints are large and consequently those scenarios are more difficult to solve given a fixed span of available frequencies.

- **The largest vertex degree**

  If one or more vertices are highly connected then there will be a large number of constraints which must be mutually satisfied to enable a solution with zero interference to be found. The difficulty is increased if one or more vertices with large vertex degree are themselves connected. In particular, if there is a clique (a complete subgraph) of large size then the difficulty can be no better than the problem of finding a frequency assignment for the clique.

- **Size of the search space ($|F|^N$)**

  A larger value of $|F|$ would increase the search space. If the problem has a unique solution, then even if $|F|$ is increased (within a fixed span) then that solution is harder to find because the search space is larger.

- **Span(F)**

  If the span of available frequencies is increased, and the difference between frequencies is maintained, then there is an increased likelihood of being able to satisfy the constraints and so the scenario becomes easier.

- **The (approximate) value $\frac{C}{\frac{N^2}{2}} * 100$.**

  Average connectivity of a vertex. If this value is high then there will be many constraints which must be mutually satisfied to enable a solution with zero interference to be found.

- **L.co-site C_{ij}**

  If the largest required co-site frequency separation is close in value to the span of the available frequencies then the scenario will be more difficult. This is because

the links involved in those co-site constraints will be limited to taking values from a very small number of frequencies at either end of the range.

- **The number of global optima**

  This would only be known if complete enumeration were possible! It is reasonable to suppose that in general a scenario with a large number of optima is easier than a scenario with one.

The results presented show that 13 of the group A scenarios were solved optimally in less than 1 second. By looking at tables 5 and 6 in chapter 6 it can be seen that all of the 13 scenarios had either a very small search space or had a low percentage of co-site constraints. The values for these two metrics were significantly smaller than for the remaining scenarios. This observation leads to a tentative conclusion that scenarios with a low percentage of co-site constraints or small search space are easier to solve (on average). The idea of reducing the search space size provided the initial motivation for investigating the divide and conquer technique described in the next chapter. The greedy algorithm in stage 2 of the divide and conquer algorithm aims to spread evenly those links involved in large numbers of co-site constraints.

For ease of reference the metrics for the 13 'easy' scenarios are reproduced here.

Table 9. Group A Data - 'Easy' Scenarios

Scenario	|F|	N	C	%co-site	span(F) MHz	Largest V.degree	L.co-site C_{ij}	L.far-site C_{ij}	Search Space
2*	40	60	244	16.4	169	25	27.1	6.1	10^{96}
3*	80	98	193	35.8	209	18	40.1	8.1	10^{187}
6*	40	12	20	0.3	169	6	27.1	0.6	10^{19}
9*	40	16	30	26.67	169	5	27.1	0.6	10^{26}
13*	40	4	4	50.00	169	2	27.1	6.1	10^{6}
16*	40	4	2	100.0	169	1	27.1	0.0	10^{6}
20*	40	12	27	22.2	169	7	27.1	0.3	10^{19}
25*	40	12	16	37.5	169	6	27.1	6.1	10^{19}
28*	40	328	670	0.0	399	14	—	5.5	10^{525}
30*	40	16	50	16.0	169	10	27.1	6.1	10^{26}
33*	40	16	19	52.6	169	4	27.1	0.6	10^{26}
36*	40	422	1191	0.0	399	31	—	5.5	10^{676}
37*	40	448	1782	0.0	399	33	—	5.5	10^{717}

7.5.4 Parameters

Some improvement to the results could be expected by altering the parameters. However a sample of scenarios was originally used to set the parameters and then they remained constant throughout. One of the aims of this research was to find a robust, general method for solving the frequency assignment problem and so using a method which requires fine tuning for each scenario is undesirable.

7.6 Conclusions

Both algorithms solved the 13 'easy' scenarios quickly and using few resources.

For group A, TS outperformed SA for the majority of scenarios but for those scenarios for which TS did not perform well on average the SA algorithm was far superior. The poor achievement of TS for six of the scenarios can be attributed to the premature termination of the algorithm (SA ran for approximately 3.5 times longer on average). At the time when the computations for this comparison were executed it was not known which of the termination criteria for TS was satisfied.

For the group B scenarios the best solutions were found when the full run-time allowance was used. Both algorithms obtained the same cost solution for scenario 1 and each obtained the lowest cost for half of the remaining scenarios. Once again SA obtained lower cost (better) solutions than TS when TS stopped before the maximum run time had elapsed.

In view of the general difficulty in getting heuristic algorithms to run the full allocated time and the particular problems encountered with TS described in this section, it was decided to re-run TS with different parameters for a selected subset of the scenarios with a view to discovering if the early termination was a significant factor in the poorer results. This work is described in section 7.7.

In conclusion: both algorithms are able to obtain good quality solutions for the frequency assignment problem. Tabu search obtains lower cost solutions using fewer resources for most scenarios. However, TS terminated early on a number of occasions and so SA has proven to be the more predictable of the two algorithms.

7.7 Termination Criteria Investigation

This section describes the further investigation into the premature termination of TS for a selected subset of scenarios. Group A (4, 11, 18, 22, 23 and 29) and all scenarios in Group B.

7.7.1 Early Termination of TS

After investigation it was discovered that for some scenarios TS terminated early, the possible termination criteria are given in section 7.3.6. For each of the scenarios investigated the TS algorithm terminated because a given gumber of iterations had yielded solutions of identical cost. The allowable number of iterations yielding identical cost is variable and determined by the user; by altering this value it was possible to enable TS to run for a longer period of time. The final solution cost of these longer runs was considerably reduced for some scenarios. This indicates that TS is capable of obtaining lower cost solutions than the original results suggested. For ease of reference the longer run TS shall be referred to as LRTS. The results of this latest test run are given against the previous SA and TS results for comparison (Figs 29, 30, 31 and 32).

Figure 29. Graph showing SA & TS vs. LRTS Average Cost (Group A subset)

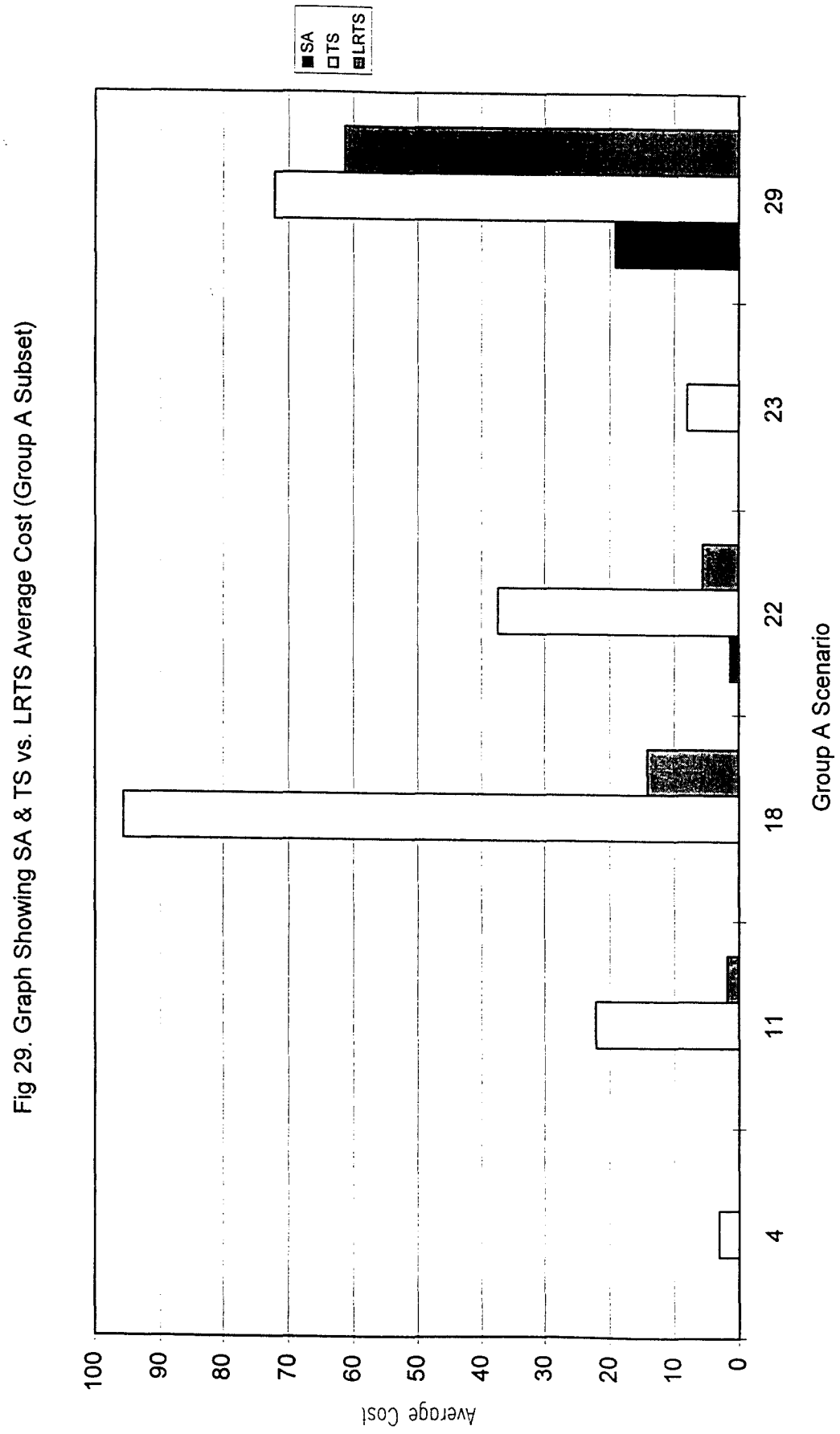


Fig 29. Graph Showing SA & TS vs. LRTS Average Cost (Group A Subset)

Figure 30. Graph showing SA & TS vs. LRTS Average Time (Group A subset)

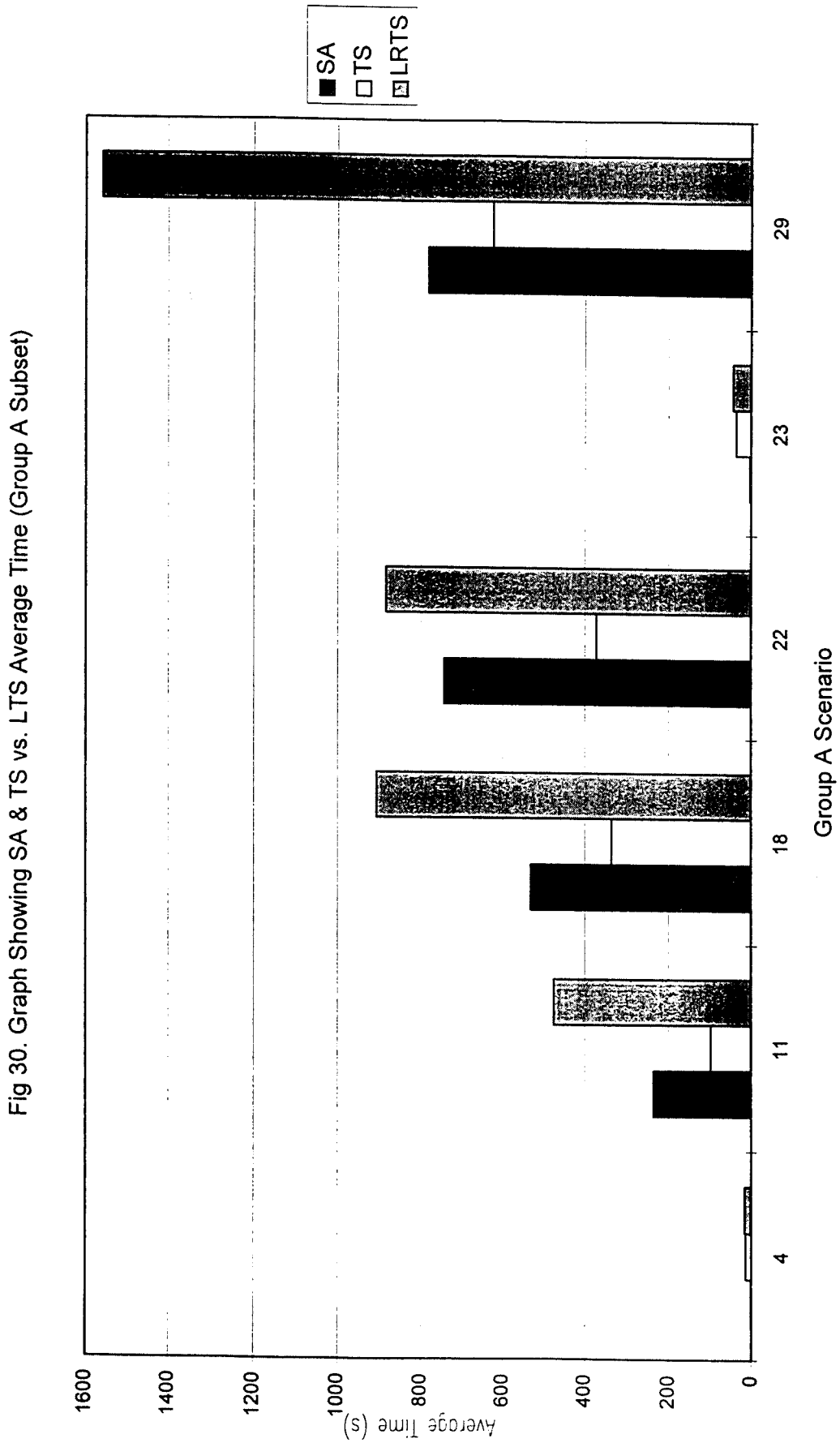


Figure 31. Graph showing SA & TS vs. LRTS Average Cost (Group B)

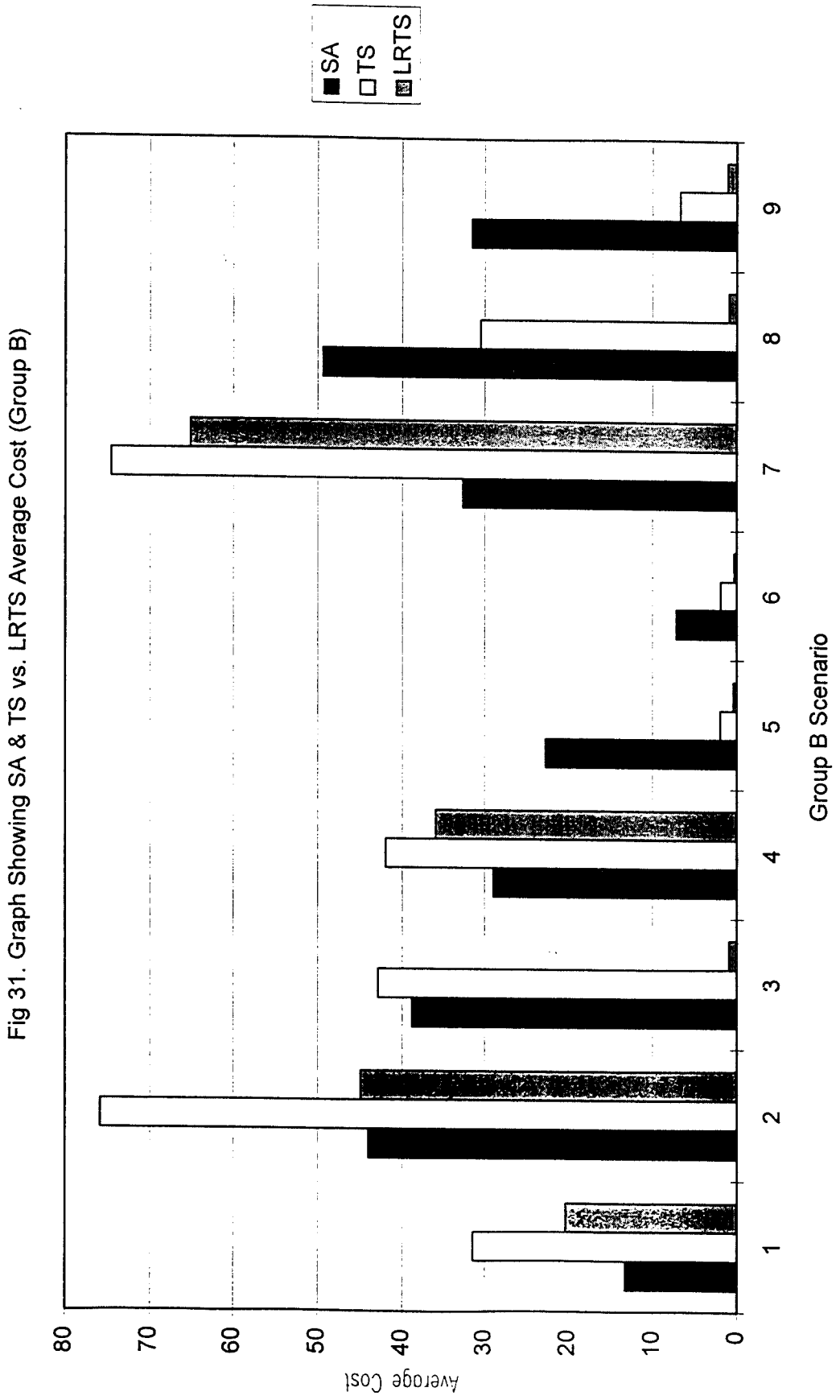
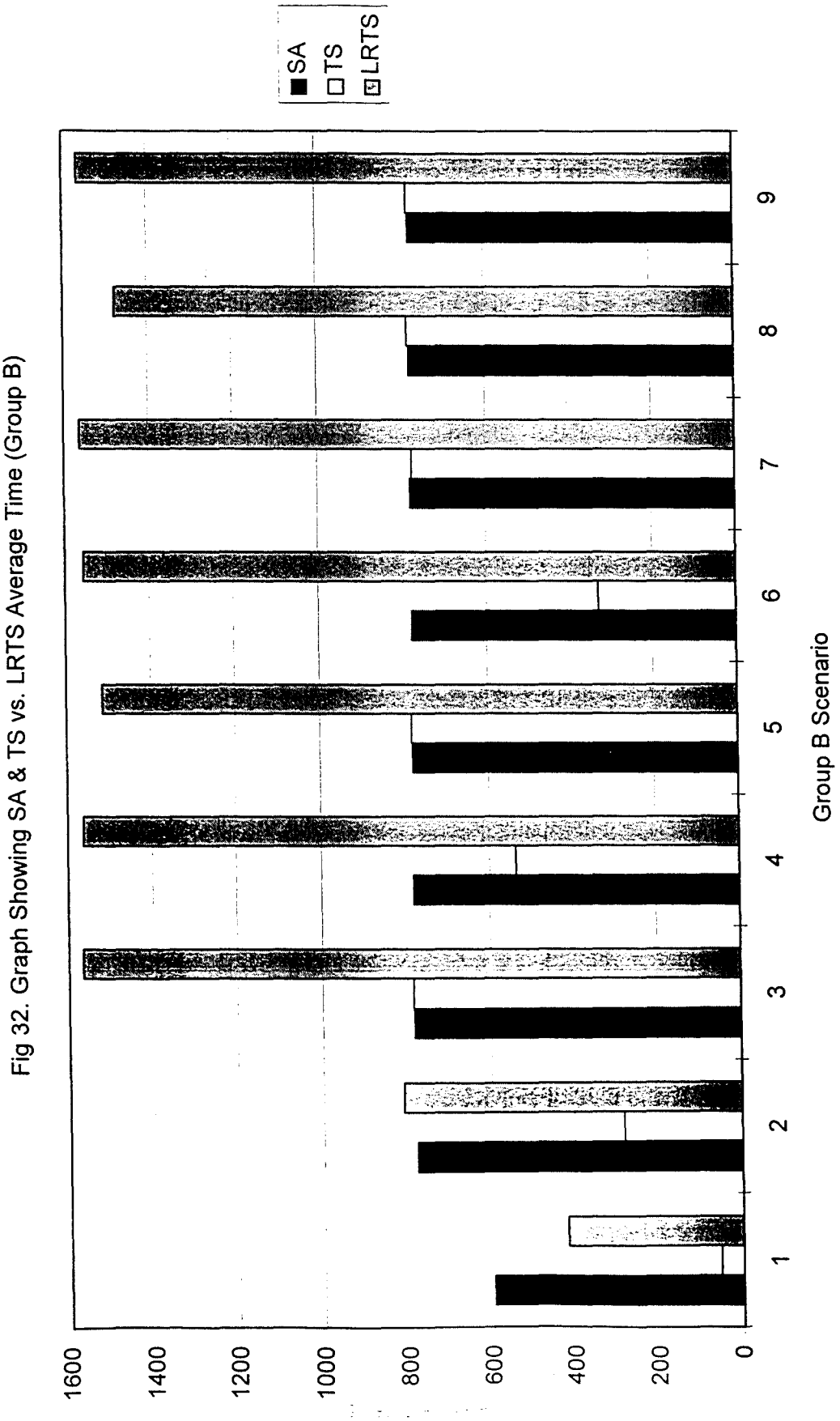


Fig 31. Graph Showing SA & TS vs. LRTS Average Cost (Group B)

Figure 32. Graph showing SA & TS vs. LRTS Average Time (Group B)



Group A

For the group A scenarios there were only six scenarios which TS did not solve optimally on average originally and so the graphs given show only these six scenarios. When tabu search was able to run for longer the same results were obtained for the remaining 31 scenarios as for the original (shorter) run time.

Fig 29 clearly shows that the LRTS obtained lower cost solutions on average than the TS algorithm. For two of the scenarios (4 and 23) LRTS obtained optimal solutions previously unobtained by TS — only slightly more time was taken to reduce the low cost solution to an optimal one. For three of the remaining four scenarios the time taken was approximately 1.5 times that originally used and the solution cost using LRTS was markedly reduced (although still higher than the corresponding SA solution). For the final scenario the LRTS algorithm ran for considerably longer, approximately 16 minutes (2.5 times) longer. For this scenario the cost was reduced although only slightly — and still did not approach the low cost solution found by SA in approximately half the time.

Group B

For all scenarios LRTS obtained lower cost solutions than TS. For the first two scenarios in this group the LRTS ran for approximately 2.5 times longer than TS and obtained solutions with costs approximately 60% of those obtained previously (and still greater than the corresponding SA solutions). Simulated annealing ran for longer than LRTS for the first scenario. For the final seven scenarios LRTS used twice the length of time used by SA and TS (for scenarios 4 and 6 this margin was greater). For five of the scenarios the LRTS algorithm obtained very low cost solutions (near-optimal) and considerably lower than either of the SA or TS solutions. For the remaining two scenarios LRTS obtained lower cost than TS but this value was still higher than the corresponding SA solution.

Conclusions

For the group A scenarios there was improvement in solution cost from TS to LRTS although the SA algorithm still obtained the better solutions (scenarios 4 and 23 were

comparable). For the group B scenarios the improvement in solution cost obtained by the longer run times was significant for all scenarios, for five scenarios the LRTS results were much improved on the TS and SA algorithms, for the remaining four scenarios LRTS was still unable to obtain solutions to rival SA.

To conclude: greater run times for the group A scenarios yielded little improvement whilst for over half of the harder scenarios the improvement was significant. There were still four scenarios for which SA obtained lower cost solutions than LRTS (Group B scenarios 1, 2, 4, 7).

The only parameter which was changed for this investigation was the variable indicating entrapment in a local optimum. However, this had a significant impact on the length of time for which the algorithm ran. This is due to the timing parameter being checked outside of a loop. It is possible that the parameters originally set for the algorithms are not optimal and that tweaking the TS parameters may yield better results. However, it was the intention of this study to try and find a robust technique which is why this has not been investigated.

7.7.2 Lower Bound for Group B Scenario 1?

The results from both algorithms provide a list of all constraints that were violated along with the required and actual frequency separation values. By investigating the violations of the group B scenario (no. 1) for which both algorithms obtained the same best-cost it was anticipated that a lowerbound might be observed. A lowerbound, in this case, might be a triangle of constraints which have a combined separation requirement greater than the available frequency span. Such a triangle would therefore be unsatisfiable with the available resources.

The constraint violations for both SA and TS were similar in nature, but did not confirm the expectations of an unsatisfiable trio of constraints as previously described.

7.8 Further Research

It would be interesting to run the algorithms for longer on the Group B data to see if any further improvement to the best cost values could be obtained. The practical

application of this data set would prevent run-times in excess of 15 minutes. However, from an investigative view it would be interesting to gauge the algorithms' performance over a longer time span.

Chapter 8

Divide and Conquer

The divide and conquer technique was described briefly in Chapter 4. In this chapter more specific implementation details are given. The divide and conquer technique described here has a number of improvement phases which use a metaheuristic algorithm. Simulated annealing and tabu search have been used and the results are given and some conclusions drawn. This chapter is divided into six parts: part one is an introduction to the divide and conquer technique, part two discusses ways of dividing the frequency assignment problem, part three suggests ways of overcoming non-independent subproblems, part four gives the implementation detail of the technique, part five gives the results, and finally, part six gives some conclusions for the chapter.

8.1 Introduction

The divide and conquer technique was investigated to try and reduce the search space of the problem without the use of problem specific knowledge. The literature cites many examples of successful applications of heuristics to the frequency assignment problem (see Chapter 3 section 3.1.8), many of which have extensive preprocessing elements that reduce the size of the problem or build up stores of problem-specific data that is used later in the algorithm [Aar et al 95] [BB95b] [SH97] [THL95].

The nature of the scenarios provided for military applications is more random than interference networks for mobile telephones, for example. The well-known Philadelphia

problems (see Chapter 3 section 3.2.2) for mobile telephone applications have graphs based on a regular hexagonal pattern and this knowledge of the graph can be exploited in solving the frequency assignment problem. In contrast the antennae and locations of the soldiers in a military scenario are far from regular (and constantly changing for the incremental assignment problem). Although the bulk assignment problem has been studied here, a general approach to the problem has been used to allow for extension of the program into the update assignment problem in future.

The only preprocessing of the scenario in this implementation is the ordering of the links into descending co-site vertex degree order prior to the initial greedy assignment in stage three. This ordering was suggested by earlier results outlined in Chapter 5.

8.2 Dividing the Problem

The divide and conquer strategy is simple to understand; the difficulty is finding a way to divide the initial problem into subproblems. The subproblems should ideally be independently solvable and their recombination should provide a solution to the initial problem.

Two possible division criteria were considered for the frequency assignment problem; dividing the frequency set into separate bands or dividing the links to be assigned.

To divide the links successfully the graph representing the problem would need to be divided into one or more disjoint subgraphs of approximately equal difficulty. Detailed knowledge of the constraint network would need to be exploited to work out how best to divide the graph. In the event that the constraint graph could not be separated into disjoint graphs an approximation could be obtained using a min-cut algorithm. Division of the problem by dividing the links depends on the fact that the constraint graph can be decomposed into one or more smaller graphs. One way to divide the links is to split the graph using arc consistency. The arc consistency preprocessing was one of the key factors in the success of heuristic approaches explored in the CALMA project [CAL95]. An arc consistency algorithm was applied to the TNET scenarios. The results indicated that the graph could rarely be simplified owing to the high connectivity. The constraint graphs representing a tactical communications FAP are typically highly interconnected and rarely reducible and so division of the problem by dividing the frequencies is more

appropriate.

Division of the problem into subproblems by dividing the frequencies does not require knowledge of the constraint graph, is simple to execute and applicable to all types of graph. The implementation described here divides the frequency set to provide subproblems.

Having decided to divide the frequency set a further decision needs to be made; whether to divide F into bands having approximately the same *order* (same number of frequencies) or *range*. The links are first placed in descending order of co-site vertex degree as described in chapter 7 section 7.1.5. Then they are evenly distributed between the subproblems as described in section 8.4.3 stages 1 and 2. The set F closely resembles a set of random discrete frequencies from a given range.

Dividing the frequency set into bands of approximately the same *order* ensures that the search space of each of the subproblems is comparable. However, the range of frequencies in each band may vary. Selecting the number of bands would depend on the desired search space size.

Dividing the frequency set into bands of approximately the same *range* creates subproblems having varying search space sizes although it potentially facilitates the efficient assignment of links involved in co-site constraints. Selecting the number of bands is related to the average co-site channel separation and the range of frequencies available.

Early experimentation showed the division of the frequency set F into bands having approximately the same *order* gave assignments with less interference than division of F into bands having a similar *range*.

In stage one of the algorithm the band of frequencies, F, is partitioned into approximately the same order, non-overlapping bands $F_1, F_2, ..., F_k$, for some k which was found by experimentation. Kauffman, Macready and Dickinson [KMD95] used a similar technique for the spin glass problem and found that the value for k had a significant effect on the quality of the final solutions.

8.3 Overcoming Non-Independent Subproblems

Dividing the frequency set introduces artificial boundaries. In the initial problem the links could take any value from the entire frequency range. Since the graph is highly interconnected, it is unlikely that restricting links to a limited range of frequencies will yield optimum assignments. However, it does enable solutions of the resulting subproblems to be found quickly. The implementation described here follows the band improvement stage with a global improvement stage which allows the links to take frequencies outside of their allocated band. This enables the algorithm to overcome any initial decisions, made with incomplete knowledge, that later prove to be prohibitive to finding good solutions.

8.3.1 Solving Subproblems

Once the problem has been suitably divided it is necessary to obtain an initial solution for each of the subproblems and also to improve these solutions further. The tabu search and simulated annealing metaheuristics, described in Chapter 7, proved relevant to the solution of the frequency assignment problem and so they have been used within the divide and conquer technique to improve the subproblems once an initial solution has been obtained. The parameter values used in chapter 7 (section 7.2.2 and section 7.3.3) were also used for the global stages of the divide and conquer implementation, the only change to the parameters for the local stages was that MaxIters was divided by k ($k = 4$). the number of bands.

8.3.2 Combining Partial Solutions

The solution of each of the subproblems is an assignment of links to frequencies. Each link belongs to one and only one subproblem and so the combination of the subproblem solutions to form an assignment for the initial problem is trivial, representationally, in this case. Of course, there may still be unsatisfied constraints which exist between links in different bands.

8.3.3 Alternating Local and Global Phases

During early experimentation the subproblems were solved using the chosen metaheuristic, the subproblem solutions were combined and then the metaheuristic was applied to the whole assignment as a final improvement phase. This idea was extended: local and global improvement phases were applied alternately until no further improvement was found.

8.4 Implementation

All of the code was written in ANSI C using Borland C++ version 4.5 software. The results were obtained on a Pentium desktop computer (100MHz).

In this section, a more detailed description of the divide and conquer implementation is given. There are eight stages:

1. Division of the set of frequencies into bands.
2. Allocation of a band to each link.
3. An initial assignment of a frequency within its allocated band to each link.
4. An application of a steepest descent algorithm.
5. A sequence of local changes to the frequencies.
6. A sequence of global changes to the frequencies.
7. If an improved solution was found in stage 5 or 6 then update data structures and return to step 5, otherwise continue with step 8.
8. A further application of the steepest descent algorithm to enhance the overall quality of the solution.

8.4.1 Pseudocode Algorithm

The pseudocode for the algorithm is given in Fig 33.

Figure 33. Divide and Conquer Pseudocode

```
Initialise data structures
Generate frequencies (given number and range)
Sort links into descending co-site vertex degree order
Chosen heuristic := simulated annealing OR tabu search
step 1 Divide the problem into k bands according to division criteria
step 2 Greedy algorithm : allocate each link to a band
    For each link (in desc order of difficulty) Do
        For each band Do
            Evaluate stage2_cost
            If stage2_cost := 0 Then allocate link to band
        EndFor
        If link unallocated Then allocate to band with fewest links
    EndFor
step 3 Greedy algorithm : allocate each link to a frequency from band
    For each band Do
        For each link in band Do
            For each frequency in band Do
                Evaluate cost of assigning this frequency to this link
                Assuming the WORST of any unassigned links
                Note frequency with minimum cost
            EndFor
            Assign frequency with minimum cost
        EndFor
    EndFor
step 4 Local Steepest descent
step 5 Local improvement (chosen heuristic)
step 6 Global improvement (chosen heuristic)
step 7 Local improvement (chosen heuristic)
step 8 Global Steepest descent
Report the results to file and close the results file
```

8.4.2 Rationale

The results given in Chapter 7 showed that the scenarios with a smaller search space or with a low percentage of co-site (larger required frequency separation) constraints were more quickly solved by both tabu search and simulated annealing heuristics. The rationale for using a divide and conquer technique was to reduce the scenarios to subproblems that had a smaller search space. The links involved in co-site constraints were initially divided between the subproblems to distribute the more difficult constraints evenly. The idea of solving the subproblems in a number of different stages was based on a multi-phase approach described by Thompson and Dowsland [TD96] whereby decisions made in earlier phases could be undone later.

Example (data from TNET scenario 1)

N = number of links, $|F|$ = number of frequencies, search space = $|F|^N$

For example, $N = 158, |F| = 40, k = 4$

- original search space size = $|F|^N = 40^{158} = 10^{250}$
- stage two search space = $k^N = 4^{158} = 10^{95}$
- local stages search space = $(\frac{|F|}{k})^{\frac{N}{k}} = 10^{40}$

8.4.3 Detailed Explanation

Stage 1

In stage one of the algorithm the band of frequencies, F, is partitioned into approximately the same order, non-overlapping bands $F_1, F_2, ..., F_k$, for some k which was found by experimentation.

$F = F_1 \cup F_2 \cup F_3 ... \cup F_k$ where $Fi \cap Fj = \{\}$, $1 \leq i < j \leq k$ and also
$a_i \in F_i$, $a_{i+1} \in F_{i+1} \Rightarrow a_i < a_{i+1}$ for $i = 1, 2, ...k-1$

Early experimentation suggested that $k = 4$ would yield good quality solutions. The maximum range of frequencies available for any of the scenarios was approximately 400 MHz and the maximum co-site frequency separation was approximately 200 MHz. Thus, by dividing the range into 4 bands the largest co-site constraints could be guar-

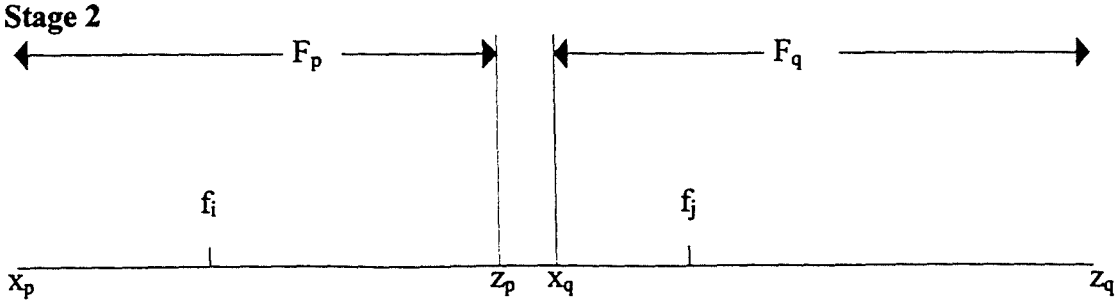
anteed to be solved if one link involved in that constraint was placed in band 1 and the other in band 4. There is clearly a trade-off between the overhead required to maintain a number of bands and the benefit gained by having subproblems with smaller search spaces.

Stage 2

In stage two each link was allocated to one of the bands. In allocating a link to a band, consideration was given to the satisfaction of two types of constraints; the *external* constraints between that link and other links which have been allocated *other* bands and *internal* constraints between that link and other links allocated the *same* band. A constraint is *feasible* if it is possible for the links to be assigned frequencies within the appropriate bands which satisfy the constraint. Allocation of the links to the bands uses a greedy algorithm based on an estimate of the number of unsatisfiable constraints which is given below. For any particular link, the band allocated the fewest links is considered first. If all constraints, external and internal are feasible, that band is chosen. Otherwise, the next band is considered. If no band is suitable, the band with the fewest links is chosen.

To determine whether a constraint is feasible the following equations were devised. Recall that the constraints are of the form $|f_i - f_j| \geq C_{ij}$, note that if $f_i \in F_p$ and $f_j \in F_q$ and $p < q$, the constraint simplifies to $f_j - f_i > C_{ij}$ Consider Fig 34. $x_p = min(f \in F_p)$, $z_p = max(f \in F_p)$.

Figure 34. Stage 2 Representation of Bands



If $C_{ij} \leq x_q - z_p$ then $f_j - f_i \geq x_q - z_p \geq C_{ij}$ the constraint was satisfied

If $C_{ij} > z_q - x_p$ then $f_j - f_i \leq z_q - x_p < C_{ij}$ the constraint was not satisfiable

If $x_q - z_p < C_{ij} \leq z_q - x_p$ then the constraint may have been satisfied.

The 'objective function' for the greedy algorithm used in this stage simply counted the number of unsatisfiable constraints using the equations above.

$$Stage2\_Cost(A) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \delta_{ij}$$

$$\delta_{ij} = \begin{cases} 0 \text{ if } C_{ij} \leq x_q - z_p \\ 1 \text{ if } C_{ij} > z_q - x_p \\ \lambda \text{ if } x_q - z_p < C_{ij} < z_q - x_p \end{cases}$$

and λ is such that (this is an approximate λ found by linear interpolation).

$$\begin{aligned} C_{ij} &= (1-\lambda)(x_q - z_p) + \lambda(z_q - x_p) \\ \lambda &= \frac{C_{ij} + (z_p - x_q)}{(z_q - x_p) + (z_p - x_q)} \\ 0 &\leq \lambda \leq 1 \end{aligned}$$

At this stage the links are being allocated to bands rather than to specific frequencies and so the search space is k^N, where k is the number of bands and N is the number of links. However, since the algorithm used to allocate links to the bands is greedy, the time complexity for this stage is bounded by $O(Nk * \frac{C}{N}) = O(kC)$ and in practice requires negligible time.

Stage 3

For this stage the links were assigned a frequency from within their allocated band. The aim of this stage was to obtain a good assignment based on the partitions $F_1, F_2, ..., F_k$ that could be used as an initial assignment for the metaheuristic improvement stages. A greedy algorithm was used. The strategy was to place a link in the band in such a way that, whenever possible, a constraint with a link outside the band was guaranteed to be satisfied. Only after this step had been completed were links with no outside commitments considered. The links with commitments outside the band were considered first as these were more likely to be links involved in co-site constraints. Co-site constraints are more difficult to solve because of the larger frequency separations required. The

greedy approach lacks any look ahead facility and so large costs are incurred towards the end of the search. As only one pass is made the links that are assigned frequencies last are more likely to cause constraint violations owing to the highly connected nature of the graph. The greedy algorithm rarely provides low-interference assignments. However the assignment will, in the majority of cases, be better than a random initial assignment(see Chapter 5 section 5.5).

Stage 4

There are now k frequency assignment subproblems — each subproblem has approximately $\frac{N}{k}$ links to be assigned frequencies from a band of size approximately $\frac{|F|}{k}$. This gives a search space of size $(\frac{|F|}{k})^{\frac{N}{k}}$. A steepest descent algorithm, described in Chapter 4 section 4.3.2, is applied to each band in turn to improve on the initial assignment. Since the initial solution is likely to be far from optimal a steepest descent algorithm is used to improve the solution rapidly. In the early stages of optimising, it is wasteful to use an algorithm which is 'expensive' in terms of the amount of calculations done per iteration. Improvements are easily found and by using the incremental objective function the total time spent in steepest descent is very small (approximately 1 second) whilst still yielding high improvement.

Stage 5

The strategy now uses a local improvement stage trying to solve each subproblem in turn using the chosen metaheuristic: simulated annealing or tabu search. When generating a move only links and frequencies allocated to the band under consideration may be used. The implementation modules developed previously (described in Chapter 7) were used for both the local (stage 5) and global (stage 6) improvement stages.

Stage 6

The links were originally assigned to bands using a greedy algorithm in stage 3. It is possible that links were placed in bands where constraint violations were unavoidable in order to even out the number of links in each subproblem. Achieving subproblems with similar search space sizes was the primary aim of stage 3. Now that the links have

been assigned frequencies and a more accurate objective function is being used, it is important not to limit the search by decisions made in the earlier stages with incomplete knowledge. For this reason a second application of the chosen metaheuristic is used, where this time the whole scenario is acted on thereby allowing a candidate frequency to be any value in the whole set F.

Stage 7

If either of stages 5 or 6 improved the quality of the best solution found so far then the algorithm is repeated from stage 5. During the global improvement stage (stage 6) links could be assigned any frequency from the whole set F. Therefore a link originally allocated to band X may no longer be assigned a frequency falling in that range. It is necessary to update the data structures holding information about which links are allocated to which bands. Also as a consequence of the global stage, the number of links allocated to each of the bands may not be approximately equal. Some further adjustment is necessary: the band boundaries are altered so that there are approximately equal numbers of links allocated to each band. Once the data structures have been updated the local-global improvement stages (stages 5 and 6) are repeated.

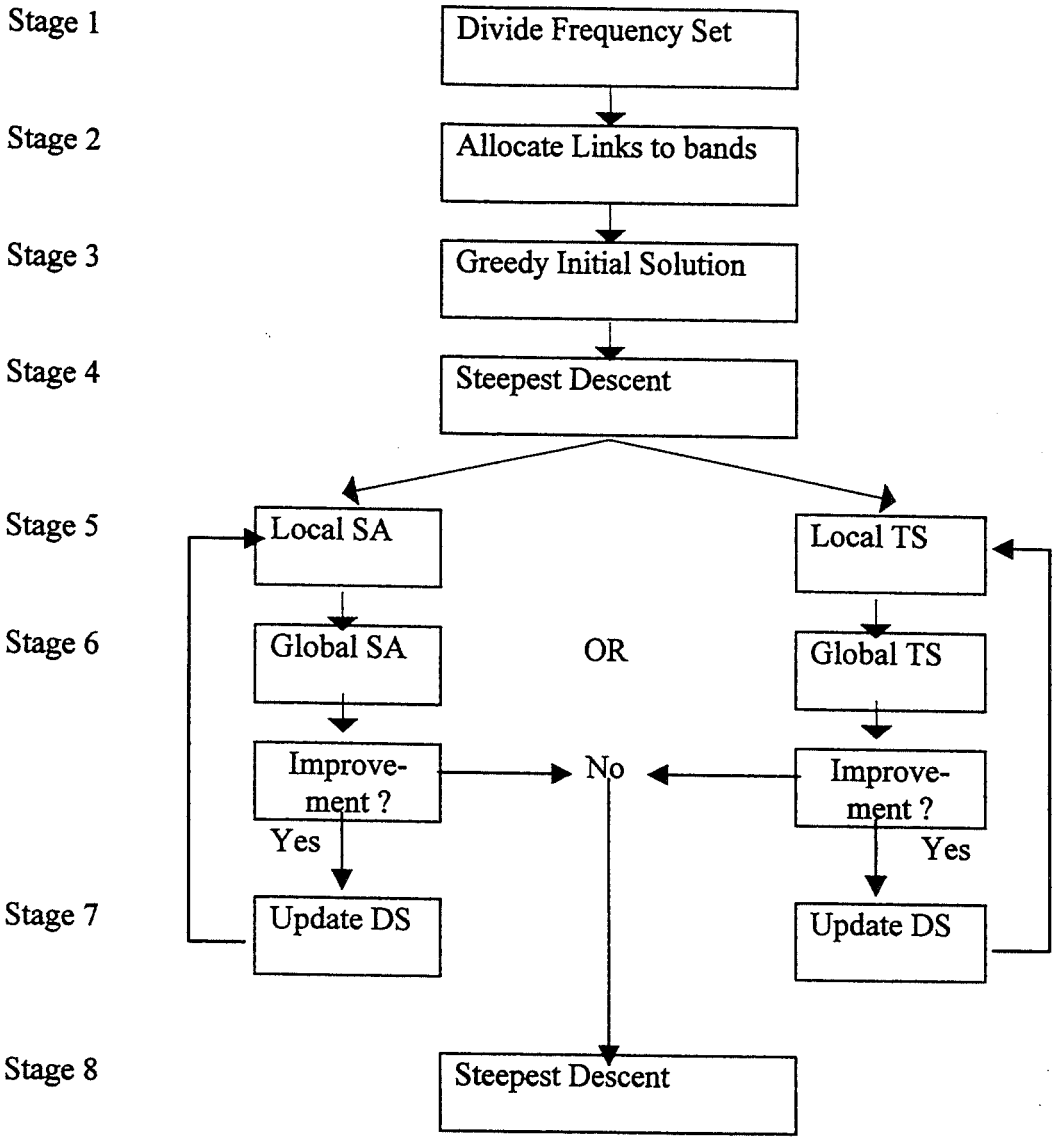
When stages 5 and 6 no longer provide any improvement on the best solution found the loop is terminated and execution continues from stage 8.

Stage 8

Finally a global improvement phase is executed: a steepest descent algorithm is used to improve the final assignment. Candidate links and frequencies were randomly selected from all those available to generate suggested moves.

A diagram representing the stages of the algorithm is given in Fig 35.

Figure 35. Diagram showing DC stages



8.4.4 Alternating Local and Global Phases

The following abbreviations have been used in this, and later, sections :
DCTS=Divide and Conquer algorithm with tabu search in improvement stages
DCSA=Divide and Conquer algorithm with simulated annealing improvement stages

Preliminary results showed that the local-global improvement phases were rarely repeated more than once. On closer inspection it was discovered that no improvement was made after the second application of the local improvement phase and so the loop was removed and a further application of the local-improvement was added to the end of stage 7.

By looking at the graph (Fig 45 page 151) showing the average cost of solutions at the end of each stage averaged over ten runs it can be seen that both DCSA and DCTS obtained almost identical solution costs at each stage. The graph also shows that the average cost of a solution after stage 6 (global improvement stage) is very close to the (optimal) final solution cost. It is well known that algorithms plateau as they approach the optimal solution. Fig 45 shows that, in this case, DCSA obtained the lower cost solutions — there was no improvement during the second local improvement stage (Stage 7) as the optimal solution was obtained at the end of stage 6. Conversely, there was a slight improvement during stage 7 for the DCTS algorithm although this had a noticeably more shallow gradient than the first local improvement stage (Stage 5). A possible reason for the drop in improvement during the second local improvement stage is that the co-site constraints have been satisfied (hence the low cost) by the end of stage 6 (global improvement stage). Both TS and SA are less likely to accept moves which are significantly worse than the current solution towards the end of the search and so the final local stage is used to fine tune the far-site constraints which have a relatively small impact on the total solution cost.

Possible reasons for the ineffectiveness of a second global improvement stage:

A further SA global improvement stage after stage 7 could potentially destroy the current solution unless the starting temperature was sufficiently low to prevent any major deviations from the current cost. Reducing the temperature and restricting the algorithm to moves whose solution cost is similar to the current cost prevents the algorithm from escaping a local minima.

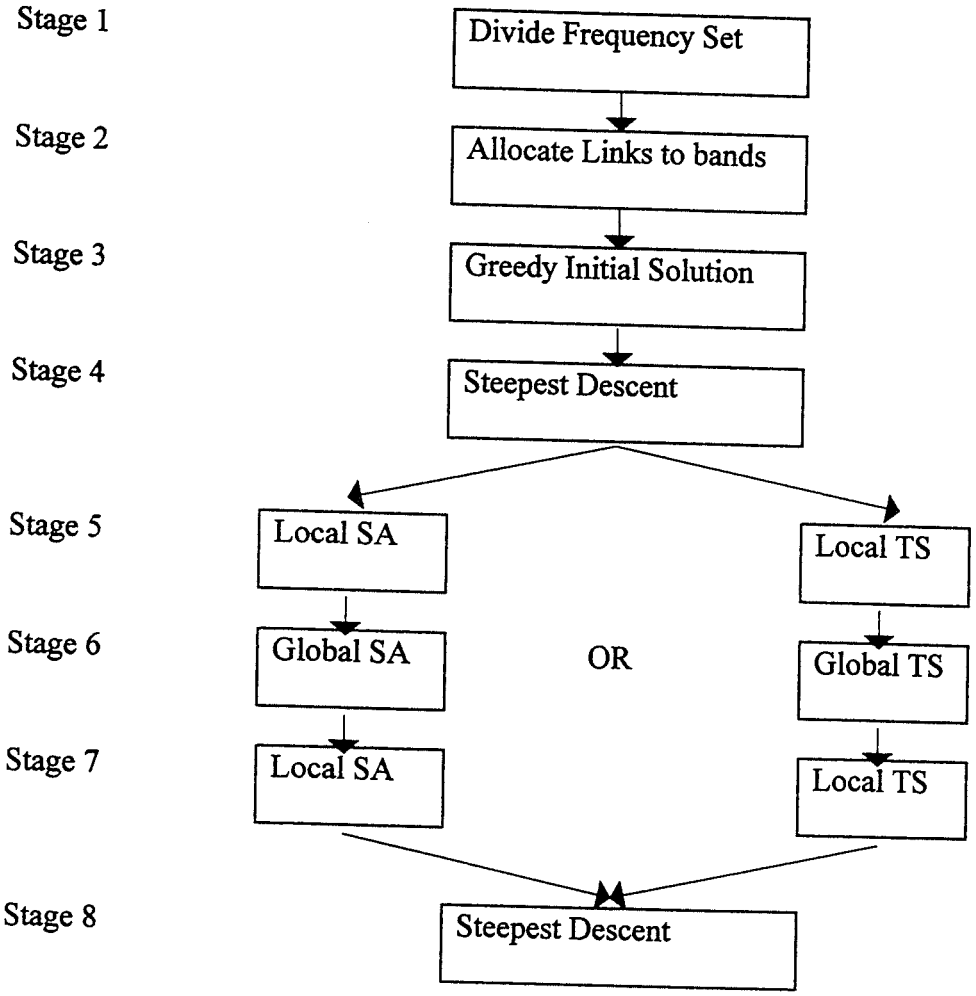
A further TS global improvement stage after stage 7 would consider all possible moves,

assigning a tabu or non-tabu status accordingly and accepting the best move. When the current cost is close to optimal (as it is by this stage), moves are likely to concentrate on eliminating the remaining far-site constraint violations.

Owing to the observed behaviour of the two metaheuristics when a further global stage was applied, it was decided that a steepest descent algorithm would behave similarly and take less time.

The resulting change to the figure previously presented is given in Fig 36.

Figure 36. Diagram showing DC stages - revised



8.5 Results

The TNET scenarios were provided by D.E.R.A., but it is not known whether zero interference assignments are possible for all of the data sets. The results provided in this thesis show that zero interference assignments have been found for 80% of the scenarios. For ease of comparison between the benchmark results (in Chapter 7) and the divide and conquer results (given here) the graphs are again divided into Group A (known zero-cost solutions) and Group B (no known zero-cost solutions).

The two implementations described above were tested using the 46 TNET scenarios. Both algorithms were given an upper time limit of approximately 780 seconds (13 minutes) but each had several termination criteria which meant that the full time allocation was not always required. Each scenario was solved using 10 different seeds for the random number generator. For each scenario the frequency set and initial solution were constant over all runs.

Graphs are given to show the average cost of solutions obtained for the simulated annealing and tabu search implementations of the divide and conquer algorithm (Graphs 37 and 41). Graphs showing the average time taken are also given (Graphs 38 and 42).

Graphs comparing the cost of the best solution found by each of the divide and conquer algorithms is also given (Graphs 39 and 43) along with the time taken to obtain that best solution (Graphs 40 and 44).

The results used to create these graphs are given in Appendix b.

Figure 37. Graph showing DCSA vs. DCTS Average Cost (Group A)

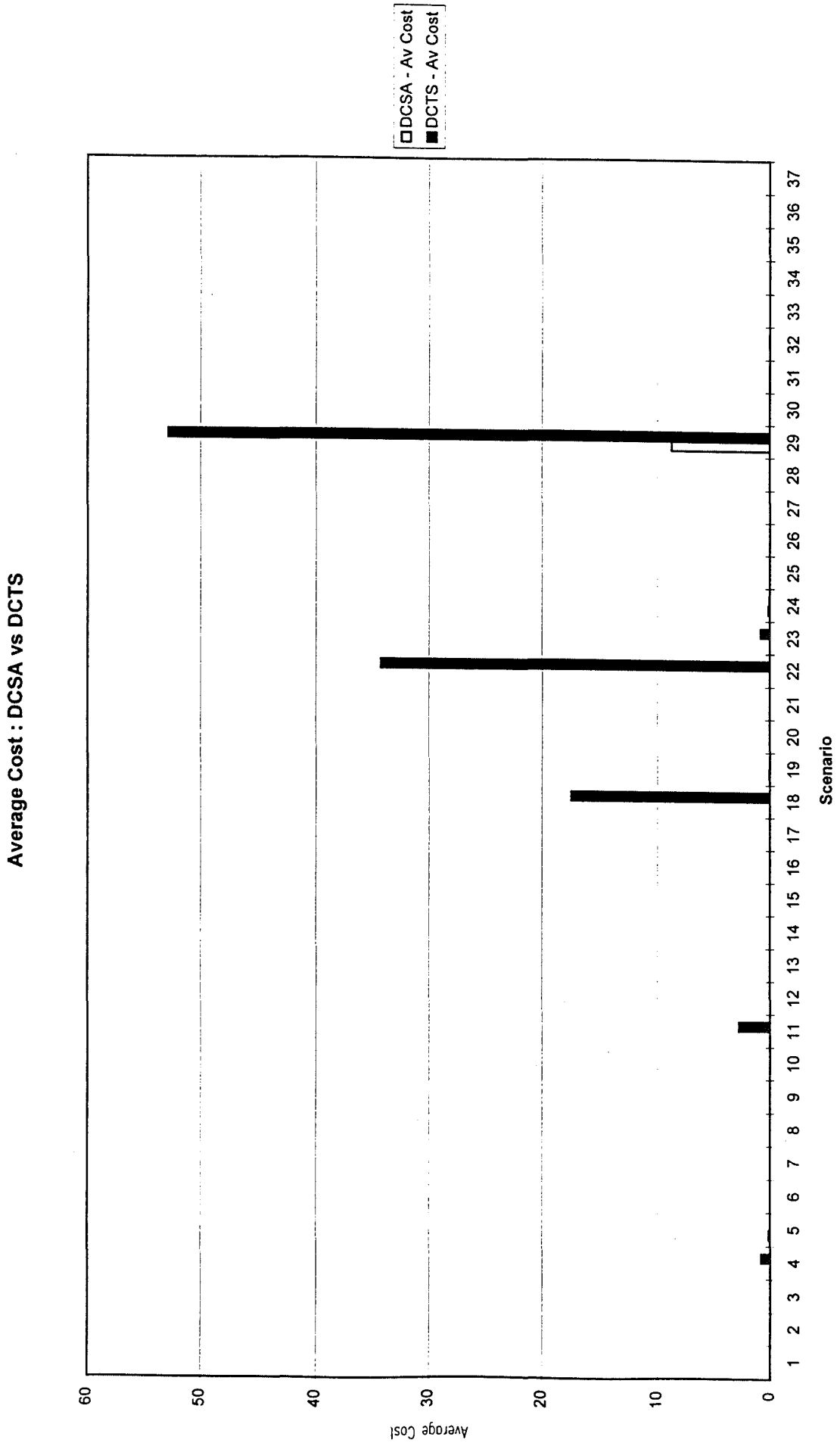


Figure 38. Graph showing DCSA vs. DCTS Average Time (Group A)

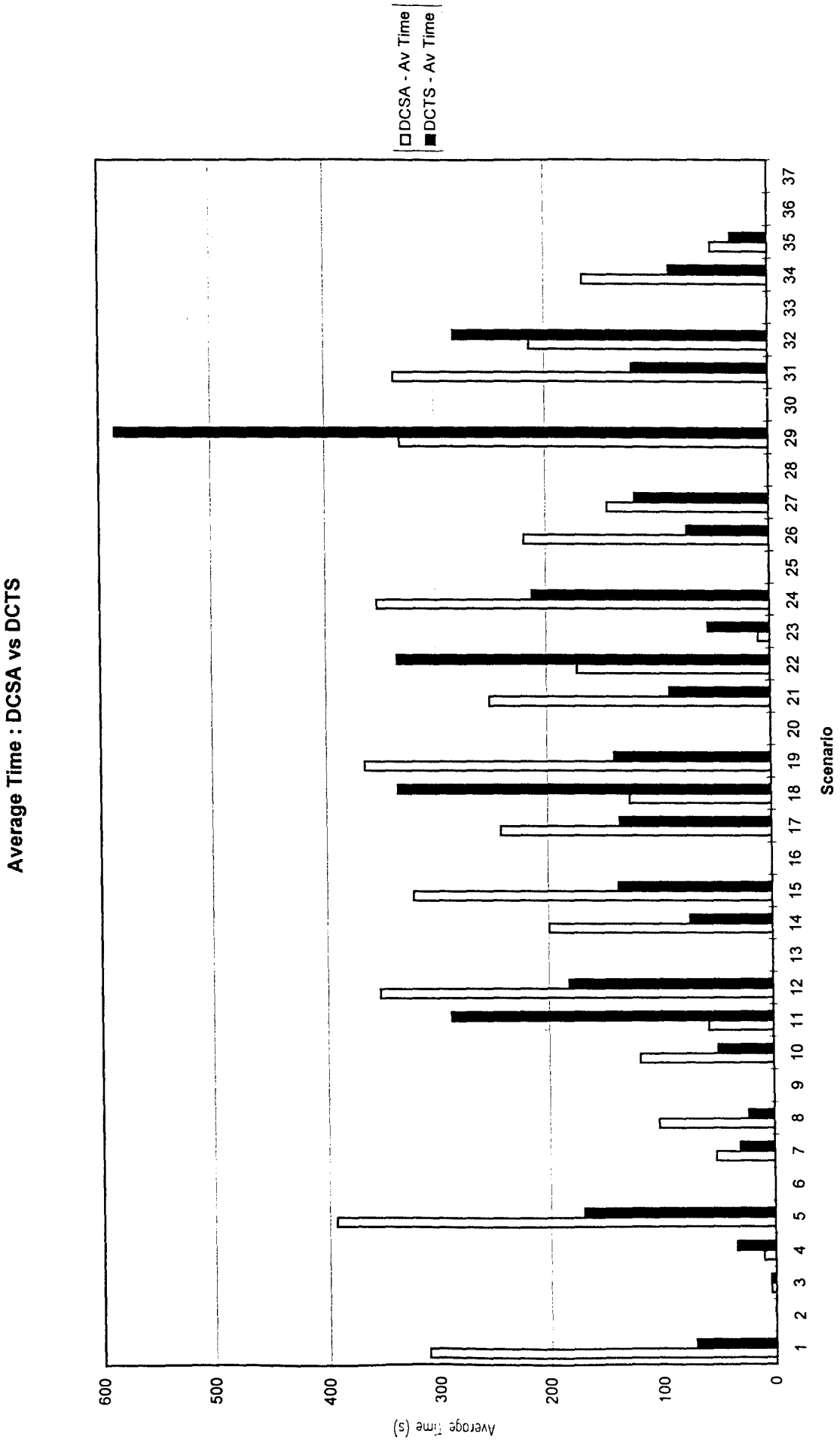


Figure 39. Graph showing DCSA vs. DCTS Best Cost (Group A)

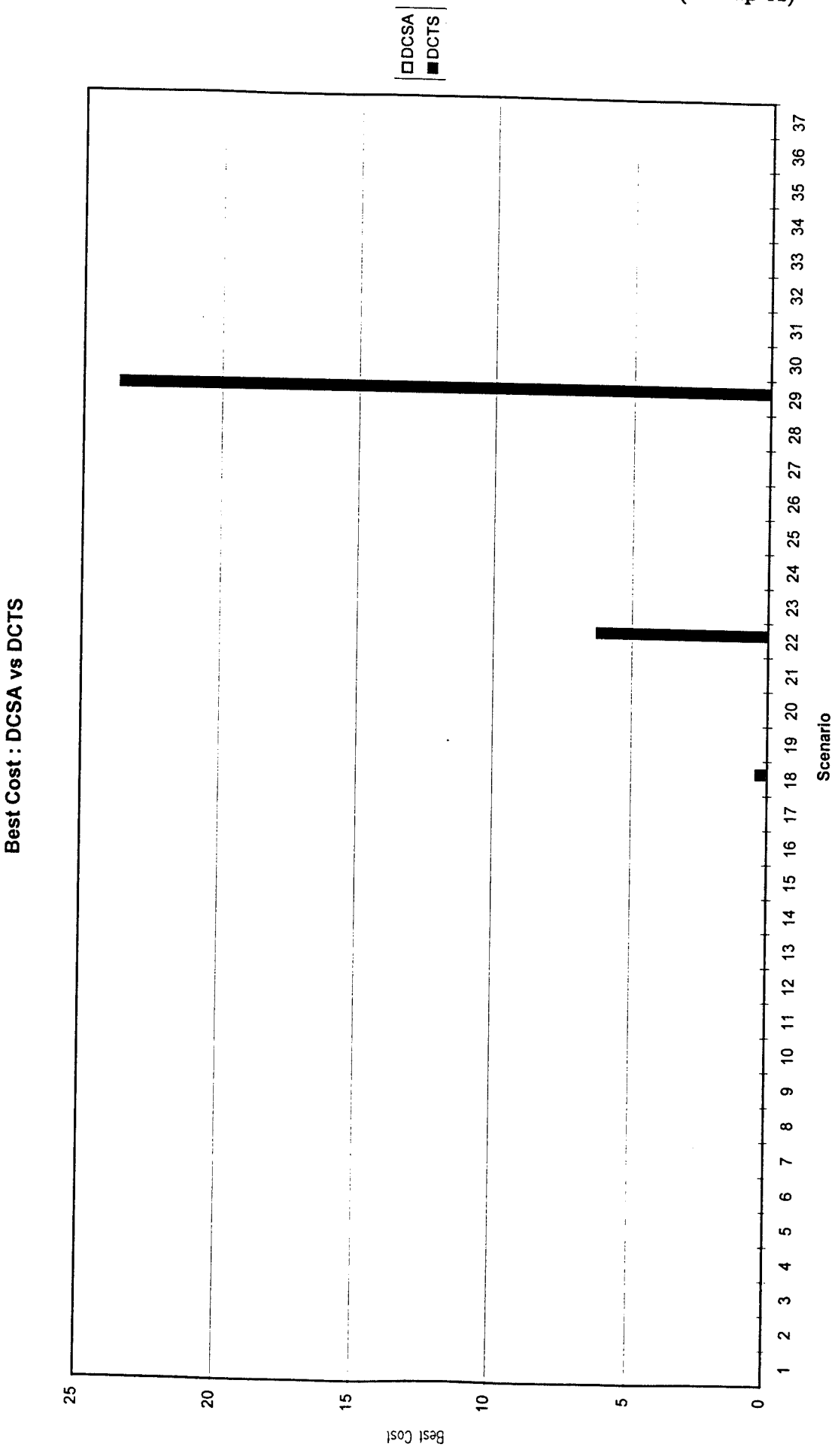


Figure 40. Graph showing DCSA vs. DCTS Time to Best Cost (Group A)

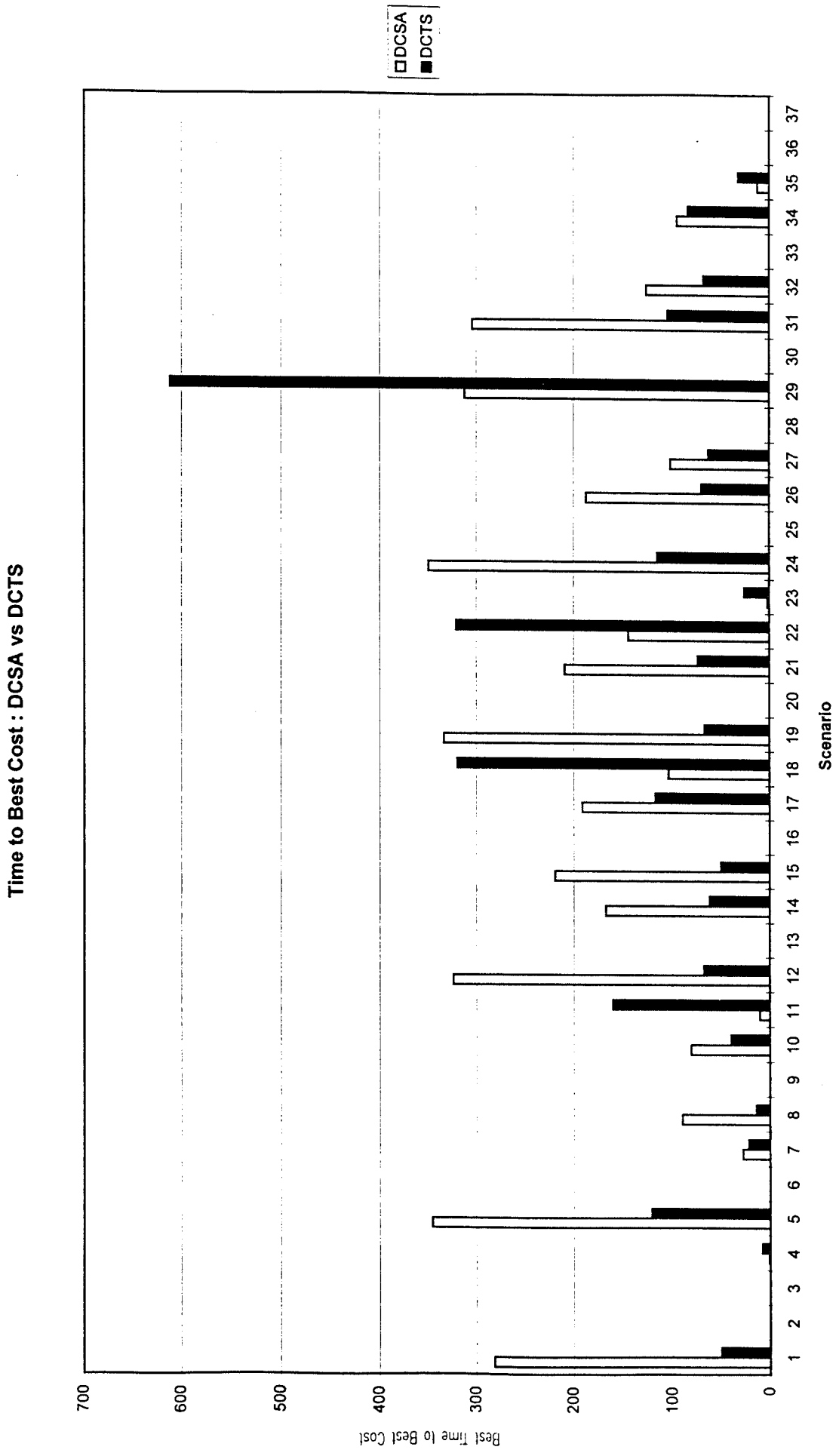


Figure 41. Graph showing DCSA vs. DCTS Average Cost (Group B)

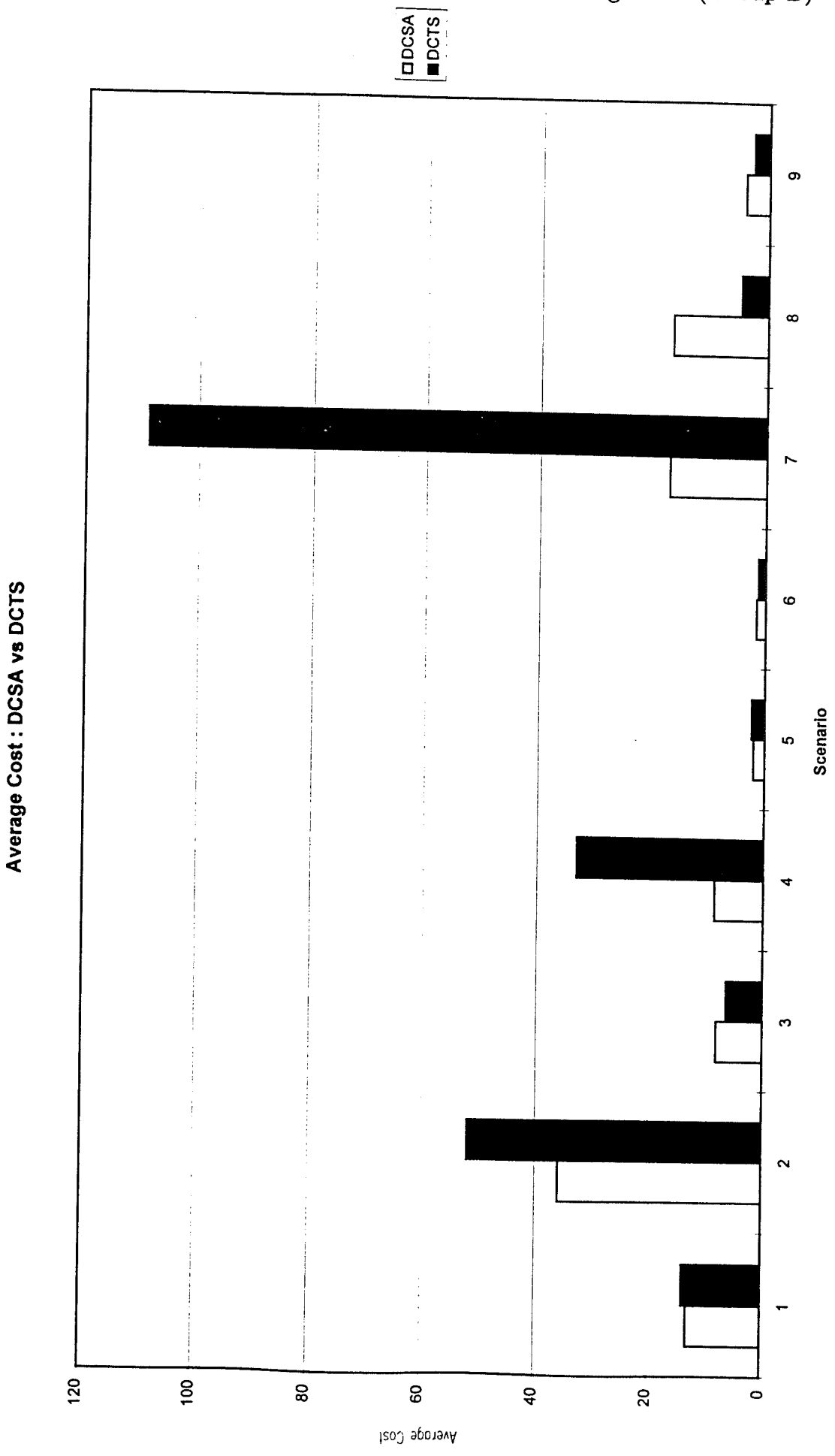


Figure 42. Graph showing DCSA vs. DCTS Average Time (Group B)

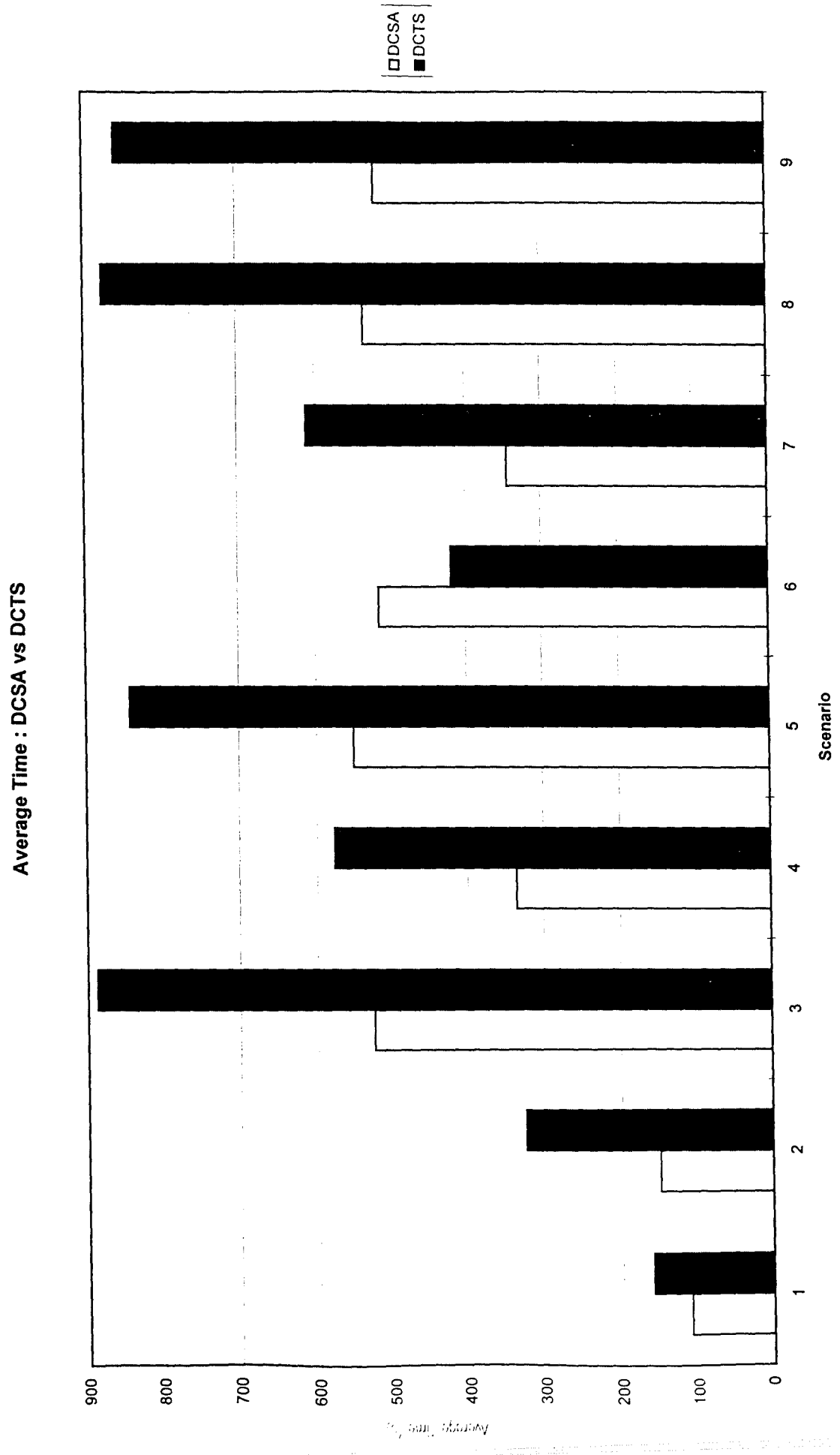


Figure 43. Graph showing DCSA vs. DCTS Best Cost (Group B)

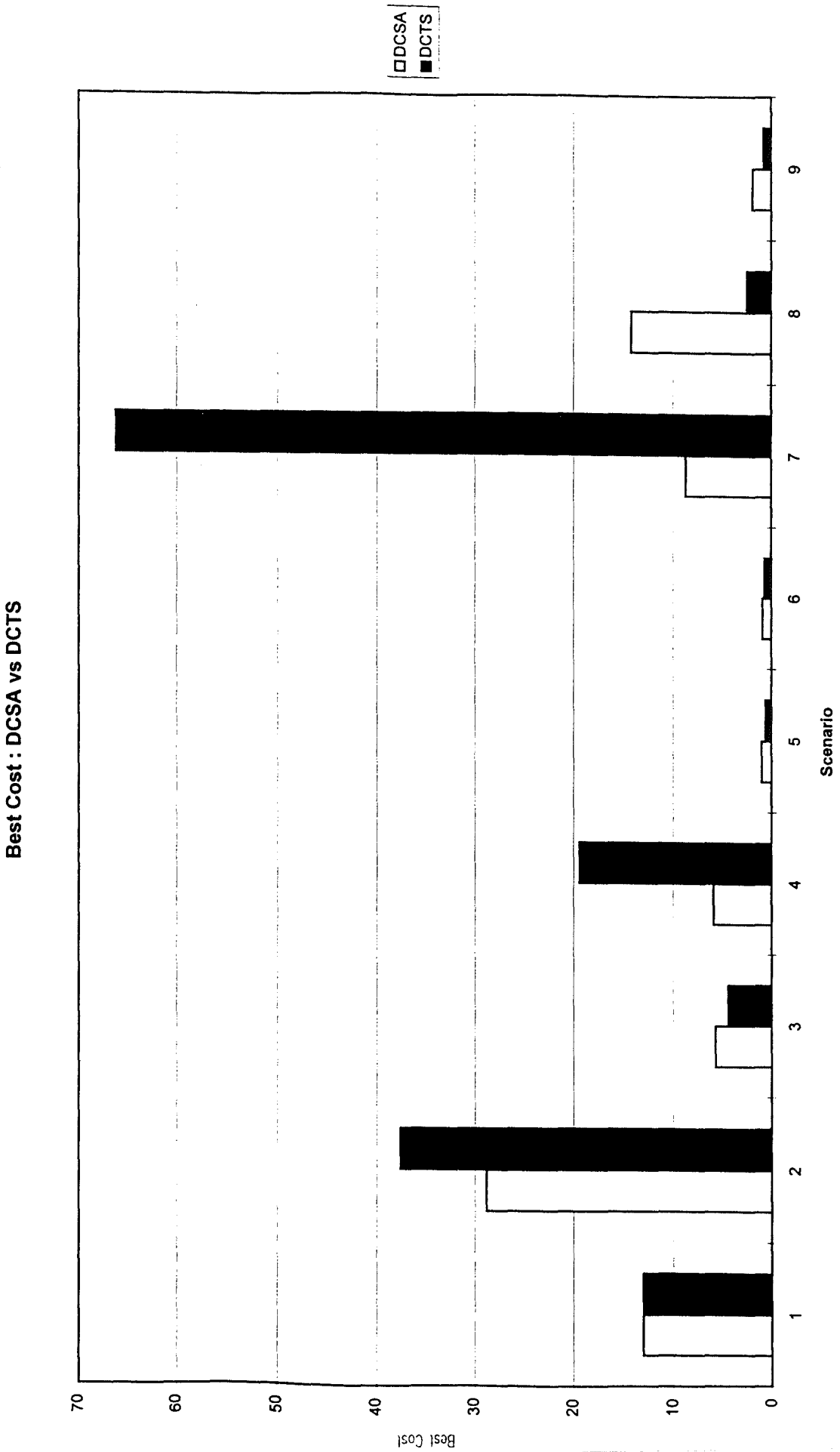
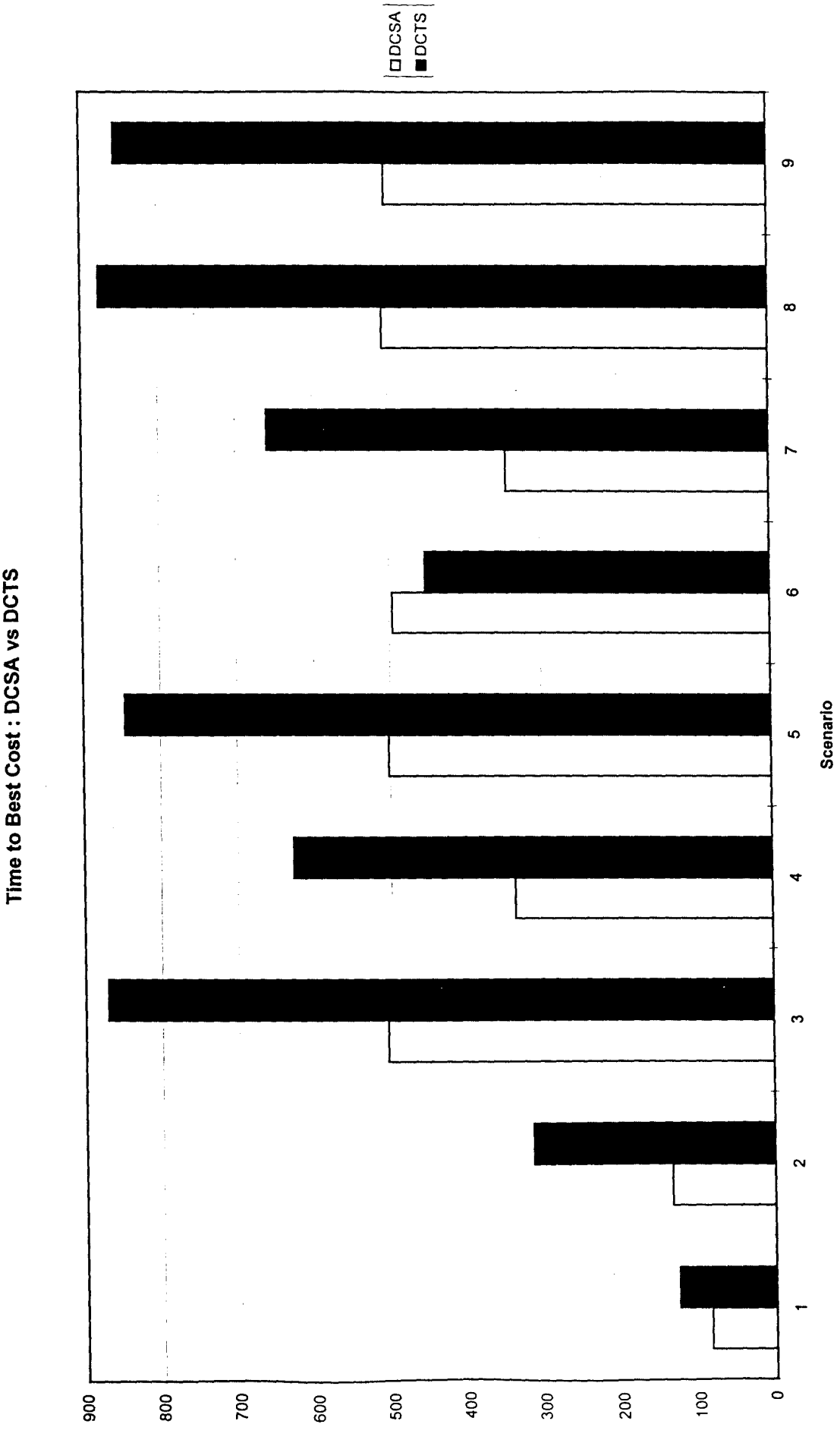


Figure 44. Graph showing DCSA vs. DCTS Time to Best Cost (Group B)



8.6 Discussion

8.6.1 Group A Data

By looking at the average cost graph for these scenarios (Fig 37) it can clearly be seen that the DCSA algorithm obtained lower cost solutions than the DCTS algorithm for 6 scenarios (4, 11, 18, 22, 23 and 29). These were the same six scenarios which TS also found difficult (Chapter 7 section 7.5.1). For the remaining scenarios the costs were zero (or negligible hence not showing on the graph). DCSA obtained optimal solutions to 81% of the scenarios and DCTS obtained optimal solutions to 78% of the scenarios in the average case.

By looking at the average time graph (Fig 38) for the scenarios for which DCTS did not obtain an optimal solution it can be seen that the DCSA algorithm obtained its superior solutions in less time than DCTS obtained its inferior solutions. The DCTS algorithm obtained lower cost solutions more quickly than DCSA for all other scenarios except scenario 32 for which both implementations obtained near-optimal solutions.

The best cost solutions (Fig 39) for these scenarios saw a marked reduction in the cost of the solutions. DCSA optimally solved 100% of the scenarios whereas DCTS obtained zero cost solutions for 34/37 of the scenarios (18, 22 and 29 not solved optimally). For those scenarios which DCTS did not solve optimally the time taken was longer than that for DCSA to find an optimal solution.

The graph showing the time to reach these best cost solutions (Fig 40) had the same trend as the average times; the scenarios solved optimally on average by DCTS were generally solved more quickly than the corresponding DCSA attempts. The scenarios not solved optimally on average by DCTS had optimal solutions found more quickly by DCSA.

The percentage range and percentage frequency values were comparable for both algorithms. The solutions for 24 of the 37 scenarios used 100% range, the remaining scenarios used 80-91% of the available range. The six scenarios which DCTS did not solve optimally on average also had solutions using 100% range for both algorithms.

Sixteen scenarios used approximately 100% of available frequencies, the remaining scenarios varied in the percentage of the available frequencies used, from 10-90%. For

those scenarios not solved optimally by DCTS on average the lower cost solutions were found using marginally fewer frequencies by DCSA.

Summary

- DCSA obtained optimal solutions to 81% of the scenarios on average and 100% in the best case.
- DCTS obtained optimal solutions to 78% of the scenarios in the average case, for all of these scenarios DCTS obtained its solutions far quicker than the corresponding DCSA attempt.
- DCTS obtained solutions for 92% of the scenarios in the best case. For the 3 remaining scenarios DCSA obtained lower cost solutions in less time than DCTS.
- On average the DCSA algorithm obtained lower cost solutions than the DCTS algorithm for 6 scenarios (4, 11, 18, 22, 23 and 29) and these solutions were also obtained in less time and using marginally fewer frequencies than the corresponding DCTS attempt.
- The average and best time graphs exhibited the same pattern.
- The percentage range and percentage frequency values were comparable for both algorithms.
- 24 of the 37 scenarios used 100% range, the remaining scenarios used 80-91% of the available range.
- Sixteen scenarios used approximately 100% of available frequencies.

8.6.2 Group B Data

By looking at the graph of average cost (Fig 41) it can be seen that the DCSA algorithm obtained lower cost solutions than DCTS for 5 of the 9 scenarios (scenario 6 has a comparable cost solution for both algorithms but is marginally less for DCTS). If these five scenarios are cross-referenced with the average time graph (Fig 42) it can be seen that DCSA obtains these lower cost solutions in less time than DCTS obtains its inferior solutions.

For scenario 6 both algorithms obtained similar average cost solutions; DCTS arrived at its marginally lower cost solutions more quickly than the solutions found by DCSA. For the remaining scenarios DCTS obtained lower cost solutions on average than DCSA. It can be seen that DCTS also ran for approximately five minutes longer on average. DCSA stopped after approximately the same time for all three scenarios. This may imply that the DCSA algorithm had finished its cooling schedule and therefore terminated prematurely. It would be interesting to see if altering the parameters of the cooling schedule enabled the algorithm to continue for longer and obtain lower cost solutions. However, because of the time limitations (maximum run-time of 15 minutes) imposed by the D.E.R.A this was not investigated. DCSA also stopped at approximately the same time for scenario 6.

The trends described above for the average cost and average time graphs also hold for the best cost and time to best cost graphs (Figs 43 and 44) without exception. The values for best cost are smaller and the time taken to reach the best solutions is comparable to the average time taken. The scenarios for which DCSA obtained higher cost solutions than DCTS had a consistent run-time, less than the time used for DCTS to obtain its superior solutions.

All scenarios used 100% range available for both algorithms. Scenarios 1 and 2 used 85-90% of the available frequencies and the remaining scenarios used approximately100% of frequencies. Since these scenarios are more difficult to solve it is expected that all available resources would be used.

8.6.3 Summary

- DCSA obtained lower cost solutions in less time than DCTS for 5 of the 9 scenarios.
- For scenario 6 both algorithms obtained similar average cost solutions; DCTS arrived at its marginally lower cost solutions more quickly than the solutions found by DCSA.
- For the remaining scenarios DCTS obtained lower cost solutions (taking approx 5 minutes longer) on average than DCSA. DCSA stopped after approximately the same time for all three scenarios. DCTS exceeded the 13 minute allowance for several scenarios.

- All scenarios used 100% range available for both algorithms.
- Scenarios 1 and 2 used 85-90% of the available frequencies the other scenarios used approximately100% of frequencies.

8.7 Conclusions

For the group A scenarios there were 6 scenarios which TS did not solve optimally on average. For these scenarios lower cost solutions were found by DCSA more quickly than the corresponding DCTS attempt. DCSA solved all of the group A scenarios optimally in the best case, DCTS optimally solved 92% of scenarios in the best case. For the remaining 27 scenarios DCTS obtained lower cost solutions more quickly than DCSA on average.

For Group A : Both DCSA and DCTS used comparable resources in obtaining the solutions, for the majority of scenarios almost the full range and number of frequencies were used. There were 13 scenarios which were solved in less than one second by both algorithms and the solutions for these scenarios used fewer resources.

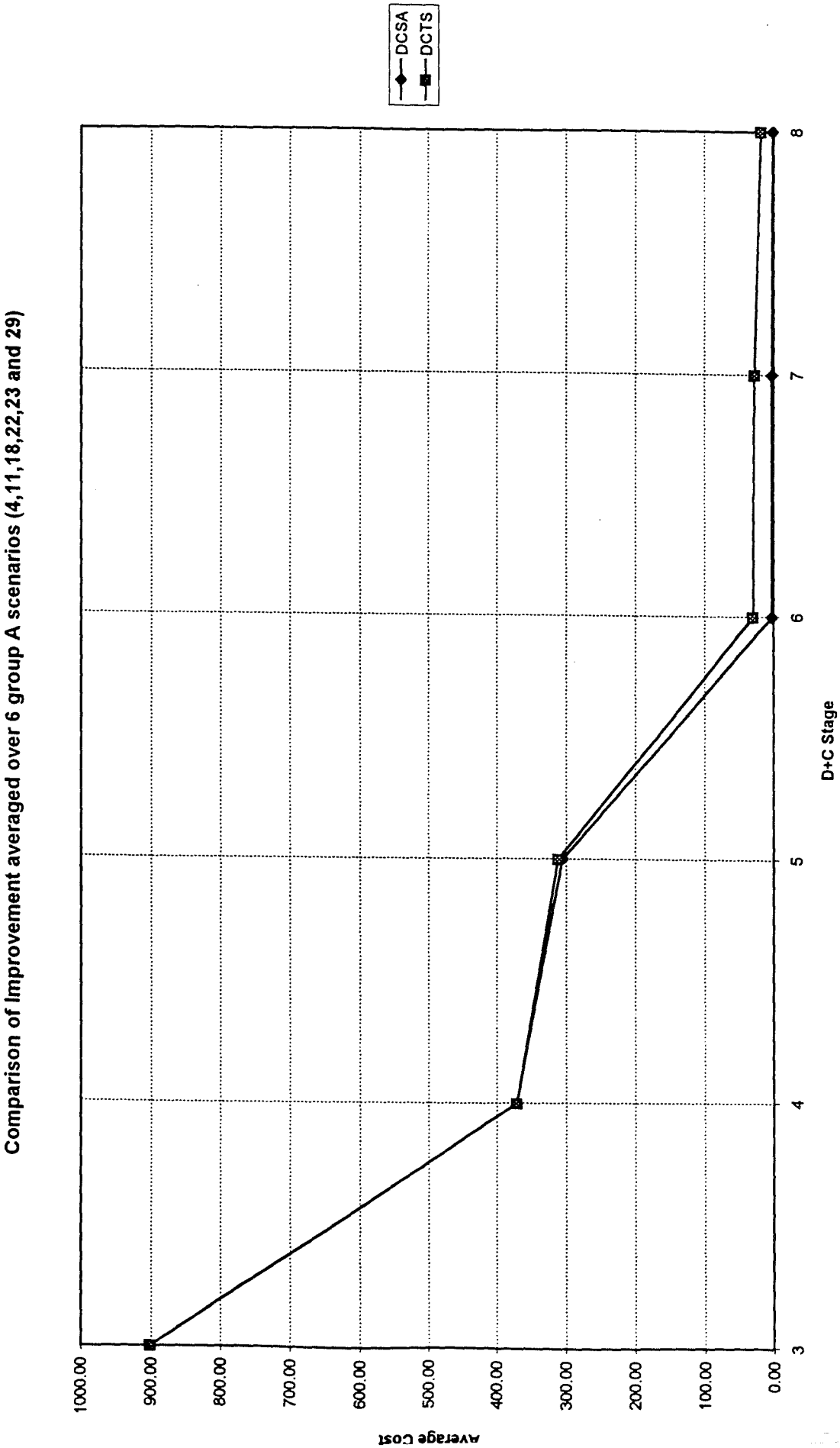
For the group B scenarios almost all resources were used in the solution of all scenarios, the algorithms also ran for longer than the group A scenarios, this was expected as the group B scenarios are harder to solve. DCSA and DCTS each obtained the lowest cost solutions for half of the scenarios. However, DCSA appeared to stop consistently prematurely for those scenarios for which DCTS obtained a lower cost. This could imply that the DCSA algorithm had finished its cooling schedule and therefore terminated before the maximum run time.

Considering those scenarios for which both implementations obtain zero cost solutions: DCTS converges very quickly to an optimal solution and finds its solutions in significantly less time than DCSA. A possible explanation for this observation is as follows: once in a fruitful area of the search space the tabu search algorithm will move efficiently towards the local minimum whereas simulated annealing may make a move to a different area if the temperature is sufficiently high.

There are six scenarios in group A which are solved optimally by DCSA on average but for which DCTS struggles to obtain low cost solutions. DCTS optimally solves three of these scenarios in the best case. Fig 45 shows the improvements made at

each stage for both implementations averaged over the six scenarios (4, 11, 18, 22, 23 and 29). Clearly the paths are identical to the end of stage 4 (same initial solution followed by deterministic steepest descent algorithm). For each of the improving stages DCSA obtains a slightly better solution than DCTS ultimately leading to the difference between obtaining an optimal or non-optimal solution.

Figure 45. Graph showing DCSA vs. DCTS Stage Improvements



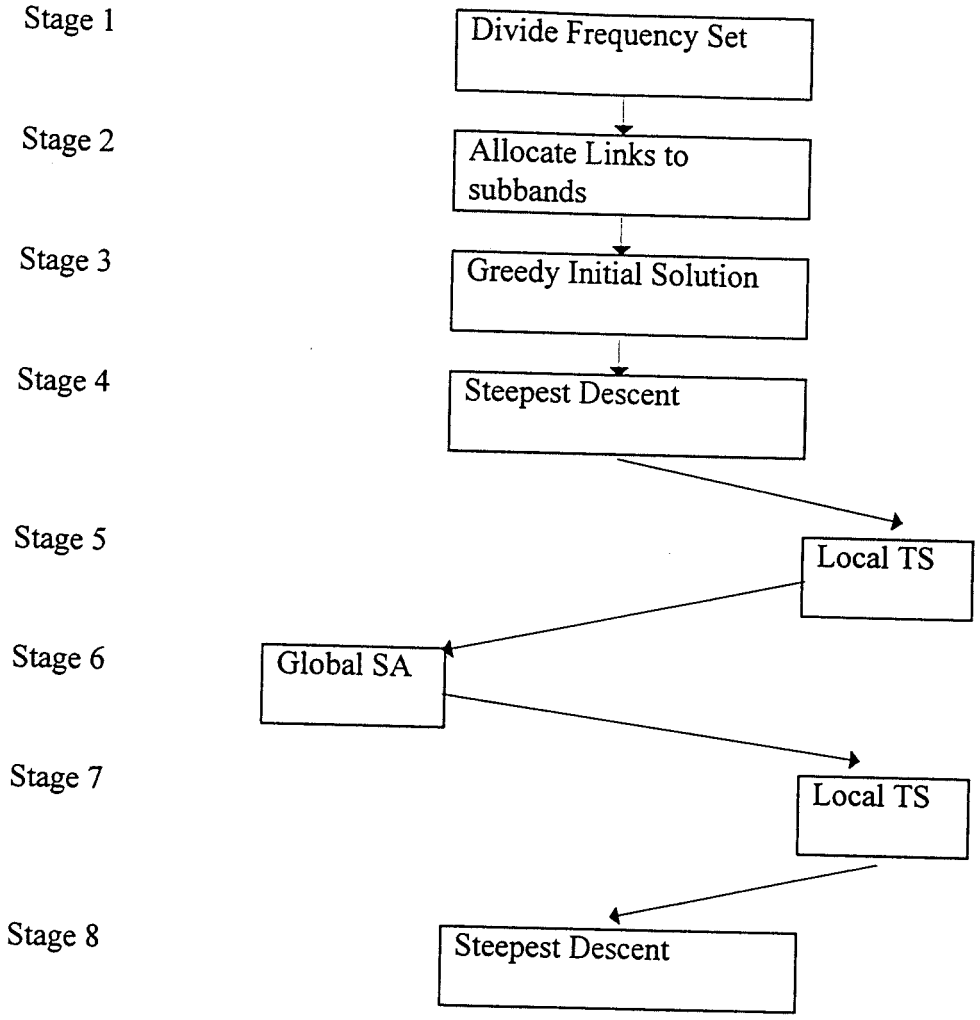
The short time available for solving the group B scenarios appears to be the most limiting factor in the search for good solutions. This is further aggravated by the early termination of the DCSA algorithm. Unless both algorithms run for the same amount of time it is difficult to assess their relative success on this subset of the data.

8.8 Further Work

DCTS obtained lower cost solutions than DCSA for some of the Group B scenarios. However, the run time for DCSA for these scenarios appears 'capped'. It is possible that DCSA would obtain similar cost solutions to DCTS if the algorithms ran for comparable times. This could be investigated by altering the parameters of the cooling schedule to enable the algorithm to continue for longer.

Another possible investigation would combine the advantages of DCTS with those of DCSA. DCTS generally obtained lower cost solutions more quickly than DCSA for the smaller group A scenarios. DCSA obtained lower cost solutions than DCTS for the harder (and larger) group B scenarios. A potential hybrid would therefore use the divide and conquer framework but use tabu search during the local improvement stages and simulated annealing during the global improvement stage. This is shown diagrammatically in Fig 46.

Figure 46. Diagram showing DC stages - Further Work



Chapter 9

Effectiveness of Using Divide & Conquer

The simulated annealing (SA) and tabu search (TS) algorithms were fully described in Chapter 7 and the results discussed. The divide and conquer (DC) algorithm was explained in Chapter 8. Results were given for the two implementations: divide and conquer with simulated annealing (DCSA) and divide and conquer with tabu search (DCTS). This chapter is divided into five parts: part one gives the implementation details, part two compares the results for SA with the results obtained by DCSA, part three compares the results for TS with the results obtained by DCTS, and finally, some conclusions are drawn and ideas for further work given.

The following abbreviations have been used in this chapter : SA = Simulated annealing algorithm, TS = Tabu search algorithm, DCSA = Divide and Conquer algorithm with simulated annealing in improvement stages, DCTS = Divide and Conquer algorithm with tabu search in improvement stages.

9.1 Implementation

The full implementation details are given in Chapters 7 and 8 as previously mentioned. For each scenario, all of the algorithms used the same frequency set and the same initial

solution over all runs. The parameters (chapter 8 section 8.3.1) for each algorithm were constant for all 46 TNET scenarios on which the programs acted. The programs were run 10 times each, they all used the same seeds to prime the random number generator. All of the results were obtained on a desktop PC, 100MHz. As many of the variables as possible were the same for all algorithms to enable a fair comparison of the algorithms [Bar et al.95]. Despite these measures there remains some difficulty in comparing the results obtained, these difficulties were given in Chapter 7 section 7.4.1.

The results used to create these graphs are given in Appendix c.

9.2 DCSA vs SA Results

9.2.1 Group A Data

By looking at the graphs showing average cost and time (Figs 47 and 48) it can clearly be seen that the DCSA algorithm obtained lower cost solutions than the SA algorithm for 12 of the 37 scenarios. For the remaining scenarios both algorithms obtained a zero cost solution. It can also be seen that DCSA obtained its superior solutions in significantly less time than SA obtained its inferior solutions.

The graphs of best cost and time to best cost (Figs 49 and 50) exhibit a similar pattern. It is significant that in the best case DCSA obtained zero cost solutions for all of the scenarios whereas SA obtained zero cost solutions for just 31 of the 37. Again it can be clearly seen that DCSA consistently arrived at its solutions more quickly than SA.

The average percentage of range used by both DCSA and SA were identical, again 100% range was used for all scenarios, except the 13 scenarios which were solved in less than one second by both algorithms.

The average percentage of frequencies used by both algorithms was comparable in most cases, for 6 scenarios DCSA used a smaller percentage than SA, for scenario 4 the reverse was true. Smaller percentage frequency values were found for the 13 easiest scenarios.

Figure 47. Graph showing DCSA vs. SA Average Cost (Group A)

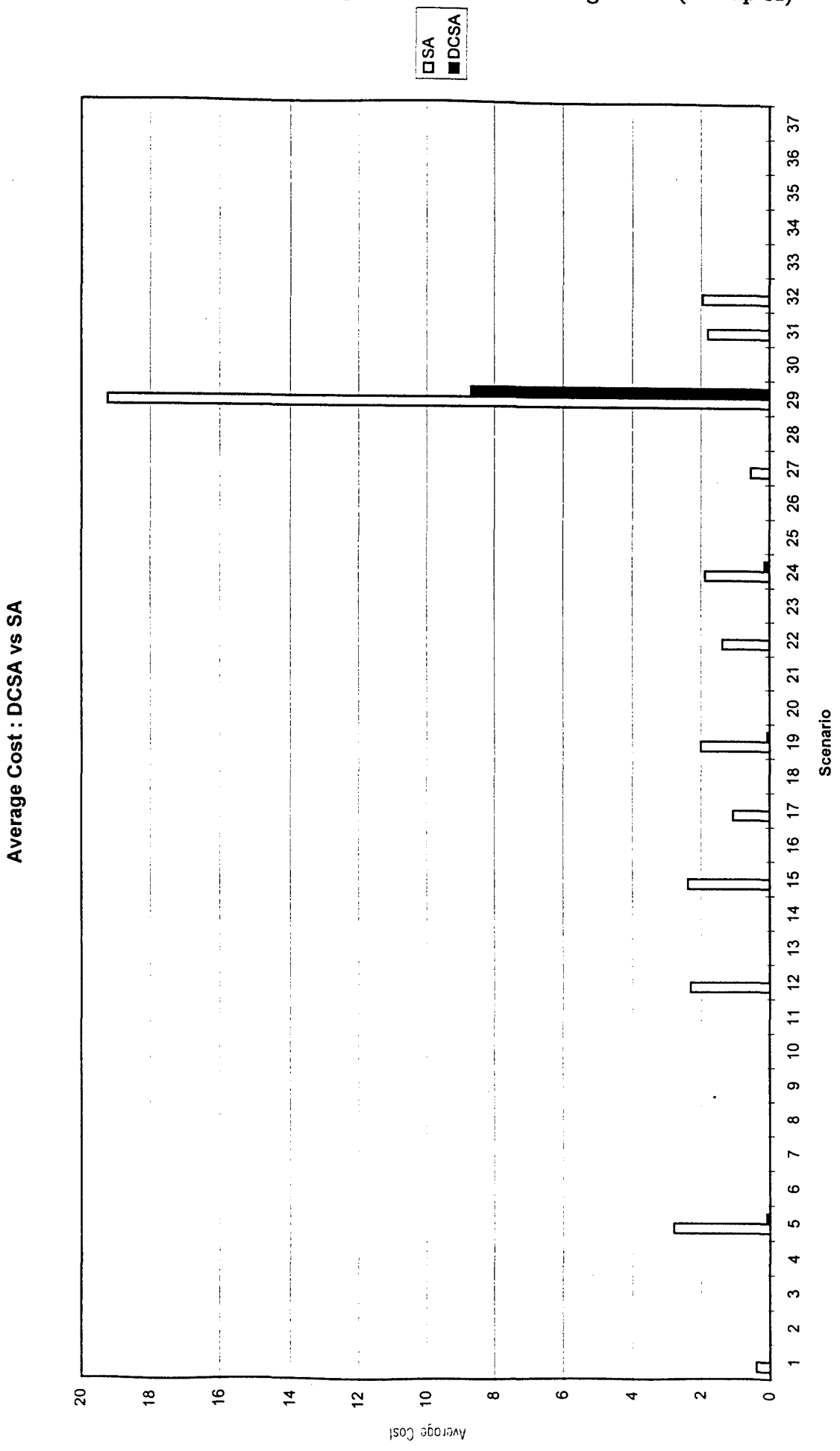


Figure 48. Graph showing DCSA vs. SA Average Time (Group A)

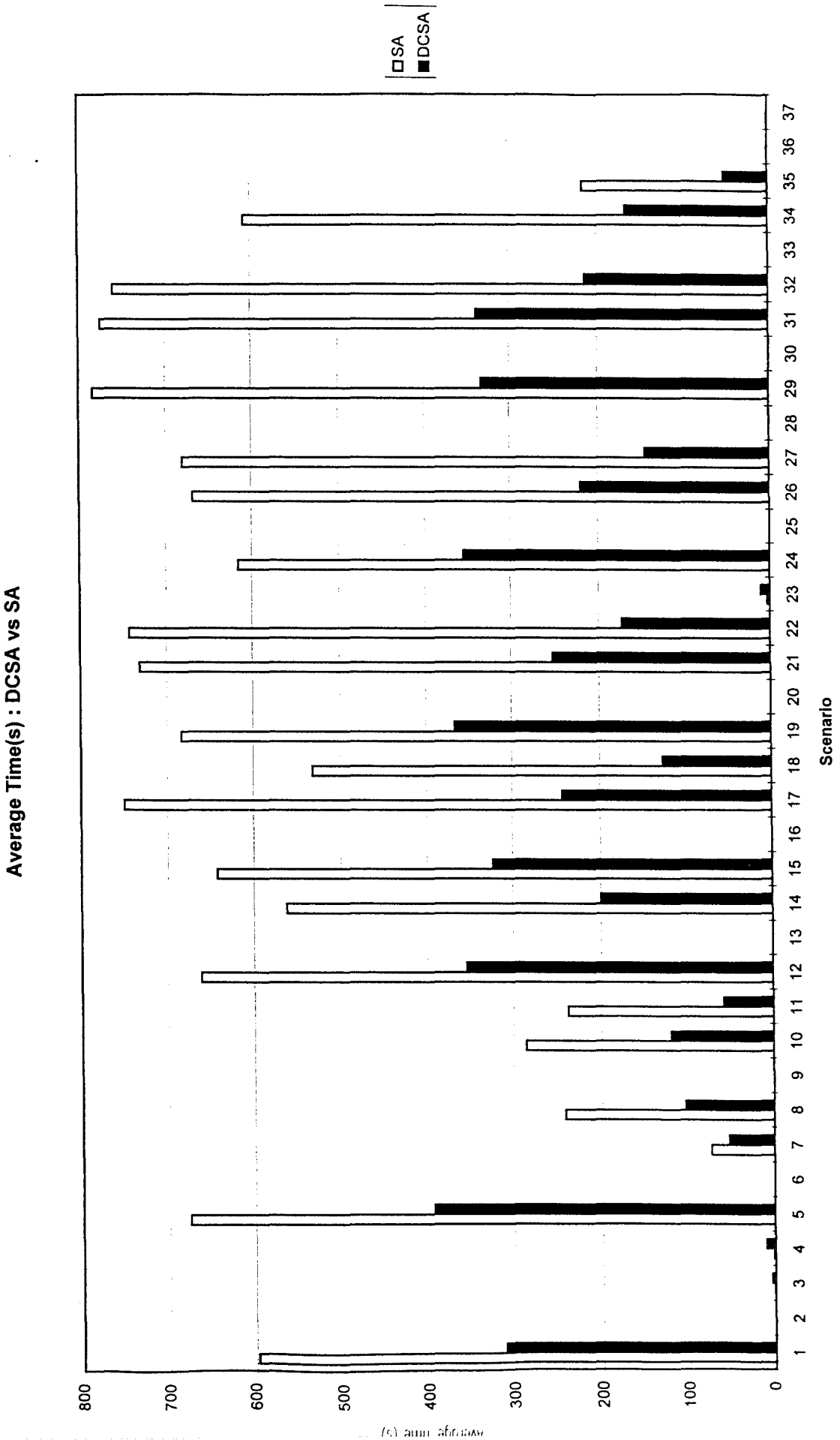


Figure 49. Graph showing DCSA vs. SA Best Cost (Group A)

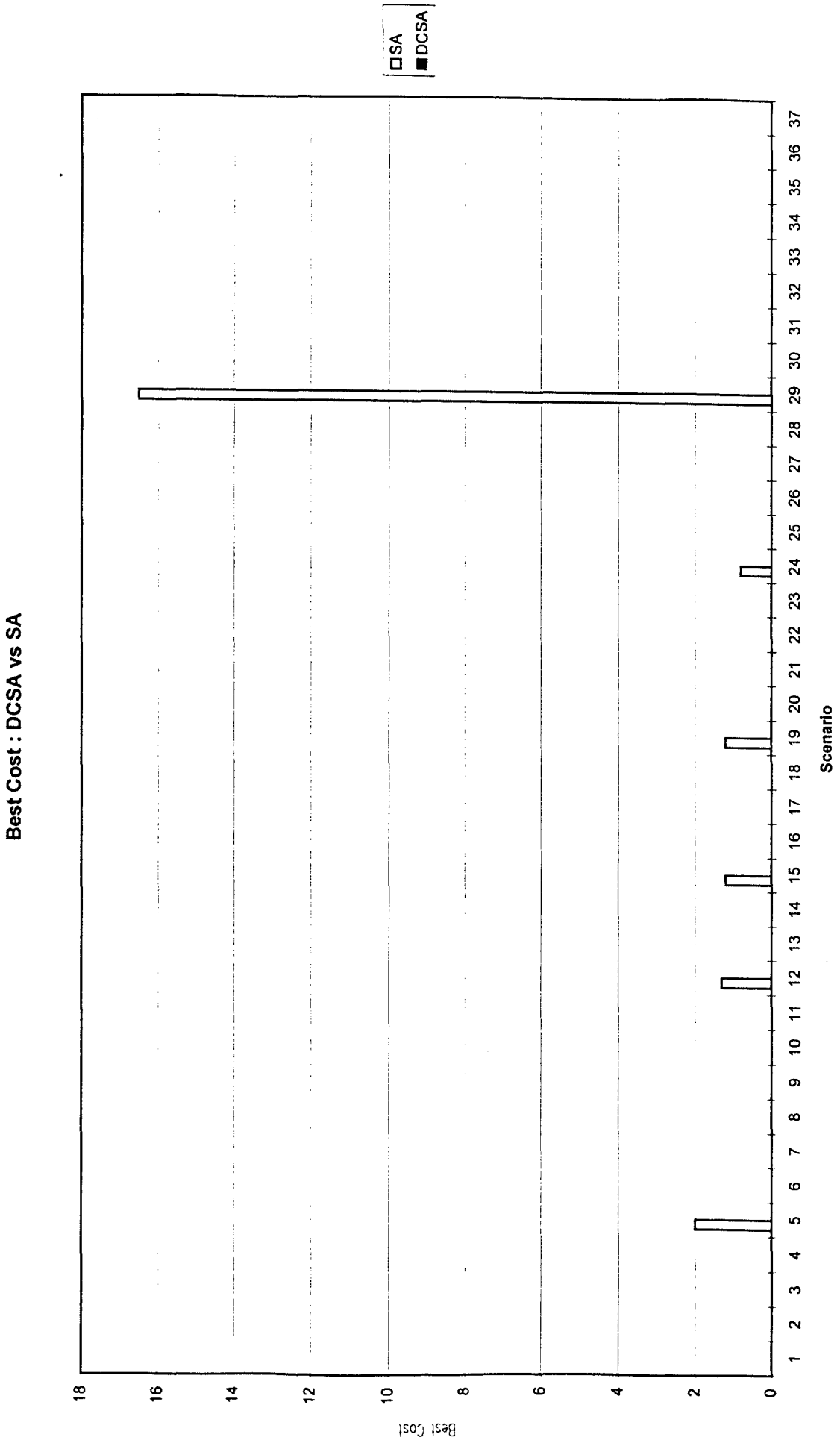
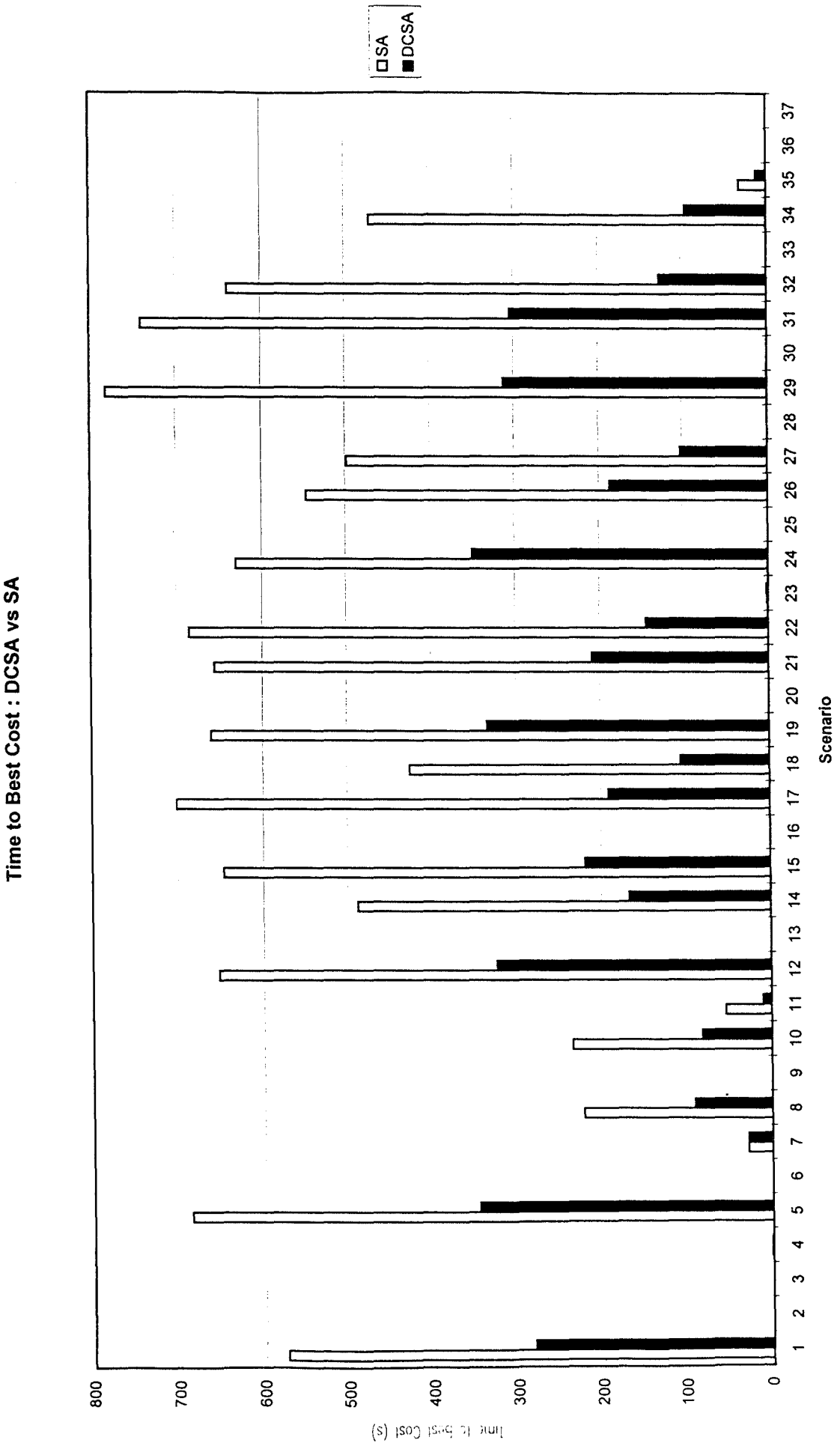


Figure 50. Graph showing DCSA vs. SA Time to Best Cost (Group A)



9.2.2 Group B Data

The graphs of average cost and time (Figs 51 and 52) show clearly that for all scenarios DCSA obtains lower cost solutions than SA, it also obtains these superior solutions in less time.

The graphs of best cost and time to best cost (Figs 53 and 54) reinforce the evidence that DCSA obtains lower cost solutions than SA for all scenarios (the best cost is identical for scenario 1). For all scenarios DCSA uses significantly less time than SA.

All scenarios were solved by both algorithms using 100% range. Sixty-six percent of scenarios used 100% of the available frequencies with the remaining scenarios (1 and 2) using 85-90% frequencies.

To summarise: despite the additional overhead of the DCSA algorithm it obtains superior solutions in less time than SA. This is true for all scenarios (Group A and Group B) in both the average and best case. Both algorithms used comparable resources in reaching their solutions (except DCSA using a lower percent frequencies for Group A for a small number of scenarios). These results are significant and consistent. The DCSA algorithm has been shown to be robust and efficient in its solution of the frequency assignment scenarios presented, furthermore the results are significantly better than those obtained by the sole metaheuristic, SA.

Figure 51. Graph showing DCSA vs. SA Average Cost (Group B)

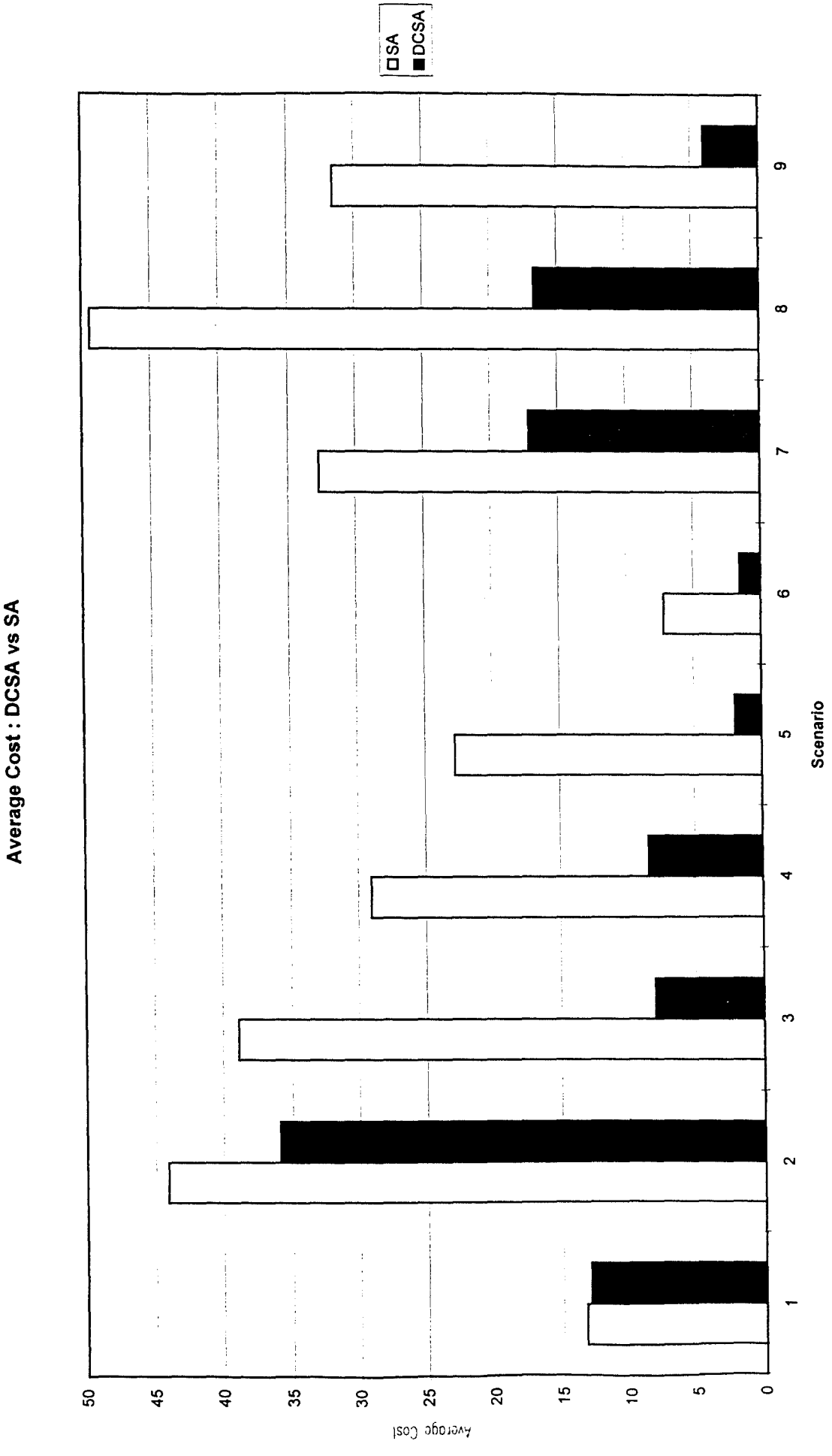


Figure 52. Graph showing DCSA vs. SA Average Time (Group B)

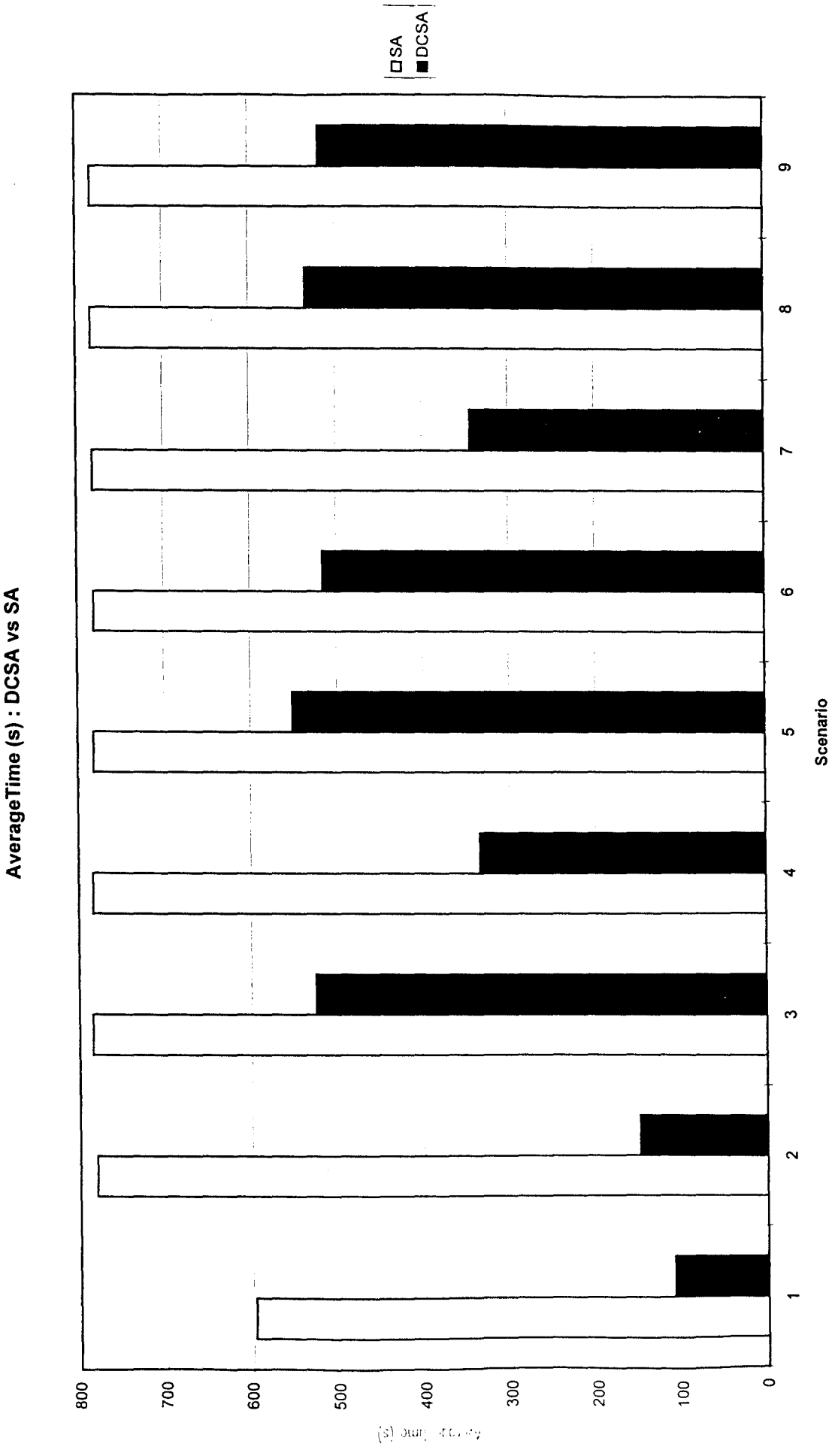


Figure 53. Graph showing DCSA vs. SA Best Cost (Group B)

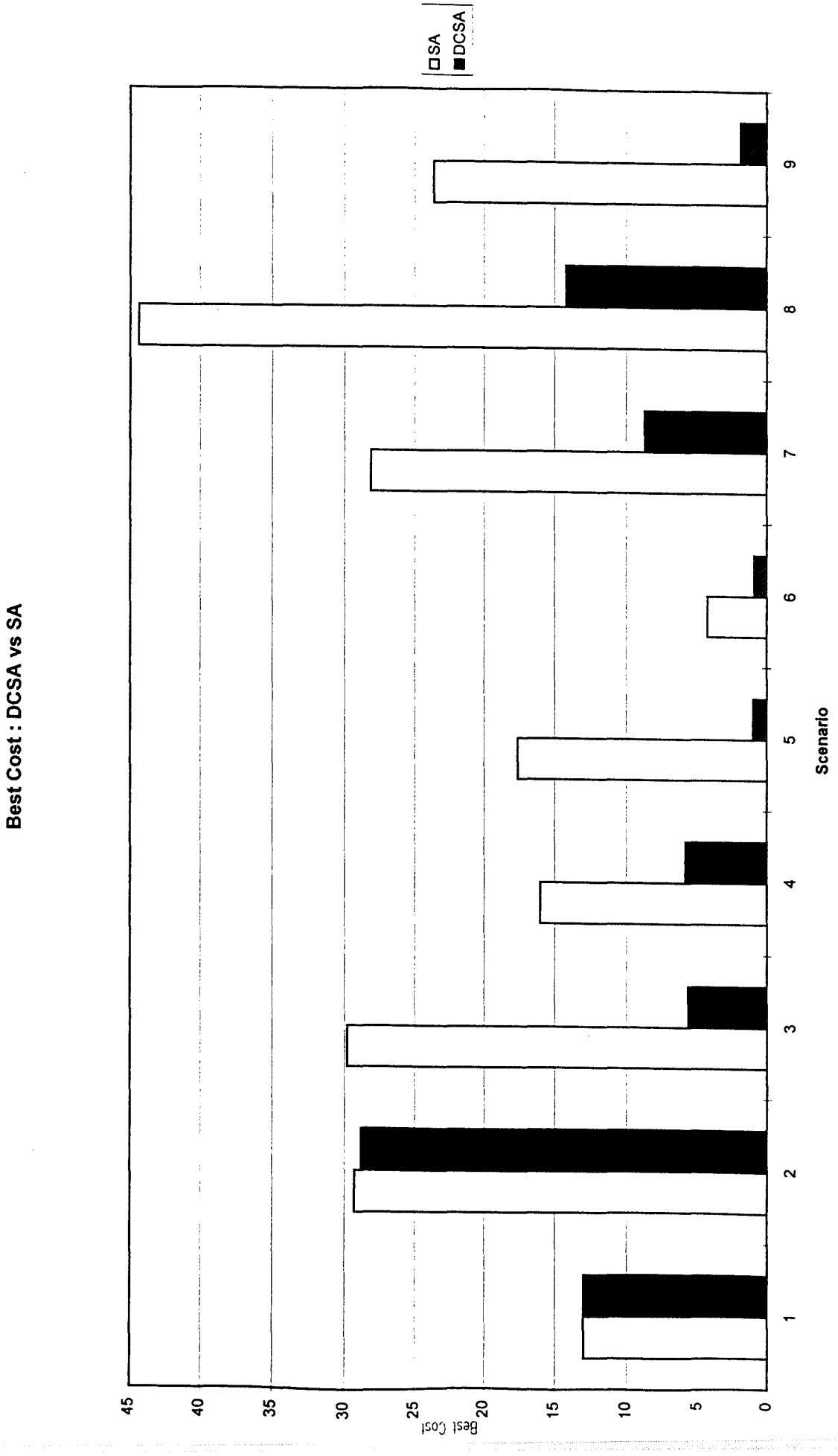
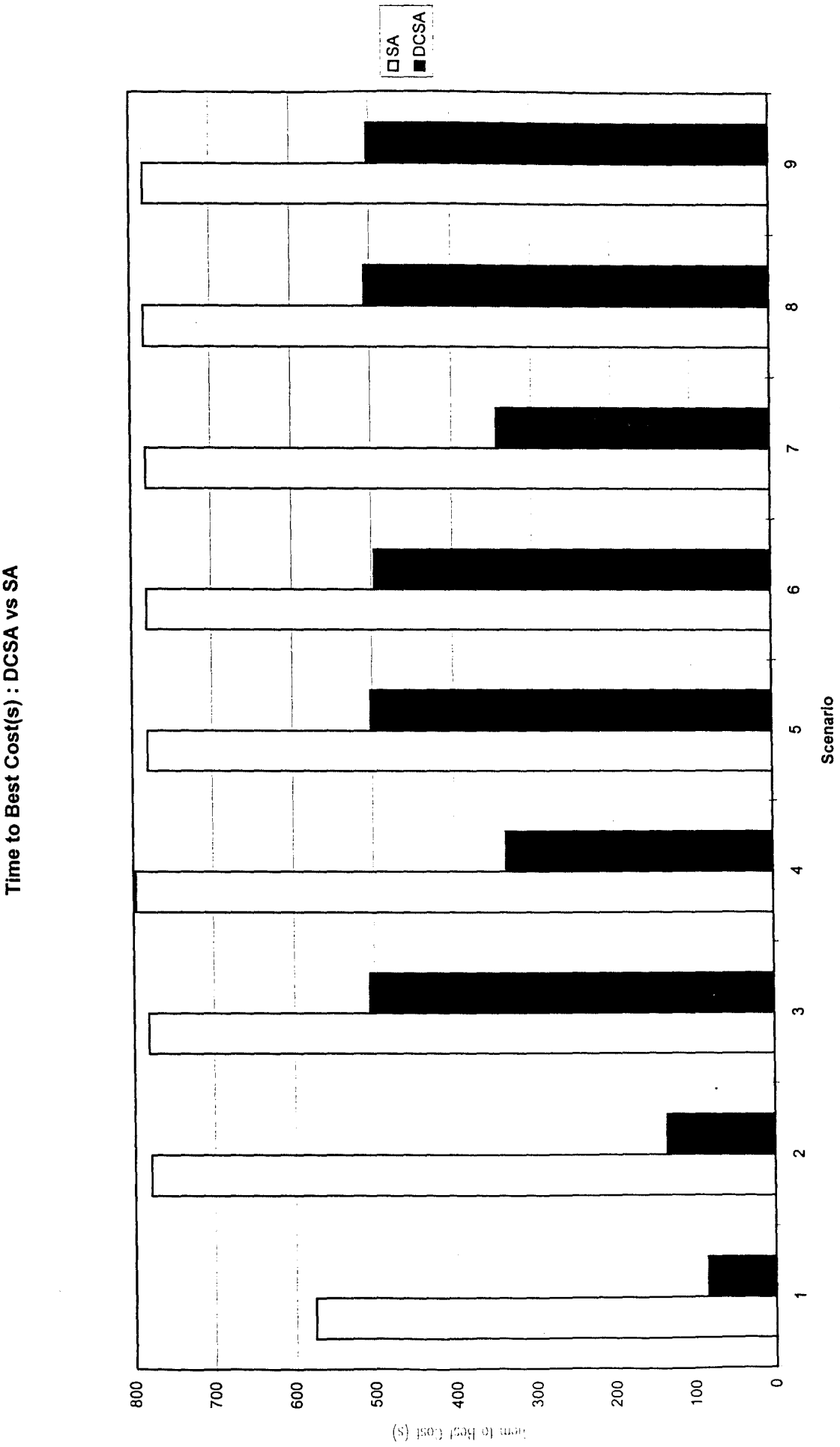


Figure 54. Graph showing DCSA vs. SA Time to Best Cost (Group B)



9.3 DCTS vs TS Results

9.3.1 Group A Data

Fig 55 showing the average cost of the DCTS and TS algorithms clearly shows that DCTS obtains lower cost solutions than TS for the 6 scenarios not solved optimally. In fact there are a further 3 scenarios which DCTS obtained lower cost solutions than TS, although both yielded solutions of low cost. Conversely TS obtained a lower cost solution than DCTS for one scenario, again the difference in cost was negligible. For the remaining 27 scenarios both algorithms obtained optimal solutions on average.

Fig 56 shows the average time taken by both algorithms and it can clearly be seen that DCTS takes longer than TS for all but 2 scenarios. It should also be noted that neither algorithm used the full time available for those scenarios which were not solved optimally by either algorithm. It would appear that using TS within the divide and conquer framework enables TS to overcome the premature termination experienced by the solo metaheuristic. However, the DCTS algorithm still terminates before the maximum run time for those algorithms not solved optimally. For those scenarios which were solved optimally and very quickly (less than one minute) by TS optimal solutions were also found by DCTS although it took slightly longer due to the additional overhead.

Figs 57 and 58 show the best cost and time to best cost for Group A scenarios. It can clearly be seen that DCTS optimally solves 3 of the 6 scenarios which were not solved optimally on average and TS solves 2 optimally. For these 6 scenarios the best cost is significantly reduced from the average. Scenario 22 has a lower best cost for TS than DCTS, the time taken was significantly longer.

The average percentage range used by both algorithms was identical. The average percentage of frequencies was identical for the easiest 13 scenarios. For the majority of the remaining scenarios DCTS used a higher percentage frequencies than TS. Some of the differences in values were significant.

To summarise, DCTS obtained lower cost solutions than TS for the six tricky scenarios. The time taken was comparable in most cases; the percentage range used and the percentage frequencies used was slightly higher. For the remaining scenarios DCTS and TS obtained optimal solutions on average although TS obtained its solutions more

quickly, using the same or fewer frequencies and the same range as DCTS. The divide

Figure 55. Graph showing DCTS vs. TS Average Cost (Group A)

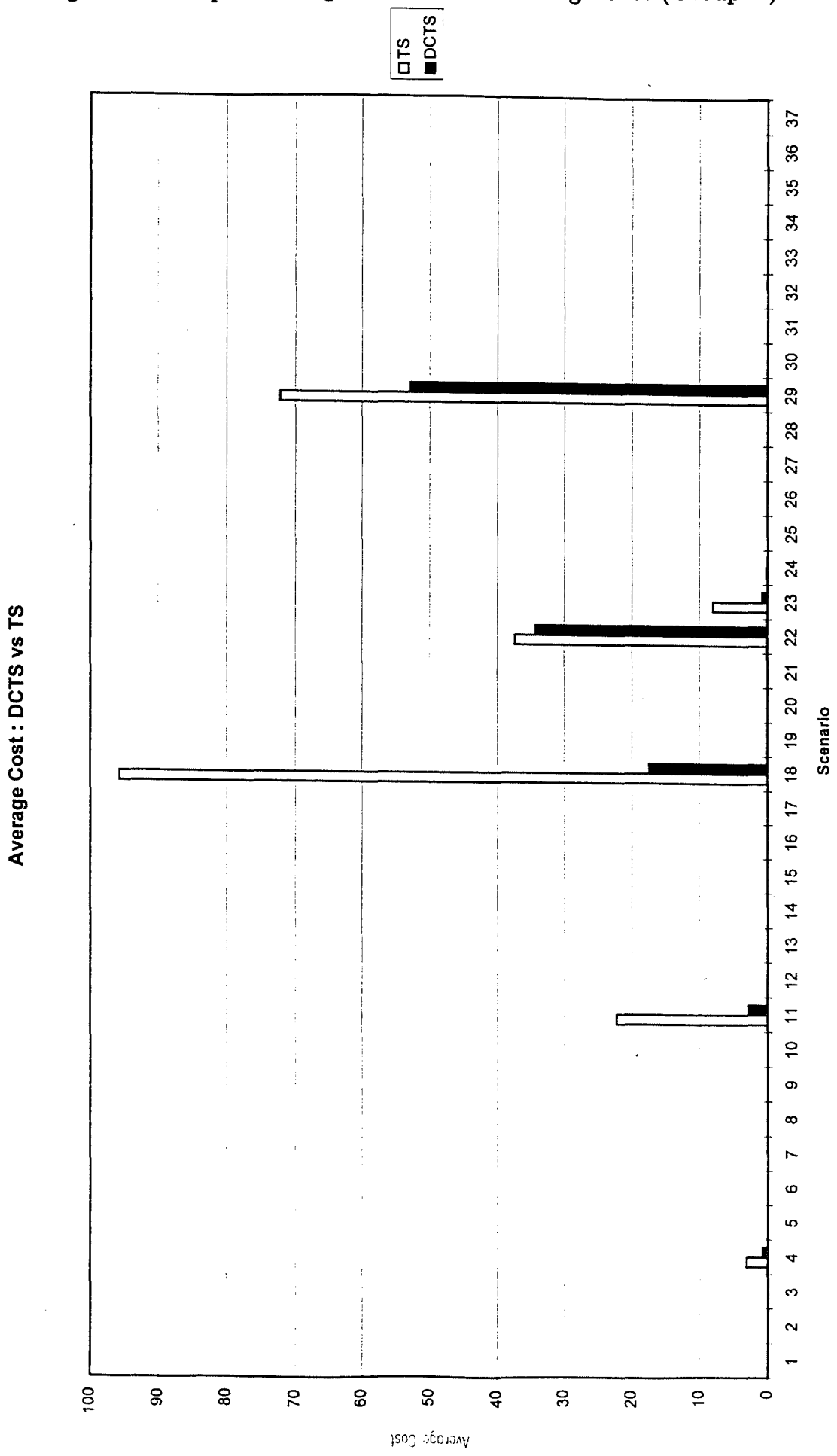


Figure 56. Graph showing DCTS vs. TS Average Time (Group A)

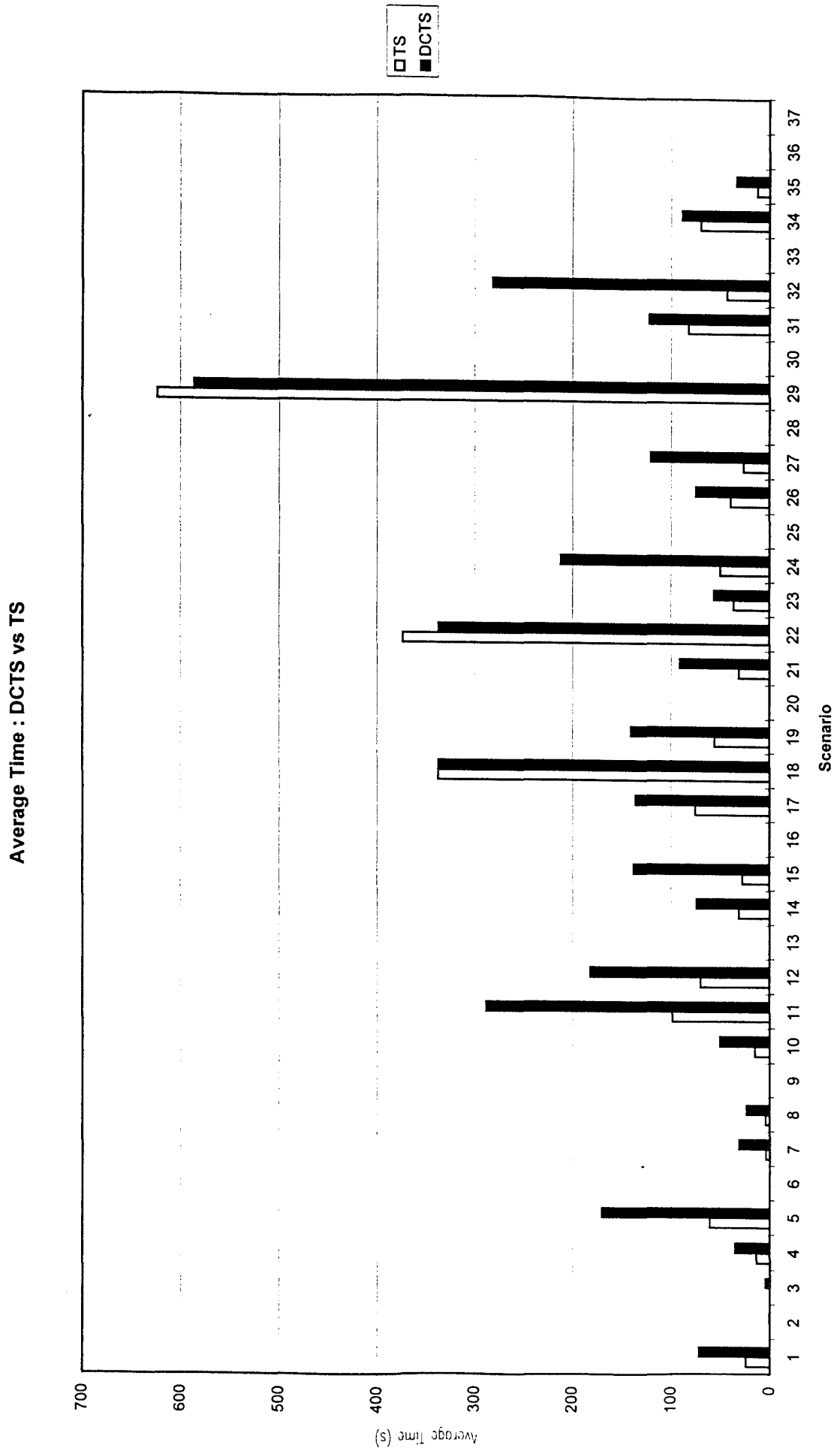


Figure 57. Graph showing DCTS vs. TS Best Cost (Group A)

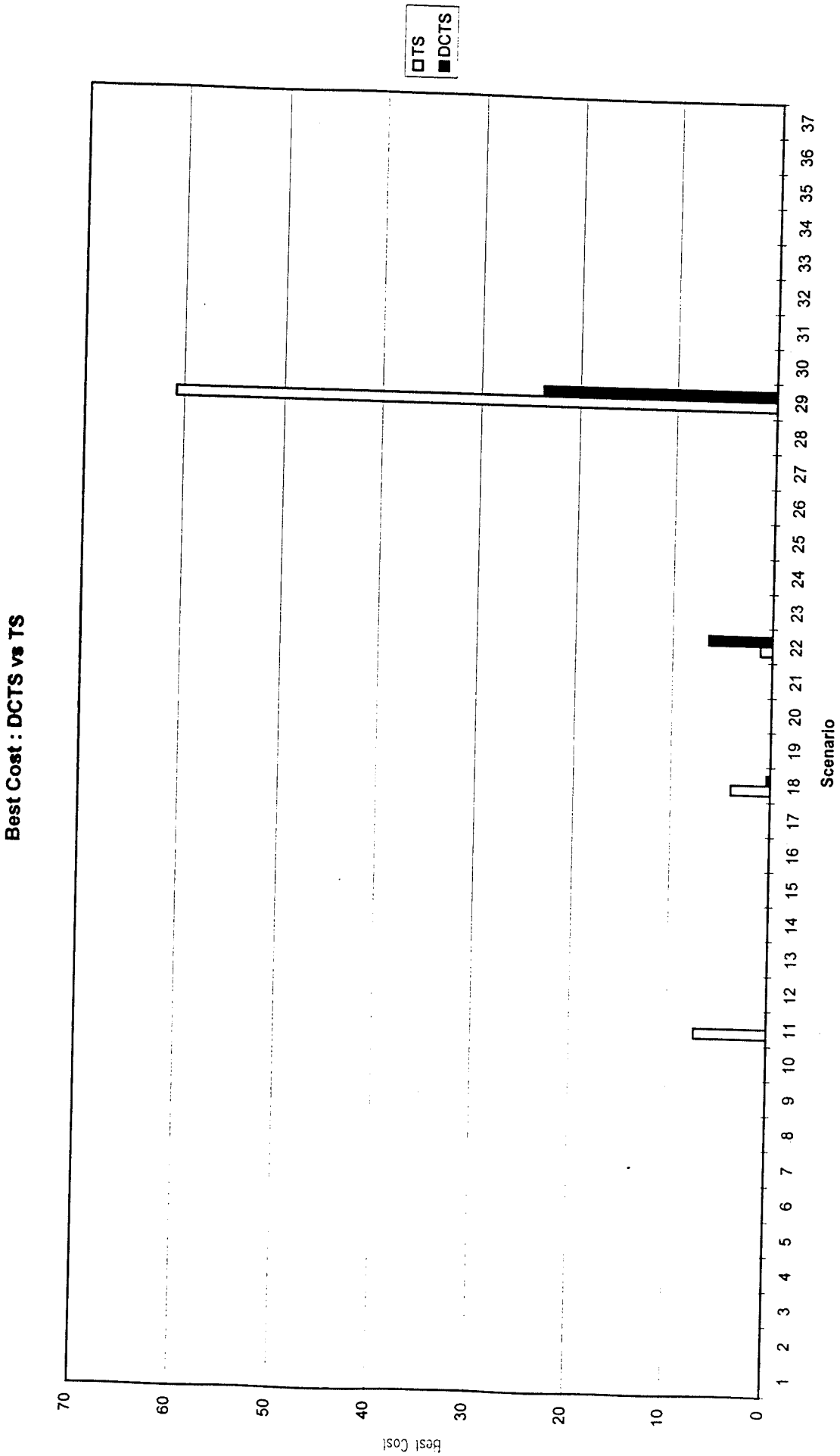
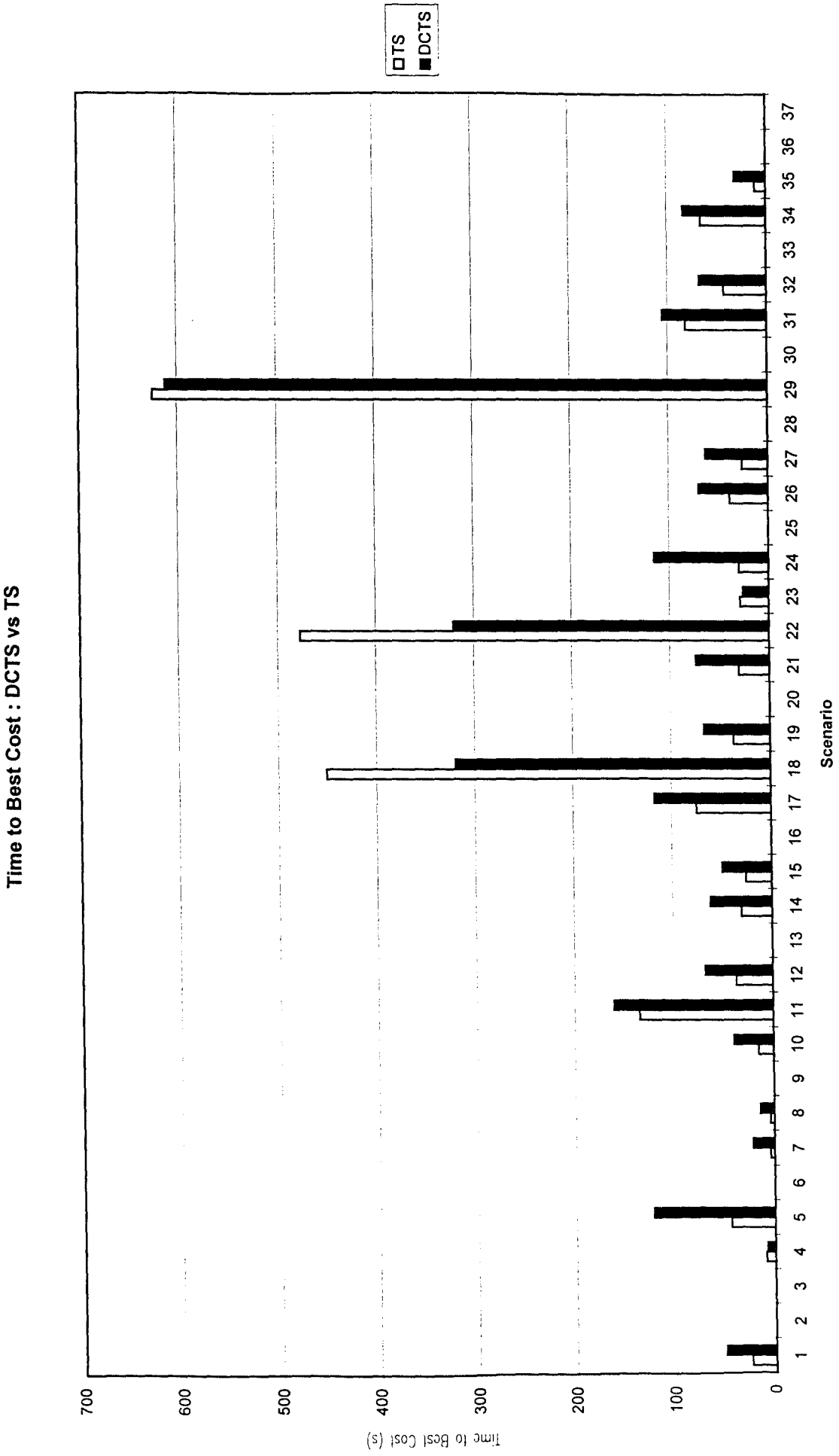


Figure 58. Graph showing DCTS vs. TS Time to Best Cost (Group A)



and conquer framework assisted TS in finding lower cost solutions than the sole meta-heuristic for the six tricky scenarios but the overhead meant that worse solutions (time, % Frequencies) were obtained by DCTS than TS on its own for those scenarios which were readily solved by TS.

9.3.2 Group B Data

By looking at Figs 59 and 60 showing average cost and time for the two algorithms it can clearly be seen that DCTS obtains lower cost solutions than TS for 7 of the 9 scenarios and obtains them using slightly more time. For scenario 7 TS obtains a lower cost solution than DCTS and it also runs for longer. Scenario 5 has a marginally lower cost for the TS algorithm and it also runs for slightly less time on average.

Figs 61 and 62 show the best cost and time to best cost. Scenario 1 has an identical best cost for both DCTS and TS; TS obtains its solution more quickly. For the remaining 8 scenarios DCTS obtains lower cost solutions than TS. It also takes marginally longer in 7 out of 8 cases.

The percentage range used for all scenarios and both algorithms was 100%. On average TS used fewer frequencies than DCTS. DCTS used 100% frequencies for 6 of the 9 scenarios (and obtained lower cost solutions than TS)

To summarise DCTS generally solves all of the group B scenarios using slightly more time and a higher percentage of frequencies than TS but obtains lower cost solutions. For these harder problems DCTS enables TS to overcome premature termination and go on to obtain better solutions.

Figure 59. Graph showing DCTS vs. TS Average Cost (Group B)

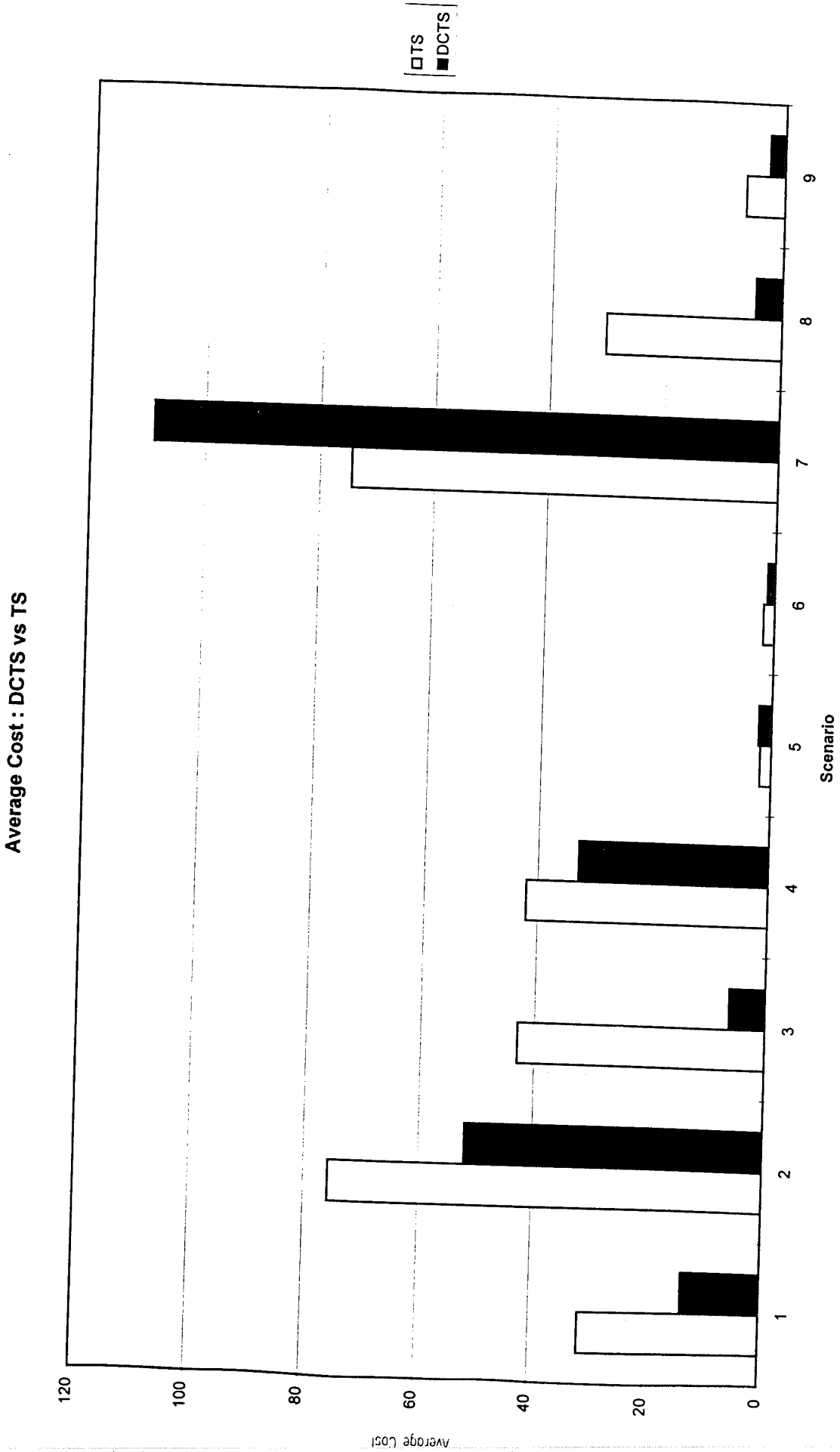


Figure 60. Graph showing DCTS vs. TS Average Time (Group B)

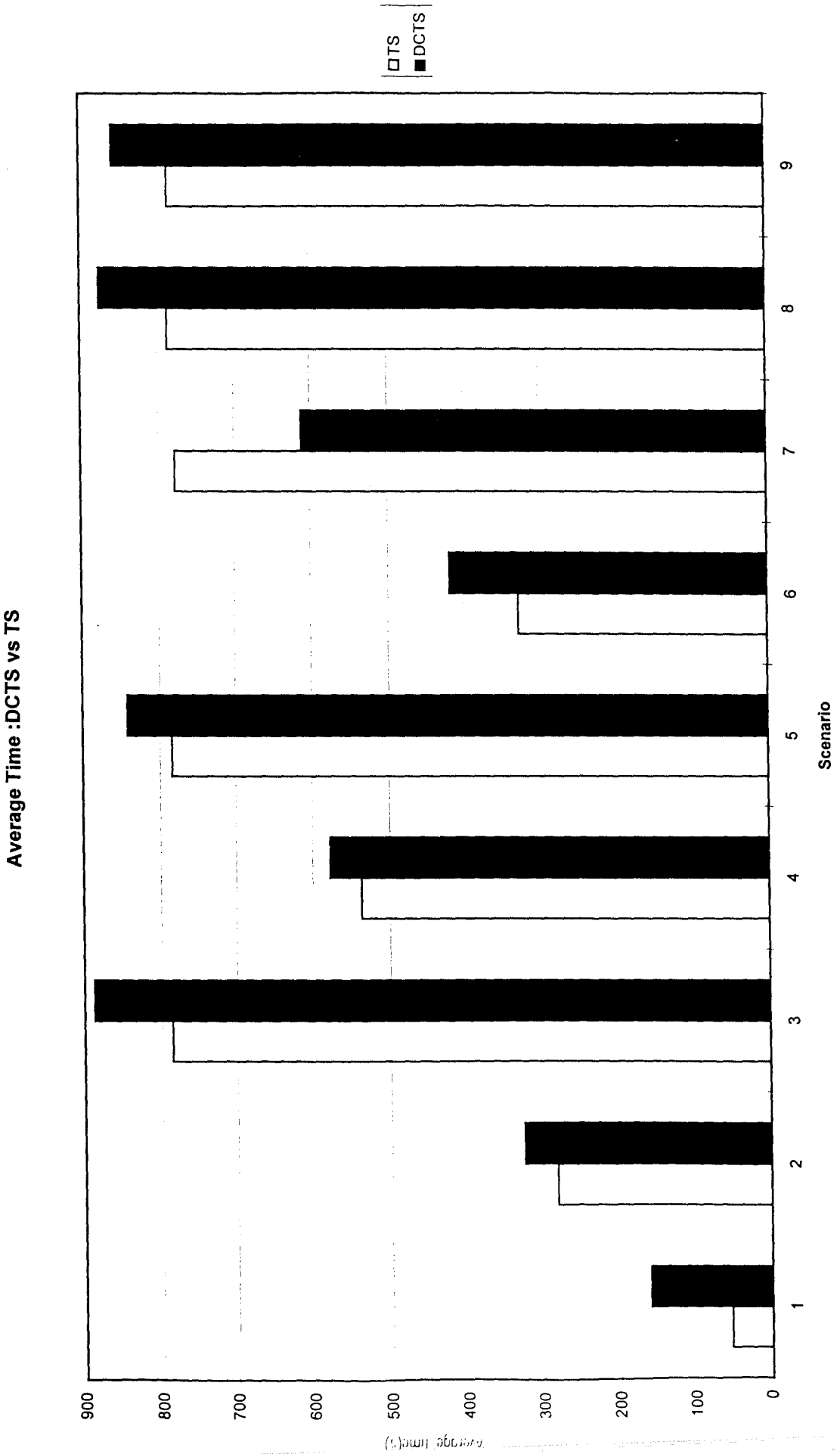


Figure 61. Graph showing DCTS vs. TS Best Cost (Group B)

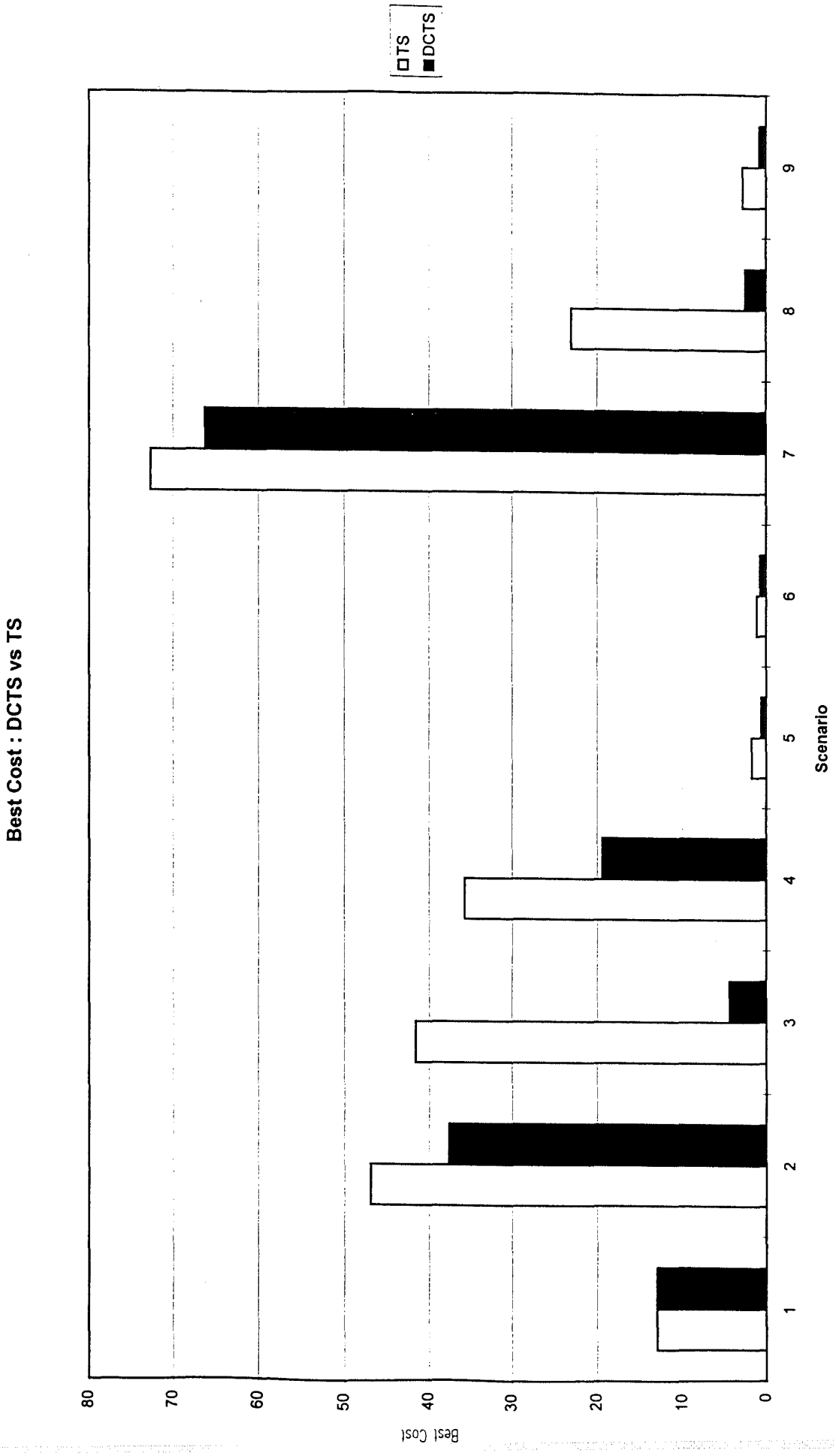
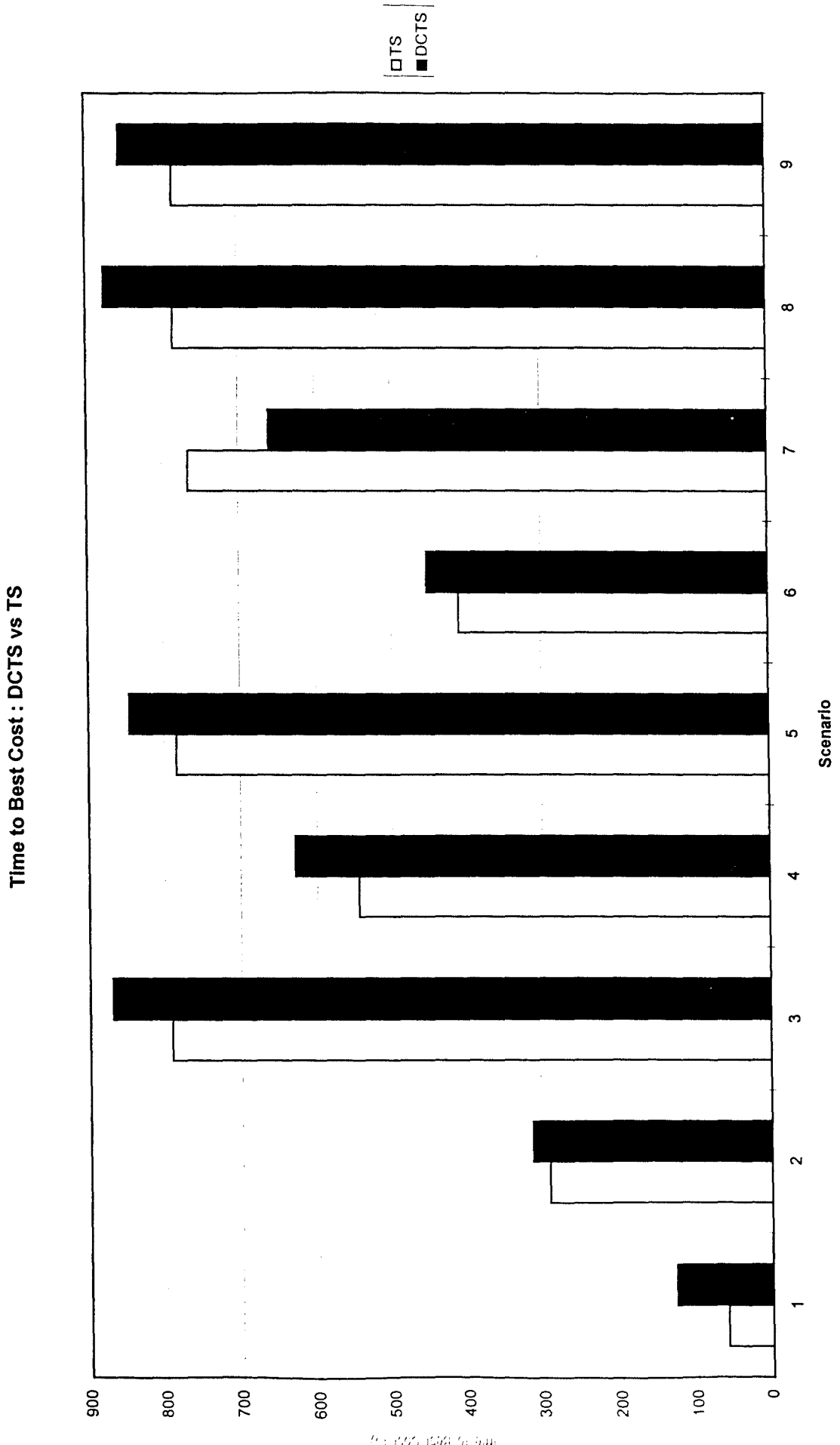


Figure 62. Graph showing DCTS vs. TS Time to Best Cost (Group B)



9.4 Summary

9.4.1 Group A

- DCSA obtained optimal solutions for 100% of scenarios in the best case and also obtained lower cost solutions than SA on average. These lower cost solutions were obtained in less time, using the same percentage range and comparable (or slightly fewer) frequencies than SA obtained its inferior solutions.
- DCTS improved on TS for the 6 scenarios which TS did not solve optimally. The solutions obtained had lower cost, used the same percentage range but were found in slightly longer time and using a slightly higher percentage of frequencies.
- For the scenarios solved optimally by TS on average the corresponding DCTS solutions were also solved optimally but took longer and used more frequencies.

9.4.2 Group B

- DCSA obtains lower cost solutions than SA in less time and using comparable percent range and frequencies for all scenarios.
- DCTS generally solves all of the group B scenarios using slightly more time and a higher percentage of frequencies than TS but obtains lower cost solutions.

9.5 Conclusions

Both the divide and conquer implementations obtained superior solutions to their solo heuristic counterparts for the majority of scenarios. The exception was that TS obtained optimal solutions more quickly and using fewer resources than DCTS for the most easily solved scenarios in Group A. This was due to the additional overhead involved in the divide and conquer framework.

The Group A scenarios were generally solved best by TS (with the exception of the six scenarios in the average case and 3 scenarios in the best case). The solutions to these exceptional cases was improved by using DCTS, although there was a slight time overhead incurred. The Group B scenarios were generally solved better by DCSA and

100% range was required throughout. A range of 80-100% of frequencies were also used when solving these scenarios.

The superiority of the divide and conquer technique (for non-trivial problems) over both metaheuristics when used alone is significant. The divide and conquer technique consistently produced solutions of higher quality than either TS or SA. The parameters were not fine-tuned for any of these algorithms and so the advantages in terms of cost are due to the methods employed by the divide and conquer technique. The divide and conquer algorithms have been tested on a wide range of scenarios and have demonstrated that they are robust and efficient in their solution of the frequency assignment scenarios presented.

For those non-trivial scenarios having comparable cost, the resources, span and order, used by the divide and conquer algorithms were less than the solo metaheuristic counterparts. The superiority of the divide and conquer method with respect to the span and order has very important practical applications. The frequencies that are not used for the assignment can be freed for use by other users of the electromagnetic spectrum. Since demand on the spectrum is increasing rapidly this advantage will become essential in the future.

Chapter 10

DC vs. DERA Results

In this chapter the results obtained using the divide and conquer algorithm are compared with those provided by the Defence, Evaluation and Research Agency (D.E.R.A). The results are not directly comparable and the reasons for this are explained. Despite the potential inconsistencies in the comparisons, the DC algorithm appears to obtain lower cost solutions in less time than methods currently in use at D.E.R.A. Some information which is needed to aid comparisons is unobtainable due to confidentiality constraints and these areas are highlighted in the relevant discussions.

The following abbreviations have been used in this chapter :
DCSA = Divide and Conquer algorithm with simulated annealing
DCTS = Divide and Conquer algorithm with tabu search
DERA = program used by D.E.R.A.

10.1 Reforming the Data Sets

The 46 TNET scenarios were provided by D.E.R.A. In the original format of the data there were 16 tactical communications problems. However, these were composed of one or more independent problems. The independent problems described constraint networks to which different frequency ranges were available. The frequency ranges were sufficiently separated that it was impossible for the constraint networks assigned to each of the ranges to cause inter-network interference. Each of the original 16 tactical

communications problems described one or more such networks and gave 46 separate scenarios in total. The results given throughout this thesis use the 46 scenarios format. This was agreed with D.E.R.A.

To compare the results obtained using the divide and conquer algorithm with those provided by D.E.R.A. it was necessary to recombine the individual scenario results to obtain solutions to the 16 tactical communications problems provided.

10.2 Comparison Difficulties

The programs being compared are DC (SA and TS versions) and DERA. The implementation of the divide and conquer program is given in Chapter 8, the DERA algorithm is completely unknown[1] but is thought to be based on a simulated annealing approach. To compare the two programs fairly each scenario should have used the same frequency set and initial solution over all runs. This was not possible, as indicated below.

10.2.1 Frequency Set

The number of frequencies to be generated randomly from a given frequency range was provided as part of the data set. The frequency range was selected by D.E.R.A. to maintain confidentiality but the range of frequencies and the number of distinct frequencies to generate was realistic. The objective function measured the deviation from the required separation and so the use of a different frequency range did not affect the credibility of the results obtained. To enable a fair comparison of the programs the DERA program used the same frequency set as that generated by the DC program. For each scenario, the frequency set was constant over all runs.

10.2.2 Initial Solution

The same initial solution was used over all runs and all four algorithms discussed in this thesis (TS, SA, DCSA and DCTS). The method used to obtain this initial solution is described in Chapter 8 section 8.4.3. Unfortunately the DERA program did not use the same initial solution and so this limits the direct comparison of the two programs.

[1] Confidentiality reasons

It is not known how the DERA program arrived at its initial solution except that it used a ' *very clever*' algorithm taking '*several minutes at least*' [2] to arrive at the initial solution. D.E.R.A. were unable to provide their initial solution for me to use [3]. In addition a change of supervisor at D.E.R.A. made it impossible for them to run their program again using my initial solution.

10.2.3 Machine

The results in this thesis were obtained on a desktop PC, 100MHz; the code was written in ANSI C using Borland C++ software, version 4.5. The DERA results were obtained on a VAX 4000; the code was written in ADA and ran on a UNIX platform. There are obvious difficulties in comparing the run-times of the programs due to the very different platforms on which the results were obtained.

10.2.4 Maximum Run Time

During the investigations leading to the results obtained in this thesis the maximum run time was set (by D.E.R.A.) at 15 minutes. However, the results recently obtained from D.E.R.A. were obtained by program runs of up to 40 minutes. Since only the average time taken has been provided in the DERA results comparison of the run-times of the two programs has limited value. In addition to this, the times provided for the DERA program do not include any preprocessing time (creation of data structures) or the time taken to arrive at the initial solution (which takes 'several minutes at least'). The run-times for the DC program include the time to create all the data structures and the time to obtain the initial solution.

10.2.5 Objective Function

The agreed objective function for this investigation was the sum of the positive discrepancies (explained in Chapter 2 section 2.5). Both parties have exchanged solutions for all of the 16 problems in order to confirm that the solution evaluation proceedures for both programs are in agreement. The constraint networks provided by D.E.R.A.

[2] comments from Roger Edwards at D.E.R.A.

[3] this was due to the risk of confidential information being derivable from the solution provided.

in the 16 data sets are a worst-case description of the constraints that they use and as such are harder to satisfy. This means that the DC evaluation of the DERA solutions tends to highlight several additional constraint violations than those declared by D.E.R.A. These additional constraint violations show only minor discrepancies in the objective function (typically < 5), which are thought to be due to 'harmonics'. Since these discrepancies are small the results of the two programs can still reasonably be compared. For problems where the cost is less than or equal to five the discrepancies make it difficult to compare the results realistically.

Table 11 shows the cost of the final solutions obtained for all 16 scenarios by each of the DC and DERA programs. The cost calculated for the DC program satisfies harder constraints than those used to calculate the final solution cost results obtained by the DERA program and so will often produce a marginally higher cost when evaluating the same solution.

10.3 Results

Table 10 gives the average run-time of the DC and DERA programs. Table 11 compares the cost of the final solution obtained by the DC and DERA programs.

Table 10. Comparison of DC and DERA : Average Time (s)

TNET	DCSA	DCTS	DERA	TNET	DCSA	DCTS	DERA
1	325	111	365	9	719	408	453
2	555	362	575	10	551	842	335
3	281	362	514	11	333	586	665
4	703	581	622	12	1067	824	649
5	692	611	625	13	344	611	677
6	791	568	630	14	218	122	100
7	536	943	633	15	534	875	475
8	335	577	639	16	519	858	462

Table 11. Comparison of DC and DERA : Cost

TNET	DCSA			DCTS			DERA		
Problem	Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
1	0.0	0.1	0.0	0.0	8.0	0.8	0.0	27.8	2.8
2	13.0	13.3	13.1	13.0	21.0	13.8	0.0	53.9	18.0
3	0.0	0.0	0.0	0.0	7.9	2.7	0.0	90.6	30.3
4	28.8	57.2	35.9	37.6	96.5	51.8	1.6	275.3	144.7
5	0.0	0.0	0.0	0.4	48.4	17.4	0.2	182.1	63.2
6	0.0	0.2	0.1	6.3	60.4	34.2	8.7	191.2	64.1
7	5.7	10.0	8.0	4.4	15.6	7.1	0.6	16.8	6.6
8	5.9	14.1	8.5	19.4	43.8	33.0	25.5	69.2	48.9
9	0.0	0.3	0.1	0.0	0.2	0.0	0.0	7.8	1.5
10	1.0	3.0	1.9	0.6	5.3	2.3	0.0	1.2	0.3
11	0.0	23.0	8.7	23.7	96.2	52.8	20.6	76.4	45.3
12	1.0	2.0	1.6	0.8	1.9	1.3	2.1	10.7	6.3
13	8.7	47.7	17.1	66.2	135.1	108.9	48.6	123.4	95.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.1
15	14.2	19.3	16.7	2.5	10.2	4.7	0.0	3.8	1.7
16	1.9	6.8	4.0	0.8	5.0	2.7	0.0	2.3	1.0

Figure 63. Graph showing DCTS, DCSA and DERA Best cost

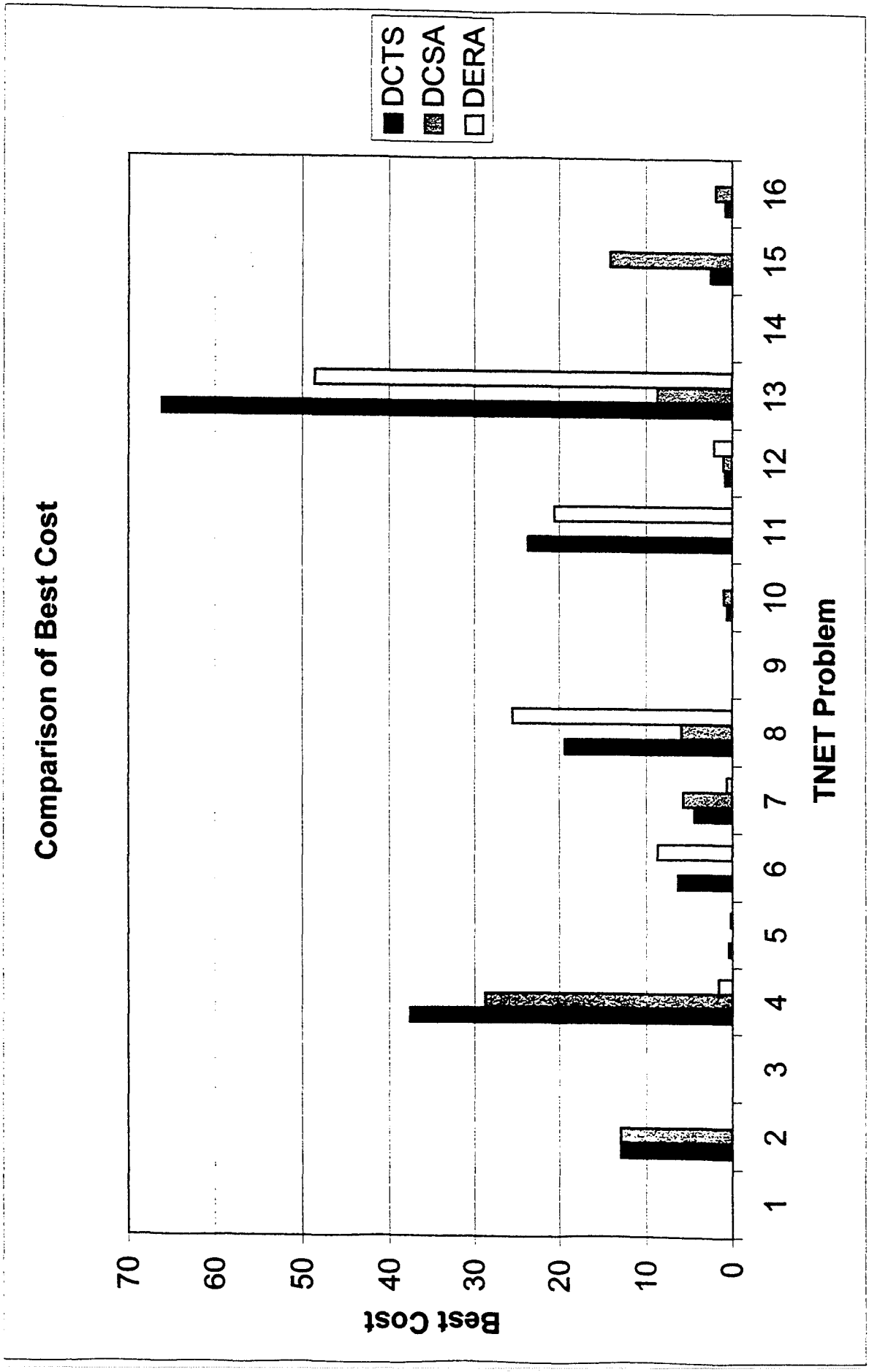


Figure 64. Graph showing DCTS, DCSA and DERA Average cost

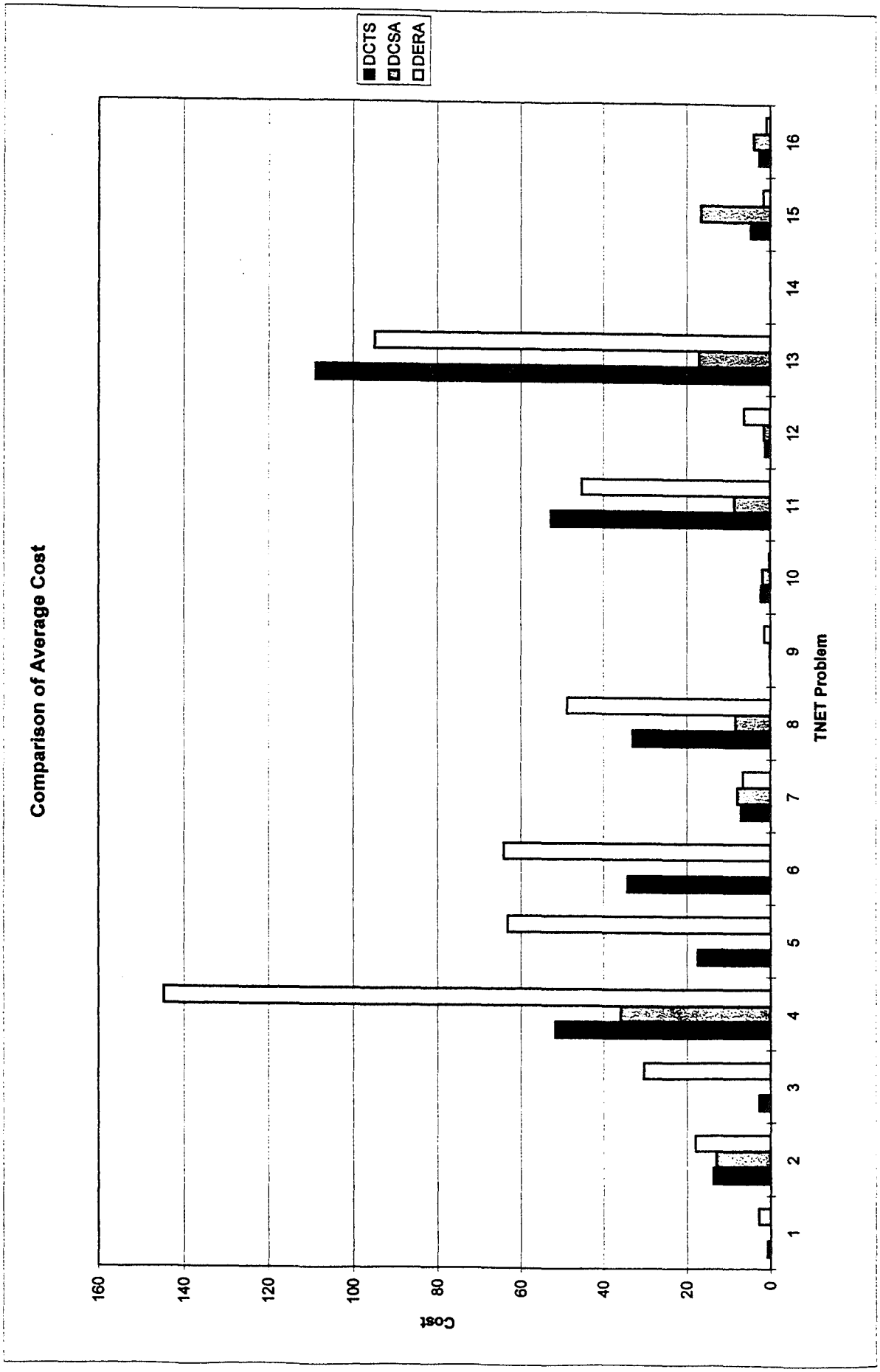


Figure 65. Graph showing DCTS, DCSA and DERA Average Time(s)

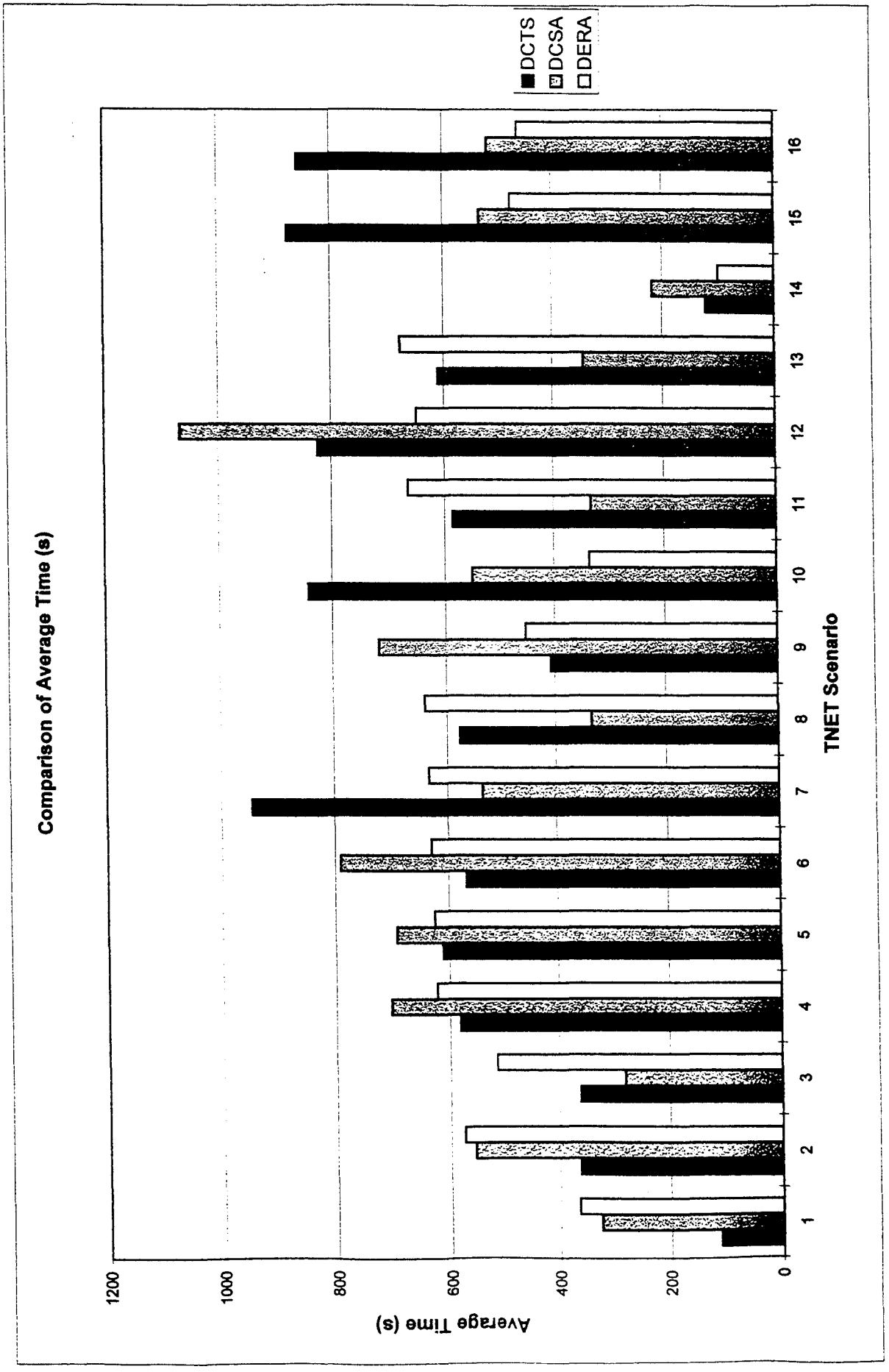
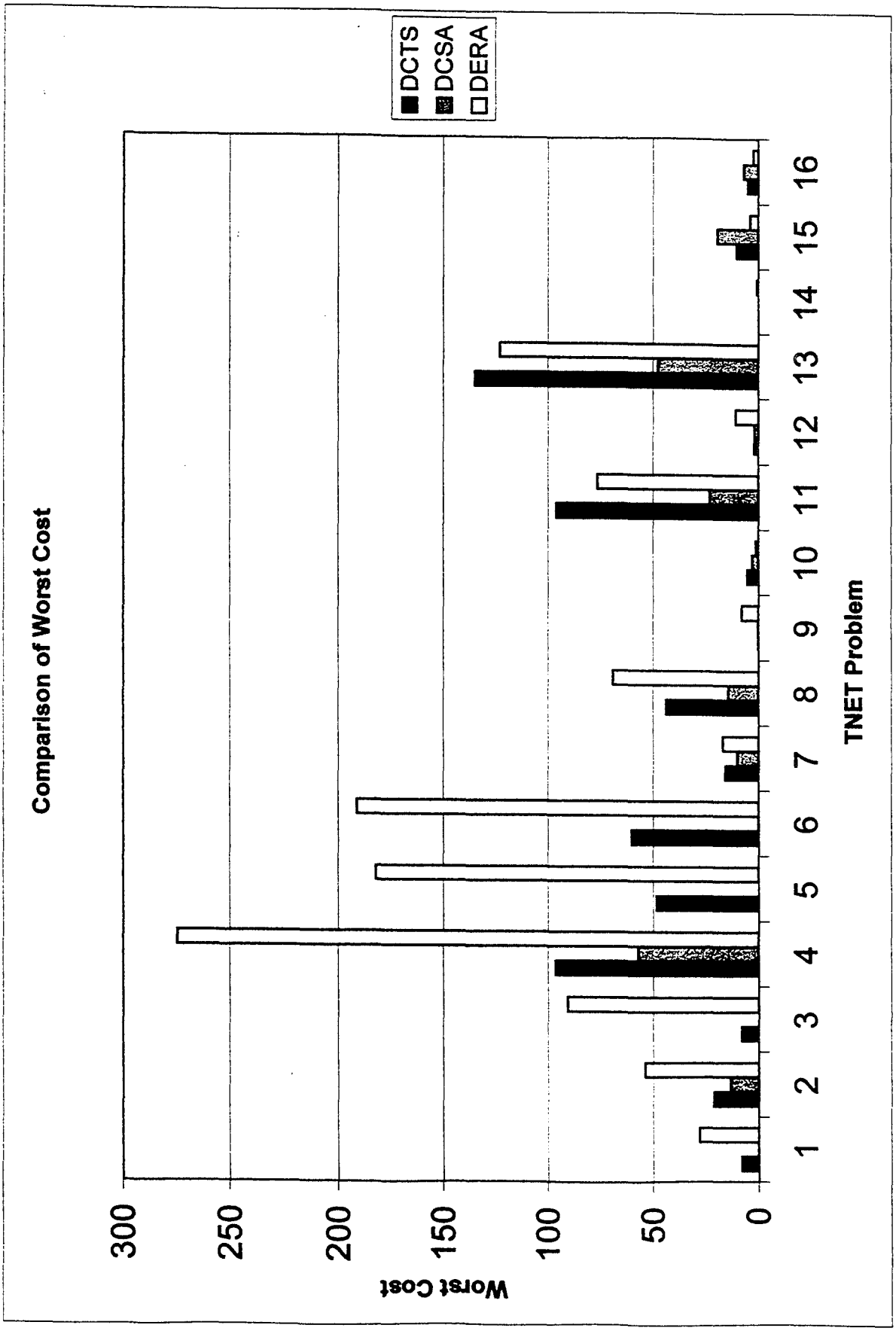


Figure 66. Graph showing DCTS, DCSA and DERA Worst cost



10.4 Discussion

10.4.1 Cost

Best Case (Fig 63)

If the DC results with cost < 5 are treated as zero then there are 6 problems for which all three algorithms obtain zero-cost solutions. For the remaining 10 problems the following observations can be made:

- DERA obtains lower cost solutions than DCSA and DCTS for 40% of problems
- DCSA obtains lower cost solutions than DERA for 60% of problems, 30% have zero-cost solutions.
- DCTS obtains lower cost solutions than DERA for 30% of problems
- DCTS obtains comparable cost solutions (< 5) with DERA for a further 30% of problems.
- DERA obtains a zero-cost solution for $TNET_2$, neither of the DC programs obtain optimal solutions.
- DERA obtains a significantly better solution for $TNET_4$ than DC.

Average Case (Fig 64)

- DERA obtains significantly higher cost (worse) solutions for 75% of problems than DC.
- DCSA generally obtains lower cost solutions than DCTS (except for 3 comparable, low cost problems)

Worst Case (Fig 66)

- DERA obtains the highest cost solutions for 81% of problems
- DERA obtains comparable (with DC), low cost solutions for the remaining 19% of problems.

- DCSA generally obtains lower cost solutions than DCTS (except for 2 comparable, low cost problems)

10.4.2 Time (Fig 65)

When comparing the average times of the programs the comparison difficulties discussed in Chapter 7 section 7.4.1 should be borne in mind. The average run-times are compared alongside the average cost results:

- There were 9 problems which had comparable low cost solutions for all three algorithms; the times for DERA and DCSA were comparable for 4 problems and for the other 5 DERA took less time.
- For 7 problems DERA obtained significantly higher cost solutions than DCSA; for 3 of these problems the times were comparable, for the remaining 4 DERA took much longer.

10.5 Conclusions

To reiterate: The cost values given in table 10 represent the values obtained by two different objective functions, the constraints used for the DC program were harder to satisfy and therefore may show a marginally higher (< 5) cost than an identical solution evaluated by the DERA program. Therefore, comparison of solutions where the DC cost is < 5 has limited value.

There were 6 problems solved optimally (zero-cost solutions) by both DC and DERA. For the remaining 10 problems: the DERA program obtains better solutions than DC for 3 problems in the best case, $TNET_2$, $TNET_4$ and $TNET_{15}$. It is not known how long the DERA program ran to obtain these solutions, but may have been up to 40 minutes. The best case is interesting because it highlights two problems for which DERA obtained zero-cost solutions where DC did not.

The average and worst case results are the most useful when comparing algorithms. For the average case DC obtained lower cost (better) solutions than DERA for (75%) of problems, in the worst case this rose to (81%). The DCSA algorithm generally obtained better solutions than DCTS.

DCSA obtained the lowest cost solutions of the three algorithms, obtaining zero-cost solutions for 4 problems where DERA did not. Conversely DERA obtained zero-cost solutions for 2 problems where DCSA did not. For the 9 low-cost problems the DERA program took comparable or less time than DCSA. For the 7 higher cost problems the DERA program took comparable or significantly more time than DCSA

For the nine low cost problems DCSA and DERA obtained comparable cost solutions (DERA used less time on average). For the seven higher cost problems DCSA obtained significantly lower cost solutions than DERA and also found these superior solutions in less time on average. For harder problems DCSA seems to obtain the better results. For easier problems the results are comparable.

The differences between the cost values of the best/average/worst solutions obtained by the DERA algorithm are quite significant. This shows that the DERA program has a higher probability of providing a poor solution compared to the relatively less variable solution costs obtained by the divide and conquer algorithms. The divide and conquer algorithm is more reliable than the DERA algorithm.

10.6 Further Work

For the two problems ($TNET_2$ and $TNET_{15}$) for which DERA obtained zero-cost solutions and DC did not it would be interesting to allow DC to run for a longer time period to see if a zero-cost solution could be obtained.

Result : the above problems were solved to optimality by DCSA in 16 and 20 minutes respectively.

Chapter 11

Conclusions

11.1 Introduction

The main objective of this research was to investigate how to split the frequency assignment problem into subproblems for more rapid solution, and to establish a software library of techniques based on existing and novel algorithms (comparing the performance of the alternative techniques).

This chapter is divided into three parts : part one provides a summary of the main findings of the research, part two states some conclusions which can be drawn from this work, and finally, part three gives some concluding remarks. This chapter shows that all the objectives that were stated at the outset of the research have been achieved, and that all the metaheuristic algorithms which have been developed prove to be efficient and effective solution techniques for the FAP. In particular, the divide and conquer technique has been shown to obtain superior solutions when compared with classic implementations of TS and SA, and when compared with results obtained by the program currently in use at D.E.R.A.

11.2 Summary of the Research

The main contribution of this research is the detailed design, development, implementation and empirical analysis of four metaheuristics based on SA, TS and divide and

conquer strategies for solving the FAP.

In addition to fulfilling the research objectives an up-to-date literature review of the frequency assignment problem and its variants was given (Chapter 3). This review (and discussions in Chapter 2) established why the FAP is important at the current time and outlined its practical applications. This was followed by an overview of recent developments in the field of metaheuristics (Chapter 3) with particular attention being paid to techniques used to solve the frequency assignment problem. The aim of this review was to highlight both the significance and relevance of the heuristics used in this research.

Chapter 4 gave a brief introduction to heuristics, explained why exact algorithms are insufficient for large-scale commercial problems and defined some general terms. Neighbourhood search was then explained with the relevant terminology. These ideas were used later in the introduction to the various algorithms used in this research.

For ease of reference the original research objectives are repeated here. The research contributions are given with references to the sections of the thesis in which the results were first presented.

1 *To establish a software library of techniques based on existing and novel algorithms and to compare the performance of the alternative techniques.*

A suite of programs have been developed using Borland C. The following (existing) algorithms have been implemented for solving the MATRIX test data: random descent, steepest descent, greedy and backtracking. The results of these preliminary investigations, on a small data set, is described in Chapter 5.

For the realistic (TNET) data set a suite of heuristic and metaheuristic algorithms was developed. The available algorithms are steepest descent, greedy algorithm, simulated annealing and tabu search. In addition a (novel) divide and conquer algorithm was developed which is able to utilise either simulated annealing or tabu search in the improvement stages. The implementation details and results for the simulated annealing and tabu search algorithms were given in Chapter 7. These two metaheuristics were developed as benchmarks for later comparison with the novel divide and conquer

approach. The divide and conquer framework was described fully in Chapter 8 and results were presented. In Chapter 9 the results from the solo metaheuristics were compared with the divide and conquer algorithm. The divide and conquer algorithm was found to obtain significantly better solutions than either of the solo metaheuristics when acting on non-trivial scenarios. The DCSA algorithm in particular was shown to be robust and efficient in its solution of the FAPs presented.

2 *To develop metrics for measuring the goodness of assignments.*

The initial suggestion of counting the number of constraint violations has been superceded by a new objective function which calculates the sum of the positive discrepancies of the violated constraints. This evaluation function is thought to more accurately represent the interference suffered by a communications network. In addition, preliminary investigations (described in Chapter 5) indicated that there was a close association between the vertex ordering criterion used and success of the algorithm when using a particular evaluation function. Ideally the ordering criterion and evaluation function should be chosen to complement one another.

3 *To investigate how to split the problem into subproblems for more rapid solution.*

The divide and conquer program (Chapter 8) divides the problem by dividing the frequencies available into disjoint bands. The smaller versions of the problem are then solved before their solutions are recombined to obtain a solution to the whole problem. Several division criteria were investigated and division of the frequency set (into approximately equal order bands) yielded the higher quality results. A particular advantage of this division method is that it does not depend on knowledge of the constraint graph. This method has proven to yield high quality solutions even with rich connectivity between the subproblems.

4 *Investigate the benefits of hybrid techniques compared with the application of a single type of algorithm.*

The divide and conquer program using either metaheuristic in the improvement stages outperforms the corresponding solo metaheuristic (Chapter 9). The divide and conquer algorithm obtains solutions with less interference, in less time and often using fewer spectral resources than the solo metaheuristic program for realistic non-trivial data sets. The divide and conquer method has proven to be robust and efficient in solving the frequency assignment problem.

5 *To study the significant factors which influence performance, e.g. frequency allotment, frequency selection and the provision of an incremental objective function.*

Early experimentation (Chapter 5) showed that the best results were obtained by pre-ordering the links in descending order of co-site vertex degree when used in conjunction with the objective function summing the positive discrepancies.

The results in this chapter also suggested that the selection of an equivalent best-cost frequency at random from those available reduced the amount of spectral resources used in the final solution.

Finally, the provision of an incremental objective function was shown to improve solution quality. Data structures were described (Chapter 5 for MATRIX, Chapter 7 for TNET data) which enabled a very fast incremental objective function to be employed. This was found to give a significant reduction in the computation time for the algorithms, enabling them to evaluate more solutions in the time allocated. The average time complexity of the objective function (Chapter 7) was reduced from $O(N^2)$ to $O(\frac{C}{N})$.

The findings of the preliminary investigation were considered when developing the metaheuristic algorithms.

6 *To devise suitable experiments to assess the performance of each algorithm in isolation, and with other available techniques for solving a class of simulated but realistic problems.*

Extensive computational results are provided (Chapters 7-9) to compare the effectiveness of each of the algorithms developed. All of the programs

were developed and executed on the same machine. In addition program modules were developed which enabled the metaheuristics implemented in Chapter 7 to be used directly within the divide and conquer framework. For each scenario, all of the algorithms used the same initial solution and the same frequency set. By controlling as many environment factors as possible, a realistic comparison of the algorithms was facilitated.

Chapter 10 compared the solutions found by the divide and conquer technique with results obtained from the program currently in use at D.E.R.A. Some comparison difficulties were highlighted and some reasons for these offered. Despite the various caveats (including that the divide and conquer results were obtained on a far slower machine) the divide and conquer algorithms were found to obtain far superior solutions to the existing program. In addition the divide and conquer algorithm performed more consistently (variations between best, worst and average values was much smaller).

Chapter 6 described the data sets used in the course of this research. The primary data set, on which most of the metaheuristics acted, comprised of 46 realistic networks generated to simulate real-life tactical communications scenarios, the largest of these had 1090 links, 40 frequencies and 9325 constraints giving a search space of 10^{1746}. The test data was provided by Roger Edwards at D.E.R.A., Malvern. The test problems were sufficiently varied to highlight the difference between the performance of the heuristics.

7 *To compare the difference between problems considered in this thesis and others in the literature; for example the CELAR data.*

There is a set of FAP instances known as the CELAR data. In chapter 6 these instances were briefly discussed and the differences between those problems and the TNET data used in this research were highlighted. Chapter 4 described two of the metaheuristic techniques (SA and TS) which were developed during the 18 month CALMA project and tested using the CELAR data. It would be unfair to compare the metaheuristics developed for the CALMA project with the ones developed in this thesis. Many of the algorithms in the CALMA project employed consistency techniques to reduce the size of the problem instances, and this led to an improvement in

their efficiency. In addition to this a great deal of time was spent fine-tuning the parameters of the search technique for each problem considered — this goes against the ideals behind the research described here. In addition the CELAR data describes a fundamentally different type of assignment problem due to the existence of multiple frequency domains.

11.3 Conclusions

In this section several points are enumerated which outline findings that have been well researched in the literature, and conclusions which may be drawn from the research described in this thesis.

1. TS and SA are often used for solving the FAP and produce results which are better for large problems than many alternatives, in particular hill climbing and genetic algorithms [HTS96] [HST97].

2. TS and SA have both been used effectively for the data supplied by D.E.R.A. It was found that SA was the more reliable of the two heuristics but that TS was better for the easier scenarios.

3. DC combined with simulated annealing or tabu search has never been considered before for any combinatorial problem (as far as an extensive literature search and consultation with experts has revealed).

4. DC is able to improve upon the results obtained by both TS and SA. Barr et al listed seven indicators to establish whether a new heuristic method could be deemed to have made a contribution to the subject [Barr et al.95]. The first seven indicators below are from their paper, all of which are satisfied by the divide and conquer algorithm. In addition, two further important features of the divide and conquer have been added to the list.

    - Fast : The DC algorithms generally obtain better quality solutions in less time than the corresponding solo metaheuristic for non-trivial problems.
    - Accurate : The DC algorithm has been shown to provide equivalent or higher quality solutions than simulated annealing, tabu search or the program in use at the D.E.R.A.

- Robust : The DC programs, DCSA in particular, has proven to be less sensitive to problem characteristics than the SA, TS or DERA programs.
- Simple : The method is easy to implement — researchers may already have existing simulated annealing or tabu search algorithms which can be used within the divide and conquer framework.
- High-Impact : The divide and conquer algorithm developed is able to solve an important combinatorial optimisation problem, the FAP, faster and more accurately than other existing methods.
- Generalizable : The division of the FAP is not reliant on knowledge of the constraint graph (compared to CELAR problems being solved with pre-processing using arc consistency and the property that some graphs could be partitioned into disjoint subgraphs). The framework could be readily adapted to accept other FAPs.
- Innovative : A 'new' method (combining DC with heuristics) providing improvement without fine-tuning of parameters.
- Efficient : Even though spectral resources are not minimised by way of inclusion in the objective function the results show that often fewer spectral resources are used by the solutions obtained by the divide and conquer algorithm compared to solo metaheuristics. Efficient use of the available spectrum will become essential as demands continue to increase.
- Parallelisable : Method is ammenable to parallelisation

5. It is difficult to compare the DC results with the DERA results. However the results indicate that DC is better with respect to robustness, quality of solution (with some caveats) and is also faster.

6. Divide and conquer incorporating heuristics may well be effective for other problems.

   - New
   - Robust
   - Parallelisable
   - Effective.

11.4 Final Remarks

Finally, it may be concluded that all the initial objectives posed at the outset have been achieved. The FAP is an NP-hard combinatorial optimisation problem and since exact algorithms are generally impractical for dealing with large realistic problems, the development of effective metaheuristic methods is extremely important. The divide and conquer technique described in this thesis is more effective and efficient than the SA and TS metaheuristics. This thesis has shown that the design of efficient data structures and the use of pre-ordering and efficient selection techniques can further improve the performance of algorithms — these issues should be considered when developing any metaheuristic. Also since there are no strict guide-lines on how to measure the performance of a heuristic, comparison with available heuristic techniques is also important. To this end implementations of all four algorithms have been developed so that as many variables as possible can be kept constant and therefore enable a fair comparison of the algorithms [Bar et al.95].

The divide and conquer technique has been shown to provide high quality solutions for this class of FAPs. The technique is flexible enough to incorporate other types of constraints, objective functions and indeed improving heuristics in the local and global improvement stages. The divide and conquer technique is easily adaptable for solving other variants of the FAP. Although conclusions were based on a specific class of FAPs, these problems are more general than some FAPs that occur in practice. Therefore the divide and conquer technique deserves further investigation, both for the FAP and for other difficult combinatorial optimisation problems.

This research has established the usefulness of employing a divide and conquer strategy to improve upon the results obtained by metaheuristics used alone and to take the emphasis away from parameter tuning into a more general framework. The research also shows that enhanced solutions for the FAP can be found by subdividing the problem as described. The subproblems are successful even with rich interconnections between the subproblems.

Overall, DCSA appears to be the most efficient and effective approach (of those investigated) for solving this class of FAPs. The divide and conquer method described here is novel, robust and effective. In addition it lends itself to parallelisation. To conclude : the divide and conquer framework presented here shows promise and it is worthwhile

investigating the suggestion of TS in the local improvement phases and SA in the global stages in anticipation of further improving the results.

This research has gone some way to identifying factors which influence the performance of algorithms used for solving the FAP. The divide and conquer framework has been shown to out-perform solo metaheuristics without the need for excessive parameter tuning or reliance on knowledge of the constraint network. It is hoped that this research will be of use to computer scientists, operation researchers and practitioners investigating solution methods for the FAP. This research will also be of interest to those considering heuristic approaches in general. Knowledge of how well this divide and conquer framework can perform is encouraging and provides some valuable ideas and information.

Bibliography

[Aar et al. 95] K.I. Aardal et al (1995), A Branch-and-Cut Algorithm for the Frequency Assignment Problem. aardal@cs.uu.nl

[And73] L.G. Anderson (1973), A Simulation Study of Some Dynamic Channel Assignment Algorithms in a High Capacity Mobile Telecommunications System, *IEEE Trans. Commun.* COM-21 : 1294-1301

[AC96] M.J. Alves and J. Climaco (1996), An Experimental Comparison of Simulated Annealing and Tabu Search in 0-1 Linear Programs, Discussion Report, Combinatorial Optimisation 1996, London.

[Bar et al.95] R. Barr et al (1995), Designing and Reporting on Computational Experiments with Heuristic Methods, *Journal of Heuristics*, **1**(1) : 9-32.

[Bat96] R. Battiti (1996), Reactive Search : Towards Self-Tuning Heuristics, In : *Modern Heuristic Search Methods.* V.J. Rayward-Smith, I.H. Osman et al (eds.), John Wiley and Sons. England.

[Bat et al.98] J. Bater et al., Are There Effective Frequency Separation Constraints for FAPs with Irregular Tx Placements?, submitted to : *NATO Symposium in Alborg* 1998.

[Bay82] I. Baybars (1982), Optimal Assignment of Broadcasting Frequencies, *European Journal of Operations Research*, **9** : 257-263.

[Bea90] . Beasley (1990), OR Library: Distributing Test Problems by Electronic Mail. JORS, **41**, 1069-1072.

[BB95] J.A. Bland and D.J. Baylis (1995), A Tabu Search Approach to the Minimum Distance of Error-Correcting Codes, *Int Journal of Electronics*, **79(6)**: 829-837.

[Bou et al.95b] A. Boujou et al. (1995), Tabu Search for the Radio Links Frequency Assignment Paper, Presented in : Applied Decision Technologies [ADT95], Brunel, London, 3-5 April, organised by UNICOM

[BBM93a] Beasley, Bull and Martin (1993), An Overview of Genetic Algorithms : Part 1 Fundamentals, *Univeristy Computing*, **15** : 58-69

[BBM93b] Beasley, Bull and Martin (1993), An Overview of Genetic Algorithms : Part 2 Research Topics, *Univeristy Computing*, **15** : 170-181

[BBM93c] Beasley, Bull and Martin (1993), Reducing Epistasis in Combinatorial Problems by Expansive Coding, *Fifth International Conference on Genetic Algorithms*, pp400-407

[BBM94] Beasley, Bull and Martin (1994), Complexity Reduction using Expansive Coding, Evolutionary Computing AISB workshop '94. *Lecture Notes in Computer Science* 865.

[BJC98] J.E. Bater, P.G.Jeavons and D.A. Cohen, Non-binary Modelling of Frequency Assignment Leads to Significant Reduction in Predicted Spectrum Requirements, submitted to *EMC'98 Nroclaw.*

[BMP95] C. Brind, C. Muller and P. Prosser (1995), Stochastic Techniques for Resource Management, *BT Technical Journal.***13**(11) : 55-63

[Cas97] D.J. Castelino (1997), Meta-heuristics for the Radio Link Frequency Assignment Problem, PhD Thesis, University of Wales College of Cardiff.

[Cos93] D. Costa (1993), On the Use of Some Known Methods for T-Coloirings of Graphs, *Annals of Operations Research*, **41** : 3443-358.

[CAL95] *ftp://ftp.win.tue.nl/pub/techreports/CALMA/*

[CEL] *ftp.win.tue.nl/pub/techreports/Instances*

[CHS93] W. Crompton, S.Hurley and N.M. Stephens (1993), Frequency Assignment Using a Parallel Genetic Algorithm, *Proceedings of the IEE/IEEE Natural Algorithms in Signal Processing Workshop*, **2** : 26/1-26/8.

[CHS94a] W.Crompton, S.Hurley and N.M. Stephens (1994), Applying GAs to Frequency Assignment Problems, *SPIE conference on Neural and Stochastic Methods in Image and Signal Processing III*, San Diego.

[CHS94b] W. Crompton, S.Hurley and N.M. Stephens (1994), A Parallel Genetic Algorithm for Frequency Assignment Problems, *Proceedings of the IMACS/IEEE Conference on Signal Processing, Robotics and Neural Networks*, pp81-84, Lille, France.

[CHS96] D.J. Castelino, S. Hurley and N.M. Stephens (1996), A Tabu Search Algorithm for Frequency Assignment, *Annals of Operations Research* **63** 301-319.

[CMB92] D.J. Chadwick, J.C. Moody and F. Box (1992), An Approach to Transitioning FAA air/ground Communications to a Digital System, *IEEE/AIAA 11th Digital Avionics Systems Conference Proceedings*, 5-8 Oct, Seattle WA.

[CR96] W. Chiang and R.A. Russell (1996), Simulated Annealing Metaheuristics for the Vehicle Routing Problem with Time Windows, *Annals of Operations Research* **63** : 3-27

[CS95] D.J. Castelino and N.M. Stephens (1995), Solving Frequency Assignment Problems using Tabu Thresholding, *Proceedings of Metaheuristics International Conference*, I.H. Osman and J.P. Kelly, (eds.), Breckenridge, Colorado, pp. 88-93.

[CS98] T. Clark and G.D. Smith (1998), A Practical Frequency Planning Technique for Cellular Radio, In : *Artificial Neural Nets and Genetic Algorithms*, G. Smith, N. Steele, R.A. Albrecht, (eds.), Springer Computer Science.

[CSK93] Chakrapani, Skorin-Kapov (1993), Massively Parallel Tabu Search for the Quadratic Assignment Problem, *Annals of Operations Research*, **41** : 385-403

[Dow95] K.A. Dowsland (1995), Simulated Annealing, In: *Modern Heuristic Techniques for Combinatorial Problems*, C.R.Reeves (ed.), McGraw Hill, U.K. pp 20-69.

[Dow96] K.A. Dowsland (1996), Simulated Annealing Solutions for Multi-Objective Scheduling and Timetabling, In: *Modern Heuristic Search*

Methods. V.J. Rayward-Smith, I.H. Osman et al (eds.), John Wiley and Sons. England.

[Dun et al. 98] N. Dunkin et al (1998), Towards High Order Constraint Representations for the Frequency Assignment Problem, Technical Report, CSD-TR-98-05. June, Royal Holloway and Bedford New College, London.

[DKR93] M. Duqueanton, D. Kunz and B. Ruber (1993), Channel Assignment for Cellular Radio Using Simulated Annealing, *IEEE Transactions on Vehicular Technology*, **42(1)** : 14-21.

[DV93] F. Dammeyer and S. VoB (1993), Dynamic Tabu List Management Using the Reverse Elimination Method, *Annals of Operations Research*, 41 : 31-46.

[EUC] Combinatorial Algorithms for Military Applications (Appendix 3), In: *Euclid CEPA6 - AI RTP6.4 CALMA Project Specification.*

[Fox92] B.L. Fox (1992), Uniting Probabilistic Methods for Optimisation, *Proceedings of WSC'92*, Arlington VA, U.S.A. ch 182 pp500-505.

[Fox93] B.L. Fox (1993), Integrating and Accelerating TS, SA and GAs, *Annals of Operations Research*, **41** : 47-67.

[FF96] C. Fleurent and J.A. Ferland (1996), Genetic and Hybrid Algorithms for Graph Colouring, *Annals of Operations Research*, **63** : 437-461.

[FK92] B. Friesleben and T. Kielmann, (1992), Automatic Parallelisation of Divide and Conquer Algorithms, *Lecture Notes in Computer Science*, **634** : 849-850.

[FM96] B. Friesleben and P. Merz (1996), New Genetic Local Search Operators for the Travelling Salesman Problem.
from http://www.informatik.uni-siegen.de/ pmerz/publications.html

[Gam86] A. Gamst (1986), Some Lower Bounds for a class of Frequency Assignment Problems, *IEEE Transactions on Vehicular Technology*, **35** : 8-14.

[Glo89] F. Glover (1989), Tabu Search - Part 1, *ORSA Journal Comp.* **1** : 190-206.

[Glo90] F. Glover (1990), Tabu Search - Part 2, *ORSA Journal Comp.* **2** : 4-32.

[GH94] J. Gu and X. F. Huang (1994), Efficient Local Search with Search Space Smoothing - A Case Study of the Travelling Salesman Problem, *IEEE Transactions on Systems Management and Cybernetics*, **24(5)**: 728-735.

[GJ79] M.R. Garey and D.S. Johnson (1979), *Computers and Intractability*, Freeman Co., 1979.

[GK93] M.X. Geomans and M.S. Kodialam (1993), A Lower Bound on the Expected Cost of an Optimal Assignment, *Maths Oper Res*, **18(2)** : 267-274.

[GL95] F. Glover and M. Laguna (1995), Tabu Search In : *Modern Heuristic Techniques for Combinatorial Problems*, C.R.Reeves (ed.), McGraw Hill, U.K. pp 70–150.

[GL97] Gower and R.A. Leese (1997), The Sensitivity of Channel Assignment to Constraint Specification, *Proceedings of IEEE Symposium on Electromagnetic Compatibility*, Zurich pp131-136.

[GM82] J.G. Gardiner and M.S.A Magaza (1982), Some Alternative Frequency Assignment Procedures for Mobile Radio Systems, *Radio and Electronic Engineer*, **52(4)** : 193-197.

[GSS93] Gendreau, Soriano and Salvail (1993), Solving the Maximum Clique Problem using a Tabu Search Approach, *Annals of Operations Research*, **41** : 327-341.

[GTdW93] F. Glover, E.D. Taillard and D. de Werra (1993), A Users Guide to Tabu Search. *Annals of Operations Research* **41** : 3-28.

[Hal80] W.K. Hale (1980), Frequency Assignment : Theory and Applications, *Proceedings of the IEEE*, **68** : 1497-1514.

[Hal93] M.M. Hallorsson (1993), A Still Better Performance Guarantee for Approximate Graph Colouring, *Inf Process Letters*, **45(1)** : 19-23.

[Hof94] A.G. Hoffman (1994), Dynamic Locking Heuristic - A New Graph Partitioning Algorithm, *IEEE International Symposium on Circuits and Systems* CAD(ISCAS94), London, **1(122)**:173-176.

[Hol75] J.H. Holland (1975), Adaption in Natural and Artificial Systems, *University of Michigan Press*, Ann Arbor, 1975.

[HS78] E. Horowitz and S.Sahni (1978), Branch and Bound, In : *Fundamentals of Computer Algorithms.* E. Horowitz and S. Sahni, Pitman Publishing ltd, U.S.A.

[HST97] S. Hurley, D.H. Smith and S.U. Theil (1997), FASoft : A System for Discrete Channel Frequency Assignment, *Radio Science*, **32.**

[HTS96] S. Hurley, S.U. Theil and D.H. Smith (1996), A Comparison of Local Search Algorithms for the Frequency Assignment Problem, *ACM Symposium on Applied Computing*, Philadelphia, pp251-257.

[HdW90] A. Herz and D. de Werra (1990), The Tabu Search Heuristic : How We Used It, *Annals of Maths and Artificial Intelligence*, **1** : 111-121.

[Joh et al.89] D.S. Johnson et al (1989), Optimisation by Simulated Annealing : An Experimental Evaluation; Part 1, Graph Partitioning, *Operations Research*, **37(6)** : 865-892.

[Joh et al 91] D.S. Johnson et al (1991), Optimisation by Simulated Annealing : An Experimental Evaluation; Part 2, Graph Colouring and Number Partitioning, *Operations Research*, **39(3)** : 378-406.

[JDB98] P. Jeavons, N. Dunkin and J. Bater, Why Higher Order Constraints are Necessary to Model Frequency Assignment Problems, *The 13th Biennial European Conference on Artificial Intelligence (ECAI'98)*, Brighton, U.K.

[JM96] D.S. Johnson and L.A. McGeoch (1996). The Travelling Salesman Problem. A Case Study in Local Search, In : *Local Seach in Optimisation,* E.H.L. Aarts and J.K. Lenstra (eds). Wiley, Chichester.

[KGA93] J. Kelly, B. Golden and A Assad (1993), Large Scale Controlled Rounding Using Tabu Search with Strategic Oscillation, *Annals of Operations Research*, **41** : 69-84.

[KGV83] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi (1983), Optimisation by Simulated Annealing, *Science*, **220(May)** : 671-680.

[KK94] S. Kim and S. Kim (1994), A Two-Phase Algorithm For Frequency Assignment in Cellular Mobile Systems, *IEEE Transactions on Vehicular Technology*, **43** : 542-549.

[KMD95] S.A. Kauffman, W.G. Macready and E. Dickinson (1995), Divide to Coordinate : Optimization Reconsidered. from http://www.santafe.edu/sfi/publications/Abstracts/94-06-031/abs.html

[KN96] I. Katzela and M. Naghshineh, Channel Assignment Schemes for Cellular Mobile Phone Telecommunications Systems : A Comprehensive Survey, *IEEE Personal Communications*, June 96, pp10-31.

[KT97] M. Kolonko and M.T. Tran (1997), Convergence of Simulated Annealing with Feedback Temperature Schedules, *Probability in the Engineering and Informational Sciences*, **11(3)** : 279-304.

[Lan89] T.A. Lanfear (1989), Graph Theory and Radio Frequency Assignment, *NATO EMC Analysis Project No. 5*, NATO Headquarters, 1110 Brussels.

[Lee94] R.A. Leese (1994), A Unified Approach to Problems in Radio Channel Assignment, *IMA CACM*, Oxford.

[Lee97] R.A. Leese (1997), A Fresh Look at Channel Assignment in Uniform Networks, presented at *EMC97*, Zurich.

[Lee99] R.A. Leese (1999), A Unified Approach to the Assignment of Radio Channels on a Regular Hexagonal Grid, to appear : *IEEE Transactions on Vehicular Technology.*

[Leh93] A. Lehtola (1993), Heuristic Search within Strict Time Limits, *Proceedings of Artificial Intelligence '93*, Sweden 4-7 May, pp 240-247.

[Les98] D.Lesaint (1998), Local Search and Constraint Programming for Workforce Scheduling : A Case Study, *INFORMS National Meeting Session*, Montreal, 27-29 April.

[LG93] M. Laguna and F. Glover (1993), Bandwidth Packing, a Tabu Search Approach, *Management Science* **39(4)** : 492-500.

[LH94] S.C. Lin and M.C. Hseueh (1994), Nearest Neighbour Heuristics in Accelerated Algorithms for Optimisation, *Physica A* **203(3-4)** : 369-380.

[LSM96] S. Lor H. Shen and P. Maheshwari (1996), Divide and Conquer Minimal Cut Bisectioning of Task Graphs, *Computer Systems Science and Engineering*, **11(4)** : 227-234.

[Met70] B.H. Metzger (1970), Spectrum Management Technique, presented at *38th National ORSA meeting* (Detroit, MI).

[Mor89] C. Morgernitem, Algorithms for General Graph Colouring, Dept Comp Sci, University of New Mexico Technical Report C589-16.

[MG86] C. McMillan and F. Glover (1986), The General Employee Scheduling Problem : An Integration of MS and AI, *Comp and Operations Research*, **13(5)** : 563-573.

[MKS95a] J.W.Mann, A.Kapsalis and G.D. Smith (1995), The GAmeter Toolkit, In: Applications of Modern Heuristic Techniques, V.J. Rayward-Smith (ed) Alfred Waller. pp 195-209.

[MM93] R. Mathar and S. Mattfield (1993), Channel Assignment in Cellular Radio Networks, *IEEE Transactions on Vehicular Technology*, **42(4)** : 647-656.

[MO96] O.C. Martin and S.W. Otto (1996), Combining Simulated Annealing with Local Search Heuristics, *Annals of Operations Research*, **63** : 57-75.

[MR93] E. Mooney and R. Rardin (1993), Tabu Search for a Class of Scheduling Problems, *Annals of Operations Research*, **41** : 253-278.

[MS96] J.W. Mann and G.D. Smith (1996), A Comparison of Heuristics for Telecommunications Traffic Routing, In : *Modern Heuristic Search Methods*. V.J. Rayward-Smith, I.H. Osman et al (eds.), John Wiley and Sons. England.

[MS98] J.W. Mann and G.D. Smith (1998), The Ring-loading and Ring-sizing Problem, In : *Artificial Neural Nets and Genetic Algorithms*, G. Smith, N. Steele, R.A. Albrecht, (eds.), Springer Computer Science.

[Osm93] I.H. Osman (1993), Metastrategy SA and TS algorithms for Vehicle Routing Problems, *Annals of Operations Research*, **41** : 421-451.

[OL96] I.H. Osman and G. Laporte (1996), Metaheuristics - A Bibliography, *Annals of Operations Research*, **63** : 513-623.

[Pee91] R. Peeters (1991), The Frequency Assignment Problem as a Weighted Graph Colouring Problem, Masters Thesis, Eindhoven University of Technology.

[Pro93] P. Prosser (1993), Hybrid Algorithms for the Constraint Satisfaction Problem, *Computational Intelligence*, pp 268-299.

[PL96] T. Park and C.Y. Lee (1996), Application of the Graph-Colouring Algorithm to the Frequency Assignment Problem, *Journal of the Operations Research Society of Japan*, **39(2)** : 258-265.

[PS98] A.T. Potter and N.M. Stephens (1998), A Divide and Conquer Technique to Solve the Frequency Assignment Problem, In : *Artificial Neural Nets and Genetic Algorithms*, G. Smith, N. Steele, R.A. Albrecht, (eds.), Springer Computer Science.

[Ran et al.93] R.P. Rankin et al. (1993), Considerations for Rapidly Converging GAs Designed for Application to Problems with Expensive Evaluation Functions, University of Missouri, Rolla, MO'93, UMI Order GAX93-26599.

[Ray92] A. Raychaudhuri (1992), Optimal Multiple Interval Assignments in Frequency Assignment and Traffic Phasing, *Discrete Applied Mathematics*, **40** : 319-332.

[Ray94] A. Raychaudhuri (1994), Further Results on T-colouring and Frequency Assignment Problems, *Siam Journal on Discrete Maths.* 74(4) : 605-613.

[Ree95b] C.R. Reeves (1995), Evaluation of Heuristic Performance, In : *Modern Heuristic Techniques for Combinatorial Problems*, C.R.Reeves (ed.), McGraw Hill, U.K.

[Ree95c] C.R. Reeves (1995), Genetic Algorithms, In : *Modern Heuristic Techniques for Combinatorial Problems*, C.R. Reeves (ed.), McGraw Hill, U.K.

[Ree96a] C.R.Reeves (1996), Modern Heuristic Techniques, In : *Modern Heuristic Search Methods.* V.J. Rayward-Smith, I.H. Osman et al (eds.), John Wiley and Sons. England.

[Ree96b] C.R. Reeves (1996), Heuristic Search Methods : A Review, In : *Operational Research, Keynote Papers,* D.Johnson and F.O'Brien (eds).

[Ree98] C.R. Reeves (1998), Landscapes, Operators and Heuristic Search. to appear *Annals of Operations Research* 1998

[Rob91] F.S. Roberts (1991), T-Colourings of Graphs : Recent Results and Open Problems, *Discrete Mathematics*, **93** : 229-245.

[RB95] C.R. Reeves and J. Beasley (1995), Introduction, In : *Modern Heuristic Techniques for Combinatorial Problems*, C.R.Reeves (ed.), McGraw Hill, U.K.

[RPG96] E. Rolland, H Pirkul and F. Glover (1996), Tabu Search for Graph Partitioning, *Annals of Operations Research*, 63 : 209-232.

[RS95] N.J. Radcliffe and R.D. Surry (1995), Fundamental Limitations on Search Algorithms : Evolutionary Computing in Perspective, In : *Computer Science Today : Recent Trends and Developments*, J van Leeuwen (ed.), Springer-Verlag, pp275-291.

[Sch98] A. Shaerf (1998), Local Search Techniques for Scheduling Problems, Tutorial in The 13th Biennial European Conference on Artificial Intelligence (ECAI'98), Brighton, U.K.

[Sen et al 94] M. Sengoku et al (1994), Development in Graph-Theoretic and/ or Network-Theoretic Research of Cellular Mobile Communication Channel Assignment, *IEICE Trans of Fundamentals of Electronics*, Comms and Comp Sci **E77a(7)** : 1117-1124

[Sin93] M. Sinclair (1993), Comparison of the performance of Modern Heuristics for Combinatorial Optimisation on Real Data, *Comput Operational Research*, **20(7)** : 687-695.

[Smi88] D.H Smith, Graph Colouring and Frequency Assignment, *Ars Combinatorica*, **25C** : 205-212.

[Smi et al.95] G.D Smith, A Kapsalis, V.J. Rayward-Smith and A.Kolen (1995), Report 2.1.1 : Implementation and Testing of Genetic Algorithm Approaches, technical report, EUCLID CALMA Radio Link Frequency Assignment Project, Dep. of Comput. Sci., Univ. of East Anglia, Norwich, England.

[Soh96] A. Sohn (1996), Generalized Speculative Computation of Parallel Simulated Annealing, *Annals of Operations Research*, 63 : 29-55.

[SH88] D.H. Smith and A.R. Hutchings (1988), Spectral Resource Requirements of a Fielded Network, Report - Vol2.

[SH97] D.H. Smith and S. Hurley (1997), Bounds for the Frequency Assignment Problem, *Discrete Mathematics*, **167** : 571-582.

[SHT98] D.H. Smith, S. Hurley and S.U. Theil (1998), Improving Heuristics for the Frequency Assignment Problem, *European Journal of Operations Research*, **107** : 76-86.

[Tuz92] Z.J Tuza (1992), *Graph Colouring in Linear Time, Comb Theory Series B55*, **2** : 236-243.

[TD96] J.M. Thompson and K.A. Dowsland (1996), Variants of Simulated Annealing for the Examination Timetabling Problem, *Annals of Operations Research*, 63 : 105-128.

[THL95] S. Tiorine, C. Hurkens, J.K. Lenstra (1995), Report 2.5.4 : A Review of Algorithmic Approaches to Frequency Assignment Problems, Technical Report, EUCLID CALMA RLFAP, Eindhoven University of Technology, Netherlands.

[VJH98] C.L. Valenzuela, A. Jones and S. Hurley (1998), Breeding Permutations for Minimum Span Frequency Assignment, In : *Artificial Neural Nets and Genetic Algorithms*, G. Smith, N. Steele, R.A. Albrecht, (eds.), Springer Computer Science.

[VT96] C. Voudouris and E.P.K. Tsang (1996), Partial Constraint Satisfaction Problems and Guided Local Search, In : *Proceedings of the 2nd International Conference on the Practical Application of Constraint Technology*, Practical Application Company Ltd, U.K.

[Whi98] A.T. Whitford (1998), Solving Frequency Assignment Problems with a Divide and Conquer Approach. UNICOM Modern Heuristics for Decision Support, London, December 98.

[WF96] X.D. Wang and Q.X. Fu (1996), A Frame for General Divide and Conquer Recurrences, *Information Processing Letters*, **59(1)** : 45-51.

[dWG94] D. de Werra and Y. Gay (1994), Chromatic Scheduling and Frequency Assignment, *Discrete Applied Mathematics,* **49** : 165-174.

[WR96] W. Wang and C.K. Rushforth (1996), An Adaptive Local-Search Algorithm for the Channel Assignment Problem, *IEEE Transactions on Vehicular Technology*, **45(3)** : 459-466.

[WZ93a] D. Woodruff and E. Zemel (1993), Hashing Vectors for Tabu Search, *Annals of Operations Research*, **41** : 123-137

[WZ93b] W.D. Wallis and G. Zhang (1993), On the Partitioning and Colouring of a Graph by Cliques, *Discrete Mathematics*, **120(1-3)** : 191-203.

[You] D.A. Youngs (19??), Frequency Assignment for Cellular Radio Networks, ? p233-241.

[YR98] T.Yamada and C.R. Reeves (1998), Solving the Csum Permutation Flowshop Scheduling Problem by Genetic Local Search. *Proceedings of the IEEE Evolutionary Computation.*

[ZB77] J.A. Zoellner and C.L. Beal (1977), A Breakthrough in Spectrum Conserving Frequency Assignment Technology, *IEEE Transactions on Electromagnetic Compatibility*, **19** : 313-319.

Appendix a : Results for Chapter 7 : SA vs. TS

SA : % Range

		Group A	Data	
Scenario	best	worst	average	Std Devn
1	100	100	100	0.000
2	86	86	86	0.000
3	100	100	100	0.000
4	100	100	100	0.000
5	100	100	100	0.000
6	80	80	80	0.000
7	100	100	100	0.000
8	100	100	100	0.000
9	80	80	80	0.000
10	100	100	100	0.000
11	100	100	100	0.000
12	100	100	100	0.000
13	80	80	80	0.000
14	100	100	100	0.000
15	100	100	100	0.000
16	80	80	80	0.000
17	100	100	100	0.000
18	100	100	100	0.000
19	100	100	100	0.000
20	80	80	80	0.000
21	100	100	100	0.000
22	100	100	100	0.000
23	100	100	100	0.000
24	100	100	100	0.000
25	82	82	82	0.000
26	100	100	100	0.000
27	100	100	100	0.000
28	86	86	86	0.000
29	100	100	100	0.000
30	86	86	86	0.000
31	100	100	100	0.000
32	100	100	100	0.000
33	80	80	80	0.000
34	100	100	100	0.000
35	100	100	100	0.000
36	86	86	86	0.000
37	91	91	91	0.000
		Group B	Data	
Scenario	best	worst	average	Std Devn
1	100	100	100	0.000
2	100	100	100	0.000
3	100	100	100	0.000
4	100	100	100	0.000
5	100	100	100	0.000
6	100	100	100	0.000
7	100	100	100	0.000
8	100	100	100	0.000
9	100	100	100	0.000

SA : % Frequencies

		Group A	Data	
Scenario	best	worst	average	Std Devn
1	98	100	99	0.904
2	32	32	32	0.000
3	55	76	68	6.478
4	65	80	73	4.333
5	100	100	100	0.000
6	15	15	15	0.000
7	85	91	88	2.050
8	95	100	98	1.678
9	15	15	15	0.000
10	92	99	96	2.330
11	80	85	82	2.321
12	100	100	100	0.000
13	10	10	10	0.000
14	98	100	98	0.728
15	100	100	100	0.000
16	10	10	10	0.000
17	98	100	99	0.756
18	78	88	83	3.523
19	98	100	100	0.700
20	15	15	15	0.000
21	96	100	99	1.278
22	82	92	86	3.375
23	78	88	81	3.653
24	100	100	100	0.000
25	15	15	15	0.000
26	98	100	99	0.833
27	100	100	100	0.000
28	38	38	38	0.000
29	100	100	100	0.000
30	20	20	20	0.000
31	98	100	99	0.833
32	100	100	100	0.000
33	10	10	10	0.000
34	98	100	99	0.639
35	100	100	100	0.000
36	45	45	45	0.000
37	45	45	45	0.000
		Group B	Data	
Scenario	best	worst	average	Std Devn
1	85	95	87	3.642
2	85	92	88	1.884
3	100	100	100	0.000
4	100	100	100	0.000
5	99	100	100	0.452
6	100	100	100	0.000
7	100	100	100	0.000
8	100	100	100	0.000
9	100	100	100	0.000

Tabu Cost : TNET Data

		Group A Data		
Scenario	best	worst	average	std deviation
1	0.0	0.0	0.00	0.000
2	0.0	0.0	0.00	0.000
3	0.0	0.0	0.00	0.000
4	0.0	16.4	3.10	5.730
5	0.0	0.0	0.00	0.000
6	0.0	0.0	0.00	0.000
7	0.0	0.0	0.00	0.000
8	0.0	0.0	0.00	0.000
9	0.0	0.0	0.00	0.000
10	0.0	0.0	0.00	0.000
11	7.2	30.2	22.19	7.113
12	0.0	0.1	0.05	0.050
13	0.0	0.0	0.00	0.000
14	0.0	0.0	0.00	0.000
15	0.0	0.0	0.00	0.000
16	0.0	0.0	0.00	0.000
17	0.0	0.0	0.00	0.000
18	3.9	343.8	95.75	107.763
19	0.0	0.1	0.01	0.033
20	0.0	0.0	0.00	0.000
21	0.0	0.0	0.00	0.000
22	1.2	80.2	37.40	22.522
23	0.0	24.0	8.00	8.000
24	0.0	0.7	0.11	0.232
25	0.0	0.0	0.00	0.000
26	0.0	0.0	0.00	0.000
27	0.0	0.0	0.00	0.000
28	0.0	0.0	0.00	0.000
29	60.8	94.4	72.21	11.048
30	0.0	0.0	0.00	0.000
31	0.0	0.0	0.00	0.000
32	0.0	0.0	0.00	0.000
33	0.0	0.0	0.00	0.000
34	0.0	0.0	0.00	0.000
35	0.0	0.0	0.00	0.000
36	0.0	0.0	0.00	0.000
37	0.0	0.0	0.00	0.000
		Group B Data		
Scenario	best	worst	average	std deviation
1	13.0	45.7	31.49	10.278
2	46.9	112.9	75.73	19.170
3	41.6	44.5	42.89	0.949
4	35.7	44.8	41.99	3.641
5	1.7	2.5	1.96	0.255
6	1.1	2.6	1.90	0.480
7	72.7	79.1	74.48	2.493
8	23.0	68.0	30.53	14.202
9	2.8	24.0	6.73	6.629

Tabu Time : TNET Data

		Group B	Data	
Scenario	best	worst	average	std deviation
1	24	26	24.38	0.696
2	0	0	0.00	0.000
3	0	0	0.00	0.000
4	9	18	13.63	2.955
5	44	89	60.50	18.214
6	0	0	0.00	0.000
7	4	4	4.00	0.000
8	4	4	4.00	0.000
9	0	0	0.00	0.000
10	15	15	15.00	0.000
11	58	134	99.00	29.479
12	37	105	69.50	31.068
13	0	0	0.00	0.000
14	31	32	31.13	0.331
15	26	29	27.25	0.829
16	0	0	0.00	0.000
17	75	76	75.13	0.331
18	87	453	337.00	109.736
19	37	104	55.25	25.371
20	0	0	0.00	0.000
21	31	31	31.00	0.000
22	251	477	373.13	59.501
23	29	48	36.13	6.936
24	30	91	49.50	25.045
25	0	0	0.00	0.000
26	39	40	39.13	0.331
27	26	27	26.13	0.331
28	0	0	0.00	0.000
29	609	639	624.00	10.595
30	0	0	0.00	0.000
31	82	83	82.13	0.331
32	43	43	43.00	0.000
33	0	0	0.00	0.000
34	66	86	69.13	6.392
35	11	15	12.38	1.317
36	0	0	0.00	0.000
37	0	0	0.00	0.000
		Group B	Data	
Scenario	best	worst	average	std deviation
1	33	77	52.63	15.386
2	220	365	281.13	40.160
3	783	791	785.75	2.332
4	491	626	536.75	38.065
5	782	787	783.88	2.027
6	231	409	328.25	70.567
7	765	784	777.00	7.089
8	784	790	786.13	2.204
9	783	788	785.25	1.714

Tabu % Freqs : TNET Data

		Group A	Data	
Scenario	best	worst	average	std deviation
1	95	100	97.50	1.581
2	32	32	32.00	0.000
3	20	24	21.75	1.479
4	65	80	73.63	5.023
5	98	100	99.50	0.866
6	15	15	15.00	0.000
7	35	38	35.88	0.927
8	78	90	82.88	3.219
9	15	15	15.00	0.000
10	48	55	49.88	2.147
11	75	85	80.63	3.314
12	98	100	99.75	0.661
13	10	10	10.00	0.000
14	58	66	60.13	2.421
15	95	100	97.88	1.269
16	10	10	10.00	0.000
17	66	74	70.63	2.176
18	82	90	86.00	2.784
19	98	100	98.25	0.661
20	15	15	15.00	0.000
21	51	58	54.63	2.342
22	80	90	87.25	3.419
23	75	88	79.88	3.480
24	98	100	99.75	0.661
25	15	15	15.00	0.000
26	60	66	61.63	2.118
27	78	90	84.13	4.594
28	38	38	38.00	0.000
29	98	100	99.50	0.866
30	20	20	20.00	0.000
31	66	81	73.00	4.664
32	88	95	93.00	2.345
33	10	10	10.00	0.000
34	61	74	69.50	3.708
35	68	82	77.25	4.323
36	45	45	45.00	0.000
37	45	45	45.00	0.000
		Group B	Data	
Scenario	best	worst	average	std deviation
1	80	90	84.00	3.428
2	82	90	85.25	2.947
3	84	89	86.25	1.479
4	95	98	97.25	1.299
5	94	98	95.38	1.218
6	100	100	100.00	0.000
7	95	100	98.88	1.691
8	84	94	90.63	3.238
9	91	98	93.75	2.278

Tabu % Range : TNET Data

			Group A	Data
Scenario	best	worst	average	std deviation
1	100	100	100.00	0.000
2	86	86	86.00	0.000
3	100	100	100.00	0.000
4	100	100	100.00	0.000
5	100	100	100.00	0.000
6	80	80	80.00	0.000
7	100	100	100.00	0.000
8	100	100	100.00	0.000
9	80	80	80.00	0.000
10	100	100	100.00	0.000
11	100	100	100.00	0.000
12	100	100	100.00	0.000
13	80	80	80.00	0.000
14	100	100	100.00	0.000
15	100	100	100.00	0.000
16	80	80	80.00	0.000
17	100	100	100.00	0.000
18	100	100	100.00	0.000
19	100	100	100.00	0.000
20	80	80	80.00	0.000
21	100	100	100.00	0.000
22	100	100	100.00	0.000
23	100	100	100.00	0.000
24	100	100	100.00	0.000
25	82	82	82.00	0.000
26	100	100	100.00	0.000
27	100	100	100.00	0.000
28	86	86	86.00	0.000
29	100	100	100.00	0.000
30	86	86	86.00	0.000
31	100	100	100.00	0.000
32	100	100	100.00	0.000
33	80	80	80.00	0.000
34	100	100	100.00	0.000
35	100	100	100.00	0.000
36	86	86	86.00	0.000
37	91	91	91.00	0.000
			Group B	Data
Scenario	best	worst	average	std deviation
1	100	100	100.00	0.000
2	100	100	100.00	0.000
3	100	100	100.00	0.000
4	100	100	100.00	0.000
5	100	100	100.00	0.000
6	100	100	100.00	0.000
7	100	100	100.00	0.000
8	100	100	100.00	0.000
9	100	100	100.00	0.000

SA vs TS : Average Cost and Time : TNET Data

	Average Cost Data Group A		Average Time Data Group A	
Scenario	SA	TS	SA	TS
1	0.40	0.00	597	24
2	0.00	0.00	0	0
3	0.00	0.00	0	0
4	0.00	3.10	2	14
5	2.80	0.00	675	61
6	0.00	0.00	0	0
7	0.00	0.00	73	4
8	0.00	0.00	241	4
9	0.00	0.00	0	0
10	0.00	0.00	285	15
11	0.00	22.19	237	99
12	2.31	0.05	661	70
13	0.00	0.00	0	0
14	0.00	0.00	562	31
15	2.39	0.00	642	27
16	0.00	0.00	0	0
17	1.06	0.00	750	75
18	0.00	95.75	532	337
19	2.01	0.01	683	55
20	0.00	0.00	0	0
21	0.00	0.00	732	31
22	1.37	37.40	743	373
23	0.00	8.00	3	36
24	1.87	0.11	615	50
25	0.00	0.00	0	0
26	0.00	0.00	667	39
27	0.54	0.00	680	26
28	0.00	0.00	0	0
29	19.21	72.21	784	624
30	0.00	0.00	0	0
31	1.80	0.00	774	82
32	1.94	0.00	760	43
33	0.00	0.00	0	0
34	0.00	0.00	607	69
35	0.00	0.00	215	12
36	0.00	0.00	0	0
37	0.00	0.00	0	0
	Average Cost Data Group B		Average Time Data Group B	
Scenario	SA	TS	SA	TS
1	13.30	31.49	596	53
2	44.07	75.73	780	281
3	38.89	42.89	784	786
4	29.01	41.99	783	537
5	22.79	1.96	782	784
6	7.17	1.90	780	328
7	32.71	74.48	781	777
8	49.39	30.53	782	786
9	31.59	6.73	782	785

SA vs TS : Best Cost and Time : TNET Data

	Best	Cost	Time 2	Best Cost
	Group A	Data	Group A	Data
Scenario	SA	TS	SA	TS
1	0.00	0.00	574	24
2	0.00	0.00	0	0
3	0.00	0.00	0	0
4	0.00	0.00	1	9
5	2.00	0.00	685	44
6	0.00	0.00	0	0
7	0.00	0.00	28	4
8	0.00	0.00	222	4
9	0.00	0.00	0	0
10	0.00	0.00	235	15
11	0.00	7.20	53	134
12	1.30	0.00	652	37
13	0.00	0.00	0	0
14	0.00	0.00	488	31
15	1.20	0.00	646	26
16	0.00	0.00	0	0
17	0.00	0.00	701	75
18	0.00	3.90	426	451
19	1.20	0.00	660	37
20	0.00	0.00	0	0
21	0.00	0.00	655	31
22	0.00	1.20	685	477
23	0.00	0.00	1	29
24	0.80	0.00	630	30
25	0.00	0.00	0	0
26	0.00	0.00	547	39
27	0.00	0.00	498	26
28	0.00	0.00	0	0
29	16.50	60.80	781	625
30	0.00	0.00	0	0
31	0.00	0.00	740	82
32	0.00	0.00	639	43
33	0.00	0.00	0	0
34	0.00	0.00	470	66
35	0.00	0.00	32	11
36	0.00	0.00	0	0
37	0.00	0.00	0	0
	Best	Cost	Time 2	Best Cost
	Group B	Data	Group B	Data
Scenario	SA	TS	SA	TS
1	13.00	13.00	576	59
2	29.30	46.90	780	293
3	29.80	41.60	782	791
4	16.00	35.70	797	545
5	17.60	1.70	781	783
6	4.30	1.10	781	409
7	28.10	72.70	781	765
8	44.40	23.00	783	784
9	23.60	2.80	783	784

TS and SA : Average % Frequencies

%Freqs	Group A	Data
	TABU	SA
1	98	100
2	32	32
3	21.5	69
4	75	72
5	100	100
6	15	15
7	36	88
8	82	98
9	15	15
10	49	96
11	81	82
12	100	100
13	10	10
14	59.5	98
15	98	100
16	10	10
17	71	99
18	86.5	82
19	98	100
20	15	15
21	54.5	99
22	89	88
23	80	80
24	100	100
25	15	15
26	60.5	99
27	83.5	100
28	38	38
29	100	100
30	20	20
31	72	99
32	93.5	100
33	10	10
34	70	99
35	78	100
36	45	45
37	45	45
Group B	Average	Average
1	83.5	85
2	85	88
3	86	100
4	98	100
5	95	100
6	100	100
7	100	100
8	90.5	100
9	93	100

SA, TS vs. LRTS : Average Cost and Time

Group A	Average	Cost		Average	Time	
Scenario	SA	TS	LRTS	SA	TS	LRTS
4	0.00	3.10	0.00	2	14	17
11	0.00	22.19	1.80	237	99	475
18	0.00	95.75	14.26	532	337	907
22	1.37	37.40	5.62	743	373	885
23	0.00	8.00	0.00	3	36	45
29	19.21	72.21	61.42	784	624	1562
Group B	Average	Cost		Average	Time	
Scenario	SA	TS	LRTS	SA	TS	LRTS
1	13.30	31.49	20.38	596	53	417
2	44.07	75.73	44.98	780	281	812
3	38.89	42.89	0.94	784	786	1565
4	29.01	41.99	36.02	783	537	1563
5	22.79	1.96	0.40	782	784	1513
6	7.17	1.90	0.30	780	328	1556
7	32.71	74.48	65.14	781	777	1562
8	49.39	30.53	0.94	782	786	1477
9	31.59	6.73	1.08	782	785	1565

SA : Cost

SA		Group A	Data	
Scenario	best	worst	average	Std Devn
1	0	1.7	0.40	0.593
2	0	0	0.00	0.000
3	0	0	0.00	0.000
4	0	0	0.00	0.000
5	2	3.8	2.80	0.670
6	0	0	0.00	0.000
7	0	0	0.00	0.000
8	0	0	0.00	0.000
9	0	0	0.00	0.000
10	0	0	0.00	0.000
11	0	0	0.00	0.000
12	1.3	3.8	2.31	0.759
13	0	0	0.00	0.000
14	0	0	0.00	0.000
15	1.2	3.8	2.39	0.743
16	0	0	0.00	0.000
17	0	2.6	1.06	1.227
18	0	0	0.00	0.000
19	1.2	3.4	2.01	0.687
20	0	0	0.00	0.000
21	0	0	0.00	0.000
22	0	4.5	1.37	1.754
23	0	0	0.00	0.000
24	0.8	3.2	1.87	0.684
25	0	0	0.00	0.000
26	0	0	0.00	0.000
27	0	2.5	0.54	0.916
28	0	0	0.00	0.000
29	16.5	23.8	19.21	2.367
30	0	0	0.00	0.000
31	0	3.8	1.80	1.054
32	0	5.3	1.94	1.733
33	0	0	0.00	0.000
34	0	0	0.00	0.000
35	0	0	0.00	0.000
36	0	0	0.00	0.000
37	0	0	0.00	0.000
Group B				
Scenario	best	worst	average	Std Devn
1	13	14.1	13.30	0.385
2	29.3	58.7	44.07	10.897
3	29.8	51.5	38.89	6.952
4	16	41.5	29.01	8.280
5	17.6	28.9	22.79	3.873
6	4.3	10.9	7.17	1.826
7	28.1	37	32.71	3.448
8	44.4	55.2	49.39	4.036
9	23.6	38.6	31.59	5.173

SA : Time(s)

		Group A	Data	
Scenario	best	worst	average	Std Devn
1	574	615	597	15.406
2	0	0	0	0.000
3	0	1	0	0.495
4	1	3	2	0.700
5	654	691	675	14.696
6	0	0	0	0.000
7	28	122	73	30.773
8	222	255	241	10.780
9	0	0	0	0.000
10	235	333	285	28.104
11	53	308	237	81.426
12	643	680	661	13.260
13	0	0	0	0.000
14	488	666	562	53.992
15	612	668	642	19.719
16	0	0	0	0.000
17	701	781	750	34.272
18	426	650	532	87.323
19	653	703	683	18.359
20	0	0	0	0.000
21	655	774	732	37.317
22	685	780	743	40.897
23	1	5	3	1.678
24	598	632	615	13.266
25	0	0	0	0.000
26	547	777	667	64.009
27	498	780	680	94.760
28	0	1	0	0.350
29	781	799	784	6.299
30	0	0	0	0.000
31	740	781	774	14.060
32	639	781	760	49.399
33	0	0	0	0.000
34	470	678	607	60.952
35	32	353	215	106.033
36	0	1	0	0.495
37	0	1	0	0.495
		Group B	Data	
Scenario	best	worst	average	Std Devn
1	574	619	596	19.820
2	780	781	780	0.350
3	782	799	784	5.949
4	781	797	783	5.599
5	781	782	782	0.495
6	780	781	780	0.452
7	781	781	781	0.000
8	782	783	782	0.495
9	782	783	782	0.452

Appendix b : Results for Chapter 8 : DCSA vs. DCTS

DCSA Cost : TNET Data

			Group A	Data
Scenario	Best	Worst	Average	std deviation
1	0.0	0.1	0.01	0.030
2	0.0	0.0	0.00	0.000
3	0.0	0.0	0.00	0.000
4	0.0	0.0	0.00	0.000
5	0.0	0.3	0.10	0.089
6	0.0	0.0	0.00	0.000
7	0.0	0.0	0.00	0.000
8	0.0	0.0	0.00	0.000
9	0.0	0.0	0.00	0.000
10	0.0	0.0	0.00	0.000
11	0.0	0.0	0.00	0.000
12	0.0	0.1	0.01	0.030
13	0.0	0.0	0.00	0.000
14	0.0	0.0	0.00	0.000
15	0.0	0.0	0.00	0.000
16	0.0	0.0	0.00	0.000
17	0.0	0.0	0.00	0.000
18	0.0	0.0	0.00	0.000
19	0.0	0.2	0.07	0.064
20	0.0	0.0	0.00	0.000
21	0.0	0.0	0.00	0.000
22	0.0	0.0	0.00	0.000
23	0.0	0.0	0.00	0.000
24	0.0	0.3	0.14	0.102
25	0.0	0.0	0.00	0.000
26	0.0	0.0	0.00	0.000
27	0.0	0.0	0.00	0.000
28	0.0	0.0	0.00	0.000
29	0.0	23.0	8.69	8.460
30	0.0	0.0	0.00	0.000
31	0.0	0.0	0.00	0.000
32	0.0	0.1	0.01	0.030
33	0.0	0.0	0.00	0.000
34	0.0	0.0	0.00	0.000
35	0.0	0.0	0.00	0.000
36	0.0	0.0	0.00	0.000
37	0.0	0.0	0.00	0.000

			Group B	Data
Scenario	Best	Worst	Average	std deviation
1	13.0	13.0	13.00	0.000
2	28.8	57.2	35.92	9.541
3	5.7	10.0	8.03	1.356
4	5.9	14.1	8.46	2.909
5	1.0	3.0	1.97	0.648
6	0.9	2.0	1.55	0.326
7	8.7	47.7	17.09	10.624
8	14.2	19.3	16.67	1.423
9	1.9	6.8	4.05	1.384

b

			Group A	Data
Scenario	Best	Worst	Average	std deviation
1	281	399	310.7	33.25
2	0	0	0.0	0.00
3	0	40	4.0	12.00
4	1	82	10.6	23.83
5	346	465	393.6	33.19
6	0	0	0.0	0.00
7	28	123	52.6	26.74
8	90	166	103.5	21.84
9	0	0	0.0	0.00
10	81	200	119.7	29.41
11	10	93	58.1	22.95
12	324	397	353.7	18.99
13	0	0	0.0	0.00
14	167	245	199.3	21.50
15	219	401	322.6	43.79
16	0	0	0.0	0.00
17	191	326	242.3	36.31
18	104	169	127.4	18.30
19	329	476	366.1	40.35
20	0	0	0.0	0.00
21	209	319	251.5	37.94
22	144	227	173.0	22.90
23	2	81	10.7	23.45
24	329	375	354.4	12.49
25	0	0	0.0	0.00
26	187	256	219.1	20.08
27	102	194	145.4	26.04
28	0	1	0.1	0.30
29	312	345	332.5	13.50
30	0	0	0.0	0.00
31	304	408	337.9	27.69
32	126	249	213.2	31.84
33	0	0	0.0	0.00
34	95	201	166.5	29.12
35	12	95	51.6	21.98
36	0	1	0.1	0.30
37	0	1	0.1	0.30
			Group B	Data
Scenario	Best	Worst	Average	std deviation
1	85	230	109.2	40.61
2	121	242	150.0	32.44
3	505	593	525.6	33.61
4	264	386	335.0	27.80
5	502	590	551.0	30.53
6	495	551	515.6	18.08
7	342	345	343.9	0.83
8	507	616	533.9	42.63
9	501	585	518.5	27.45

DCSA % Range : TNET Data

			Group A	Data
Scenario	Best	Worst	Average	std deviation
1	100	100	100.0	0.000
2	86	86	86.0	0.000
3	100	100	100.0	0.000
4	100	100	100.0	0.000
5	100	100	100.0	0.000
6	80	80	80.0	0.000
7	100	100	100.0	0.000
8	100	100	100.0	0.000
9	80	80	80.0	0.000
10	98	100	99.8	0.600
11	100	100	100.0	0.000
12	100	100	100.0	0.000
13	80	80	80.0	0.000
14	100	100	100.0	0.000
15	100	100	100.0	0.000
16	80	80	80.0	0.000
17	100	100	100.0	0.000
18	100	100	100.0	0.000
19	100	100	100.0	0.000
20	80	80	80.0	0.000
21	100	100	100.0	0.000
22	100	100	100.0	0.000
23	100	100	100.0	0.000
24	100	100	100.0	0.000
25	82	82	82.0	0.000
26	100	100	100.0	0.000
27	100	100	100.0	0.000
28	86	86	86.0	0.000
29	100	100	100.0	0.000
30	86	86	86.0	0.000
31	100	100	100.0	0.000
32	100	100	100.0	0.000
33	80	80	80.0	0.000
34	100	100	100.0	0.000
35	100	100	100.0	0.000
36	86	86	86.0	0.000
37	91	91	91.0	0.000
			Group B	Data
Scenario	Best	Worst	Average	std deviation
1	100	100	100.0	0.000
2	100	100	100.0	0.000
3	100	100	100.0	0.000
4	100	100	100.0	0.000
5	100	100	100.0	0.000
6	100	100	100.0	0.000
7	100	100	100.0	0.000
8	100	100	100.0	0.000
9	100	100	100.0	0.000

b

			Group A	Data
Scenario	Best	Worst	Average	std deviation
1	98	100	99.4	0.92
2	32	32	32.0	0.00
3	21	21	21.0	0.00
4	72	88	76.5	4.84
5	100	100	100.0	0.00
6	15	15	15.0	0.00
7	36	92	73.7	21.99
8	92	100	96.9	2.43
9	15	15	15.0	0.00
10	92	100	95.0	2.19
11	78	85	81.0	2.53
12	100	100	100.0	0.00
13	10	10	10.0	0.00
14	96	100	98.8	1.17
15	98	100	99.6	0.80
16	10	10	10.0	0.00
17	99	100	99.5	0.50
18	78	88	83.1	3.65
19	98	100	99.8	0.60
20	15	15	15.0	0.00
21	96	100	98.6	1.11
22	82	90	86.0	3.07
23	70	85	77.6	4.10
24	100	100	100.0	0.00
25	15	15	15.0	0.00
26	99	100	99.9	0.30
27	100	100	100.0	0.00
28	38	38	38.0	0.00
29	100	100	100.0	0.00
30	20	20	20.0	0.00
31	96	100	99.4	1.20
32	100	100	100.0	0.00
33	10	10	10.0	0.00
34	98	100	99.4	0.66
35	100	100	100.0	0.00
36	45	45	45.0	0.00
37	45	45	45.0	0.00
			Group B	Data
Scenario	Best	Worst	Average	std deviation
1	80	92	86.5	3.83
2	85	95	89.1	3.01
3	99	100	99.9	0.30
4	100	100	100.0	0.00
5	100	100	100.0	0.00
6	100	100	100.0	0.00
7	100	100	100.0	0.00
8	100	100	100.0	0.00
9	99	100	99.9	0.30

DCTS Cost : TNET Data

			Group A	Data
Scenario	Best	Worst	Average	std deviation
1	0.0	0.0	0.00	0.000
2	0.0	0.0	0.00	0.000
3	0.0	0.0	0.00	0.000
4	0.0	8.0	0.80	2.400
5	0.0	0.0	0.00	0.000
6	0.0	0.0	0.00	0.000
7	0.0	0.0	0.00	0.000
8	0.0	0.0	0.00	0.000
9	0.0	0.0	0.00	0.000
10	0.0	0.0	0.00	0.000
11	0.0	7.9	2.75	3.093
12	0.0	0.0	0.00	0.000
13	0.0	0.0	0.00	0.000
14	0.0	0.0	0.00	0.000
15	0.0	0.0	0.00	0.000
16	0.0	0.0	0.00	0.000
17	0.0	0.0	0.00	0.000
18	0.4	48.4	17.48	18.023
19	0.0	0.0	0.00	0.000
20	0.0	0.0	0.00	0.000
21	0.0	0.0	0.00	0.000
22	6.3	60.4	34.29	24.394
23	0.0	8.0	0.80	2.400
24	0.0	0.2	0.05	0.081
25	0.0	0.0	0.00	0.000
26	0.0	0.0	0.00	0.000
27	0.0	0.0	0.00	0.000
28	0.0	0.0	0.00	0.000
29	23.7	96.2	52.86	20.286
30	0.0	0.0	0.00	0.000
31	0.0	0.0	0.00	0.000
32	0.0	0.1	0.03	0.046
33	0.0	0.0	0.00	0.000
34	0.0	0.0	0.00	0.000
35	0.0	0.0	0.00	0.000
36	0.0	0.0	0.00	0.000
37	0.0	0.0	0.00	0.000
			Group B	Data
Scenario	Best	Worst	Average	std deviation
1	13.0	21.0	13.80	2.400
2	37.6	96.5	51.86	15.925
3	4.4	9.8	6.36	1.591
4	19.4	43.8	33.01	7.652
5	0.6	5.3	2.30	1.440
6	0.7	1.9	1.29	0.413
7	66.2	135.1	108.93	19.408
8	2.5	10.2	4.71	2.312
9	0.8	5.0	2.72	1.317

b

			Group A	Data
Scenario	Best	Worst	Average	std deviation
1	50	126	71.6	25.742
2	0	1	0.4	0.490
3	0	40	4.4	11.876
4	8	118	34.8	30.544
5	121	226	170.6	38.066
6	0	1	0.1	0.300
7	22	101	31.3	23.242
8	14	94	23.3	23.576
9	0	1	0.1	0.300
10	40	117	50.3	22.325
11	160	399	288.2	53.734
12	68	324	182.2	80.533
13	0	0	0.0	0.000
14	62	147	74.1	24.394
15	50	268	138.3	81.728
16	0	0	0.0	0.000
17	117	210	136.6	26.192
18	317	451	336.5	38.656
19	67	266	140.9	50.240
20	0	0	0.0	0.000
21	74	160	91.0	23.660
22	320	442	336.3	35.355
23	26	167	55.9	42.500
24	89	343	212.2	95.028
25	0	0	0.0	0.000
26	70	80	74.4	2.871
27	63	337	121.1	106.987
28	0	0	0.0	0.000
29	555	615	585.8	19.338
30	0	0	0.0	0.000
31	105	138	122.8	8.987
32	68	345	282.1	105.831
33	0	0	0.0	0.000
34	84	96	88.7	3.551
35	32	38	33.7	2.326
36	0	0	0.0	0.000
37	0	0	0.0	0.000
			Group B	Data
Scenario	Best	Worst	Average	std deviation
1	125	270	160.3	42.800
2	306	439	325.1	38.072
3	852	932	887.1	25.816
4	513	651	577.3	39.085
5	816	867	841.8	15.728
6	403	452	418.8	16.290
7	572	659	610.7	23.431
8	870	880	875.2	3.156
9	844	871	857.5	7.446

DCTS % Range : TNET Data

			Group A	Data
Scenario	Best	Worst	Average	std deviation
1	100	100	100	0.000
2	86	86	86	0.000
3	100	100	100	0.000
4	100	100	100	0.000
5	100	100	100	0.000
6	80	80	80	0.000
7	100	100	100	0.000
8	100	100	100	0.000
9	80	80	80	0.000
10	100	100	100	0.000
11	100	100	100	0.000
12	100	100	100	0.000
13	80	80	80	0.000
14	100	100	100	0.000
15	100	100	100	0.000
16	80	80	80	0.000
17	100	100	100	0.000
18	100	100	100	0.000
19	100	100	100	0.000
20	80	80	80	0.000
21	100	100	100	0.000
22	100	100	100	0.000
23	100	100	100	0.000
24	100	100	100	0.000
25	82	82	82	0.000
26	100	100	100	0.000
27	100	100	100	0.000
28	86	86	86	0.000
29	100	100	100	0.000
30	86	86	86	0.000
31	100	100	100	0.000
32	100	100	100	0.000
33	80	80	80	0.000
34	100	100	100	0.000
35	100	100	100	0.000
36	86	86	86	0.000
37	91	91	91	0.000
			Group B	Data
Scenario	Best	Worst	Average	std deviation
1	100	100	100	0.000
2	100	100	100	0.000
3	100	100	100	0.000
4	100	100	100	0.000
5	100	100	100	0.000
6	100	100	100	0.000
7	100	100	100	0.000
8	100	100	100	0.000
9	100	100	100	0.000

b

			Group A	Data
Scenario	Best	Worst	Average	std deviation
1	98	100	99.8	0.60
2	32	32	32.0	0.00
3	21	21	21.0	0.00
4	68	85	78.1	5.43
5	100	100	100.0	0.00
6	15	15	15.0	0.00
7	71	80	74.6	2.76
8	90	100	96.1	3.42
9	15	15	15.0	0.00
10	79	85	81.1	1.70
11	80	90	85.8	3.82
12	98	100	99.8	0.60
13	10	10	10.0	0.00
14	88	92	89.5	1.20
15	98	100	99.8	0.60
16	10	10	10.0	0.00
17	89	96	92.2	2.04
18	85	92	88.8	2.36
19	98	100	99.6	0.80
20	15	15	15.0	0.00
21	82	86	84.2	1.33
22	88	92	89.8	1.08
23	70	92	82.0	5.20
24	98	100	99.8	0.60
25	15	15	15.0	0.00
26	89	92	90.7	1.19
27	100	100	100.0	0.00
28	38	38	38.0	0.00
29	100	100	100.0	0.00
30	20	20	20.0	0.00
31	89	92	90.4	1.02
32	100	100	100.0	0.00
33	10	10	10.0	0.00
34	86	94	90.7	2.28
35	98	100	99.8	0.60
36	45	45	45.0	0.00
37	45	45	45.0	0.00
			Group B	Data
Scenario	Best	Worst	Average	std deviation
1	80	92	88.8	3.37
2	85	98	91.6	4.10
3	96	100	98.7	1.19
4	100	100	100.0	0.00
5	99	100	99.8	0.40
6	98	100	99.8	0.60
7	100	100	100.0	0.00
8	99	100	99.8	0.40
9	99	100	99.7	0.46

DCSA vs DCTS :Average Cost and Time : TNET Data

	Average Cost		Average Time	
	Group A Data		Group A Data	
Scenario	DCSA	DCTS	DCSA	DCTS
1	0.01	0.00	311	72
2	0.00	0.00	0	0
3	0.00	0.00	4	4
4	0.00	0.80	11	35
5	0.10	0.00	394	171
6	0.00	0.00	0	0
7	0.00	0.00	53	31
8	0.00	0.00	104	23
9	0.00	0.00	0	0
10	0.00	0.00	120	50
11	0.00	2.75	58	288
12	0.01	0.00	354	182
13	0.00	0.00	0	0
14	0.00	0.00	199	74
15	0.00	0.00	323	138
16	0.00	0.00	0	0
17	0.00	0.00	242	137
18	0.00	17.48	127	337
19	0.07	0.00	366	141
20	0.00	0.00	0	0
21	0.00	0.00	252	91
22	0.00	34.29	173	336
23	0.00	0.80	11	56
24	0.14	0.05	354	212
25	0.00	0.00	0	0
26	0.00	0.00	219	74
27	0.00	0.00	145	121
28	0.00	0.00	0	0
29	8.69	52.86	333	586
30	0.00	0.00	0	0
31	0.00	0.00	338	123
32	0.01	0.03	213	282
33	0.00	0.00	0	0
34	0.00	0.00	167	89
35	0.00	0.00	52	34
36	0.00	0.00	0	0
37	0.00	0.00	0	0

	Average Cost		Time	Average
	Group B	Data	Data	Group B
Scenario	DCSA	DCTS	DCSA	DCTS
1	13.00	13.80	109	160
2	35.92	51.86	150	325
3	8.03	6.36	526	887
4	8.46	33.01	335	577
5	1.97	2.30	551	842
6	1.55	1.29	516	419
7	17.09	108.93	344	611
8	16.67	4.71	534	875
9	4.05	2.72	519	858

b

	Best	Cost	Time 2	Best Cost
	Group A	Data	Group A	Data
Scenario	DCSA	DCTS	DCSA	DCTS
1	0.0	0.0	281	50
2	0.0	0.0	0	0
3	0.0	0.0	0	0
4	0.0	0.0	1	8
5	0.0	0.0	346	121
6	0.0	0.0	0	0
7	0.0	0.0	28	22
8	0.0	0.0	90	14
9	0.0	0.0	0	0
10	0.0	0.0	81	40
11	0.0	0.0	10	160
12	0.0	0.0	324	68
13	0.0	0.0	0	0
14	0.0	0.0	167	62
15	0.0	0.0	219	50
16	0.0	0.0	0	0
17	0.0	0.0	191	117
18	0.0	0.4	104	320
19	0.0	0.0	334	67
20	0.0	0.0	0	0
21	0.0	0	209	74
22	0.0	6.3	144	321
23	0.0	0.0	2	26
24	0.0	0.0	350	115
25	0.0	0.0	0	0
26	0.0	0.0	187	70
27	0.0	0.0	102	63
28	0.0	0.0	0	0
29	0.0	23.7	312	612
30	0.0	0.0	0	0
31	0.0	0.0	304	105
32	0.0	0.0	126	68
33	0.0	0.0	0	0
34	0.0	0.0	95	84
35	0.0	0.0	12	32
36	0.0	0.0	0	0
37	0.0	0.0	0	0
	Best	Cost	Time 2	Best Cost
	Group B	Data	Group B	Data
Scenario	DCSA	DCTS	DCSA	DCTS
1	13.0	13.0	85	128
2	28.8	37.6	136	315
3	5.7	4.4	506	870
4	5.9	19.4	336	628
5	1.0	0.6	502	846
6	0.9	0.7	496	452
7	8.7	66.2	344	659
8	14.2	2.5	507	876
9	1.9	0.8	503	855

Appendix c : Results for Chapter 9 : DCSA vs. SA and DCTS vs. TS

DCSA vs SA : Best Cost and Time : TNET Data

		Group A Data		
	Best Cost		Time 2 Best Cost	
Scenario	SA	DCSA	SA	DCSA
1	0.0	0.0	574	281
2	0.0	0.0	0	0
3	0.0	0.0	0	0
4	0.0	0.0	1	1
5	2.0	0.0	685	346
6	0.0	0.0	0	0
7	0.0	0.0	28	28
8	0.0	0.0	222	90
9	0.0	0.0	0	0
10	0.0	0.0	235	81
11	0.0	0.0	53	10
12	1.3	0.0	652	324
13	0.0	0.0	0	0
14	0.0	0.0	488	167
15	1.2	0.0	646	219
16	0.0	0.0	0	0
17	0.0	0.0	701	191
18	0.0	0.0	426	104
19	1.2	0.0	660	334
20	0.0	0.0	0	0
21	0.0	0.0	655	209
22	0.0	0.0	685	144
23	0.0	0.0	1	2
24	0.8	0.0	630	350
25	0.0	0.0	0	0
26	0.0	0.0	547	187
27	0.0	0.0	498	102
28	0.0	0.0	0	0
29	16.5	0.0	781	312
30	0.0	0.0	0	0
31	0.0	0.0	740	304
32	0.0	0.0	639	126
33	0.0	0.0	0	0
34	0.0	0.0	470	95
35	0.0	0.0	32	12
36	0.0	0.0	0	0
37	0.0	0.0	0	0
		Group B Data		
	Best Cost		Time 2 Best Cost	
Scenario	SA	DCSA	SA	DCSA
1	13.0	13.0	576	85
2	29.3	28.8	780	136
3	29.8	5.7	782	506
4	16.0	5.9	797	336
5	17.6	1.0	781	502
6	4.3	0.9	781	496
7	28.1	8.7	781	344
8	44.4	14.2	783	507
9	23.6	1.9	783	503

DCSA vs SA : Average Cost and Time : TNET Data

		Group A	Data	
	Average	Cost	Average	Time
Scenario	SA	DCSA	SA	DCSA
1	0.40	0.01	597	311
2	0.00	0.00	0	0
3	0.00	0.00	0	4
4	0.00	0.00	2	11
5	2.80	0.10	675	394
6	0.00	0.00	0	0
7	0.00	0.00	73	53
8	0.00	0.00	241	104
9	0.00	0.00	0	0
10	0.00	0.00	285	120
11	0.00	0.00	237	58
12	2.31	0.01	661	354
13	0.00	0.00	0	0
14	0.00	0.00	562	199
15	2.39	0.00	642	323
16	0.00	0.00	0	0
17	1.06	0.00	750	242
18	0.00	0.00	532	127
19	2.01	0.07	683	366
20	0.00	0.00	0	0
21	0.00	0.00	732	252
22	1.37	0.00	743	173
23	0.00	0.00	3	11
24	1.87	0.14	615	354
25	0.00	0.00	0	0
26	0.00	0.00	667	219
27	0.54	0.00	680	145
28	0.00	0.00	0	0
29	19.21	8.69	784	333
30	0.00	0.00	0	0
31	1.80	0.00	774	338
32	1.94	0.01	760	213
33	0.00	0.00	0	0
34	0.00	0.00	607	167
35	0.00	0.00	215	52
36	0.00	0.00	0	0
37	0.00	0.00	0	0
		Group B	Data	
	Average	Cost	Average	Time
Scenario	SA	DCSA	SA	DCSA
1	13.30	13.00	596	109
2	44.07	35.92	780	150
3	38.89	8.03	784	526
4	29.01	8.46	783	335
5	22.79	1.97	782	551
6	7.17	1.55	780	516
7	32.71	17.09	781	344
8	49.39	16.67	782	534
9	31.59	4.05	782	519

DCTS vs TS : Average Cost and Time : TNET Data

	Group A Average Cost		Data	Group A Average Time (s)	Data
Scenario	TS		DCTS	TS	DCTS
1	0.00		0.00	24.38	71.6
2	0.00		0.00	0.00	0.4
3	0.00		0.00	0.00	4.4
4	3.10		0.80	13.63	34.8
5	0.00		0.00	60.50	170.6
6	0.00		0.00	0.00	0.1
7	0.00		0.00	4.00	31.3
8	0.00		0.00	4.00	23.3
9	0.00		0.00	0.00	0.1
10	0.00		0.00	15.00	50.3
11	22.19		2.75	99.00	288.2
12	0.05		0.00	69.50	182.2
13	0.00		0.00	0.00	0.0
14	0.00		0.00	31.13	74.1
15	0.00		0.00	27.25	138.3
16	0.00		0.00	0.00	0.0
17	0.00		0.00	75.13	136.6
18	95.75		17.48	337.00	336.5
19	0.01		0.00	55.25	140.9
20	0.00		0.00	0.00	0.0
21	0.00		0.00	31.00	91.0
22	37.40		34.29	373.13	336.3
23	8.00		0.80	36.13	55.9
24	0.11		0.05	49.50	212.2
25	0.00		0.00	0.00	0.0
26	0.00		0.00	39.13	74.4
27	0.00		0.00	26.13	121.1
28	0.00		0.00	0.00	0.0
29	72.21		52.86	624.00	585.8
30	0.00		0.00	0.00	0.0
31	0.00		0.00	82.13	122.8
32	0.00		0.03	43.00	282.1
33	0.00		0.00	0.00	0.0
34	0.00		0.00	69.13	88.7
35	0.00		0.00	12.38	33.7
36	0.00		0.00	0.00	0.0
37	0.00		0.00	0.00	0.0
		Group B	Data		
	Average	Cost		Average	Time
Scenario	TS		DCTS	TS	DCTS
1	31.49		13.80	52.63	160.3
2	75.73		51.86	281.13	325.1
3	42.89		6.36	785.75	887.1
4	41.99		33.01	536.75	577.3
5	1.96		2.30	783.88	841.8
6	1.90		1.29	328.25	418.8
7	74.48		108.93	777.00	610.7
8	30.53		4.71	786.13	875.2
9	6.73		2.72	785.25	857.5

DCTS vs TS : Best Cost and Time : TNET Data

		Group A	Data	
	Best	Cost	Time 2	Best Cost
Scenario	TS	DCTS	TS	DCTS
1	0.00	0.00	24	50
2	0.00	0.00	0	0
3	0.00	0.00	0	0
4	0.00	0.00	9	8
5	0.00	0.00	44	121
6	0.00	0.00	0	0
7	0.00	0.00	4	22
8	0.00	0.00	4	14
9	0.00	0.00	0	0
10	0.00	0.00	15	40
11	7.20	0.00	134	160
12	0.00	0.00	37	68
13	0.00	0.00	0	0
14	0.00	0.00	31	62
15	0.00	0.00	26	50
16	0.00	0.00	0	0
17	0.00	0.00	75	117
18	3.90	0.40	451	320
19	0.00	0.00	37	67
20	0.00	0.00	0	0
21	0.00	0.00	31	74
22	1.20	6.30	477	321
23	0.00	0.00	29	26
24	0.00	0.00	30	115
25	0.00	0.00	0	0
26	0.00	0.00	39	70
27	0.00	0.00	26	63
28	0.00	0.00	0	0
29	60.80	23.70	625	612
30	0.00	0.00	0	0
31	0.00	0.00	82	105
32	0.00	0.00	43	68
33	0.00	0.00	0	0
34	0.00	0.00	66	84
35	0.00	0.00	11	32
36	0.00	0.00	0	0
37	0.00	0.00	0	0
		Group B	Data	
	Best	Cost	Time 2	Best Cost
Scenario	TS	DCTS	TS	DCTS
1	13.00	13.00	59	128
2	46.90	37.60	293	315
3	41.60	4.40	791	870
4	35.70	19.40	545	628
5	1.70	0.60	783	846
6	1.10	0.70	409	452
7	72.70	66.20	765	659
8	23.00	2.50	784	876
9	2.80	0.80	784	855