Venables, Harry (2011) Ant Colony Optimisation – A Proposed Solution Framework for the Capacitated Facility Location Problem. Doctoral thesis, University of Sunderland.

Ant Colony Optimisation – A Proposed Solution Framework for the Capacitated Facility Location Problem

Harry Venables

A thesis submitted in partial fulfilment of
the requirements of the

University of Sunderland

for the degree of

Doctor of Philosophy
June 2011

Abstract

This thesis is a critical investigation into the development, application and evaluation of ant colony optimisation metaheuristics, with a view to solving a class of capacitated facility location problems. The study is comprised of three phases.

The first sets the scene and motivation for research, which includes; key concepts of ant colony optimisation, a review of published academic materials and a research philosophy which provides a justification for a deductive empirical mode of study. This phase reveals that published results for existing facility location metaheuristics are often ambiguous or incomplete and there is no clear evidence of a dominant method. This clearly represents a gap in the current knowledge base and provides a rationale for a study that will contribute to existing knowledge, by determining if ant colony optimisation is a suitable solution technique for solving capacitated facility location problems.

The second phase is concerned with the research, development and application of a variety of ant colony optimisation algorithms. Solution methods presented include combinations of approximate and exact techniques. The study identifies a previously untried ant hybrid scheme, which incorporates an exact method within it, as the most promising of techniques that were tested. Also a novel local search initialisation which relies on memory is presented. These hybridisations successfully solve all of the capacitated facility location test problems available in the OR-Library.

The third phase of this study conducts an extensive series of run-time analyses, to determine the prowess of the derived ant colony optimisation algorithms against a contemporary cross-entropy technique. This type of analysis for measuring metaheuristic performance for the capacitated facility location problem is not evident within published materials. Analyses of empirical run-time distributions reveal that ant colony optimisation is superior to its contemporary opponent.

All three phases of this thesis provide their own individual contributions to existing knowledge bases: the production of a series of run-time distributions will be a valuable resource for future researchers; results demonstrate that hybridisation of metaheuristics with exact solution methods is an area not to be ignored; the hybrid methods employed in this study ten years ago would have been impractical or infeasible; ant colony optimisation is shown to be a very flexible metaheuristic that can easily be adapted to solving mixed integer problems using hybridisation techniques.

Acknowledgements

Initially, my sincere thanks must go to Dr. Mitchell Andrews, a former colleague from the University of Sunderland Business School, who coerced me into starting a Ph.D. study. Next, I would like to thank my original supervisor Prof. Alfredo Moscardini. He gave me that initial spark about *agent based modelling*, that made me think of *ants* and how they solved everyday problems by their social interactions; I owe him a great deal. His encouragement for letting me just get on with it, allowing me to develop the confidence to present my work to academic peers and use the experience as a vital part of the research apprenticeship. Upon Alfredo's retirement from the university, Dr. Valentina Plekhanova, had the task of taking over. Thankyou for your thoughts and guidance through those initial writing stages at a vulnerable time of life.

However, my biggest thanks must go to my wife Laura for all of the emotional support through the highs and lows of this study. This work would not have been worth it or even completed without her being there for me when I needed more than just kind words. Finally, my two children Louis and Rachel their infectious giggles and laughs make everything worthwhile.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Many behavioural characteristics of ants have been studied by scientists including their social interactions, brood sorting, colony welfare, hierarchical systems, division of labour, cooperative transport and adaptive foraging strategies. These same topics have provided substance for many humorous children's stories and even several animated big-screen adventures. A fascinating statistic is that:

> "Ants are everywhere on earth. When combined, all ants in the world weigh about as much as all humans ..." (Hölldobler and Wilson, 1994)

A simple fact is that ants are very successful which is primarily due to their adaptive nature. There are literally tens of thousands of different species of ants that have evolved throughout the natural world. Their intrinsic behaviour to work and search for food all for the good of the colony to which they belong is incredible. Their peculiar foraging behaviour often incites laughter from both adult and child observers. If we can use what is known about their incessant quest for food and their success at delivering it safely back to the colony nest, then we ought to able to solve many logistical and transportation problems. In essence, it is the ants desire for food that motivates and provides a rationale for using Ant Colony Optimisation on problems that can be modelled as a network of pathways, such as those associated with facility location.

Initially, this chapter provides a research study outline and then gives a brief background and overview of Ant Colony Optimisation to demonstrate that an observation in nature can be used artificially, in a computing environment, to solve hard combinatorial optimisation problems. The method exploits the fact that ants have an adaptive ability to forage for food and return it to the colony nest in an effective and efficient manner by discovering and using shortest pathways to fetch and carry food supplies. Their ability to find a series of shortest pathways is analogous to minimum-cost optimisation problems, particularly those types of problems encountered in discrete combinatorial optimisation.

1.1 Research Study Outline

This research study is primarily comprised of three phases, relating to the application of Ant Colony Optimisation to facility location. These are a study rationale phase, research design and development phase, with the final phase being a critical evaluation.

The first phase is organised as follows: Chapter 1, which provides a useful background to Ant Colony Optimisation. Chapter 2 concentrates on various published methods that have been used within facility location, with an emphasis on applications to capacitated facility location problems. Chapter 3 discusses and rationalises the need for an empirical study, to determine performance related measures for Ant Colony Optimisation that are required for comparative analyses.

The second phase of this study, initially compares two common ant based algorithms to determine the suitability of Ant Colony Optimisation in its standard format to solve the capacitated facility location problem and is discussed in Chapter 4. Hybrid algorithms that exploit solution structure and make use of local search routines are developed and investigated in Chapter 5.

The final phase presents a rigorous series of run-time analyses, in Chapter 6, of algorithms developed during the second phase alongside a Cross-Entropy algorithm obtained from a research source (Caserta and Quiñonez Rico, 2009). These analyses contain empirical probability distribution profiles and bootstrapped statistical inferences, that are unknown to metaheuristic approaches in facility location. Conclusions, suggestions for future research and contributions to existing knowledge are summarised in Chapter 7.

1.2 Ant Colony Optimisation

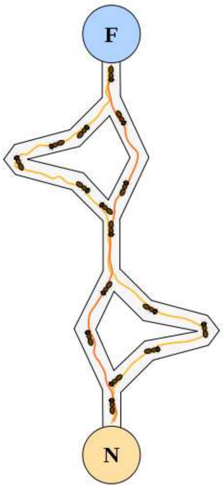
Ant Colony Optimisation (ACO) is a metaheuristic technique that is based on natural scientists' behavioural observations of foraging ants (Deneubourg et al., 1990). They observed that colonies of ants whilst searching for food from a nest site intially behaved in swarm-like random manner to locate a food source. However, once food was returned to the nest, then most of the ants then quickly converged to a single route to both fetch and carry food back to the nest site. Despite these insects being relatively simple creatures, that are almost blind, they seemed to be able to communicate efficiently and effectively for the well-being of the colony. Also, it was observed that if a food source became depleted or the convergent pathway unusable then the ants would adapt by returning to their swarm-like behaviour to find an alternative solution. This type of self-organising behaviour or intelligence is crucial to their dominant success and stems for their ability to react and respond to any environmental change.

To assist the ants reactive systems they have a very acute sense of smell that enables them to detect and respond to any environmental change. Each ant also has a facility for producing and laying chemical or odour based substances known as pheromones. Different types and intensities of pheromone deposits can occur and are used to indicate different types of food sources, or types of nest-based problems that need to be overcome. Their ability to smell and lay various types of pheromones acts as an indirect communication system, that ants may respond to in different ways depending upon their hierarchy within the colony. This process of communication via environmental change is known as *"stigmergy"* (Bonabeau et al., 1999, Theraulaz and Bonabeau, 1999).

When ants search for a new food source they have no overall vision or insight of prior knowledge towards the problem that they are faced with. However, through their ability to lay pheromones whilst searching for food they can communicate to other ants promising pathways by the type or intensity of pheromone that they lay. Thus, those pathways that have high intensities of pheromones are deemed more likely to used by other ants. Once a source is found then food is initially carried back along the original pathway. A colony requires the ants to be able to locate food and take it back to the nest efficiently, i.e. as quickly as possible. To achieve this a stigmatic process is applied when returning food to the nest and higher intensities of pheromones are laid on the most promising or shortest pathways. This process continues until a near complete convergence of the ants locating the source and carrying food along the same pathway emerges (Beckers et al., 1992, Deneubourg et al., 1990, Dréo et al., 2006).

An important observation is that not all of the ants converge to the same pathway, it is the behaviour of these non-conformists that give rise to the adaptive nature of a colony of ants. These rogue ants may find new or better food sources nearer to the nest site, or should the convergent pathway become blocked they may have already discovered an alternative route both of which could then be communicated via stigmergy. Figure 1.1 depicts a typical modelling represen-

Figure 1.1: Random Search and Convergence

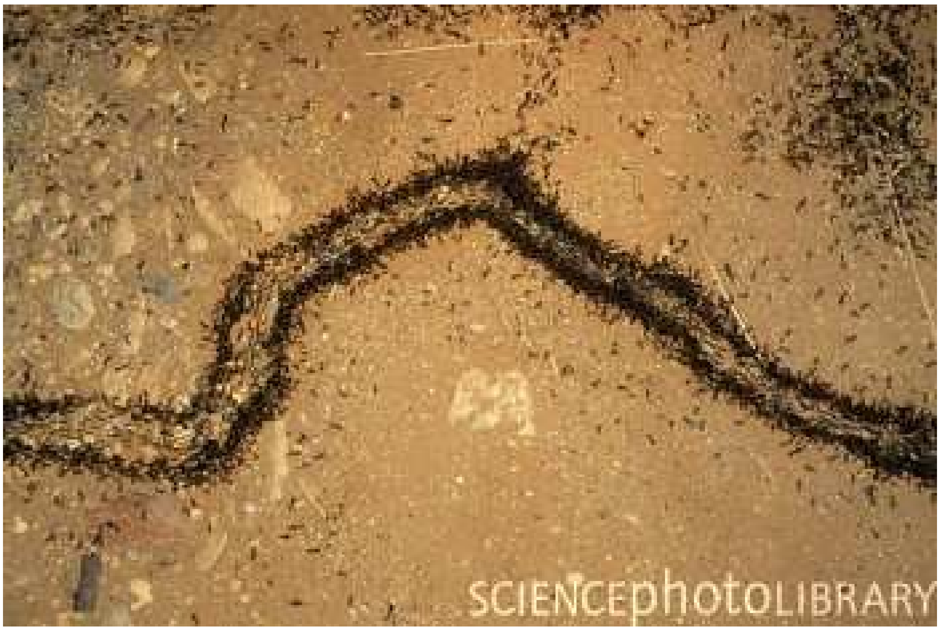


Initially ants randomly search various pathways, then converge to a shortest route (Bonabeau et al., 1999).

tation of foraging behaviour: initially ants search through the pathways laying pheromones and intensifying the trails taken on the return journey to the nest, most ants then converge to a common shortest path. Whereas, Figure 1.2 displays foraging behaviour of army ants observed in nature and Figure 1.3 displays a well-established path from a nest to highly significant and stable food source.

The emergence of a shortest or optimal pathway from foraging behaviour was the impetus for the then contemporary works of Colorni et al. (1992) and Dorigo (1992); which paved the way for future ACO developments. Originally the metaphor of ACO and its algorithmic application to the travelling salesman problem gave results that were a little disappointing. The methodology displayed asymptotic convergence to shortest paths but suffered from long computational run-times. However, researchers realised the potential of such a strong metaphor and its general application, so much so that it later became a popular and accepted metaheuristic technique. It was in the late 1990's that ACO started to show a trend in research works for solving hard combinatorial optimisation problems.

Figure 1.2: A foraging strategy by African army ants



Food is sent back to the nest along the central highway, soldier ants protect the flanks whilst some other army ants look for alternative foods sources and pathways *(Courtesy of Science Photo Library).*

These research areas lent themselves towards solving purely discrete classical decision based optimisation problems: the travelling salesman and quadratic assignment problem. The most significant contributions made to ACO algorithmic research and development at the time were the Max-Min Ant System by Stützle and Hoos (1997, 2000) and Ant Colony System by Gambardella et al. (1999), which are still cited as amongst the best performing algorithms by Blum et al. (2008).

Although ACO algorithms were inspired by foraging ants, it is important to remember that these techniques do not try to completely model their behaviour, moreover they model their key features in an artificial intelligent manner; i.e. ACO uses principles of stigmergy and pheromones artificially to help make stochastic based decisions to find shortest paths within a given search area. ACO is a metheuristic that was developed as a procedure to solve hard combinatorial op-

Figure 1.3: A mature ant trail

A well established trail of the chaco leafcutter ant in Argentina *(Courtesy of http://www.alexanderwild.com).*

timisation problems in a global sense using a probabilistic local step or decision process and thus can also be described as a stochastic local search optimisation technique (Hoos and Stützle, 2005).

ACO has played a significant role in solving many types of combinatorial optimisation problems and at times achieved state-of-art performances on various classical operational research style problems and applications; examples can be found in the works of Dorigo and Stützle (2004), Dréo et al. (2006) and Blum et al. (2008). Interestingly, ACO like some other types of metaheuristic techniques is described as an 'incomplete algorithm' by Hoos and Stützle (2005). As an optimal solution may be found but it can not be guaranteed that the algorithm will find an optimal solution within specific run-time bounds. Typically these bounds may include combinations of clock run-time limits and/or a limit on the maximum number of iterations permitted. However, asymptotic convergence of ACO algorithms are usually observed, i.e. an optimal solution is likely to be found if the algorithm is allowed to run for a long enough period of time. Theoretical concepts of convergence and asymptotic behaviour are discussed by Dorigo and Blum (2005) and suggestions for empirical research is given by Hoos and Stützle (2005). Consequently, as indefinite computational run-times are impractical then caution should be heeded when using metaheuristic solution techniques to determine inferences and decisions based on experimentation.

ACO solution methods are usually designed to be applied to true/false type combinatorial optimisation problems. However, Dorigo and Stützle (2004) state that the ACO metaphor can be applied to other types of optimisation problems; such as optimisation problems containing a mixed-integer or continuous variables. Research in these areas is in its infancy and evidence is somewhat limited (Dorigo et al., 2008, 2006). The *capacitated facility location problem* (CFLP) is a mixed-integer problem that had remained unsolved by various metaheuristic techniques including Simulated Annealing, Genetic Algorithms and Tabu Search, (Arostegui et al., 2006, Bornstein and Azlan, 1998, Bornstein and Campelo, 2004, Filho and Galváo, 1998, Sörensen, 2008) until recent hybrid methods were presented by Caserta and Quiñonez Rico (2007), Venables and Moscardini (2008) and Caserta and Quiñonez Rico (2009). Two primary aims of this research are to be able to solve the CFLP using ACO and to provide a corresponding critical evaluation of the proposed techniques.

Chapter 2

A Review of Facility Location

This chapter provides a review of research materials relevant to this study and is compiled from the fields of facilities location, ACO and metaheuristics. The specific aims of this chapter are to determine the validity of conducting a Ph.D. research study into the use of ACO as a potential generic platform to solve capacitated location problems, and to identify what area should the research focus on to ensure that contributions to existing literature and knowledge are attainable.

Initially an overview of facility location is presented, which then procedes to focus on advancements in heuristic solution techniques to a particular class of theoretical facility location problems. The final section summarises the key points associated with capacitated location problems, identifies a focused theme for research purposes and presents a general research question.

2.1 Facility Location

Facility location, which is also referred to as location analysis or location science, is concerned with the siting of facilities on a plane or within a network. The location of a facility is dependant upon various attributes such as demand placed upon a facility by a set of customers, the cost of supplying those customers and the costs of opening facilities at potential locations. Customer supply costs are measurable, yet variable across the set of available facilities and are often presented in terms of transportation or allocation costs. Objectives of facility location problems are usually dependent upon the type of scenario being modelled, e.g. minimise fixed and variable costs (Daskin, 1995); minimise customer's travelling distances or times to facilities (Mirchandani and Francis, 1990), or maximise customer coverage with various distance criteria (Drezner, 1995). Applications of facility location have been used in various domains: public sector (recreation and leisure, health centres, etc.), private sector (supermarkets, distribution centres, factories, etc.) and in environmentally sensitive areas (production and disposal of obnoxious waste, chemicals, etc.), (Agar and Salhi, 1998, Church and Murray, 2008, Daskin, 1995, Drezner, 1995, Klose and Drexl, 2004, Love et al., 1988, Smith et al., 2009, Zanjirani and Hekmmatfar, 2009).

A recent paper of Smith et al. (2009) provides a review of milestone contributions to facility location applications and theoretical models. The authors categorise these works into three time periods; Period 1: Early contributions, Period 2: Coming of age and Period 3: Fruitfulness with new models and applications. The first period attributes the works of Weber (1909) and Hotelling (1929) as having significant influences on early and present-day facility location development. This view is also shared within the review works of Daskin (2008), Owen and Daskin (1998), ReVelle and Eislet (2005), ReVelle et al. (2008). The second period provides insight into a twenty year period during the 1960s and 1970s, that produced theoretical models which form the backbone for many modern-day theoretical location problem formulations. These problems include median and covering problems (Hakimi, 1964, 1965, Kariv and Hakimi, 1979a,b), plant and warehouse location problems (Balinski, 1966, Kuehn and Hamburger, 1963) and the quadratic assignment problem (Lawler, 1963). The third period refers to the development of problems and new applications from the 1980s to present day which includes important works on algorithmic solution techniques within facility location, such as enhanced developments of Lagrangean relaxation techniques that were based on the original DUALOC algorithm of Erlenkotter (1978) for the uncapacitated facility location problem.

Recent facility location reviews by Daskin (2008) and ReVelle et al. (2008) categorise location problems into four classes of models: analytical, continuous, network and discrete. The first class of problems refers to simple contrived problems, that have some useful analytical value and are solved using classical mathematical optimisation techniques. However, their simplicity leads to limited uses in practical applications. Continuous problems allow for the location of facilities anywhere within a region or on a plane, yet customer demand is usually restricted to fixed points or nodes within the region or plane. These types of problems are referred to as Weber-type problems and are often solved using analytical techniques. A review of algorithmic techniques and applications to solve these types of problems is given by Drezner et al. (2001). A recent paper by Altlnel et al. (2009) gives an interesting recent development of this problem, that considers demand nodes within the plane to have stochastic locations. This is an attempt to find optimal facility locations when customers change their location, such as the relocation of manufacturing process that requires raw materials via a supply chain. The third and fourth classes are described separately. However there is a great deal of similarity between these types of problems as network models can be used to describe discrete location models and vice versa. Indeed the two classical p-median and p-centre problems described by Hakimi (1964, 1965) were originally described as graph or network problems but are now often referred to as discrete location problems (Daskin, 2008, Mirchandani and Francis, 1990, ReVelle et al., 2008). In a taxonomy of location models, by Daskin (2008), discrete location models fall into one of three sub-classes: covering models, median-based models and other discrete models. Covering based models include set covering, max covering and p-centre problems. Median based models include p-median and fixed-charge problems. Whilst, the other discrete models include the classic quadratic assignment problem (Lawler, 1963), p-dispersion and various types of combined max-min problems. All three classes may or may not include extra modelling constraints placed upon them such as capacity and distance-covering constraints. Furthermore, some of these problems may also incur strict binary constraints associated not only with facilities selection but also whether a customer's demand is fulfilled from a single or multiple facilities.

2.2 P-Median Facility Location Problems

This thesis concentrates on a derivative of the p-median problem namely the *fixed-charge capacitated facility location problem* and the rest of this sub-section presents a review of advancements made with associated problems. The p-median problem is an adaption of the classical Weber problem and is attributed to Hakimi (1964). The main difference between the two problems is that in a Weber problem customers, or demand nodes, are allocated to their nearest facility whereas in a median problem this allocation is made by considering least weighted distances; where the weighted distance is a product of distance from a facility and customer demand. Should a 1-median problem be considered then this is analogous to applying the method of moments to obtain the centre of gravity for the problem. Hakimi (1964), also proved that if a p-median location problem can be modelled as a network, consisting of customer and potential facility nodes connected by edges, then an optimal set of p-medians occur at p-nodes of the network. Consequently, if the number of potential facilities is say a handful and the number of customers is of a similar size then the problem could be solved by complete enumeration. However, Kariv and Hakimi (1979a,b) went on to prove that on a network the p-median problem is generally **NP**-hard (Garey and Johnson, 1979). Thus, it is particularly difficult to solve when considering problems of a more practical size where the number of variables may run into the order of hundreds or even thousands. Hence heuristic solution methods are often employed, as the complete enumeration time component increases exponentially with the number of binary variables to be considered (facilities and customers). The combinatorial nature of the p-median problem meant that advances in solution techniques were often made using combinations of mathematical programming and heuristic search techniques such as branch and bound and Lagrangean relaxation. The more successful of these solution techniques provided a means of solving to other types of location problems. Beasley (1982), Christofides and Beasley (1982) applied Lagrangean relaxation and subgradient optimisation to a tree search procedure for the p-median problem. These works provided the instigation for further developments for solving median based location problems, (Beasley, 1988, 1990, Christofides and Beasley, 1983), with the culmination of a Lagrangean based framework for solving a variety of location problems (Beasley, 1993). This significant work, along with it's results, is still often cited today by many authors.

Reese (2006), provides an annotated bibliography for the p-median problem which indicates that since the late 1980s there has been a significant shift from using traditional mathematical programming relaxation techniques, such as Lagrangean and surrogate relaxation, towards the use of more modern metaheuristics techniques. The use of these methods have resulted in not only obtaining more accurate solutions efficiently but have also allowed the consideration of, and solution generation to, much larger problem instances. Some of these methods for the p-median problem are based on well known metaheuristics: Simulated Annealing (Al-khedhairi, 2008, Levanova and Loresh, 2004), Genetic Algorithms (Alp et al., 2004, Fathali, 2006), Tabu Search (Rolland et al., 1997, Salhi, 2002), greedy adaptive search procedure (Resende and Werneck, 2004) and ACO (Fathali et al., 2006, Kaveh and Shojaee, 2008, Levanova and Loresh, 2004). Whatever metaheuristic procedure is used, the quality of solution and the time-efficiency is dependant on appropriate applications of local search structures and strategies embedded within the algorithms. Mladenovic et al. (2007) concluded that it was too difficult at the time to determine which was the more dominant metaheuristic for solving these type of problems, but stated that “the advent of metaheuristics has advanced the state-of-the-art significantly”. Hoos and Stützle (2005) suggested that it is necessary to conduct more rigorous testings of metaheuristic procedures, that use stochastic steps, than is often accepted within academic literature to gain more general insights into algorithmic behaviour and enable unbiased comparisons.

2.3 Uncapacitated Facility Location Problems

Historically, the next derivative of the p-median problem was the *simple plant location problem*. Which considers each facility (median) to include a one-off fixed opening cost and the number of facilities to be used is such that the overall costs are minimised; i.e. the number of medians or facilities required are not pre-fixed. This problem was first formulated by Balinski (1965, 1966) and is often referred to as the *uncapacitated facility location problem* (UFLP) or the *warehouse location problem* (Beasley, 1993, ReVelle and Eislet, 2005). Like its predecessor, the p-median, the uncapacitated facility location problem is **NP**-hard and various heuristic solution methods have made significant advances into solving these types of problems. Lagrangean dual methods to this problem have been used successfully by Barahona and Chudak (2005), Beasley (1993), Erlenkotter (1978) and Guignard (1988). A useful survey is given by Karup and Pruzan (1983). More recently advances have been made using Lagrangean relaxation and branch and peg techniques (Canovas et al., 2007, Goldengorin et al., 2004, Lu et al., 2005). Metaheuristics and hybrid methods have also had significant roles to play in improving solution techniques: Simulated Annealing (Aydin and Fogarty, 2004), Genetic Algorithms (Jaramillo et al., 2002, Sun, 2006), Tabu Search (Michel and Hentenryck, 2004), variable neighbourhood search (Ghosh, 2003), greedy adaptive search (Resende and Werneck, 2006) and particle swarm algorithms have also been considered by Guner and Sevkli (2008), Sevkli and Guner (2006). Hoefer (2003) performed empirical analyses on five different types of local search algorithms and concluded that Tabu Search was the most promising solution method. This was tested against a hybrid method designed by Resende and Werneck (2006) on the same computational platform. The results obtained were very similar with the hybrid method being slightly more favourable. However, although the tests were carried out on the same set of problem instances the statistical measures obtained were only concerned with ten randomised runs on each instance, which may have led to incorrect inferences.

2.4 Capacitated Facility Location Problems

Both of the p-median and uncapacitated location problems consider facilities that have sufficient supply or capacity to supply all of their assigned customers. In practise this may not be the case as facilities may only have a finite supply or capacity available. Thus, most customers may be assigned to their nearset facility until the supply runs out, whilst some customers may be partially assigned to their nearest and/or nearest available facilities. Two scenarios arise, firstly where a binary constraint is placed on the assignment of customers to facilities to ensure that a customer is served from a single facility, and secondly the customer demand constraint is allowed to be fractional to ensure demand is satisfied from one or more facilities. As a consequence studies have been conducted into *capacitated p-median* and *capacitated facility location* problems. The introduction of capacity constraints effectively makes these problems theoretically more difficult to solve and are generally **NP**-hard. Contributions to solving the capacitated p-median problem are primarily metaheuristic or hybrid based. Lorena and Senne (2003, 2004) had some success with applications of local search and Lagrangean relaxation. Olivetti et al. (2005) made a useful contribution by considering a hybrid ACO procedure which involved a new type of embedded local search routine and França et al. (2006) provided a successful Tabu Search algorithm. A scatter search algorithm was presented by Scheuerer and Wendolsky (2006), and claimed that the algorithm out-performed those available at the time. Recently, Fleszar and Hindi (2008), Osman and Ahmadi (2007) have applied various variable neighbourhood search techniques which include elements of hybridisation, both gave better results than previously published efforts by other authors.

The *capacitated facility location problem* can either be viewed as an extension of the capacitated p-median problem that includes fixed facility opening costs, or more appropriately as extension to the uncapacitated facility location problem that considers each facility to have a finite supply/capacity constraint. The capacitated facility location problem is often referred to be **NP**-hard and literature works generally cite either Kariv and Hakimi (1979a,b) or Garey and Johnson (1979); neither of whom considered this problem. However, as the problem is modelled on a network and then it is highly likely to be at least as hard as the p-median problem. This thesis presents a formal problem specification for the capacitated facility location problem, with its mathematical formulation and demonstrates that its complexity is **NP**-hard. Solution techniques presented within the literature are generally heuristic based algorithms. ReVelle and Eislet (2005) indicated that the introduction of capacity constraints destroy the property that all the demand of a customer ought to be assigned to a single facility, which makes problem much more difficult to solve. They also claimed that available literature conclusions in this area were ambiguous, i.e. there was no evidence of a dominant or state of the art solution technique at that time. Introducing the capacity constraints leads to two possible scenarios. The first is that if each customer's demand has to be completely satisfied then a customer's demand may be supplied from more than one facility; which gives a mixed-integer optimisation problem. Whilst the second, considers that a customer's demand has to be supplied from a single facility; which gives a pure 0–1 integer or binary decision optimisation problem.

2.4.1 Lagrangean Relaxation

Although early attempts at solving the *capacitated facility location problem* (CFLP) were based on branch and bound techniques and linear relaxation of the integral constraints and they were only applied to contrived small-scale problems (Baker, 1982, Sa, 1969). Later, major developments in solution techniques are attributed to the use of approximate techniques based on the application of Lagrangean relaxation (Barceló and Casanovas, 1984, Christofides and Beasley, 1983). Initially these ideas were applied to the mixed integer formulation of the CFLP and the term 'large problems' appeared (Beasley, 1988). Other contemporary and mathematical programming solution techniques, together with recent applications of the time were documented in the text by Love et al. (1988) and briefly discussed in the theoretical text of Mirchandani and Francis (1990). Sridharan (1993) applied Lagrangean relaxation to both the mixed-integer (CFLP) and discrete versions

(0–1 CFLP). However, Beasley (1993) provided the most significant advancement by using Lagrangean relaxation as a framework for solving a host of facility location problems. This work involved not only the development of Lagrangean relaxation but also implemented ideas of approximate local strategies and was applied to a series of test problems that are part of the OR-Library and almost twenty years later are still often used and cited today (Beasley, 1990). A review of the CFLP was published by Sridharan (1995) that gave researchers a good insight into the best known techniques of the time and possibilities for future research directions. A sub-drop local search method was proposed by Salhi and Atkinson (1995). This starts with a potential set of facilities then attempts to reduce the overall costs by dropping or closing facilities and reassigning customers to their nearest available facility. This technique is still very relevant today in more contemporary metaheuristic methods. Daskin (1995) also published his facility location text that gave detailed expositions of Lagrangean heuristic approaches to uncapacitated and capacitated fixed-charge location problems. This text also offered insights that embraced and integrated ideas of ‘add’, ‘’drop and ‘swap’ local search improvement methods. A review and perspective on future directions of facility location including those issues associated with the CFLP was discussed by ReVelle (1997).

A major problem with Lagrangean relaxation is that even the best lower bound obtained may not provide an optimal solution or feasible solution. During the procedure, lower bounds need to be checked against upper bounds. Both of these require solutions to sub-problems of the original problem that may be difficult to solve in their own right and consequently need to be approximated. Indeed, lower bound calculations are required at each step of the Lagrangean process, which may take thousands of iterations with many sub-problem solution required at each iteration. The difference between upper and lower bounds is known as the duality gap. Should this gap be zero then an optimal solution is observed. The types of sub-problems that need to be solved for the CFLP are dependent upon the constraints that are relaxed and the type of problem being solved. Beasley (1993) gave a good overview of these issues. Typically, if the mixed-integer CFLP is considered, then for a set of known facilities the problem reduces to solving an unbalanced transportation problem. Similarly, for the 0–1 CFLP the problem reduces to a special form of generalised assignment problem which is **NP**-hard. Although efficient specialist algorithms for transportation problems, based on linear programming existed at the time, for example see (Goldberg, 1997), they were considered computationally too expensive to solve these problems at each iteration of a CFLP solver. Consequently, in both cases the transportation problem and generalised assignment problems were approximated. Agar and Salhi (1998) tackled the issue of obtaining solutions to large scale CFLPs by implementing a new interchange local search heuristic and step size criteria within their Lagrangean heuristic. This approach allowed for the consideration of some considerably larger problems containing one hundred facilities and one thousand customers. Results obtained were very encouraging and solution quality was comparable and marginally better on the small to medium sized problems as used by Beasley (1993) and significantly better for the larger instances.

2.4.2 Metaheuristics

Metaheuristic techniques started to make some advancements over Lagrangean techniques in the mid-to-late period of the 1990s. Two promising techniques to solve the CFLP were derived using Simulated Annealing and Tabu Search (Bornstein and Azlan, 1998, Filho and Galváo, 1998). The Simulated Annealing approach used by Bornstein and Azlan (1998) tested a technique that incorporated approximate and exact solution methods for any transportation sub-problems that were encountered. The results obtained for the approximate method indicated that the Lagrangean relaxation method of Beasley (1993) was a better method. However, the exact method gave results comparable to the the Lagrangean method in terms of the average relative error. The Tabu Search method of Filho and Galváo (1998) gave a slight improvement over the Lagrangean, and Simulated Annealing methods methods for the concentrator problem which is similar to the CFLP. Attention then turned towards the 0–1 CFLP, Hindi and Pieńkosz (1999) designed an efficient Lagrangean relaxation algorithm that outperformed the method used by Beasley (1993). A hybrid branch and bound method that was embedded within Lagrangean relaxation by Holmberg et al. (1999), out-performed a CPLEX integer programming technique, but was not tested against any of the main published methods. The trend for Lagrangean based approaches to solve the 0–1 CFLP continued for some time. Some of the more noteworthy contributions, for various reasons, that often appear within the literature are: Ahuja et al. (2004), Barahona and Chudak (2005), Cortinhal and Captivo (2003), Díaz (2001), Díaz and Fernádez (2002), Rönnqvist et al. (1999). The CFLP was also considered within the combinatorial optimisation vehicle routing problem using an effective algorithm by Bramel and Simchi-Levi (1995).

During the first five years of the new millennium interest in the UFLP and CFLP dwindled. However, some useful work was carried out. A Genetic Algorithm for the CFLP was developed by Jaramillo et al. (2002). This performed well on the UFLP, finding optimal solutions for all of the test problems in relatively small computational times, but suffered with excessively long run-times for the CFLP and thus was abandoned as a potential solution technique. A very successful problem-reduction local search technique based on a facility dominance criteria was developed by Bornstein and Campelo (2004). This algorithm gave results similar to previous Simulated Annealing and Lagrangean relaxation techniques of Bornstein and Azlan (1998) and Beasley (1993) for the CFLP, but with significantly improved run-times. It was during this period that ACO became a popular technique for solving a variety of discrete combinatorial optimisation problems. Stützle and Hoos (2000) published a paper on an ant algorithm to solve the travelling salesman problem and the quadratic assignment problem; the latter being a facility location problem. Their algorithm gave results that were classed as state-of-the-art for both problems. The generalised assignment problem was successfully tackled by two location experts, Lourenço and Serra (2002), using a hybrid algorithm based on ACO and Tabu Search. They also indicated that it may be possible to adapt their ideas into a solution techniques for the 0–1 CFLP. Although Kumweang and Kawtummachai (2005) attempted to solve this problem and claimed that the problem could be solved effectively and efficiently using ACO, their results were somewhat misleading as insufficient experimentation had been carried out. However, Olivetti et al. (2005) did manage to produce a successful ant based algorithm to solve the capacitated p-median problem, that was an adaption and extension to the algorithms presented by Stützle and Hoos (2000). Montemanni et al. (2005) went on to successfully apply ACO to the vehicle routing problem, which has since been adapted into an industrial-based planning solution technique.

A recent publication by Arostegui et al. (2006) concluded from empirical research testing that Tabu Search was more reliable than Genetic Algorithms and Simulated Annealing for solving different types of CFLPs. A multiple ant system algorithm, that used two colonies, was implemented to solve the 0–1 CFLP by Chen and Ting (2006). The first colony was used to derive what facilities to open and the second to solve the underlying general assignment problems for those facilities opened in the first stage. Although, this method gave some encouraging results the process was prone to long run-times. The same authors improved on this method by using a hybrid scheme based on combining Lagrangean relaxation to identify what facilities to use and then apply an ant system to the generated sub-problems (Chen and Ting, 2008). Levanova and Loresh (2006) designed an ant algorithm, similar to that of Venables et al. (2005) for the CFLP, to solve the 0–1 version based on traversing a bipartite graph; which incorporated a swap local search technique and gave some encouraging results. Although the two techniques were similar, the results for the CFLP indicated that solution convergence was very slow and prone to stagnation. Consequently, this method was abandoned for the CFLP and various ant hybrid metaheuristics were later developed to solve this problem (Venables and Moscardini, 2006, 2008)

A Tabu Search procedure for the CFLP was developed by Sörensen (2008) that managed to find optimal solutions to all but two of the small to medium sized OR-Library test problems. Although those instances were not identified within their paper, relative errors were reported to be less than 0.1% which was an improvement over any other previously published works for the CFLP. Indeed, this work not only backed up the conclusions made by Arostegui et al. (2006) but also suggests that Tabu Search may be a dominant methodology to use when solving the CFLP. However, a contemporary paper that used a *Cross-Entropy* approach by Caserta and Quiñonez Rico (2009) claimed to obtain optimum solutions to all of the CFLP test problems in the OR-Libray; including all of the larger problems (100 facilities and 1000 customers), which is something previously unseen or reported on within the literature. The Cross-Entropy method was originally designed to model the occurrence of rare events, within network systems, and was later adapted to be used in combinatorial optimisation (Rubinstein, 2002, Rubinstein and Krose, 2004). The analogy between rare events and combinatorial optimisation is that the probability of selecting an optimal solution for a decision problem with many variables is very small, and thus selecting an optimal solution is considered to be a rare event. Caserta and Quiñonez Rico (2009) used the basic ideas of Cross-Entropy in a hybrid manner by integrating a local search mechanism based on multiple ‘add’ and ‘drop’ type procedures. The methodology of Cross-Entropy is very similar to a specific type of ACO algorithm which was recognised by Dorigo et al. (2002).

It appears that any future work to solve the CFLP effectively and efficiently using metaheuristics would have to be of a hybrid nature. This is something that is now being recognised by the academic community and the term *hybrid metaheuristics* is being used and qualified within literature sources and dedicated international conferences. Although the idea of combining or integrating different heuristic or metaheuristic techniques is not entirely new, authors have been busy identifying and defining the concepts and implications of these approaches (Blum et al., 2008, Jourdan et al., 2009).

2.5 Summary

It is obvious from the literature that Lagrangean relaxation has played a key role in research into the solution of facility location problems and is still actively used nowadays (Ahuja et al., 2004, Beasley, 1993). What is more important is the wealth of information available on the use of various metaheuristic techniques to solve these types of problems. What is not clear is if there exists a particularly dominant type of heuristic method to generally solve the network based facility location problems. This leads to a certain amount of ambiguity as discussed by ReVelle et al. (2008) for the CFLP. The p-median in both uncapacitated and capacitated forms along with the 0–1 CFLP have acquired a great deal of attention from researchers. However, there is the opportunity for further developments to be made by using hybrid techniques, such as those discussed by Blum et al.

(2008), because more needs to be known about the reliability of these techniques when applied to the CFLP. Tabu search is seen as a good technique to use that generates many optimal solutions to a variety of facility location problems, and has successfully been integrated with other heuristics such as Lagrangean relaxation and ACO (Chen and Ting, 2008, Levanova and Loresh, 2006, Lourenço and Serra, 2002, Sörensen, 2008). Hybridisation of ACO procedures is an emerging technique for solving facility locations that have purely discrete decision variables, such as the p-median, capacitated p-median, UCFLP, and the 0–1 CFLP. Ant colony optimisation is given a thorough overview by Dorigo and Stützle (2004) and appropriate analyses of these types of algorithms are discussed by Hoos and Stützle (2005). However, when presented with a mixed-integer problem such as the CFLP then hybrids that integrate with either approximate or exact methods in an attempt to efficiently exploit the solution space are worth exploring. Although, in theoretical terms mixed-integer problems are not as difficult to solve than pure 0–1 integer problems, (Dréo et al., 2006), it may be practically more difficult to solve these mixed-integer problems. This conundrum was also inferred from the results obtained by Agar and Salhi (1998), who commented that they had observed this behaviour during their study.

Research and literature evidence presented in this chapter suggests that ACO can be successfully used to derive solutions to pure discrete facility location problems such as the p-median and discrete capacitated problems, yet there is very little evidence of its application to mixed-integer location problems. This Ph.D. research focuses on the use of ACO as a platform to solve the CFLP, to gain further knowledge about the behaviour and suitability of using these metaheuristic solution techniques. Some of the research undertaken during the development and testing phases has been either published or presented at several international conferences and shall be discussed in detail in subsequent chapters. The main objective of this research project is to determine if ACO can be used as a potential solution technique for the CFLP. If so, then determine whether it can be competitive with contemporary methods, such as the Cross-Entropy method (Caserta and Quiñonez Rico, 2007, 2009). Initial research conducted for this thesis (Venables et al., 2005, Venables and Moscardini, 2006) suggests that ACO used in its original format of traversing a graph/network (Dorigo, 1992, Dorigo and Stützle, 2004) may not be an appropriate technique to solve the CFLP. Thus, the hybridisation of ant algorithms may benefit from the integration of approximate heuristic techniques and/or some type of exact methods that exploit the structure of the CFLP. These types of designs are collectively termed as hybrid-metaheuristics (Blum et al., 2008). Consequently, research output from this Ph.D. study shall make contributions to three areas; location analysis, ACO and hybrid metaheuristics.

2.5.1 Research Question

Clearly, the ambiguity associated with published results of heuristic solution methods raised by ReVelle and Eislet (2005) and ReVelle et al. (2008) indicated that there was a gap in the existing knowledge base. Furthermore, ACO had not been applied to the CFLP. A study into the use of ACO as a solution technique would provide sufficient information to determine if it was a suitable solution method for the CFLP and identify if a dominant heuristic method existed.

A general research question is: *Does ACO provide a suitable solution framework platform for solving capacitated location problems?*

Chapter 3

Research Methodology

The previous chapter indicated that there was a gap within research materials associated with the development and application of ACO to solve mixed-integer optimisation problems, such as the *capacitated facility location problem* (CFLP). Evidence showed, (Agar and Salhi, 1998, Bischoff and Dächert, 2007, Chen and Ting, 2008, Lorena and Senne, 2003), that metaheuristics had been applied to pure discrete forms, but there had only been a varied amount of success in solving mixed-integer forms of the CFLP, (Arostegui et al., 2006, Bornstein and Azlan, 1998, Bornstein and Campelo, 2004, Sörensen, 2008), on those test instances available from the OR-Library. The best results obtained were accomplished using metaheuristics or some type of hybrid technique based on local search or mathematical programming techniques (Arostegui et al., 2006, Sörensen, 2008). Recent research conducted by Caserta and Quiñonez Rico (2007, 2009) based on the Cross-Entropy method of Rubinstein (2001) had a great deal of success in tackling the CFLP, and they openly claimed to have solved all of the test problems from the OR-Library.

This thesis investigates the mixed-integer form of the CFLP and critically evaluates the use of ACO as a solution framework. The latter part of this thesis presents a rigorous series of comparative tests on a variety of ant based algorithms against the aforementioned Cross-Entropy method presented by Caserta and Quiñonez Rico (2007, 2009).

3.1 Introduction

This thesis addresses the solution of the CFLP, which is a well-known theoretical problem within the field of operational research (OR). OR is the discipline of analytical decision making that can be employed by business organisations to assist in their strategic decision making policies. Although the CFLP is a theoretical problem, it belongs to the strategic business field of supply-chain management. Consequently, research output associated with this thesis can be used in both algorithmics and a business context.

This chapter presents a methodological approach that incorporates the essence of a philosophical standing which is often applied within business research and relates it specifically to this study. Once the research philosophy is discussed, a section detailing an appropriate research strategy is presented along with a series of research questions. A breakdown of the design, methods and structure required to conduct the proposed research is rationalised to address the research questions. The final section gives a key-point summary.

3.2 Research Philosophy

The act of conducting research requires some initial intellectual thought about why, what and how the research subject and processes are to be implemented and justified. A theoretical research methodology framework generally includes branches of philosophy that relate to the nature of knowledge, i.e its existence and how it is acquired. Research can be described as the search for knowledge or a critical investigation required to establish facts about a certain problem or scenario.

There are three philosophical areas associated with research methodology; epistemology, ontology and axiology (Bryman and Bell, 2007, Saunders et al., 2007). The first, epistemology, concerns itself with the theory and nature of knowledge including; what is knowledge, the quest for knowledge, how should any knowledge acquired be tested and validated. This is often referred to as the natural scientist's model (Saunders et al., 2007). The second term, ontology, is referred to as meta-physics that deals with the existence and knowledge of entities and their hierarchical social groupings. Whilst the final term, axiology, concerns itself to the philosophical study of value in terms of morality, emotional and ethical issues.

The research project at hand involves analysing results at various stages of development of algorithmic procedures and giving a critical investigation and evaluation of the designed features. This will also include comparisons with contemporary published works within the field of study. Although this study relates to operational and strategic business management, the research and development sides of this project domain belong to a computing environment. This lends itself to processes of observation, measuring and testing. Thus an epistemological approach is the most suitable philosophy.

This research focuses upon a specific class of facility location problems, namely CFLPs, and the integrated use of an artificial intelligence optimisation modelling solution procedure. ACO is a technique derived from phenomena observed in natural science and is based on the ability and efficiency in which ants forage for food (Bonabeau et al., 1999, Dorigo, 1992, Dorigo and Stützle, 2004). The natural scientist's methodology of observe, measure, test and infer is appropriate and essential to this research as it enables a thorough testing of any derived procedures.

This methodological view point reflects the principles of positivism (Bryman and Bell, 2007, Saunders et al., 2007). Furthermore, the nature of this thesis relies upon creating new knowledge from existing knowledge within relevent areas of facility location and ACO. To achieve this scrutiny of theory, research statements and testing of hypotheses are necessary, and by collecting data and rigorous testing, the theory is either accepted or rejected. This process of gaining knowledge is thus empirical and deductive. Knowledge is enhanced or gained from experience generated from a thorough testing and then theory is either accepted or revised.

3.3 Research Strategy

A successful research strategy not only relies upon the selection, implementation and inference of relevent methods but also a justification of any methods to be used and how they relate to the study's research aims and objectives. ACO is a stochastic technique that randomly selects moves on a graph or network, where the probability of making a move is based upon a feedback system that uses a combination of pheromone and problem instance information. As the algorithm progresses iteratively these combined levels may change and moves are made with some bias based on these levels at the beginning of each iteration. Consequently, any underlying ACO system is very complex to consider theoretically and an empirical study is an appropriate methodology to use.

The primary aims of this research project are concerned with a critical investigation and evaluation of an adaptive search technique, and its application to a class of combinatorial optimisation problems encountered in the area of capacitated facility location. The objectives of the study are to determine the effectiveness, reliability and potential of using ACO as a solution framework for solving capacitated facility location problems, in particular this research is focused on solving the CFLP.

3.3.1 Key Features for Empirical Analysis of the CFLP

To be able to perform a suitable empirical analysis of this case study, various aspects of performance measures for ACO algorithms need to be considered that will help to outline characteristics of algorithmic performance:

- **Solution variability** due to the randomness of moves made at each iteration the quality of solutions in terms of relative errors of final costs may vary.
- **Algorithmic robustness** with respect to *problem instance*, in terms of fixed costs, assignment costs, capacity and demand.
- **Problem size and run-time issues** related to numbers of potential facilities and customers.

Furthermore, the analysis will need to:

- Provide suitable metrics for comparison with other algorithmic approaches such as Lagrangean Relaxation and Cross-Entropy (Beasley, 1990, 1993, Caserta and Quiñonez Rico, 2007).
- Characterise algorithmic behaviour such as solution convergence and stagnation.
- Identify areas for improvement within the ACO paradigm for facility location.
- Determine the suitability of ACO for capacitated facility location.

3.3.2 Research Hypothesis

The research hypothesis for this empirical study is derived from the general research question proposed in the previous chapter: *The ACO algorithm is a useful metaheuristic for solving capacitated facility location problems.*

To support the acceptance or rejection of this hypothesis, the following research questions need to be answered:

1. What is a suitable representation for the CFLP within an ACO modelling framework?
2. How well do any derived solution techniques perform on test problems available from the OR Library, (Beasley, 1990)?
3. Is there a dominant ACO solution technique?
4. How well does ACO compare to the successful Cross-Entropy solution method, (Caserta and Quiñonez Rico, 2009), across a range of test problems available from the OR Library?
5. Does ACO provide a suitable framework for solving the CFLP?

3.4 Research Methods

The main body of this thesis focuses on research output generated during this Ph.D. study and a critical evaluation of the final solution methods adopted. All algorithmic development is implemented using C++ and any results obtained from the study will follow the guidelines given by Barr et al. (1995) and Hoos and Stützle (2005) on reporting on computational experiments when using heuristics. Also, any statistical analysis shall be performed using the open-source R statistics package. Primarily, the main research methods to be employed are those of theoretical derivation, experimentation, statistical observation and evaluation. The main body of research development, experimentation and evaluation are presented in four key chapters:

- **Preliminary Development and Experimentation** – A formal specification of the CFLP is presented. Also details of how to map the CFLP onto the prescribed ACO framework solution space are discussed. Research is restricted to two common ACO algorithms (*Ant System* and *Max-Min Ant System*) and a sample of test problems from the OR Library are used to determine the effectiveness of these methods. Basic statistical descriptors and graphical output are used to assess solution quality in terms of computational run-time and relative percentage errors from known optimal solutions. This shall provide evidence to address issues associated with the first research question.
- **ACO Hybrid-Metaheuristics for Facility Location** – Using *Max-Min Ant System*, an alternative solution representation is proposed, which divides the process into two parts:

  (a) uses ACO to select what facilities to locate;
  (b) obtain approximate solutions to any underlying subproblems derived by (a), i.e. approximation of transportation problems.

  Initially experiments are carried out on 37 test instances from the OR-Library. These experiments are repeated for a handful of times, using the experimental strategy adopted by Lourenço and Serra (2002), and basic statistical descriptors are used for reporting on run-time and solution quality. At this stage two local search solution improvement methods similar to those used by Agar and Salhi (1998) are introduced; where local solution improvements are sought after by one method that closes open facilities, whilst a second method combines the first one with swapping open facilities for closed ones. Since the two ACO methods are developed on the same computational platform, their run-time statistics can be directly compared.

Research presented in this chapter breaks away from traditional aspects of heuristics for the CFLP, (Agar and Salhi, 1998, Beasley, 1993, Bornstein and Azlan, 1998, Bornstein and Campelo, 2004), which depend upon approximations during their iterative solution procedures as in (b), and presents a novel technique that embeds exact solution methods into the iterative solution process. The ACO technique iteratively selects feasible subsets of facilities which inturn defines many transportation problems, that are solved using an exact linear programming algorithm from the COIN-OR open-source project (Lougee-Heimer, 2003). Recent applications of hybridisation based on combining approximate and exact solution techniques can be found in Blum et al. (2008). As with the approximate hybrid method, solution improvements are sought after by the dropping and swapping of facilities. Again basic statistical techniques are applied to experimental run-times and solution quality for comparison with previous developments. Furthermore, all of the test problems available in the OR-Library are used in the experiments. These experiments are designed to give results that shall help to answer question two.

- **Hyper-Cube Framework for the CFLP** – A natural extension to the *Max-Min Ant System* algorithm is to fix the lower and upper pheromone limits to remain betwen zero and one. Blum and Dorigo (2004), Blum et al. (2001) developed a Hyper-Cube Framework algorithm in an attempt to overcome sensitivity issues that were thought to exist with *Max-Min Ant System*. A main feature of the algorithm is that pheromone levels converge to their limits as the algorithm progresses, i.e. zero or one, which is analogous to facilities either being selected to be opened or closed. Although this method has been applied to the p-median problem, (Olivetti et al., 2005), there is little evidence of applications to the CFLP, (Venables and Moscardini, 2008). Hybridisations as used with the *Max-Min Ant System* are also implemented and a series of similar experiments are carried out for comparison purposes. These experiments are also designed to give results that shall help to answer question two.

- **ACO: Run-Time Analysis and Evaluation** – ACO and Cross-Entropy methods are examples of stochastic optimisation techniques. The implication of this is that solution quality and computational run-times are likely to be random statistical distributions, that may by impossible to consider theoretically and thus an empirical analytical approach is often more appropriate. Subsequently, when performing algorithmic comparisons, it is not recommended to use a small sample of experiments and simple statistical descriptors as false inferences may be made. Consequently, techniques that need to be employed are based upon conducting many experiments over a variety of test problems. How many experiments to conduct is open to statistical debate as these types of algorithms are deemed to be incomplete, i.e. they can not be guaranteed to find optimal solutions to problems in finite time. However, a sample size can be justified experimentally.

  A run is defined as the execution of an algorithm on a particular problem instance, that is terminated upon reaching some predefined stopping criteria. The run-time of a particular problem instance is the elapsed period of time taken from the start to reaching the algorithm's stopping criteria, which can either be measured in units of CPU clock-time or computational operational counts. ACO is a stochastic process where the time taken to generate a solution is a random variable. Thus to make any inference about algorithmic performance for a particular problem instance the behaviour of its random solution run-time distribution needs to be statistically determined. To generate a run-time distribution for a particular instance a series of runs must be completed and their run-times recorded. If the algorithm being tested has any parameters that need to be set prior to execution then, to avoid parameter dependant variations, these must be identical for all of the executed runs. Run-time distributions can also be derived for ensembles of problem instances to help determine any general qualitative algorithmic run-time issues. This approach of using statistical techniques to characterise randomly generated run-time distributions is referred to as run-time analysis by Hoos and Stützle (2005).

Two run-time methods are available to determine the behaviour of these algorithms. The first is based on computational run-time measures (CPU) whilst the second uses run-time operational counts. To compare ACO and the Cross Entropy method, the former is chosen as both algorithms can be executed on the same machine and direct run-time distributions can be compared both qualitatively and quantitatively. The latter method is rejected because within the ACO algorithm an exact method is employed, where it is not possible to obtain operation counts when called upon.

Initially, a qualitative comparison of the distributions can be made by comparing graphs of the probability of solving a given problem against the time required to solve that problem. To construct a probability distribution, for a given problem consists of an experiment of one thousand runs. Guidelines for conducting these types of experiments are given in Hoos and Stützle (2005). Should any further analysis be required then non-parametric hypothesis testing based on median run-times may be conducted using the Mann-Whitney U-test. However, this test may lead to false acceptance or rejection of the appropriate null and alternate hypotheses (Type I and Type II errors). Alternatively, an approach that is becoming more acceptable than standard hypothesis testing is that of statistical bootstrapping. This method can be used to random sample from a sampling framework (large set of experiments for a given problem) to obtain a confidence interval of median run-times. This bootstrapping method is at present not evident in the application of metaheuristics to facility location research literature. Upon completion of the analysis and evaluation the answers to research questions three, four and five shall be available.

Furthermore, conducting a thorough set of run-time experiments on the OR-Library test problems and performing comparative analyses with the Cross-Entropy method shall provide insights into the behaviour and design of these hybrid-metaheuristic methods. Although Hoos and Stützle (2005) presented a thorough overview of using run-time analysis for stochastic optimisation, there is little evidence of this type of empirical approach to algorithmic performance and design within metaheuristic applications to facility location. Thus, this section will advance existing knowledge and provide substantial new material for future publications and indicate directions to follow for further research.

3.5 OR-Library Test Problems

Throughout this study the test problems that are available from the OR-library shall be used. Table 3.1 presents a summary of the test problems available, further details of which can be found in Beasley (1988). The rationale to use only use these problems is that their optimal solutions are already known and they often used as standard problem instances by researchers (Beasley, 1993, Bornstein and Azlan, 1998, Bornstein and Campelo, 2004, Caserta and Quiñonez Rico, 2007, 2009, Sörensen, 2008). Additional medium sized test instances could be generated, as discussed by Caserta and Quiñonez Rico (2009), and then optimally solved using a commercial mixed-integer solver such as CPLEX or the CBC module of the COIN-OR Library (Lougee-Heimer, 2003). However, it is the inconsistent use of test problems that go towards explaining the term *ambiguity* that is used by ReVelle and Eislet (2005) and ReVelle et al. (2008). The use of these library problems allows for a thorough series of experiments to be conducted, which are required to conduct run-time analyses and construct any corresponding empirical run-time distributions.

Run-time distribution analysis requires a great deal of computational experimentation. Consequently, run-time execution time limits need to be adhered to when collecting sample data. A CPU-clock time of ten minutes shall be set a the maximum time allowed for a single experimental run, i.e. a problem has to be solved in under ten minutes otherwise the run is terminated and the best least-cost solution is recorded along with the time when it occurred. Approximately 1000 runs per problem are required to build an empirical distribution, (Hoos and Stützle, 2005). Potentially, each problem may not be solved within the maximum time limit. There are 49 test problems available in the OR-Library, which indicates a worst-case run-time of $10 \times 1000 \times 49$ mins ≈ 341 days per algorithm to be tested.

To enable a practical experimental methodology, a handful of experiments conducted on each test problem shall help to decide which problems are to be selected for the set of substantial run-time analyses later in this study. The empirical run-time distributions will indicate any algorithmic traits, whether then positive or negative. These characteristics would also be evident in larger and more difficult problems to solve, but would be compounded by longer run-times. So, by using the OR-Library problems more control is placed over experimentation without the

Problem Set	Potential Facilities (m)	Number of Customers (n)	Facility Capacity	Facility Fixed Costs (000s)
cap41-44	16	50	5000	7.5/12.5/17.5/25.0
cap51	16	50	10000	17.5
cap61-64	16	50	15000	7.5/12.5/17.5/25.0
cap71-74	16	50	58268	7.5/12.5/17.5/25.0
cap81-84	25	50	5000	7.5/12.5/17.5/25.0
cap91-94	25	50	15000	7.5/12.5/17.5/25.0
cap101-104	25	50	58268	7.5/12.5/17.5/25.0
cap111-114	50	50	5000	7.5/12.5/17.5/25.0
cap121-124	50	50	15000	7.5/12.5/17.5/25.0
cap131-134	50	50	58268	7.5/12.5/17.5/25.0
A	100	1000	8000/10000/12000/14000	Random
B	100	1000	5000/6000/7000/8000	Random
C	100	1000	5000/5750/6500/7250	Random

Table 3.1: OR-Library test problems

loss of valuable algorithmic behaviour.

3.6 Summary

This chapter began by introducing the research topic and related it to academic fields of business, management, computing, computer science and operational research. The research philosophy section discussed how and why epistemology, positivism, deductive and empirical approaches were appropriate to this study based on a natural scientist's point of view. The research strategy section presented the research aims and objectives and focused on a particular type of capacitated facility location problem, the CFLP. Details of important areas for consideration were discussed and a series of research questions were given. The research methods section discussed four key areas of the research project and how they related to obtaining answers to the derived research questions. Also, discussions were made concerning the merits and use of various data collections and statistical analyses. Finally, the potential of contribution to knowledge and future research directions were indicated through the uses of run-time analyses.

Chapter 4

Preliminary Development and Experimentation

This chapter presents a formal specification for the CFLP and describes its relationship to the p-median problem, and shows that the complexity of the problem is **NP**-hard. Various aspects of ACO are discussed including algorithmic framework, design and execution phases with an emphasis on applications to the CFLP. Two popular ant colony algorithms are developed, a series of experiments are conducted, important results are presented and discussed. Conclusions and recommendations for further research and development for using ant colony algorithms are detailed in the final section.

4.1 Formal Specification of the CFLP

The CFLP considers the problem of selecting a subset of facilities from a set I of m available facility locations, that need to resource a set J of n customers at a minimum cost. Each customer $j \in J$ has an associated demand q_j to be resourced by at least one facility and each facility $i \in I$ has a finite amount of resource available Q_i. The transportation cost of resourcing a unit of demand to a customer j from a facility i is c_{ij}. Also, each facility i that is selected incurs a one-off fixed usage or opening charge f_i. The objective is to select facility locations that can supply all of the customers at an overall minimum cost. Define:

$x_{ij} =$ the fraction of the demand of customer j resourced from facility i,

and the decision variable associated with opening a facility i

$$y_i = \begin{cases} 1 & \text{if facility } i \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

The CFLP is formulated as

$$\text{min} \quad z = \sum_{i \in I} \sum_{j \in J} q_j c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \tag{4.1}$$

such that

$$\sum_{i \in I} x_{ij} = 1 \qquad \forall\, j \in J. \tag{4.2}$$

$$\sum_{j \in J} q_j x_{ij} \leq Q_i y_i \qquad \forall\, i \in I. \tag{4.3}$$

$$x_{ij} \leq y_i \qquad \forall\, i \in I \ \wedge\ \forall\, j \in J. \tag{4.4}$$

$$y_i \in \{0, 1\} \qquad \forall\, i \in I. \tag{4.5}$$

$$0 \leq x_{ij} \leq 1 \qquad \forall\, i \in I \ \wedge\ \forall\, j \in J. \tag{4.6}$$

$$I = \{1, 2, \ldots, m\} \ \wedge\ J = \{1, 2, \ldots, n\}. \tag{4.7}$$

Equation (4.1) is the objective function used to minimise the total fixed and supply costs associated with facility and allocation variables y_i and x_{ij}. Constraint (4.2) ensures that the demand q_j of each customer j is satisfied. Constraint (4.3) ensures that an open facility i does not supply more than its capacity Q_i. Constraint (4.4) further strengthens (4.3) by only allowing the assignment of customer j to a facility i that is open. Constraint (4.5) is a binary or integral condition, concerned with a facility i being selected as opened or closed. Constraint (4.6) refers to the fractional assignment condition that allows the demand of customer j to be allocated to more than one facility. Finally, constraint (4.7) are the sets of indexes that refer to discrete facility locations and customers.

4.1.1 Complexity of the CFLP

Researchers often describe the CFLP as being **NP**-hard because of its relationship to the p-median problem. However, although Garey and Johnson (1979) are often cited for the complexity of the uncapacitated facility location this does not appear directly within their text. Yet the p-median does appear as a network design problem, MIN-SUM MULTICENTER [ND51] (Garey and Johnson, 1979). To show that CFLP belongs to the same class as the p-median problem it is only necessary to determine an equivalent instance of the CFLP. Then the CFLP is at least as difficult to solve as the p-median problem and hence **NP**-hard.

The p-median problem is concerned with obtaining a set of p-facility locations, where each location is a median, so as to minimise the total demand-weighted travel distance between demand and facilities nodes on a network. Demand nodes are represented by a set J of n customers, where $J = \{1, 2, \ldots, n\}$. Potential median facility locations are represented by the set I , where $I = \{1, 2, \ldots, n\}$. The notation for the p-median problem is similar to that of the CFLP:

Define

$q_j =$ the demand of customer $j \in J$,

$d_{ij} =$ the distance from node $j \in J$ to facility node $i \in I$,

$p =$ number of median facilities to be located,

with the supply-demand decision variable

$$x_{ij} = \begin{cases} 1 & \text{if demand node } j \in J \text{ is supplied by facility node } i \in I, \\ 0 & \text{otherwise,} \end{cases}$$

and the facility decision variable

$$y_i = \begin{cases} 1 & \text{if facility } i \in I \text{ is opened,} \\ 0 & \text{otherwise.} \end{cases}$$

The p-median problem is formulated as

$$\text{min} \quad z = \sum_{i \in I} \sum_{j \in J} q_j d_{ij} x_{ij} \tag{4.8}$$

such that

$$\sum_{i \in I} y_i = p \tag{4.9}$$

$$\sum_{i \in I} x_{ij} = 1 \qquad \forall\, j \in J. \tag{4.10}$$

$$x_{ij} \leq y_i \qquad \forall\, i \in I \quad \wedge \quad \forall\, j \in J. \tag{4.11}$$

$$y_i \in \{0, 1\} \qquad \forall\, i \in I. \tag{4.12}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, i \in I \quad \wedge \quad \forall\, j \in J. \tag{4.13}$$

The objective function is used to minimise the total demand-weighted distance between customers and facilities is given in (4.8). Constraint (4.9) ensures that

p facilities are located. Constraint (4.10) ensures that all demand is supplied. Constraint (4.11) further strengthens (4.10) by only allowing the demand of a customer j to be served by a facility i that is open. Constraints (4.12) and (4.13) are binary or integral conditions, concerned with facilities being selected as either opened or closed and the assignment of customers to respective facilities.

To transform an instance of the CFLP into a p-median problem, initially set the capacity of each facility to be $Q = \sum_{j=1}^{n} q_j$. This eliminates the need for the capacity constraint (4.3), as each facility is capable of serving all of the customers. Furthermore, due to the property of supply and demand points being network nodes, all customers' demands will be assigned to their nearest demand-weighted facilities. This is achieved without the necessity of sharing single demand needs amongst several facilities. Hence, the constraint placed on the variable x_{ij} given in (4.6) can be replaced by a binary decision variable $x_{ij} \in \{0, 1\}$. Also, if the fixed costs of each facility are constant, i.e. $f_i = f \ \ \forall i$, then the fixed costs in the objective function (4.1) can be ignored. Finally, if there are p facilities to be opened to give an optimal solution then the equality constraint $\sum_{i=1}^{m} y_i = p$ is introduced.

The objective function (4.1) becomes that of a p-median problem given in (4.8) as the fixed costs can be ignored. Capacity constraint (4.3) is replaced by p-median constraint (4.9). Constraint (4.2) is identical to (4.10), as constraint (4.6) becomes integral it is equivalent to (4.13). Also, constraints (4.4) and (4.11) are the same. Thus, this instance is equivalent to a p-median problem. Hence, the CFLP must be at least as difficult to solve as the p-median problem and must also be **NP**-hard.

The first stage of this process is to specify the CFLP in terms of the ACO solution framework, which is the subject of the next section.

4.2 ACO Framework: Modelling Criteria

The ACO metheuristic initially allows artificial ants to construct solutions to the proposed combinatorial optimisation problem by performing steps either on a graph or network representation to create pathways for each ant. An ant's solution is a completed pathway having a length or cost associated with it, that usually represents a feasible solution although it is possible to also consider infeasible solutions (Dorigo and Stützle, 2004, Lourenço and Serra, 2002). A combinatorial minimisation problem can be described as a triplet (S, f, Ω), where S is the set of potential solutions, f is the objective cost function that assigns a cost $f(s)$ to each potential solution $s \in S$, and Ω is the set of constraints placed upon the problem. The aim is to obtain a solution $\{s^* \mid f(s^*) \leq f(s), \forall s \in S \setminus s^*\}$, whilst simultaneously satisfying all of the constraints placed on the problem i.e. $\Omega(s^*) \mapsto \top$.

The solution space S is represented as a network or completely connected graph that is referred to as a construction graph $G_C = (C, L)$, with a set nodes for components C and a set of links or edges that fully connect the nodes L. A feasible solution s is obtained by conducting a random walk on the graph G_C via the nodal components C along the nodal links L, whilst adhering to the constraints Ω. Thus to solve a given problem using ACO, the problem must first be modelled as a graph according to the conditions described earlier in this section.

4.2.1 Characteristics of an ACO Construction Graph for the CFLP

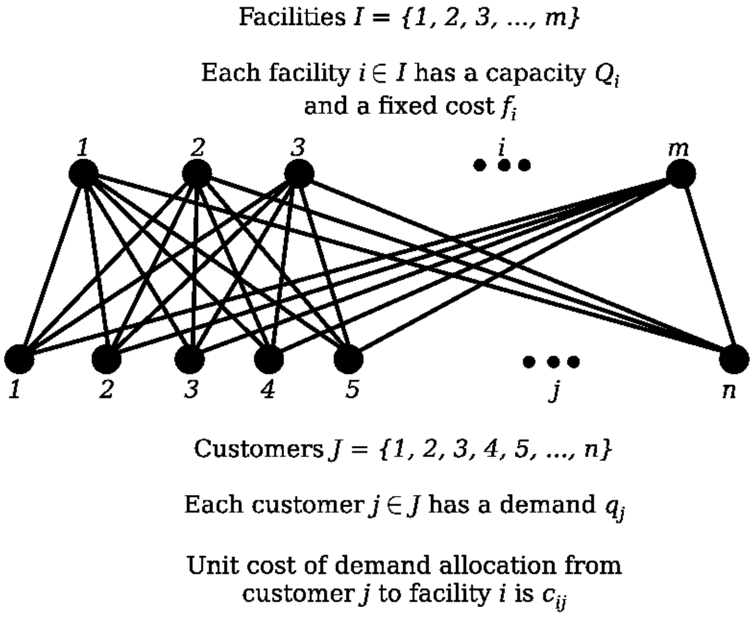
As previously discussed any problem under consideration must satisfy certain modelling criteria before an ACO solution method can be used. The distribution of potential facilities and customers within the CFLP can be easily modelled as a network or graph problem made up of nodes and links. One way of visualising this is to describe the potential assignments of customers to facilities as a bipartite graph, as shown in Figure 4.1. Other graph modelling schematics shall be presented and discussed in later chapters.

The first stage of mapping the CFLP onto a combinatorial optimisation triplet of the form (S, f, Ω), is to define the solution space to be the construction graph $S \subseteq G_C = (C, L)$. Where the set of component nodes C is the combined set of facility nodes I and demand or customer nodes J, i.e. $C = I \cup J$, and the set of links L are all the links that represent possible assignments of nodes in I with nodes in J and vice-versa. Artificial ants use this construction graph as network of various pathways that are built stochastically to provide solutions.

The next step is to define a solution $s \subseteq G_C$, i.e. a solution is a subgraph of the construction graph G_C. Each generated solution contains assignment links from customers to facilities and vice-versa which have associated costs. Collectively these costs are known as an objective function cost f, that is defined as the total fixed and variable costs for a solution s and is stated as z in equation (4.1). Thus a derived solution s will have an objective cost of $z(s)$. Consequently, any ant that searches through the construction graph has the ability to build a pathway that connects all of the customers to a subset of facilities at some overall cost.

The final step determines the feasibility of a generated solution $\Omega(s) \mapsto \top$, i.e. all feasible solutions must satisfy the constraints placed on the CFLP as given by constraints (4.2)–(4.6). Details of how to ensure that only feasible solutions are built during the construction phase of a solution s are presented in the next section. The overall aim is to use a colony of artificial ants to find an optimum solution $\{s^* \mid z(s^*) \leq z(s), \forall s \in S \setminus s^*; \Omega(s^*) \mapsto \top\}$, where each ant in the colony builds a feasible solution and at least one ant finds the best solution.

Figure 4.1: Bipartite graphical representation of potential assignments of customers to facilities.


Facilities $I = \{1, 2, 3, ..., m\}$
Each facility $i \in I$ has a capacity Q_i and a fixed cost f_i
1
2
3
i
m
1
2
3
4
5
j
n
Customers $J = \{1, 2, 3, 4, 5, ..., n\}$
Each customer $j \in J$ has a demand q_j
Unit cost of demand allocation from customer j to facility i is c_{ij}

4.3 Design of an ACO Algorithm

There are two primary phases within any ACO algorithm, the first is *solution construction* and the second is *pheromone update*. Many ACO algorithms also include an optional third phase concerned with solution improvement or *local search*, that is usually performed prior to the pheromone update phase. Initially, this research concentrates on the two primary phases. At this early stage of research it is necessary to determine if a pheromone model that is based on a bipartite graph is suitable for an ACO development and implementation. Thus a rational choice is to only consider the influence of a basic ACO algorithm without the need for local search. A basic outline of the three phases of an ACO algorithm is shown in Algorithm 4.1.

Algorithm 4.1: Outline of a basic ACO algorithm

```
initialise pheromones and heuristic information

while termination condition not met do
    Construction Phase
    Local Search Phase;                              // optional phase
    Update Phase;
end
```

4.3.1 Solution Construction Phase

In an ACO algorithm, artificial ants perform random walks on a construction graph that represents a problem's discrete solution space. Ants move from one component node to another via a link or path. Although ants make random choices when faced with a move to make, they do so with bias towards more promising links or component nodes. Effectively, this is an artificial response or reaction to an ant's local environment which is known as a stigmergy process. To aid this process pheromones (τ) are placed either on the component nodes or the connecting links of the construction graph. Ants can only move from one component node to another via a single link, i.e. to a neighbouring node. Unlike real ants, artificial ants can be programmed with some information already known about a specific problem being solved to assist further with any decisions that need to made. This *a priori* information remains fixed throughout the algorithm's run-time and is called *heuristic information* (η). A probability distribution associated with making a move to a node from a given neighbouring node is a function of current pheromones levels and heuristic information. When ants tour a construction graph they generally build feasible solutions, thus a completed ant tour for a prescribed problem shall have a length or cost and must satisfy all of the given constraints. Tours of shortest lengths or lowest costs are more likely to contain components or links belonging to an optimal solution. Each ant builds its own solution based on the pheromones and heuristic information available at the beginning of an iteration. Pheromone and solution improvement updates only occurs after all of the ants have completed their tours.

The bipartite graphical representation of Figure 4.1 for the CFLP shows links from facility nodes $i \in I$ to customer or demand nodes $j \in J$. Pheromone levels τ_{ij} and heuristic information η_{ij} are associated with links (i, j). Thus when an ant builds a solution, it does so by making alternating moves from either a facility node to a demand node or from a demand node to a facility node. Consequently, any probability distribution associated with making a move will not only depend on pheromone levels and heuristic information but also on the type of move being made. The probability of making a move from a facility node i to a customer or demand node j along link (i, j) is calculated as:

$$p_{ij} = \frac{\tau_{ij}^{\alpha}\ \eta_{ij}^{\beta}}{\sum_{l \in N_i} \tau_{il}^{\alpha}\ \eta_{il}^{\beta}}, \quad \text{if } j \in N_i, \tag{4.14}$$

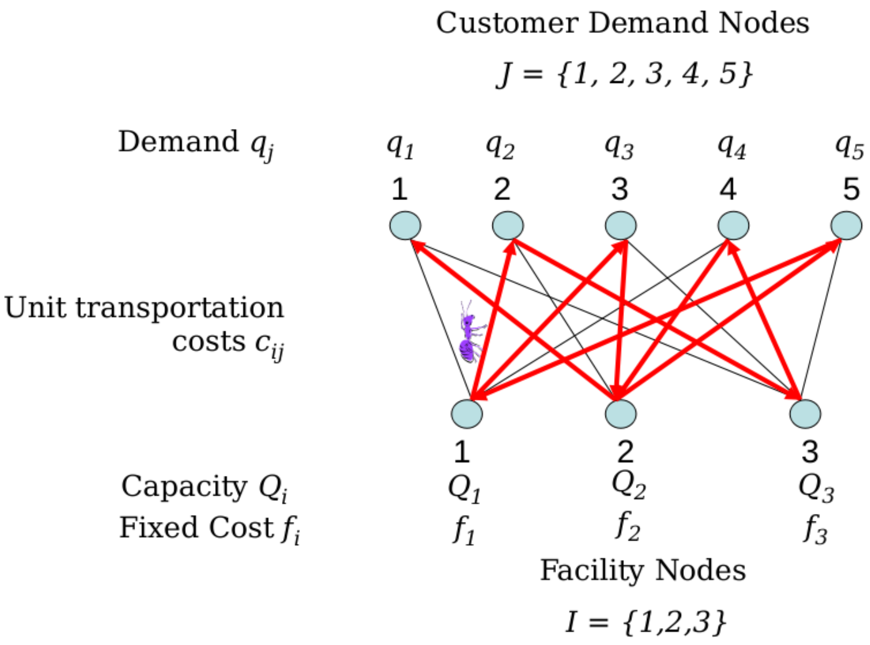
where $N_i = \{j\ |q'_j > 0,\ \forall j \in J\}$ represents all of the potential moves an ant can make from facility i to those customers that have some remaining demand, q'_j, to be supplied. Similarly, a move from a customer or demand node j to a facility node i along link (j, i), which is equivalent to a move along link (i, j), is:

$$p_{ij} = \frac{\tau_{ij}^{\alpha}\ \eta_{ij}^{\beta}}{\sum_{l \in N_j} \tau_{lj}^{\alpha}\ \eta_{lj}^{\beta}}, \quad \text{if } i \in N_j, \tag{4.15}$$

where $N_j = \{i\ |Q'_i > 0,\ \forall i \in I\}$ represents all of the potential moves an ant can make from a customer j to those facilities that have some spare capacity available, Q'_i. The parameters α and β are fixed and represent the importance of pheromone intensity and heuristic information. When a move is made, as much demand is assigned as possible, which may necessitate respective changes in facility capacity, Q'_i, and customer demand levels, q'_j. The cost of a move is computed as the cost of demand allocation and fixed facility cost if the facility needs to be opened. A completed tour is made once all of the customers' demands have been allocated. Each tour is constructed from an initial starting point or node that is randomly selected from the set of the construction graph nodes C.

Figure 4.2 displays an example of an ant's tour for a small problem consisting of five customers and three facilities. Capacities Q'_i and demands q'_j are initialised to their respective Q_i and q_j values. The tour starts by randomly selecting a facility, in this example facility 1 is selected and its one-off fixed cost f_1 is recorded as the initial total cost. Since none of the customers have been assigned then $N_i = \{1, 2, 3, 4, 5\}$ and a customer is selected according to equation (4.14), in this example customer 2 is selected. As much demand as possible is then allocated from customer 2 to facility 1 and the corresponding Q'_i and q'_j values are updated. The cost of assigning demand from $j = 2$ to $i = 1$ is computed as amount of demand assigned multiplied by the appropriate unit transportation cost, which is then added to the current total cost. The next move involves randomly selecting a facility from those facilities that have sufficient spare capacity as defined in equation (4.15). Although the customer's demand values and facility capacities are not explicitly detailed in this example, all of the facilities are assumed to have some spare capacity, so $N_j = \{1, 2, 3\}$ and facility 3 is randomly selected. The whole process of moving alternatively between facilities and customers is repeated until all of the customers' demands are assigned. The number of elements in N_i reduces in size until $|N_i| = 0$ when no further demand requires assigning to a facility, and the elements in N_j reduces as facilities reach their capacity levels.

Figure 4.2: Example of an ant's tour to assign customers to facilities



4.3.2 Pheromone Update Phase

A positive feedback or recruitment system that intensifies pheromone levels on those nodes or pathways that are likely to be in an optimum solution is implemented. However, an over amplification of pheromones may misguide the ants to converge to a poor or non-optimal solution. In an effort to overcome this some negative feedback or reduction of pheromone levels must also take place which is referred to as *pheromone evaporation* (ρ). It is the depositing and evaporation of pheromones in this phase that provides the main differences between various ACO based algorithms (Dorigo and Socha, 2006, Dorigo and Stützle, 2004). Pheromone evaporation takes place first and is applied to all of the pheromones. Whereas, pheromone deposits that reflect solution quality are made subject to some criteria depending upon the type of ACO algorithm being implemented. Two popular ACO algorithms are described in the next subsection together with their specific design attributes for case of the CFLP and pheromone update rules.

In both cases solutions are constructed on a bipartite graph as shown in Figure 4.1 and as described in section 4.3.1.

4.3.3 Ant System and Max-Min Ant System for the CFLP

Many ACO algorithms are based on extensions to the original Ant System (AS) algorithm proposed by Dorigo (1992). The Max-Min Ant System ($\mathcal{MM}$AS) algorithm (Stützle, 1999, Stützle and Hoos, 1997, 2000) is a modification of AS, which quoted as one of the most successful ACO algorithms for solving a variety of combinatorial optimisation problems by Blum et al. (2008). As described earlier in section 4.3 this initial study shall concentrate on the basic influence of ACO for solving the CFLP.

Dorigo and Stützle (2004) presented a $\mathcal{MM}$AS algorithm to solve the classical travelling salesman problem (TSP). They empirically observed that the size of the colony of ants had little impact on computational performance on instances of up to 500 cities, yet for larger instances there was evidence to support the use of a colony. However, the downside of this was as the colony size became larger then computational efficiency was lost. Through experimentation, they claimed that a suitable size of colony for larger TSP instances would be between two and ten. Also, they observed that a colony of a single ant produced solutions within 0.5% of the optimum value which displayed a slower final convergence than those of a greater colony size. Dorigo and Stützle (2004) suggest that there is evidence to support the use of a pseudorandom proportional rule for the decision of making a move on the graph G_C for the $\mathcal{MM}$AS algorithm. This was a technique originally developed as part of the *Ant Colony System* algorithm (Dorigo and Gambardella, 1997a,b). The $\mathcal{MM}$AS decision rule associated with making a move from a node $i \in I$ to a node $j \in J$ on G_C for the CFLP:

$$j = \begin{cases} \text{argmax}_{l \in N_i} [\tau_{il}]^{\alpha} [\eta_{il}]^{\beta} & \text{if } q \leq q_0, \\ J & \text{otherwise.} \end{cases} \tag{4.16}$$

Where q is a random variable uniformly distributed in interval $[0,1]$, $0 \leq q_0 \leq 1$ is a parameter, N_i is the set of possible moves from node i, and J is a random variable that is selected according to the probability distribution given in equation (4.14). Also, moves from a node $j \in J$ to a node $i \in I$ can be modelled in a similar way.

Using a single ant is a valid concept within the ACO solution construction phase, as a lone ant lays pheromone information for future ants about to leave the nest site. This is equivalent to having a large colony of ants where only one ant at a time is allowed to build a solution during each iteration of the algorithm's run-time. Consequently, the computational behaviour a simple ACO algorithm that only uses a single ant construction process can be used to determine the potential of a bipartite pheromone model to solve the CFLP.

The following update phases and initial pheromone levels τ_0 are adapted from those given by Dorigo and Stützle (2004). Initially, the pheromone levels for AS are set to $\tau_{ij} = \tau_0 = 1/z_0$; where z_0 is computed as the objective cost using a minimun link cost discrete allocation heuristic. However, initial pheromone levels for $\mathcal{MM}$AS are $\tau_0 = 1/\rho\, z_0$; where ρ is the pheromone evaporation rate.

AS Update Phase for the CFLP

Once an ant has constructed a tour then *pheromone evaporation* and an amount of *pheromone deposit* takes place. Pheromone evaporation is applied to all links of the construction graph, whilst pheromones are only deposited on those links forming the ant's tour which is defined as:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \Delta\tau_{ij}, \;\; \forall (i,j) \in L; \tag{4.17}$$

where $\Delta\tau_{ij}$ is the amount of pheromone deposited along the pathway built, T, by the ant at the current iteration:

$$\Delta\tau_{ij} = \begin{cases} 1/z & \text{if link } (i,j) \text{ belongs to ant tour } T, \\ 0 & \text{otherwise.} \end{cases} \tag{4.18}$$

Where z is the objective cost of tour T, and is computed as indicated in the right hand side of equation (4.1). The reciprocal of z given in equation (4.18) ensures that greater amounts of pheromones are applied to those links with shorter lengths or smaller objective costs. Pheromone recruitment occurs on those links that are commonly selected by a different ant at each iteration. Subsequently, as the algorithm iteratively advances any future ant will be more likely to select those links with higher pheromone levels thus converging to a shortest path.

$\mathcal{MM}$AS Update Phase for the CFLP

There are several major differences between the AS and $\mathcal{MM}$AS algorithms. Firstly, pheromones are laid on the links of the best tours, which may not be the path chosen by a current ant. If a colony is used during the search process then a choice can be made between using the best ant solution for the current iteration or the overall best solution found to date as the pathway of links on which to deposit a quantity of pheromone. However, if a colony of unit size is used then pheromones are laid on the pathway of the best solution to date. A potential downside to this process is that over recruitment may occur as the pheromone intensity of the best pathway may misguide ants to a non-optimal solution, this phenomena is referred to as *stagnation*. The second major difference is that, unlike AS where pheromone levels are unbounded, $\mathcal{MM}$AS places a restriction on the upper and lower bounds of the pheromone levels $[\tau_{min}, \tau_{max}]$ in an attempt to avoid early stagnation of the algorithm. Furthermore, the upper limit τ_{max} is initially set to $\tau_0 = 1/z_0$ as in the AS algorithm. The lower pheromone limit is set to a fraction of the upper pheromone limit $\tau_{min} = \tau_{max}/a$, where a usually represents the size of the problem in terms of the number of components C in the construction graph $G_C = (C, L)$. Also, the pheromone levels are reinitialised during the run-time of the algorithm should stagnation occur or no solution improvement occur within a pre-fixed number of iterations.

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \Delta\tau_{ij}^{best}, \;\; \forall(i,j) \in L. \tag{4.19}$$

Where $\Delta\tau_{ij}^{best}$ is the amount of pheromone deposited by an ant along the best pathway built to date, T^{best}:

$$\Delta\tau_{ij}^{best} = \begin{cases} 1/z^{best} & \text{if link } (i,j) \text{ belongs to the best to date ant tour } T^{best}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.20}$$

4.4 Computational Design and Experimentation

This section presents computational experiments and results obtained for a series of benchmark capacitated location problems whose optimal solution are known. The algorithms were coded in C++ and experiments were carried out on a Dell Inspiron 8600 with a 1.60 GHz Pentium M processor and 786Mb RAM. The problems used were taken from the OR-Library (http://people.brunel.ac.uk/ mastjjb/jeb/info.html).

Along with a suitable choice of a heuristic information function η, ACO algorithms require several parameters to be set prior to execution namely α, β and ρ which are common to both AS and $\mathcal{MM}$AS algorithms, whilst q_0 is only required for $\mathcal{MM}$AS. The function for η reflects the preference of assigning customers to their nearest facility and is implemented as $\eta = 1/c_{ij}$; where c_{ij} is the transportation cost of allocating a unit of demand from customer j to facility i. At this stage of development, it is only necessary to evaluate the usefulness of using a standard ACO based approach as a potential solution method for the CFLP. Thus, the dynamic characteristic behaviour of the two algorithms needs to be evaluated in some way. To achieve this fine tuning of the pre-run-time parameters is not required as a secondary optimisation problem may arise, i.e. optimise the parameters to give the best solutions according to some criteria.

A series of experiments were carried out to determine suitable parameters to use for each algorithm and were conducted on the same problem instance following the guidance given in Dorigo and Stützle (2004). The pheromone amplification and heuristic information importance parameters were tested over the intervals $0 < \alpha \leq 2$ and $1 \leq \beta \leq 5$. Also, the pheromone decay rate ρ was determined in a similar manner for each algorithm. The run-time termination criteria was set at 1000 iterations and a pheromone reset was applied to $\mathcal{MM}$AS if no solution improvement was observed after a non-improving period of 200 iterations. The parameter q_0 was tested on the interval $0.1 \leq q_0 \leq 0.9$, and was observed to have an insignificant influence on the solution quality (relative error from the known optimal), yet had a role to play in reducing the best solution run-time. Parameter values were tested using ten runs for each setting. Reliable and efficient parameter settings are defined as those values that give consistently greater solution accuracy and shorter run-times. Consequently, the parameter selection criteria were based on solution accuracy, and the coefficient of variation values for the relative error and solution run-time.

	α	β	ρ	q_0
AS	0.1	1.4	0.5	–
$\mathcal{MM}$AS	1.6	1.4	0.06	0.7

Table 4.1: Parameter setting for AS and $\mathcal{MM}$AS

The parameter settings that were selected to be used across a series of instances are given in Table 4.1. These parameters were then used to solve a series of test instances from the OR-Library. The problems chosen were simply selected to assess the characteristic behaviour of AS and $\mathcal{MM}$AS and do not reflect their levels of difficulty to solve. Each algorithm was executed 30 times and the best solutions with respect to smallest relative errors and computational run-times are presented in Table 4.2. Furthermore, qualitative run-time behavioural aspects are given in Figure 4.3 for two of the instances.

4.5 Initial Conclusions and Recommendations

The charts displayed in Figure 4.3 indicate that the convergence profiles for the number of iterations completed and CPU times are identical. This is because there is no local search improvement method used and the algorithm is only concerned with ACO features, thus CPU run-time and iteration counts are qualitatively equivalent. This issue is very important when deciding to use either CPU run-time or operation counts for an in-depth analysis of empirical run-time distributions, as an operation counts based approach must have the same qualitative profile as the CPU run-time profile.

Results displayed in Table 4.2 and Figure 4.3 display some encouraging results that indicate further development and investigation is required. Although none of the optimal solutions were found, $\mathcal{MM}$AS outperformed AS in terms of solution accuracy and found solutions to within 3% of the optimum in most cases, but struggled on the two larger instances. The run-times for each instance was

Instance	Size	Detail	AS	$\mathcal{MM}$AS
Cap41	16×50	% Error	8.716	2.419
		Iterations	851	955
		Time (secs)	37.284	43.773
		Run Time (secs)	43.803	45.836
Cap51	16×50	% Error	8.241	1.239
		Iterations	820	921
		Time (secs)	33.919	40.158
		Run Time (secs)	41.961	43.603
Cap61	16×50	% Error	4.098	0.696
		Iterations	925	853
		Time (secs)	37.884	37.414
		Run Time (secs)	40.909	44.014
Cap71	16×50	% Error	21.221	0.691
		Iterations	557	967
		Time (secs)	22.682	41.68
		Run Time (secs)	40.688	43.112
Cap81	25×50	% Error	13.939	2.165
		Iterations	798	900
		Time (secs)	49.922	58.955
		Run Time (secs)	62.500	65.494
Cap91	25×50	% Error	8.091	1.678
		Iterations	831	989
		Time (secs)	49.491	61.989
		Run Time (secs)	59.536	62.68
Cap101	25×50	% Error	6.186	1.886
		Iterations	348	977
		Time (secs)	20.700	61.348
		Run Time (secs)	59.355	62.790
Cap121	50×50	% Error	17.618	6.953
		Iterations	486	983
		Time (secs)	54.068	115.356
		Run Time (secs)	111.090	117.359
Cap131	50×50	% Error	16.608	7.642
		Iterations	158	939
		Time (secs)	17.625	113.032
		Run Time (secs)	118.180	120.683

Table 4.2: Experimental results using a bipartite graphical representation for the CFLP

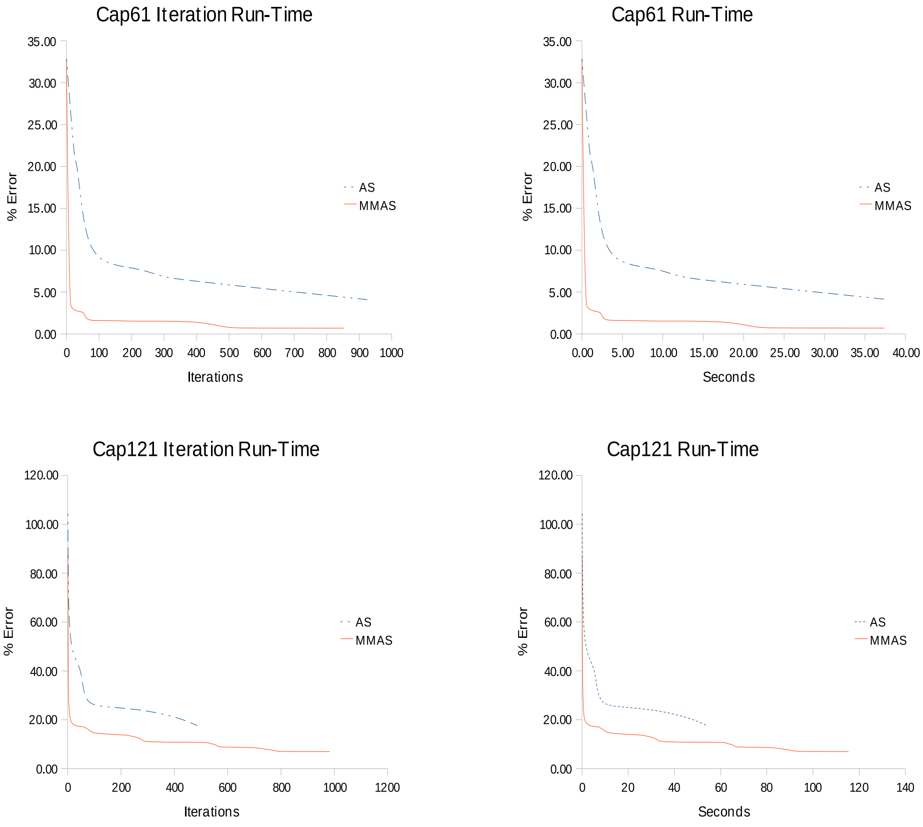


Figure 4.3: Run-time experimentation for two OR-Library Instances

comparable for both algorithms with a clock limit of 1000 iterations. Interestingly, AS found its best solutions in quicker times than $\mathcal{MM}$AS for all instances which is also evident in the run-time graphs in Figure 4.3. The reason for this is that AS is more exploratory at the beginning of the search procedure whereas $\mathcal{MM}$AS exploits solutions around the initial solution passed to the search procedure. This is a potential weakness of $\mathcal{MM}$AS, as a poor initial solution may lead to a premature stagnation of the algorithm. However, the pheromone limits and reset conditions allow the search to move away from regions of stagnation which more than makes up for its early misgivings. As the procedures progress with their search strategies $\mathcal{MM}$AS becomes the more dominant algorithm, due to the exploitation of good solutions.

At this present stage ACO does not perform at an equivalent level of other metaheuristic techniques (Arostegui et al., 2006, Bornstein and Azlan, 1998, Caserta and Quiñonez Rico, 2009, Sörensen, 2008). So it is necessary to identify where improvements may be made: use of more sophisticated *a priori* heuristic information, hybridisation, use of a different ACO model for the CFLP and finally the implementation of an appropriate local search strategy. To achieve a more competitive algorithmic design it is necessary to identify drawbacks associated with those designs discussed in this chapter.

One such area is the computation of probability distributions for making a move from a node to a different node when there may be a very large neighbourhood to choose from. A way of overcoming this is to consider a limitation of the number of nearest neighbours; where a neighbour is defined by some distance or cost metric similar to that used in the *ant colony system* algorithm for the *travelling salesman problem* (Dorigo and Stützle, 2004). These techniques can either be applied prior to the main ACO algorithm or less efficiently during the algorithmic execution phase. Unfortunately, both of these methods introduce sorting procedures and computational storage issues. Consequently, any gains in computing smaller probability distributions may be lost from the computational effort required to perform the sorting and storage prior to each move.

A more promising method would be to reduce the size of the problem in terms of its ACO representation, i.e. a smaller construction graph. To assist in this approach the structure of the CFLP can be exploited. If a set of facilities are given *a priori* which have a total capacity greater than or equal to the total customer demand, then the CFLP reduces to an unbalanced transportation problem. Thus, ACO can be used to determine the best set of facilities to be opened, where the stigmergy process is governed by a combination of opened facilities and transportation problem solutions. This approach is a hybridisation of an ACO algorithm which belongs to the current research area of hybrid-metaheuristics (Blum et al., 2008, Jourdan et al., 2009). Furthermore, a bipartite construction graph would no longer be appropriate as the nodes of an alternative pheromone model would represent facilities and not the union of facilities and customers that were used in this chapter.

A method of addressing the issue of how to compute heuristic information is to use a technique based on a linear relaxation of the problem to define those facilities most likely to be in an optimal solution. The relaxation would give a transportation type problem that can be approximated and used to provide a static heuristic information measure. A similar technique was used by Adlakha and Kowlaski (2004) to solve a source-induced transportation problem, that has a very similar structure to the CFLP.

The use of local search is seen as an optional phase within an ACO algorithm, research in the form of empirical research evidence suggests that this is essential to achieve optimal run-time performance. Indeed, Hoos and Stützle (2005) described ACO as a stochastic local search algorithm. However, during initial development stages it is best not to cloud over ACO behavioural characteristics with the inclusion of local search. It is also a poor approach to spend too much time in developing and overtune an ACO algorithm by ignoring the importance of local search within a metaheuristic approach.

The following chapters of this thesis provide details of a critically progressive investigation of the research and development of an ACO algorithmic approach to solve the CFLP, that will be competitive with existing state-of-the-art metaheuristic techniques that are currently available for solving this class of **NP**-hard problem.

Chapter 5

Hybrid-ACO Development for the CFLP

This chapter introduces a hybrid solution technique to solve the CFLP, which is achieved by creating a communication link between an ACO algorithm and a secondary sub-problem approximation technique. This hybridisation makes use of an ACO construction graph that differs to the standard one, which was discussed in the previous chapter. Also, a new technique to assist with the computation of heuristic information values is derived, that is based on a linear relaxation of the CFLP. In an attempt to improve computational run-times two solution improvement methods are implemented; *DROP* is based on closing open facilities and *SWAP* considers swapping open for closed facilities. The research works of Blum et al. (2008), Dorigo and Stützle (2004), Gambardella et al. (1999) provide a foundation for the development of a *Max-Min Ant System* ($\mathcal{MM}$AS) algorithm to solve the CFLP. The aim of this chapter is to determine if a hybrid ACO algorithm is suitable to solve the CFLP.

5.1 ACO Hybridisation

The use of a hybrid ACO technique was suggested as a way of reducing computational effort and increasing solution accuracy in the previous chapter. Exploitation of the CFLP's structure considerably reduces the size of an ACO construction graph and need only consist of facility locations. A rationale for using this approach is that if a sub-set of facilities are pre-selected that have a total capacity in excess of the total customer demand, then the CFLP reduces to a transportation problem. Transportation problems are either solved approximately using various cost reduction strategies such as Vogel's method or exactly using specialised linear programming techniques, the most common techniques are often presented in operational research texts (Hillier and Lieberman, 2005, Taha, 2006). ACO can be used to pre-select a sub-set of facilities and a transportation solution technique can implemented along side it. The solution quality of any underlying transportation problem then provides positive feedback to the ACO algorithm for pheromone update purposes, where pheromone intensities indicate the likelyhood of facilities being in a solution. The ACO selection process and any underlying transportation problem solution method needs to be applied at each ant move during each iteration.

A further hybridisation can be applied to aid the computation of heuristic information, η, which is used by artificial ants as insight to a problem instance. Ants are guided to components most likely to be in an optimal solution based on heuristic information, which is referred to as *ant visibility* (Dorigo and Stützle, 2004, Venables and Moscardini, 2006). In the case of the CFLP this refers to those facilities that are more likely to be in a final solution.

Figure 5.1 displays the communication process, that is classified as a "High-level Teamwork Hybrid" cooperation process (Jourdan et al., 2009). The lefthand side of Figure 5.1 displays the ACO metaheuristic which consists of the construction and the updates phases. A communication link is made between the ACO and transportation solver phases by passing a set of facilities which define a problem to be solved. Any solution improvements are reported back to the ACO phase to aid pheromone updates.

The local search phase is also included in this chapter, which investigates the implementation of two successfully used local search techniques for the CFLP, *DROP* and *SWAP* (Agar and Salhi, 1998, Beasley, 1993). Initially, a current solution is passed to the local search phase, where improvements are sought by changing the open/closed status of facilities in the current solution which involves the use of the transportation solver. The current solution is only updated upon successful application of the local search phase, where success is when a solution improvement is found.

A generic algorithm for the CFLP is presented in Algorithm 5.1. Each ant in the colony is defined to have a memory associated with which facilities it has chosen to locate and the corresponding objective costs which consist of fixed and transportation costs. Then while some stopping criteria has not been met, which can either be based on computational run-time and/or a maximum number of iterations, the algorithm proceeds. Firstly each ant selects which facilities to open from a complete set of closed facilities, y, ensuring that there is sufficient capacity to supply all of the customers' demand. The cost of an ant's set of facilities is computed as the sum of the opened facility fixed costs and its associated transportation costs. All transportation costs are calculated either approximately or exactly using a transportation problem solver (`TPsolve`). If the optional local search phase is chosen then the ant with the least cost is selected as a candidate for improvement testing. Should an improvement be obtained during the local search then the ant has its memory updated. It should be noted that the local search strategy also makes use of the transportation problem solver (`TPsolve`).

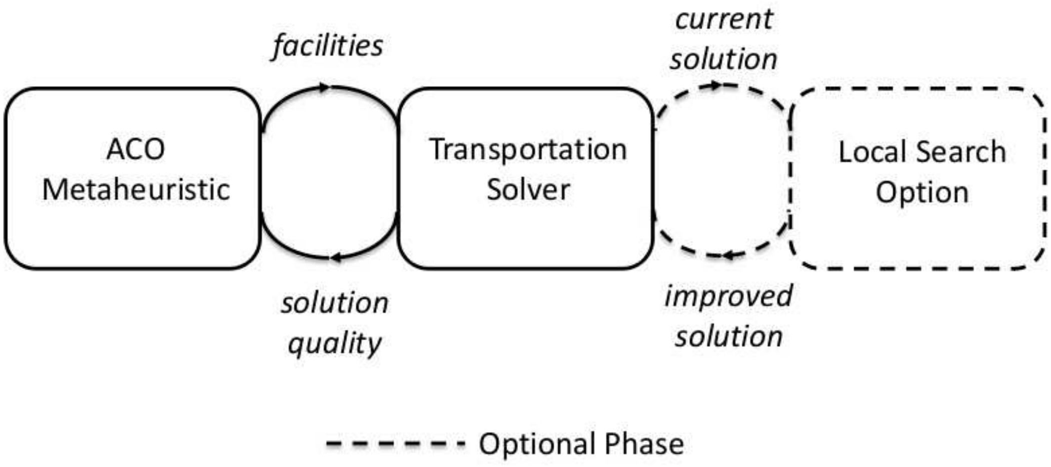


Figure 5.1: Schematic for an ACO hybrid algorithm for the CFLP

The least cost ant is then checked against a global least cost ant to determine if a global improvement update is required. The final stage of the algorithm is to perform a pheromone update phase which is based on the solution quality of the global ant solution or a combination of each of the ant's solutions.

5.2 Hybrid Construction Phase

The solution construction graph presented in the previous chapter was a bipartite graph with vertices made from the union of facilities and customer demand nodes. The edges of the graph were connections from facility to customer nodes such that each customer was reachable form each facility and vice versa. This type of approach is typical in most ACO applications (Dorigo and Stützle, 2004). However, some construction graphs take on other forms that are more advantageous when considering the structure of certain types of discrete optimisation problems (Dorigo et al., 2008, Tarrent and Bridge, 2005). Unlike real ants, artificial ants can lay pheromones on either the links connecting the solution components of the construction graph or on the solution components themselves. If the con-

Algorithm 5.1: Generic outline of a hybrid ACO algorithm for the CFLP

```
define ant:                                              // ant's structure

    ant.facilities                             // ant's facility selection
    ant.cost                         // total fixed facility and TP costs

initialise ant, bestant, LocalSearchPhase

while termination condition not met do
    // Construction Phase
    foreach ant[k] do
        // ant selects facilities using ACO solution construction
        $\mathbf{y} \leftarrow \{y_i | y_i = 1$ if selected, 0 otherwise$\}$
        $ant[k].facilities \leftarrow \mathbf{y}$
        // compute opened fixed costs and transportation costs
        $z \leftarrow \sum_{i \in I} f_i y_i +$ TPsolve($\mathbf{y}$)
        $ant[k].cost \leftarrow z$
    end

    // select ant with least cost
    $k' \leftarrow argmin_{k \in K}(ant[k].cost)$             // $K$ is the set of ants

    // Optional Local Search Phase
    if LocalSearchPhase then
        // attempt to improve current solution found by $ant[k']$
        $ant[k'] \leftarrow$ LocalSearch($ant[k']$)          // makes use of TPsolve

    end

    if $ant[k'].cost < bestant.cost$ then
        $bestant \leftarrow ant[k']$                    // update bestant solution
    end

    // Update Phase
    ApplyPheromoneUpdate()
end
```

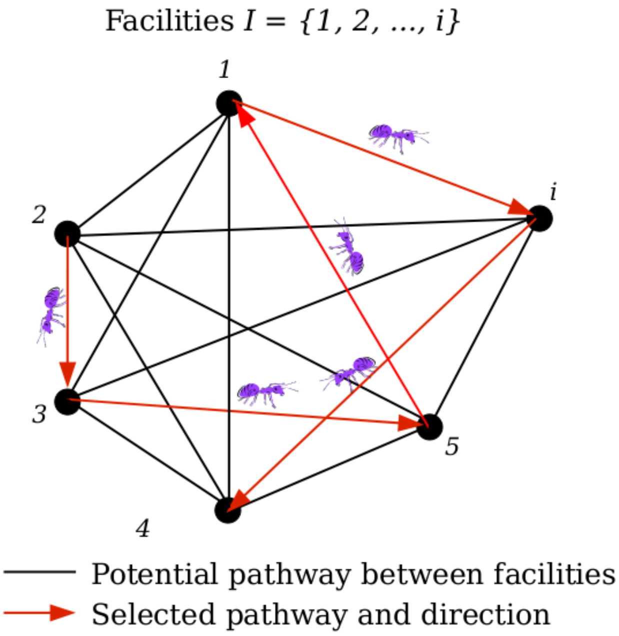

Facilities I = {1, 2, ..., i}
1
2
i
3
5
4
Potential pathway between facilities
Selected pathway and direction

Figure 5.2: Construction graph for a hybrid CFLP ACO algorithm consisting of facilities $I \in \{1, 2, \ldots, i\}$. The plain links between facility nodes represent possible pathways that an ant could take during the construction phase, whereas the arrowed links represent an example of a pathway taken by an ant.

struction graph for the CFLP only contains components that represent facilities, and the links connecting these components are merely a representation to reflect pathways from one facility to another facility, then the size of construction graph is significantly reduced. Thus, ACO can be used stochastically to identify which facilities belong to a feasible solution, i.e. from a set of potential facilities ACO would select which facilities to open or close. To ensure that the capacity constraint (4.3) is satisfied, any tour on the graph would have to open sufficient facilities to guarantee a feasible solution. Figure 5.2 shows a construction graph structure containing only facility nodes, $I \in \{1, 2, \ldots, i\}$, and links between facility nodes to demonstrate that all facilities are reachable from one another via a single link or pathway, which defines a large neighbourhood.

Initially all facilities are set to be closed, i.e. $y_i = 0, \ \forall i \in I$. A facility is then selected at random from a uniform distribution to define a starting point for the tour. The capacity and demand constraints must be adhered to during the construction phase to ensure that valid and feasible solutions are constructed. Firstly, all constructed solutions must be feasible, i.e. the total capacity of those facilities chosen to be included in a solution must be capable supplying all of the customers' demand. Secondly, to avoid recycling of facilities during the construction phase, facilities should only be visited only once to determine their open or closed status. Thirdly, any facility that is randomly selected via pheromone bias has its status set by evaluating an overall objective cost, z, which is the sum of the fixed open costs and the associated transportation costs, $f_i y_i + \texttt{TPsolve}(\mathbf{y})$. If a constructed solution improvement occurs, $z < z_c^*$, then the facility is fixed as open by setting $y_i = 1$ and the best constructed solution cost, z_c^*, is updated otherwise the facility remains closed. It is important to note that the fixed status of facilities is only applied during an ant's tour and that all further tours have facilities initially set to be closed. A mini-step algorithmic scheme that describes the solution construct phase is given in Algorithm 5.2 and an example of an ant's pathway is shown in Figure 5.2.

As discussed in Section 5.1, if a selected set of facilities satisfies the capacity constraint (4.3), then the CFLP reduces to a transportation problem. When a tour is constructed, facilities are only added to a current solution if there is an overall cost improvement. This cost evaluation requires a solution to a transportation problem at each step made on the construction graph; where the overall cost is the sum of the opened facility fixed costs and the underlying transportation problem costs. Computing solutions to transportation problems each time a subproblem is a time-costly process and is a well known issue for the CFLP, which is particularly highlighted in algorithmic approaches using Lagrangean relaxation

Algorithm 5.2: Ant solution construction phase

```
Input: $\tau, \eta$                                  // pheromone and heuristic information
Output: $\mathbf{y}$                                    // set of open/closed facilities

// initialise
forall the $i \in I = \{1, 2, \ldots, m\}$ do
    $y_i \leftarrow 0$                               // all facilities set to be closed
    $z_c^* \leftarrow \infty$                              // objective costs (fixed and TP)
    $\nu_i \leftarrow 0$                            // set all facilities to be unvisited
end

while not allvisited do
    $i \leftarrow$ ACOselect($\tau, \eta, \nu$)                   // select an unvisited facility
    $\nu_i \leftarrow 1$
    $y_i \leftarrow 1$
    // test for facility inclusion
    if $\sum_{i \in I} Q_i \, y_i >= \sum_{j \in J} q_j$ then   // ensure there is sufficient capacity

        $z \leftarrow \sum_{i \in I} f_i y_i +$ TPsolve($\mathbf{y}$)             // compute objective cost

        if $z < z_c^*$ then
            $z_c^* \leftarrow z$                             // update best solution found
        else
            $y_i \leftarrow 0$                              // do not included facility
        end
    end
end
```

(Agar and Salhi, 1998, Beasley, 1988, 1993, Christofides and Beasley, 1983, Daskin, 1995). When using iterative heuristic techniques, which may also use local search solution improvement methods, it not unusual to require the solution to hundreds or thousands of transportation problems (Agar and Salhi, 1998). Although, exact solution algorithms based on linear programming exist to solve the transportation problem, see Goldberg (1997), approximate solution techniques are usually used.

5.3 Transportation Problem Approximation

The use of approximate methods to find solutions to transportation problems defined by open facilities in the CFLP has also been applied to various metaheuristic algorithms with limited success: a Simulated Annealing algorithm that used an approximation transportation algorithm was developed by Bornstein and Azlan (1998), Bornstein and Campelo (2004). Their research provided results for twenty five of the OR-Library test instances, which found the optimum solutions for seven instances and gave an average relative error of 0.17% with a maximum error of 2.42%; Genetic Algorithm, Simulated Annealing and Tabu Search solution methods for the CFLP were the subject of a thorough empirical research by Arostegui et al. (2006). They found that Tabu Search was most promising method, which was closely followed by Simulated Annealing whilst a Genetic Algorithm performed which performed very poorly on the CFLP; support for Tabu Search was also provided by Michel and Hentenryck (2004) and Sörensen (2008).

As with Lagrangean relaxation methods, metaheuristic approaches have often adopted approximation techniques that were based on the use of Vogel's approximation method, which is a standard technique that can be found in most introductory operational research text books (Hillier and Lieberman, 2005, Taha, 2006).

However, most types of transportation problems for the CFLP turn out to be unbalanced versions where total capacity is larger than total demand. Although this issue can easily be rectified by introducing a dummy variable to balance the capacity and demand, Kirca and Satir (1990) developed a total opportunity cost method (TOM) that was superior to Vogel's method at generating solutions for unbalanced problems; see Algorithm 5.3. Recent works of Mathirajan and Meenakshi (2004) and Krishnaswamy et al. (2009) deduced that although the method of Kirca and Satir (1990) performed well on unbalanced problems, yet it struggled in comparison to Vogel's method on certain types of balanced problems. A major advantage of the algorithm outlined by Kirca and Satir (1990) is that it is computationally more efficient than Vogel's method. This computational efficiency was successfully exploited by Agar and Salhi (1998) in their Lagrangean relaxation heuristic. Consequently, TOM shall be used in the ACO algorithms developed in this chapter to compute approximate solutions to any derived transportation method. Furthermore, calculation of penalty values that are used by the TOM algorithm, T_{ij}, play a key role in providing heuristic information within the proposed ACO algorithms and are detailed in the next section.

5.4 Derivation of Ant Visibility

As in the previous chapter the use of *a priori* information is used to assist the search process of an ACO algorithm, this information is referred to as heuristic information, η. Its role is to provide bias towards solution components that are most likely to be in an optimum solution. This is analogous to artificial ants being provided with a rough plan of what tour to take prior to any ants leaving the nest and is referred to as *ant visibility* by Dorigo and Stützle (2004), Venables and Moscardini (2006). Dorigo and Blum (2005) state that the use of heuristic

Algorithm 5.3: Outline for Total Opportunity Method (TOM) of Kirca and Satir (1990)

```
Define:
    $C_{ij} =$ unit transportation cost from supply point $i$ to customer demand point $j$,
    $S_i =$ capacity of supply point $i$,
    $D_j =$ customer demand at demand point $j$,
    $E_{ij} =$ supply opportunity cost,
    $F_{ij} =$ demand opportunity cost,
    $T_{ij} =$ total opportunity cost,
    $X_{ij} =$ amount of customer demand allocated from $j$ to supply point $i$.

Compute:
    $E_{ij} = C_{ij} - C_{ij^*}$,      $i = 1, \dots, m$ and $j^* = argmin_{j \in J} \{C_{ij}\}$;
    $F_{ij} = C_{ij} - C_{i^*j}$,      $j = 1, \dots, n$ and $i^* = argmin_{i \in I} \{C_{ij}\}$;
    $T_{ij} = E_{ij} + F_{ij}$,        $i = 1, \dots, m$ and $j = 1, \dots, n$;

while spare demand do
    Allocate the maximum possible demand units $X_{ij}$ to that cell to the (next)
    smallest $T_{ij}$. In the case of a tie use the following tie breakers:
        Step 1. Make the allocations to cell $ij$ with the smallest $C_{ij}$
        Step 2. in the case of a tie in Step 1, make the allocation to cell $ij$ with the
        largest possible demand $X_{ij}$
        Step 3. in the case of a tie in Step 1 and Step 2, make the allocation to cell
        $ij$ with the smallest allocation cost
    Remove $i$ or $j$, where supply $S_i$ is depleted or demand $D_j$ is fully satisfied;
end
```

information is optional. They recommended that, for efficient computational performance, static versions are more preferable to dynamic versions, as the values can be computed prior to the main ACO algorithm. Dorigo and Stützle (2004) reported that the importance of using *a priori* instance information becomes less significant with the use of embedded local search strategies for small-scale problems, but it still has a role to play for large-scale problems. Their definition of small and large-scale is somewhat unclear, but relates to empirical observations relating to various instances solved for the travelling salesman problem. This section provides a novel approach to the application of heuristic information, by using a relaxation of the CFLP to derive heuristic information.

Relaxation techniques can be used to derive lower bounds for the CFLP and are usually based on sophisticated mathematical programming methods such as those used in Lagrangean relaxation (Agar and Salhi, 1998, Beasley, 1993). Less sophisticated methods based on linear relaxation can also be applied to the CFLP and lend themselves to theoretical development, but are prone to be computationally unreliable for anything other than small-scale instances, having only a handful of facilities (Baker, 1982, Sa, 1969). However, heuristic information is only concerned with the *likelihood of facilities being in an optimal solution and not whether they have to occur.* Thus, it is possible to use linear relaxation to help identify some of those facilities that may be in an optimum solution.

One way of applying linear relaxation to the CFLP is to relax the facility integral constraint given in equation (4.5), in a similar way to that used by Adlakha and Kowlaski (2004) for the source induced fixed-charge transportation problem:

$$y_i = \sum_{j \in J} x_{ij}/m_{ij}, \qquad \forall\, i \in I; \tag{5.1}$$

where

$$m_{ij} = min(q_j, Q_i), \qquad \forall\, i \in I \quad \wedge \quad \forall\, j \in J. \tag{5.2}$$

This relaxation results in an unbalanced transportation problem with unit costs:

$$C_{ij} = c_{ij} + f_i/m_{ij}. \tag{5.3}$$

It is worth noting that if an individuals entire demand may be supplied by a single facility, then the unit costs within the transportation problem become $C_{ij} = c_{ij} + f_i/q_j$, which is computationally more efficient to obtain. Another, way of saving computational effort is to consider the notation that those facilities most likely to be in an optimum solution will have the lowest T_{ij} values (Venables and Moscardini, 2006). This approach has the advantage of not having to completely approximate the transportation problem associated with the linear relaxation. Ant visibility for each facility i can then be defined as:

$$\eta_i = \frac{1}{T_i} \qquad \forall\, i \in I; \tag{5.4}$$

where $T_i = \sum_{j \in J} T_{ij} \qquad \forall\, i \in I$ is the total opportunity cost for facility i and T_{ij} is defined in Algorithm 5.3 (see also Kirca and Satir (1990)).

5.5 Hybridisation of $\mathcal{MM}$AS

The process of making a move on a construction graph to select a facility to include into the current solution is based on the pseudo-random proportional rule, in the same way as described in the previous chapter. Each move is defined as follows: with some probability q_0 select the facility with the greatest combined pheromone and ant visibility value, otherwise select a facility using a selective probability function. The parameter q_0 predetermines the level of exploitation of the search space, whereas exploration of the search space is chosen with probability $1 - q_0$.

A move on the construction graph to select a new facility i to potentially include in a current solution is:

$$i = \begin{cases} \text{argmax}_{l \in L} \left\{ [\tau_l]^\alpha \, [\eta_l]^\beta \right\}, & \text{if } q \leq q_0, \\ \text{I}, & \text{otherwise.} \end{cases} \tag{5.5}$$

where q is a random variable uniformly distributed in $[0, 1]$, q_0 is a predefined parameter where $(0 \leq q_0 \leq 1)$, L is the set of unopened facilities, and I is a random variable that is selected according to the following probability distribution:

$$p_i = \frac{[\tau_i]^\alpha \, [\eta_i]^\beta}{\sum_{l \in L} [\tau_l]^\alpha \, [\eta_l]^\beta}, \tag{5.6}$$

where α and β are parameters corresponding to the influential roles of pheromone intensity τ_i and ant visibility $\eta_i = 1/T_i$, whilst the set of unvisited facilities to be considered is L. Once a potential facility is selected then the corresponding transportation problem is approximated, as described in section 5.3, and the facility is added to the current solution if an overall cost improvement is observed. After the status of all potential facilities have been determined then the current ant tour is complete and a solution improvement or a local search phase, see section 5.6, is entered followed by a pheromone update phase:

$$\tau_i \leftarrow (1 - \rho)\tau_i \qquad \forall \, i \in I. \tag{5.7}$$

Pheromones are deposited on those facilities belonging to the best tour to-date:

$$\tau_i \leftarrow \tau_i + \Delta\tau_i^{\textit{best}} \qquad \forall \, i \in I \,, \tag{5.8}$$

where $\Delta\tau_i^{\textit{best}} = 1/z^{\textit{best}}$ and $z^{\textit{best}}$ is the overall cost of the best tour. Upper and lower limits τ_{max} and τ_{min} are placed on the pheromones in an attempt to avoid convergence to a local optimum. These are set as $\tau_{max} = 1/\rho z^{best}$ and $\tau_{min} = \tau_{max}/a$ where a is a parameter $(a > 1)$. Also, τ_{max} is updated whenever an improvement is made in the best overall cost z^{best}. If the procedure begins to converge to potentially a local optimum, or there is no improvement in the overall cost after a chosen number of iterations, then the pheromones are reset to the current value of τ_{max}. This is an attempt to encourage a new exploratory search away from the region of stagnation Stützle (1999), Stützle and Hoos (2000). Algorithmic stagnation occurs when the pheromone levels approach their upper and lower limits, τ_{max} and τ_{min}. A method to test if stagnation occurs is implemented, which is based upon one used by Dorigo and Stützle (2004):

$$\frac{\sum_{\tau_i \in T} min\{\tau_{max} - \tau_i, \tau_i - \tau_{min}\}}{m} \rightarrow 0, \tag{5.9}$$

as the algorithm approaches stagnation, where T are the pheromones for the current tour and m is the number of facilities. Alternatively, a maximum deviation measure could be used such as $Max_{\tau_i \in T}\ min\{\tau_{max} - \tau_i, \tau_i - \tau_{min}\}$ which would also tend towards zero as stagnation occurs. These stagnation conditions also hold true at an optimum solution. Consequently, it would be unwise to use these tests as a potential termination conditions for an ACO algorithm, as the solution obtained may not be a global optimal solution.

5.6 Local Search Methods for Solution Improvement

During the construction phase some facilities that are fixed open early on may later only play a minor role in accommodating customer demand. Thus, improvements may be made locally by closing one or more facilities in the current solution. A *DROP* heuristic is presented, which uses a best-improvement strategy. Further improvements may be obtained by swapping open facilities with closed ones using a *SWAP* heuristic in a similar manner to those used in Lagrangean techniques Agar and Salhi (1998), Beasley (1993). Consequently, a two-stage improvement method consisting of two local search procedures, *DROP* and *SWAP*, is also described. The combined *DROP-SWAP* method employs a first-improvement technique that initially relies on current pheromone intensities at each facility to help identify those most likely candidates. Existing research techniques usually employ local search strategies based on cost/structure neighbourhoods (Dorigo and Stützle, 2004, Hoos and Stützle, 2005, JI et al., 2009, Lorena and Senne, 2003, Pang et al., 2009, Stützle, 1999, Stützle and Hoos, 1997, Xu et al., 2006). Whereas using the internal stigmergy learning mechanism to help define local search regions is relatively novel (Venables and Moscardini, 2008). A disadvantage of using local search procedures for the CFLP, that consist of dropping and swapping facilities is that they require solutions to many transportation problems and are often described as being computationally too expensive (Agar and Salhi, 1998, Beasley, 1993, Bornstein and Campelo, 2004, Daskin, 1995, 2008). As a compromise, any local search strategy shall only be applied to the best-ant solution (least-cost) at each iteration.

5.6.1 Drop Facilities

Improvements are sought after by closing one or more facilities in the current solution. It is necessary to close a facility that gives the best solution improvement. The total cost of a current solution is the sum of the fixed costs of the opened facilities and the corresponding TP solution costs. Thus, if the current solution has facilities $Y = \{y_i | y_i \in \{0, 1\}\}$ with associated fixed and transportation costs z, then select an open facility $\{i^* | y_{i^*} = 1\}$ that gives the least total cost and then reset $y_{i^*} = 0$ and Y accordingly. The process is repeated until no further improvements can be made. This method was applied used with some limited success

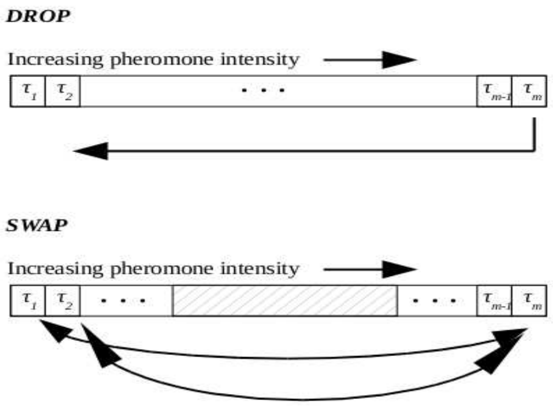


Figure 5.3: Schematic for *DROP* and *SWAP* local search mechanisms

by Venables and Moscardini (2006).

Alternatively, a first-improvement procedure that when coupled with pheromone intensity ought to provide a more efficient local search mechanism. Firstly, facilities need to be sorted into increasing order of pheromone intensity; see Figure 5.3. Then, starting with the highest pheromone intensity, facilities are sequentially closed and tested for any overall cost improvements; if an improvement occurs then that facility is closed and the current solution is updated, otherwise it remains open. The advantage of this is that once the open facilities are sorted, each facility is only considered once during the process whereas a best-improvement method requires repeated searches over the set of open facilities. Its disadvantage is that a solution improvement may not be as good as one obtained using a computationally more expensive best improvement method. Although the proposed technique may be more efficient, the overall ACO procedure may require a larger number of iterations to obtain good solutions; as shown by the experimental results given in Table 5.2.

5.6.2 Swap Facilities

In an effort to improve the *DROP* solutions a *SWAP* heuristic in a similar manner to those used in Lagrangean relaxation is implemented. Both Beasley (1993) and Agar and Salhi (1998) used an interchange or *SWAP* method that used a measure derived from the Lagrangean relaxation, which indicated whether a facility belonged to a current solution. Both considered restricting the number of interchange candidates in the closed set of facilities for a known open facility. The method presented in this section has a similar structure to that of Agar and Salhi (1998), Beasley (1993). Initially the current iterative solution is sorted into sets of opened, F, and closed facilities, $\bar{F}$, based on increasing pheromone intensity; $F = \{i|y_i = 1\}$ and $\bar{F} = \{i|y_i = 0\}$. Also, the number of candidates in both sets are restricted. *SWAP*-candidates are selected by their pheromone levels such that those opened facilities with low intensities are considered for swapping with closed facilities having high levels. The idea is to encourage the interchange of opened facilities with ones that were previously overlooked. Since the size of the complete local search space includes all of the facilities $|F \cup \bar{F}| = m$, then the complexity of the local search grows exponentially with m and is referred to as a very-large scale neighbourhood (Ahuja et al., 2002). To overcome computational inefficiency the candidate search space is restricted in size by $max(15, 0.1|S|)$ where $|S|$ is the size of the set of opened or closed facilities being considered, Agar and Salhi (1998) used the same technique. Also a first-improvement local search policy is adopted, that seeks out the first-swap which gives a solution improvement for an opened candidate facility; see Figure 5.3. The technique is then repeated for all remaining open candidates. Experimental results conducted on a small number of instances are given in Table 5.2, suggest that the sequential application of *DROP-SWAP* is worth pursuing.

5.7 Hybrid $\mathcal{MM}$AS: Initial Experimentation

This section presents computational experiments and results obtained for a variety of approximate hybrid $\mathcal{MM}$AS algorithms, that are applied to the same capacitated location problems as described in section 4.4. The algorithms were coded in C++ and experiments were carried out on the same Dell Inspiron 8600 with a 1.60 GHz Pentium M processor and 786Mb RAM as previously used. The number of experiments were limited to five per problem instance as used by Dorigo and Stützle (2004), Lourenço and Serra (2002) and the median run-time (secs), number of iterations (iters) and relative errors (% err) were recorded. All transportation sub-problems were approximated using the TOM method of Kirca and Satir (1990). At this stage of development it was felt unnecessary to optimally solve the final solution's TP, but that it ought not to be ignored for later comparisons with literature results. ACO parameters were set according to experimental guidance given in Dorigo and Stützle (2004) and the following setting were found to be robust to small changes: $\alpha = 2.5,\ \beta = 0.8,\ \rho = 0.06,\ q_0 = 0.5$ and $a = 2n$, where n is the number of customers in the problem instance being solved. The number of iterations in each experiment was limited to two hundred as testing displayed little significant change in the best solution beyond this value. To encourage exploration of the solution space, pheromones were reset to the current value of τ_{max} if there was no improvement in the best solution after fifty iterations.

5.7.1 $\mathcal{MM}$AS and Basic *DROP*

The results of the first series of experiments are given in Table 5.1. The last three rows show three basic statistical descriptors for run-times and relative errors across the set of instances (average, standard deviation and coefficient of variation). Two series of data are shown: one for the performance of the basic

$\mathcal{MM}$AS algorithm, whilst the second combines $\mathcal{MM}$AS with a best-improvement *DROP* procedure that ignores pheromone intensity.

The descriptive statistics given in Table 5.1 supports the use of the *DROP* improvement strategy, Z_D, as the relative error's coefficient of variation of 1.25 is 36% lower than that 1.94 of the basic $\mathcal{MM}$AS method, Z_{NLS}. However, there is a trade-off with the computational run-time coefficient of variation, as this experiences a 23% increase. Upon closer inspection out of the 31 problems considered, solutions were generated for 23 in a shorter time where 26 required fewer iterations and 19 benefited from improved objective values. This is further reinforced by lower errors from known optimal values being obtained. Without *DROP*, the average error is 5.83% with a standard deviation of 11.33%, whereas corresponding values using the *DROP* strategy are 3.16% and 3.94% respectively, which suggests significant improvements are made with greater reliability. Although not shown in Table 5.1, all the instance trials converged to their best solution even when there was evidence of algorithmic stagnation and the pheromones had to be reset. Solutions to 52% of the problems were obtained with an error of less than 1.0% indicating some worthy merit in the *DROP* $\mathcal{MM}$AS algorithm. These results are summarised in the published research of Venables and Moscardini (2006).

5.7.2 Pheromone Based *DROP* and *DROP-SWAP*

A second series of experiments were carried out to determine if the pre-ordering of pheromone intensities combined with proposed first-improvement local search methods, gave any performance improvements. Results of *DROP* and *DROP-SWAP* compared to *DROP* procedure given in Table 5.1. To ensure a fair comparison, the ACO parameters and the number of trials conducted per problem instance remained unchanged. However, the experiments were restricted to two

				Z_{NLS}			Z_D	
Problem	m×n	Z^*	secs	iters	% err	secs	iters	% err
cap51	16×50	1025208.225	5.217	71	60.01	0.631	2	12.45
cap61	16×50	932615.75	3.024	35	3.38	2.443	12	3.29
cap62		977799.4	4.517	55	3.12	2.313	9	3.12
cap63		1014062.05	5.38	71	3.42	1.282	8	2.89
cap64		1045650.25	2.694	35	4.03	8.001	62	4.03
cap71	16×50	932615.75	12.588	135	1.29	0.351	1	1.29
cap72		977799.4	1.19	13	0.86	0.59	3	0.86
cap73		1010641.45	0.551	6	0.45	0.29	1	0.45
cap74		1034976.975	0.431	5	0.39	0.421	1	0.39
cap81	25×50	838499.288	4.777	30	15.34	4.267	5	11.48
cap82		910889.563	4.907	31	15.04	3.605	4	9.75
cap83		975889.563	4.947	32	14.33	5.88	9	9.09
cap91	25×50	796648.438	10.886	54	0.84	45.225	107	0.20
cap92		855733.5	3.124	16	0.62	2.433	4	0.62
cap93		896617.538	3.375	18	1.61	1.452	2	1.15
cap101	25×50	896617.538	18.277	86	0.29	26.238	62	0.20
cap102		854704.2	4.236	21	0.38	0.671	1	0.38
cap103		893782.112	29.162	166	0.18	1.212	2	0.18
cap104		928941.75	8.803	52	0.10	2.544	10	0.10
cap111	50×50	826124.713	52.826	79	11.89	100.385	69	8.26
cap112		901377.213	37.774	57	12.28	20.089	7	7.23
cap113		970567.75	124.95	195	11.62	67.317	56	8.94
cap114		1063356.488	29.923	46	11.99	13.97	5	7.61
cap121	50×50	793439.563	73.556	110	0.61	61.137	54	0.21
cap122		852524.625	23.74	36	0.91	21.301	17	0.65
cap123		895302.325	18.207	29	2.32	88.537	101	1.15
cap124		946051.325	93.705	158	1.63	120.694	160	0.92
cap131	50×50	793439.562	61.368	89	0.61	129.216	115	0.33
cap132		851495.325	61.218	95	0.41	11.877	10	0.41
cap133		893076.712	57.763	94	0.70	2.593	2	0.26
cap134		928941.75	89.74	160	0.10	55.149	82	0.10
		Average	27.51		5.83	25.87		3.16
		STD	33.16		11.33	38.22		3.94
		COV	1.21		**1.94**	1.48		**1.25**

Table 5.1: Non-local search (Z_{NLS}) and facility "DROP" local search (Z_D) results for problem instances using a best-improvement technique and approximate transportation solutions; where Z^* are the known optimum solutions (Beasley, 1993)

sets of instances based on the previous results. In particular the two sets of instances were selected based on their relative errors; one displayed poor performance (Cap81-83), whilst the second displayed good performance (Cap131-134). Computational results for the selected seven instances are given in Table 5.2.

The coefficients of variation for the second set of results again support the use of local search, as there are significant improvements in the computational run-times when pheromone based local search procedures are implemented. The run-time COV for the pheromone based *DROP* local search procedure, Z_{D_τ}, indicates a reduction by 38%. Similarly, improvements solution accuracy were observed across all but one of the instances to give a COV reduction of 36%. Although the number of iterations required to find the best solutions increased and the run-times increased across most of the seven instances, the standard deviation was approximately half of that obtained for the basic *DROP* procedure, Z_D. This implies that the pheromone based local search is a more reliable method.

The results obtained for the *DROP-SWAP* method, Z_{S_τ}, show that the best solutions found for the seven instances, are the same as those found using the pheromone based *DROP* method. However, upon inspection of the instance run-times and number of iterations required to obtain the same solutions, they are far superior to those solely using a pheromone based *DROP* local search strategy. Thus due to overwhelming improvements made, the $\mathcal{MM}$AS with *DROP-SWAP* shall provide the backbone for subsequent development.

5.8 Hybrid $\mathcal{MM}$AS: Initial Evaluation

Thus far, experimentation has only considered algorithmic development that employs approximate solutions to any transportation problems that arise during dur-

			Z_D			Z_{D_τ}			Z_{S_τ}	
Problem	m×n	secs	iters	% err	secs	iters	% err	secs	iters	% err
cap81	25×50	4.27	5	11.48	6.39	20	0.51	2.65	2	0.51
cap82		3.61	4	9.75	38.50	148	0.80	2.67	3	0.80
cap83		5.88	9	9.09	5.34	21	0.92	2.71	2	0.92
cap131	50×50	129.22	115	0.33	60.38	121	0.21	5.71	2	0.21
cap132		11.88	10	0.41	47.22	124	0.28	3.58	2	0.28
cap133		2.59	2	0.26	8.76	34	0.17	1.98	2	0.17
cap134		55.15	82	0.10	4.77	22	0.10	1.08	3	0.10
	Average	30.37		4.49	24.48		0.43	2.91		0.43
	STD	43.91		4.91	21.82		0.30	1.35		0.30
	COV	1.45		1.09	0.89		0.70	0.46		0.70

Table 5.2: Results for a selection of problem instances using local search: Z_D method of Venables and Moscardini (2006) and pheromone based *DROP* and *DROP-SWAP* heuristics Z_{D_τ} and Z_{S_τ}

ing run-time. Often to solve a problem instance there may be many of these sub-problems that also need solving. Consequently, approximate sub-problem solution are sought after and different approximation techniques will have some bearing on computational run-time. Even so, approximate techniques can lead to optimal or near optimal solutions (within an acceptable degree of accuracy). A technique that is often employed, is to find the optimal solution corresponding to the final approximate solution upon completion of the algorithmic search. Thus the set of facilities belonging to the best solution found, define a transportation problem that is solved exactly by using a specific algorithm or computer software. This stage may be included in the computer program used to implement the method being tested or treated as a separate external problem.

At this stage in the development a decision was taken to solve the transportation problem defined by the best solution found, as used by Beasley (1993) and Agar and Salhi (1998), using a linear programming tool available in MATLAB. The results are presented in the final two columns of Table 5.3, where the final two rows display the relative errors from the known optimums and the time taken to find the best approximate solution using *DROP-SWAP*. The literature results given in the table are for those problems that were published by their respective authors, hence the missing data. Also, the results of Bornstein and Campelo (2004) were obtained using an exact transportation problem solver for all sub-problems encountered.

The mean and standard deviation values of the $\mathcal{MM}$AS Z_{S_τ} algorithm given in Table 5.3 are significantly lower than those corresponding values for $\mathcal{MM}$AS Z_D in Table 5.1. This reinforces earlier observations that *DROP-SWAP* is the superior ACO algorithm and ought to provide a basis for further development. However, the algorithm does not fair too well with those Lagrangean based algorithms of Beasley (1993) and Bornstein and Campelo (2004). Interestingly in terms of solution accuracy, $\mathcal{MM}$AS Z_{S_τ} outperforms the Simulated Annealing algorithm of Bornstein and Azlan (1998), that used a similar sub-problem approximation approach. Their exact version of the algorithm displayed that Simulated Annealing was able to find more optimum solutions than the developed ACO algorithm, yet the pairs of relative error statistics were very similar.

The $\mathcal{MM}$AS Z_{S_τ} algorithm managed to find 32.4% of the optimal solutions, whereas 89.2% of those solution found had a relative error of less than 0.5%. These results are very encouraging and indicate that an appropriately constructed hybrid ACO algorithm is capable of deriving optimal or near optimal solutions for the CFLP. Although these results are very supportive, it is too early to make any firm conclusions about overall behaviour. ACO is stochastic and any statistics derived by observation may change significantly with a different set of experimental trials. The next stage in the developmental process is to determine the effects on solution performance when using an exact transportation problem solver within the hybrid algorithm, in a similar manner to that of the exact Simulated Annealing algorithm developed by Bornstein and Azlan (1998).

		B93	BA98^1	BA98^2	BC04	$\mathcal{MM}$AS Z_{S_τ}	
Problem	m×n	% err	% err	% err	% err	% err	secs
cap41	16×50	0.00	0.00	0.00	0.00	0.00	0.40
cap42		0.00	0.00	0.00	0.00	0.22	0.39
cap43		0.00	0.00	0.00	0.00	0.64	0.76
cap44		0.00	0.00	0.00	0.00	0.00	0.37
cap51	16×50	0.00	0.13	0.00	0.00	0.00	0.70
cap61	16×50	0.00	0.00	0.00	0.00	1.07	0.47
cap62		0.00	0.00	0.00	0.00	0.51	0.75
cap63		0.00	0.00	0.05	0.00	0.00	0.86
cap64		0.00	0.72	0.00	0.00	0.00	0.27
cap71	16×50					0.76	0.48
cap72						0.22	0.40
cap73						0.00	0.47
cap74						0.26	0.32
cap81	25×50	0.00	0.56	0.00	0.00	0.46	2.65
cap82		0.00	1.81	0.00	0.00	0.30	2.67
cap83		0.00	2.00	0.51	0.00	0.00	2.71
cap84		0.00	1.93	0.50	0.38	0.00	1.26
cap91	25×50	0.00	0.10	0.00	0.00	0.34	1.54
cap92		0.00	0.06	0.09	0.00	0.07	2.42
cap93		0.00	1.95	0.00	0.00	0.24	0.98
cap94		0.06	0.37	0.25	0.00	0.23	0.96
cap101	25×50					0.00	2.73
cap102						0.15	1.38
cap103						0.14	0.24
cap104						0.06	1.23
cap111	50×50	0.00	0.71	0.07	0.00	0.00	5.97
cap112		0.00	0.06	0.08	0.38	0.08	8.01
cap113		0.00	2.42	0.85	0.22	0.06	7.81
cap114		0.44	1.80	0.67	0.00	0.35	9.56
cap121	50×50	0.00	0.88	0.00	0.00	0.00	8.33
cap122		0.00	0.56	0.00	0.00	0.01	4.98
cap123		0.00	1.25	0.12	0.00	0.03	3.54
cap124		0.17	0.62	0.06	0.00	0.17	3.58
cap131	50×50					0.00	5.71
cap132						0.02	3.58
cap133						0.12	1.98
cap134						0.06	1.08
Average		0.03	0.72	0.13	0.04	0.18	2.47
STD		0.09	0.81	0.24	0.11	0.24	2.60

Table 5.3: Available literature results of B93 - Beasley (1993), BA98^1 and BA98^2 - Bornstein and Azlan (1998), BC04 - Bornstein and Campelo (2004) and the approximate hybrid $\mathcal{MM}$AS ACO algorithm - $\mathcal{MM}$AS Z_{S_τ}

5.9 Hybrid $\mathcal{MM}$AS: An Alternative Approach

All of the algorithmic development, described in this chapter, has made use of obtaining approximate solutions to any TPs encountered during the run-time of a problem instance. Rationale for an approximation approach, is that during run-time many-many TPs need to be solved, which is common place with Lagrangean based heuristics and various metaheuristic solution based methods that are reported upon within academic research literature, (Agar and Salhi, 1998, Al-khedhairi, 2008, Alp et al., 2004, Arostegui et al., 2006, Barahona and Chudak, 2005, Beasley, 1993, Bornstein and Azlan, 1998, Bornstein and Campelo, 2004, Correa et al., 2004, Daskin and Melkote, 2001, Ghoseiri and Ghannadpour, 2009, Jaramillo et al., 2002, Levanova and Loresh, 2004, 2006, Lorena and Senne, 2003, Michel and Hentenryck, 2004, Sridharan, 1995). To emphasise this point consider the hybrid $\mathcal{MM}$AS that uses a pheromone based drop-swap local improvement method where two phases predominantly require the solution to TPs: a) the construction phase needs to evaluate a TP at every made step along the construction graph and b) the local search method needs to evaluate a TP initially at every step of the facility drop phase and then at every step of the facilities swap phase until no solution further improvements are observed. Consequently, the larger the number of facilities in a problem then the number of TPs needed to be evaluated grows rapidly and thus an approximate TP solution technique is generally adopted. Research published by Bornstein and Azlan (1998) effectively demonstrates computational run-time issues when using an exact TP solver; most but not all of the test instances were solved yet their run-times were excessive in comparison to those using an approximate TP method (on average $\approx$ 230 times greater).

Recently, a Cross-Entropy method (Rubinstein, 2002, Rubinstein and Krose, 2004) for the CFLP was proposed by Caserta and Quiñonez Rico (2007, 2009) which claimed to solve all of the test problems in the OR-library effectively and efficiently. Their method used an exact TP solver that was implemented via a callable routine from the open-source COIN-OR distribution (Lougee-Heimer, 2003). The routine solves the TP by using dual simplex method that is based on the minimum cost-flow algorithm of Goldberg (1997). Although no experimentation details were given, they stated that all optimal solutions to the basic 37 test instances were found in less than 2 seconds, whilst the 12 larger problems were also solved optimally within 2 minutes.

Consequently, it is worthwhile investigating the performance of the *DROP-SWAP* $\mathcal{MM}$AS algorithm across the 37 test instances by replacing the approximate TP solver with the COIN-OR exact solver. This would allow for experimentation to be conducted on the 37 test instances, to identify if any improvements in computational run-times and relative errors are possible. Table 5.4 displays a series of experiments carried out to determine the potential of using an exact TP solver in place of an approximation method. As with previous experimentation five trials were conducted on each instance and the median run-time solution was recorded. A series of three experiments were carried out on the 37 basic test instances using a single ant as described in section 5.8 and are displayed under the Table 5.4 column heading $\mathcal{MM}$AS Single Ant. The sub-columns labelled A, B and C refer to various ACO parameter settings: A - has the same settings as defined in section 5.7; B - the Cross-Entropy method uses a smoothing factor which is equivalent to an ACO pheromone decay rate and was set to $\rho = 0.9$ as used by Caserta and Quiñonez Rico (2007, 2009); C - attempts to assess the effects of a simple pheromone model that ignores the iterative use of ant visibility whilst coupled with a rapid pheromone decay rate as in B by setting $\alpha = 1.0, \beta = 0.0$ and $\rho = 0.9$.

To help facilitate the use of the COIN-OR distribution it was necessary to use a LINUX based system: the algorithms were coded in C++ and experiments were executed on a Pentium 4 3.0GHz Linux PC with 2Gb of RAM. Source codes were compiled using the GNU g++ compiler using the -O option. Furthermore observations made from the results of Z_{S_τ} given in Table 5.2, suggest that if optimal or near optimal solutions can be derived in a small number of iterations then the maximum number of iterations allowed during run-time can be set to a small number. All of the instances that were previously tested found their best solution in less than 10 iterations, so a new setting of a maximum of 20 iterations would not be unreasonable. Also, because of the small number of iterations allowed it would not be beneficial to have an algorithmic stagnation reset criteria as the intention is to find the best solution over 20 iterations.

The results for $\mathcal{MM}$AS Single Ant algorithm given in Table 5.4 indicate that the majority of the five trials derived optimal solutions. Those results shown with an asterisk (*), are where only one of the five trials failed to find the optimal solution; in each of these cases the relative error was less than 0.1%. These results are remarkable and appear to contradict previous academic research policy of using an approximate TP solver. When the means and standard deviations of the three experiments are compared then the parameter setting B comes out on top. However, the parameter settings of A was the only scheme that solved all of the instances for each of the five trials.

Interestingly, the parameter setting scheme C ($\alpha = 1.0, \beta = 0.0$ and $\rho = 0.9$), which ignores the compounded influence of heuristic information and has a high pheromone evaporation rate gives some very encouraging results. A rationale for its success is that as the evaporation rate is high, then those facilities not in the current best solution will have low pheromone levels. So, the *DROP-SWAP* procedure then actively encourages those ignored facilities with low pheromones to be tested for inclusion into a solution and thus will have a better chance of improving the objective function. By setting $\beta = 0.0$, heuristic information or ant visibility is not completely lost, as it is used to help define an initial solution from which future solutions are derived. Consequently, facilities belonging to the initial best ant will have some pheromone deposited on them, which will contain some facilities constantly belonging to improving solutions and thus experience pheromone reinforcement during the iterative procedure.

The success of using a single ant approach was very encouraging. So, the next stage is to consider the effects of using a colony of ants in sequential manner as discussed in the text by Dorigo and Stützle (2004). Their experiments on the travelling salesman and quadratic assignment problems (TSP and QAP) concluded that with a $\mathcal{MM}$AS algorithmic approach which uses an embedded local search need only use a small colony size of less than ten ants, as there was a trade-off between solution quality and computation time which worsened as the number of ants grew. ACO procedures used on the TSP and QAP were based on adding path-links and costs which did not require the need to solve any sub-problems at each step in the constructive phase, unlike the more complex structure associated with the CFLP.

Also, increasing the colony size needs more ants per iteration and would require more computation effort. So a series of experiments were carried out using a colony of five ants and as with the single ant experiments the three parameter strategies A, B and C were investigated. The results of which are shown under the column heading $\mathcal{MM}$AS 5 Ants in Table 5.4. Trials that did not find optimum solutions gave errors of less than 0.1%. Parameter setting C was the most reliable as all the optimal solutions were found. Although, parameter setting B failed to find the optimal solution for all five trails of a particular instance (Cap113), the run-times for the remaining 36 instances were the most reliable of the three parameter settings. What is evident is that the COVs for the small colony are lower

		$\mathcal{MM}$AS Single Ant						$\mathcal{MM}$AS 5 Ants					
		A		B		C		A		B		C	
Problem	m×n	secs	its	secs	its	secs	its	secs	its	secs	its	secs	its
cap41	16×50	0.09	1	0.09	1	0.09	1	0.12	1	0.12	1	0.12	1
cap42		0.09	1	0.10	1	0.09	1	0.13	1	0.12	1	0.12	1
cap43		0.10	1	0.10	1	0.09	1	0.12	1	0.12	1	0.12	1
cap44		0.10	1	0.10	1	0.09	1	0.13	1	0.13	1	0.11	1
cap51	16×50	0.15	2	0.15	2	0.08	1	0.35	3	0.77	7	0.12	1
cap61	16×50	0.06	1	0.07	1	0.06	1	0.10	1	0.11	1	0.10	1
cap62		0.20	3	0.12	2	0.06	1	0.21	3	0.21	2	0.11	1
cap63*		0.12	2	0.19*	2	0.42	7	0.21	2	0.10	1	0.11	1
cap64		0.06	1	0.06	1	0.06	1	0.09	1	0.10	1	0.10	2
cap71	16×50	0.06	1	0.06	1	0.06	1	0.10	1	0.10	1	0.10	1
cap72		0.13	2	0.06	1	0.06	1	0.10	1	0.10	1	0.11	1
cap73		0.09	2	0.06	1	0.56	13	0.26	3	0.73	9	0.58	7
cap74		0.07	2	0.04	1	0.04	1	0.08	1	0.08	1	0.08	1
cap81	25×50	0.39	2	0.40	2	0.38	2	0.27	1	0.27	1	0.52	2
cap82		1.10	5	0.44	2	0.43	2	1.21	4	0.58	2	0.56	2
cap83		1.16	5	0.68	3	0.43	2	1.24	4	0.88	3	0.56	2
cap84		1.47	6	0.47	2	0.69	3	1.55	5	0.60	2	0.91	3
cap91	25×50	0.33	3	0.33	2	0.32	2	0.48	2	0.48	2	0.25	1
cap92		0.57	4	0.29	2	0.44	3	0.44	2	0.44	2	0.44	2
cap93		0.26	2	0.37	3	0.61	5	0.79	4	0.59	3	0.22	1
cap94		0.91	8	0.91	8	0.24	2	0.72	4	1.27	7	0.39	2
cap101	25×50	0.66	4	0.32	2	1.81	11	0.71	3	0.49	2	1.68	7
cap102		0.55	4	0.28	2	0.28	2	0.42	2	0.42	2	0.43	2
cap103		0.42	4	0.80	8	1.17	11	0.86	5	0.36	2	0.20	1
cap104		0.12	2	0.07	1	0.07	1	0.13	1	0.13	1	0.14	1
cap111*	50×50	5.87	9	5.18*	8	7.02	11	5.00	5	1.69	2	1.75	2
cap112		7.07	11	1.31	2	1.96	5	5.01	6	1.70	2	1.74	2
cap113*		7.58	12	2.06*	3	2.57*	3	8.12*	5	*		1.72	2
cap114		6.70	10	1.99	3	3.22	5	6.99	8	2.48	3	2.49	3
cap121	50×50	1.95	4	0.99	2	3.83	8	3.33	5	1.30	2	1.33	2
cap122		3.82	11	0.95	3	1.05	3	2.91	6	1.48	3	1.46	3
cap123		3.67	12	0.89	3	0.90	3	3.67	8	1.39	3	1.31	3
cap124		1.80	8	2.14	9	3.02	12	1.66	4	1.51	4	1.23	3
cap131	50×50	0.93	2	0.99	2	0.95	2	1.99	3	1.26	2	1.31	2
cap132		2.27	7	0.93	3	0.95	3	2.04	4	0.97	2	1.08	2
cap133		1.15	5	0.46	2	2.60	11	1.94	5	0.79	2	0.80	2
cap134		0.30	2	0.37	3	0.25	2	0.54	2	0.30	1	0.79	3
Mean		1.42		0.67		1.00		1.46		0.70		0.70	
St. Dev.		2.10		0.94		1.42		1.96		0.61		0.68	
COV		1.48		1.40		1.42		1.34		0.87		0.97	

Table 5.4: OR-Library test problems using various $\mathcal{MM}$AS parameter settings: A – $\alpha = 2.5$, $\beta = 0.8$ and $\rho = 0.06$; B – $\alpha = 2.5$, $\beta = 0.8$ and $\rho = 0.9$; C – $\alpha = 1.0, \beta = 0.0$ and $\rho = 0.9$; secs – seconds; its – iterations

than those of the single ant method, which statistically suggests that using a small colony of ants is more favourable.

5.9.1 Larger OR-Library Instances

Observations made from statistical information for the basic test instances using an exact TP solver, shows that the proposed method works very well at finding solutions that match known optimums. A follow-on from this is to determine if the $\mathcal{MM}$AS algorithm is capable of finding optimal solutions to the 12 larger test instances from the OR-Library; each having 100 potential facilities and 1000 customers. Experiments were carried under the same conditions: obtain the best solution in 20 iterations, using the best parameter settings from the previous experiments conducted with a single ant (A) and a colony of five ants (B). To avoid excessive computational run-time issues the number of experimental trials was restricted to a maximum of five and terminated if a solution matched the known optimal solution.

Terminating an algorithm based upon knowing the optimum solution *a priori* is a contentious issue because, the optimal or best solutions are not known in advance for real problems. The rationale for using this as a stopping condition for experimental purposes was a pragmatic decision that was based on the elitist bias of pheromone laying in the ACO algorithms being tested. When pheromone depositation takes place the intensity of the pheromone levels becomes greater on the current best solution unless an improvement is found, if no improvements are found then stagnation about this solution is likely to occur. Thus, when attempting to solve a problem instance where the optimal is known *a priori* and a solution is found that matches this value, then emergent behaviour of stagnation will be evident. Knowing the optimal solution in advance can help to save on experimental time by terminating the algorithm when a matching solution is observed.

		$\mathcal{MM}$AS Single Ant				$\mathcal{MM}$AS Colony			
		A				B			
Problem	m × n	secs	its	% err	trial	secs	its	% err	trial
A-1	100 × 1000	283.66	13	0.01	1	148.11	4	0.00	1
A-2	100 × 1000	370.42	13	0.00	4	58.21	2	0.00	4
A-3	100 × 1000	358.65	14	0.00	1	66.35	2	0.00	1
A-4	100 × 1000	26.48	1	0.00	1	121.57	5	0.00	1
B-1	100 × 1000	931.50	15	0.95	3	119.35	3	0.00	4
B-2	100 × 1000	851.68	14	0.05	1	208.83	3	0.00	1
B-3	100 × 1000	253.84	6	0.21	2	176.88	4	0.21	5
B-4	100 × 1000	104.40	3	0.00	1	101.64	2	0.00	1
C-1	100 × 1000	1323.81	19	0.00	1	234.77	4	0.00	1
C-2	100 × 1000	646.08	13	0.00	1	105.11	2	0.00	1
C-3	100 × 1000	631.92	14	0.00	1	191.11	4	0.00	2
C-4	100 × 1000	499.12	13	0.00	1	155.20	4	0.00	1
Mean		523.46		0.10		140.59		0.02	

Table 5.5: Computational results of the OR-Library large test problems (100 × 1000): $\mathcal{MM}$AS Single Ant A - $\alpha = 2.5,\ \beta = 0.8$ and $\rho = 0.9$; $\mathcal{MM}$AS Colony (5 ants) B - $\alpha = 2.5,\ \beta = 0.8$ and $\rho = 0.9$; secs – seconds; its – iterations

However, should the solution to a general problem instance be sought, then the algorithm would have to run either for a fixed period of time or a maximum number of iterations as ACO derives feasible upper-bounds without the guarantee of obtaining an optimum solution and thus the best solution found would be recorded. This is a typical issue associated with experimentation of ACO and other stochastic local search algorithms, for examples see Dorigo and Stützle (2004) and Hoos and Stützle (2005).

Consequently, the time taken to obtain the best solution, the relative error from the known optimal and the number of iterations were recorded. If the best solution matched the known optimal then the experimental trial number was also recorded, otherwise the trial with the least error was recorded. The last row of the table displays the mean run times and mean relative errors. Again there is strong evidence that $\mathcal{MM}$AS has the ability to provide very high quality solutions for most problem instances.

Caution is required when regarding average solution times for this set of problems, as not all of the problems were solved the same number of times. However, they do provide some evidence of computational performance and efficiency. Results are presented in Table 5.5 that display the problem instance in the first column, problem size in the second column, the remaining columns display time taken in seconds (sec), number of iterations (its), relative error from the known optimum (% err) and the trial number for which the values were recorded (trial). The results indicate that $\mathcal{MM}$AS with a colony of five ant obtains all but one of the twelve optimum solutions with an error of 0.21% and an average computational run-time of 140.59 seconds. Whereas the single ant version, the average run-time was 523.46 seconds but one of the optimal solutions was not found which has a maximum error of 0.95%. The statistical evidence infers that the best option is to use a colony of five ants, as the average run-times are shorter.

5.10 Hyper-cube Framework for the CFLP

The *Hyper-Cube Framework* (HCF) developed by (Blum, 2004, Blum and Dorigo, 2004, Blum et al., 2001) is a natural extension to the $\mathcal{MM}$AS algorithm. The original objectives of the algorithm were to standardise the pheromone levels to belong to the interval [0,1] with the intention of pheromone levels to directly represent probabilities. Rationale behind this development was to overcome scaling problems associated with pheromone levels of $\mathcal{MM}$AS being influenced by objectives costs, where these costs were observed to have a negative influence on the efficiency of the algorithm depending on their order of magnitude. Advantages of the HCF are that due to an entropy based pheromone update phase, not only do the pheromone levels remain in the prescribed interval but they also tend towards binary values as the algorithm converges to a steady state solution, i.e. ants converge to a dominant pathway. The HCF algorithm has similar traits to another meta-heuristic technique called the Cross-Entropy method.

The *Cross-Entropy Method* (CE) was originally developed to help simulate the occurrence of rare events that can take place within stochastic networks, by Rubinstein (1997), where the probability of such rare event occurring needs to be accurately estimated. The author realised that the CE method could be easily adapted to solve combinatorial optimisation problems (Rubinstein, 1999, 2001, 2002). Initially, research centred around solving the TSP, QAP and maximal cut problems which culminated in a generalised strategy for solving combinatorial optimisation problems (Rubinstein and Krose, 2004). As with ACO the algorithm is iterative and originally had two main phases: the first phase creates a large set of randomly generated solution components, while the second phase uses common components generated by the first phase to update and guide future iterations. Similarities between the HCF ACO algorithm and CE were detailed by Blum et al. (2001) and Blum and Dorigo (2004) which concentrated on the QAP and they demonstrated that if the same update selection criteria was used, then the two methods were identical.

There is also some commonality between the $\mathcal{MM}$AS algorithm and the HCF. Primarily, the pheromone lower and upper limits of $\mathcal{MM}$AS are set to zero and one respectively in the HCF algorithm. However, the pheromone update phases differ significantly. $\mathcal{MM}$AS uses an update based only on the best solution found as described in equations (5.7) and (5.8), whereas HCF implements an entropic style update derived from the solution obtained from each ant within the colony and the best solution found. The HCF update mechanism is defined later in equations (5.10) and (5.11). The HCF pheromone update phase enables higher levels of pheromones to be placed on those facilities common to each ant at each iteration. This is a similar function to that employed in the CE method, which places future bias on common solution components from a subset of ranked solutions derived at each iteration. Consequently, both of the HCF and CE methods adopt a procedure that places emphasis or bias for future selection based on common components across a set of solutions or concentrator sets. Recently the CE method has been used to solve the CFLP by Caserta and Quiñonez Rico (2009), their method derived optimal solutions to all of the test problems in the OR-Library and conducted further successful experiments on a series of larger randomly generated problem instances. Consequently, because of the similarities between HCF and CE it is worthwhile considering the HCF as a potential solution method which would then allow for a critical comparison of HCF, $\mathcal{MM}$AS and CE to be conducted.

The development of a HCF algorithm initially takes advantage of the initialisation, construction and pheromone based local search phases that were used for the $\mathcal{MM}$AS algorithm described in the previous sections. Initially experimentation focuses on the 37 basic test instances and results are compared with those given in Table 5.4 and then moves onto the 12 larger test instances to make comparisons with results from Table 5.5.

5.11 HCF: Restricted Pheromone Interval

As previously stated, the main difference between HCF and other ACO algorithms is that pheromone levels are restricted to the interval $[0, 1]$. This restriction is imposed and guaranteed by implementing a specific pheromone update rule that is defined by information collectively gathered from each ant within the colony. This method basically favours those facilities that are commonly visited by a colony of ants, and is referred to as entropy (Blum and Dorigo, 2004, Blum et al., 2001). The method has the advantage that as the algorithm iterates the pheromone levels then tend towards binary values i.e. the limits of the interval $[0, 1]$. This is a desirable feature for the CFLP since solution components represent facilities that are modelled as binary decision variables and used by ACO to derive a set of facilities to be opened. Thus, as the algorithm converges to the binary limits then the corresponding facilities tends towards a set of optimal facilities.

5.11.1 HCF: Pheromone Update

Blum and Dorigo (2004) raised an issue concerned with the performance of $\mathcal{MM}$AS potentially being affected by instance specific data. Their rationale implied that since the pheromone update phase was inversely related to the magnitude of the best objective value obtained for minimisation problems, then algorithmic performance would be sensitive to this and a scaling of instance data or a scaled pheromone update would be required to address this potentiality. Fortunately, the HCF algorithm takes care of this by having a restricted pheromone interval and an update procedure that guarantees the pheromone limits remains within the interval $[0, 1]$ without the need to scale the original instance data.

The HCF update phase for the CFLP is defined as:

$$\tau_i \leftarrow (1-\rho)\tau_i + \rho\,\Delta\tau_i^{\textsf{best}} \qquad \forall\, i \in I\,, \tag{5.10}$$

with entropy deposit

$$\Delta\tau_i^{\textsf{best}} = \frac{1/z^{\textsf{best}}}{\sum_{h=1}^{k}\left(1/z^h\right)}, \tag{5.11}$$

where $z^{\textsf{best}}$ is the overall cost of the best solution to-date and k is the number of ants. Note that equation (5.10) is similar to the combination of equations (5.7) and (5.8), with two exceptions. Firstly, a pheromone factor ρ has been introduced into the pheromone deposit term (5.8), which is also evident in the CE method of Caserta and Quiñonez Rico (2009), and secondly the product of ρ and the HCF entropy deposit terms (5.11) ensure that the pheromone levels remain within the interval $[0, 1]$.

5.12 HCF: Experimentation

A series of experiments consisting of five trials per instance for each of the 37 basic OR-Library test instances were conducted, using the the same computational platform as used in section 5.9. Parameter settings were based on strategy C by setting $\alpha = 1.0, \beta = 0.0$ and $\rho = 0.9$. Rationale for using these setting were based on those adopted by Caserta and Quiñonez Rico (2009) for the CE method. It is worthwhile noting at this stage that should it be possible to derive optimal or near optimal solutions without the iterative use of ant visibility, then the algorithm becomes much simpler to design and implement. However, the simpler design may require a greater number of iterations and computational time to converge to optimal solutions. To reflect this, the maximum number of iterations allowed was set to 50.

Table 5.6 displays the computational results for the median run-times using a colony of five ants, these results were then compared to those given in Table 5.4. The coefficient of variation for the series of HCF experiments is 0.95 which is greater than that of $\mathcal{MM}$AS colony algorithm (0.87) with B parameter settings. However, upon closer inspection all of the optimal solutions were found using the HCF algorithm in less than 20 iterations, with a shorter average run-time ($\mathbf{0.62} < 0.70$) and a lower standard deviation ($\mathbf{0.59} < 0.61$). These observations suggest that HCF with C parameter settings performs better than the $\mathcal{MM}$AS colony algorithm.

As in section 5.9.1, a series of experiments were carried out to determine if the HCF algorithm was capable of finding optimal solutions to the 12 larger test instances of the OR-Library. The HCF algorithm was executed a maximum of five times per instance and the best solution of the five trials was recorded. The best solutions obtained for these 12 problems were found to match the known optimal solutions and are presented in Table 5.7. Comparisons were then made with the

		HCF C	
Problem	m×n	secs	its
cap41	16×50	0.11	1
cap42		0.12	1
cap43		0.12	1
cap44		0.12	1
cap51	16 × 50	0.12	1
cap61	16 × 50	0.10	1
cap62		0.11	1
cap63		0.10	1
cap64		0.19	2
cap71	16 × 50	0.10	1
cap72		0.10	1
cap73		0.09	1
cap74		0.08	1
cap81	25 × 50	0.52	2
cap82		0.56	2
cap83		0.57	2
cap84		0.59	2
cap91	25 × 50	0.25	1
cap92		0.44	2
cap93		0.22	1
cap94		0.39	2
cap101	25 × 50	0.48	2
cap102		0.43	2
cap103		0.20	1
cap104		0.14	1
cap111	50 × 50	1.72	2
cap112		1.56	2
cap113		2.50	3
cap114		2.51	3
cap121	50 × 50	1.33	2
cap122		1.40	3
cap123		1.35	3
cap124		1.21	3
cap131	50 × 50	1.30	2
cap132		1.05	2
cap133		1.11	3
cap134		0.78	3
Mean		0.62	
St. Dev.		0.59	
COV		0.95	

Table 5.6: OR-Library test problems using parameter settings: C - $\alpha = 1.0, \beta = 0.0$ and $\rho = 0.9$

	HCF			
	C			
Problem	secs	its	% err	trial
A-1	223.04	6	0.00	3
A-2	154.52	5	0.00	2
A-3	430.91	15	0.00	2
A-4	126.68	5	0.00	2
B-1	240.08	4	0.00	3
B-2	208.83	3	0.00	1
B-3	104.48	3	0.00	1
B-4	100.97	3	0.00	1
C-1	306.50	5	0.00	1
C-2	237.55	4	0.00	1
C-3	278.09	6	0.00	5
C-4	143.57	4	0.00	2
Mean	212.94		0.00	

Table 5.7: Computational results of the OR-Library large test problems (100 × 1000): HCF (5 ants) C - $\alpha = 1.0,\ \beta = 0.0$ and $\rho = 0.9$

$\mathcal{MM}$AS colony algorithm results presented in Table 5.5, which showed that the HCF method required a greater number of iterations to find a best solution and more computational run-time.

5.13 Hybrid-ACO: Conclusions and Recommendations

This chapter discussed various aspects required to exploit the structure of the CFLP with an aim to design an efficient and effective ACO algorithm, that would be capable of solving the CFLP by a hybridisation of $\mathcal{MM}$AS with a suitable TP solver. Impetus for such an approach is that, if a set of facilities are selected whose total service capacity is guaranteed to satisfy the total customer demand, then the CFLP reduces to a TP. Thus, ACO can be applied to select those facilities to use and future facility selection based on ACO would be derived from the underlying TP cost and fixed facility costs. Modelling issues associated with this hybridisation were discussed in sections 5.1–5.5.

Originally local search procedures, referred to as daemon actions (Dorigo and Stützle, 2004), were seen as an optional phase within ACO algorithms. However, these days the local search phase is accepted as an integral part of any ACO algorithmic development (Dorigo and Blum, 2005, Dorigo and Socha, 2006). Sections 5.6 and 5.7 discussed local search techniques that were based an dropping and swapping facilities in an iteratively derived solution, with an aim of decreasing overall facility-customer allocation costs. Section 5.6 described an approach new to ACO that used pheromone intensities to define local search neighbourhoods and initial results are displayed in Tables 5.2 and 5.3. By using a pheromone based neighbourhood structure, each time the local search is implemented the order in which facilities are tested for dropping or swapping are likely to be different due to depositation and evaporation within the ACO pheromone update phase. Thus providing a readily available mechanism to give a randomly distributed local search region, which is a desirable local search attribute (Hoos and Stützle, 2005).

Section 5.7 discussed the use of a hybrid $\mathcal{MM}$AS that approximated any underlying TP by using a method by Kirca and Satir (1990) that works well on unbalanced TPs (Agar and Salhi, 1998, Krishnaswamy et al., 2009, Mathirajan and Meenakshi, 2004). This TP approximation method obtained some reasonable results when applied to the OR-Library basic test instances and managed to find over 30% of the optimal solutions with almost 90% of solutions have a relative error of less than 0.5%. However, solution quality of results produced by the algorithm did not fair well against the heuristics of Beasley (1993) and Bornstein and Azlan (1998); who also used approximated TP solutions.

Section 5.9 presented a method that is usually thought of as being prone to excessive computational run-times for little gain in solution accuracy (Arostegui et al., 2006, Bornstein and Azlan, 1998). This method involved solving all of the TPs encountered with an exact solver, using a callable linear programming tool from the COIN-OR distribution (Lougee-Heimer, 2003). Analysis of the results presented in Tables 5.4 and 5.5 contradict those thoughts of previous researchers, as all the best solutions found were optimal solutions. Also, the solutions were found in quicker times using the exact TP solver and were obtainable across a variety parameter settings for the hybrid $\mathcal{MM}$AS algorithm without experiencing any excessive run-times. Furthermore, if the best solution found during an experimental trial was not optimal then the average relative error was less than 0.1%. The statistical evidence further indicated that a colony of ants was more efficient at deriving best solutions.

Recent research by Caserta and Quiñonez Rico (2009) described a CE method, which is the only published meta-heuristic research that claims to be able to solve all of the test instances available in the OR-Library: less than two seconds for each of the 37 instances and less than 2 minutes for the 12 larger problems. Their method also made use of the COIN-OR linear programming solver. Although it is evident that the developed ACO $\mathcal{MM}$AS algorithm can also solve all of these problems, it would be unwise at this stage to claim which one is the superior. Further experimentation and statistical analyses of both ACO $\mathcal{MM}$AS and CE need to be conducted to determine if a significant difference exists.

Similarities between the HCF algorithm and the CE method were discussed in sections (5.10) and (5.11). Results for the HCF algorithm are very encouraging as they indicate that all of the test instances are indeed solvable using a hybrid embedded technique. However, as the number of facilities increases then the average run-time performance favours the use of $\mathcal{MM}$AS (on average $212 > 140$ seconds). Potential reasons for this behaviour are that the pheromone evaporation rate is at 90% which will produce very low pheromone levels for some facilities during the construction phase that ought to be included. This is not an issue for the smaller test instances, as the pheromone based local search mechanism of *DROP-SWAP* is able to bring omitted facilities back into a constructed solution. Also, as the pheromone levels rapidly evaporate then the levels of omitted facilities rapidly tend towards zero which could lead to algorithmic stagnation. This issue grows as the number of decision variables increases, because the *DROP-SWAP* mechanism has a restricted neighbourhood as depicted in Figure 5.3, which can be explained by observing that some facilities may remain in the middle region of the local search space and consequently be overlooked by the search process. The computational run-times for HCF compare well with those reported by Caserta and Quiñonez Rico (2007, 2009) for the 37 basic problems, but take much longer (on average 75% longer) for the 12 larger instances.

One way of ensuring that all facilities are considered within a local search procedure is to use a combination of adding, dropping and swapping facilities as outlined by Daskin (1995). The local search strategy used by Caserta and Quiñonez Rico (2007, 2009) contained two stages; *Add One Drop Many* and *Drop One Add Many*. Blum et al. (2001) had some success using a first-level binary flip local method within the HCF algorithm to solve the QAP, which was later omitted by two of the main authors in later research using the HCF by Blum and Dorigo (2004).

The use of a 1-binary flip for the CFLP is an interesting idea as it can be easily implemented in a first-improvement strategy as used in *DROP-SWAP*. Also, it acts as a hybrid *ADD-DROP-SWAP* mechanism and ought to be considered as an alternative to the *DROP-SWAP* method used thus far. Furthermore, there is no evidence of published research within ant systems associated with this type of local search being applied to the CFLP.

As previously stated, it would be unwise to claim which is the more dominant algorithmic procedure without conducting a more in depth set of statistical analyses. As $\mathcal{MM}$AS, HCF and CE are all able to solve the OR-Library test instances, then a study need only concern itself with the generation of run-time distributions required to obtain optimal solutions.

Chapter 6

ACO: Run-Time Analysis and Evaluation

According to a location analysis survey conducted by ReVelle and Eislet (2005)

> "... the capacitated plant location problem ... Work in this area seems to offer ambiguous results ..."

There are several implications associated with this statement: there is some confusion concerning a variety algorithmic research results that have been published on CFLP; it is unclear which algorithmic approach is more suited to solving the CFLP; the mixed-integer formulation the CFLP is more difficult to solve than previously thought.

Published research for the CFLP indicates that early dominant heuristic techniques relied upon Lagrangean relaxation to provide "good" solutions to the test problems available in the OR-Library. The Lagrangean relaxation research conducted by Beasley (1993) used a Cray super-computer and provided a foundation for future researchers. However, 12 of the 37 basic problems in the OR-Library were for some reason omitted from the analysis. Research conducted by Agar and Salhi (1998), went on to publish results for only four of the basic test instances that did not display any improvement over Beasley's results. However, results for the 12 larger test instances did indicate significant improvements in terms of relative errors. Although solution run-times were given in both cases, they were not considered for comparison. The reason being disparity between the computers that the experiments were conducted on; Cray super-computer 'v' Sun Sparc workstation. Already, some ambiguity had set in and questions ought to have been raised. Even though different computational platforms were used, which is understandable and highly likely even today, why were some test instances omitted in favour of others? No rationale for test problem selection was presented in either of the papers. This behaviour associated with test instance selection continued. It is also evident even in those works considered to have made significant contributions to knowledge of solving the CFLP using metaheuristics (Arostegui et al., 2006, Bornstein and Azlan, 1998, Bornstein and Campelo, 2004).

The first clear indication that all of the 37 basic test OR-Library instances had been used with metaheuristics, was given in a series of results published for a Tabu Search application to the CFLP by Sörensen (2008). These experiments were conducted on an AMD Athlon 1100 PC with 512 Mb RAM and all but two of these instances were solved optimally. The average run-times were given with respect to their potential facility size, yet those instances that were not solved were not identified.

Until recently there was no evidence of research that had been conducted using metaheuristic techniques to solve the CFLP which had managed to solve all of the OR-Library test instances. A summary of ACO experimental results based on applications of $\mathcal{MM}$AS and HCF were reported by Venables and Moscardini (2008), which solved all of the test instances; these are explained in Chapter 5 of this thesis. Also, research by Caserta and Quiñonez Rico (2007, 2009) based on an application of CE claimed to have solved all of the OR-Library test instances for the CFLP. However, the design for the experimental analysis was not discussed and no individual run-times were given for the OR-library test instances. Their research focused on efforts to solve “large” randomly generated instances. Although the authors looked at ten randomly generated instances per problem specification and reported averages of run-time and relative errors over the ten instances, what was not made clear was the number of experiments per instance that were carried out.

The purpose of this chapter is to determine and comprehensively evaluate the differences in behaviour between the CE method, $\mathcal{MM}$AS and HCF ACO algorithms by using empirical analysis based on run-time solutions. This analysis should help address the shortcomings of ambiguity, as raised by ReVelle and Eislet (2005), and identify if a dominant solution procedure exists.

6.1 Classification of Stochastic Local Search Algorithms

A stochastic local search algorithm is a method that randomly selects candidate solutions to a combinatorial optimisation problem. In essence, ACO generates solutions to a combinatorial optimisation problem by making randomised moves on a construction graph for a given problem instance. The CE method adopts a similar approach by randomly selecting a subset of clearly defined solution components, belonging to a combinatorial optimisation problem. In both algorithmic approaches there are probabilities associated with selecting components that change during run-time. The effect of this selection procedure is an artificial learning process which aims to select the most suitable components to derive candidate solutions and determine an optimal solution. Thus, both ACO and CE are types of stochastic local search algorithms.

When applied to the CFLP both ACO and CE are required to derive candidate solutions that satisfy any constraints place upon a problem instance, i.e. both algorithmic approaches generate only feasible candidate solutions. Furthermore, since any candidate solutions are generated by making stochastic selections, then a final selection is the product of a series of embedded stochastic choices. This implies that the time taken to derive a solution is also a stochastic process, i.e. the time taken to obtain a desired solution will be a random variable. These two properties give rise to a special type of stochastic local search algorithm known as a (generalised) Las Vegas algorithm, see Definition 1.

Definition 1 ***: Las Vegas Algorithm***

An algorithm A for a problem class Π *is a (generalised) Las Vegas algorithm (LVA) if, and only if it has the following properties:*

1. *If for a given problem instance* $\pi \in \Pi$*, algorithm A terminates returning a solution* s*,* s *is guaranteed to be a correct solution of* π*.*
2. *For each given instance* $\pi \in \Pi$*, the run-time of A applied to* π *is a random variable* $RT_{A,\pi}$*.*

(Hoos and Stützle (2005), pp 150)

Since the CFLP is a combinatorial optimisation problem and the solution runtime is a random variable, it is possible that for a given fixed time neither ACO or CE may find an optimal solution in a prescribed fixed time. Although stochastic local search algorithms have asymptotic run-time behaviour, an optimal solution may not be found in a given time limit; this property is known as an algorithm's incompleteness (Dorigo and Stützle, 2004, Hoos and Stützle, 2005). Consequently, the relative error or quality of a final solution is a time dependant random variable and thus according to Definition 2 both ACO and CE are (generalised) Optimisation Las Vegas algorithms.

Definition 2 ***: Optimisation Las Vegas Algorithm***

An algorithm A for an optimisation problem Π' is a (generalised) optimisation Las Vegas algorithm (OLVA) if, and only if, it is a (generalised) Las Vegas algorithm, and for each problem instance $\pi' \in \Pi'$ the solution quality achieved after any run-time t is a random variable.

(Hoos and Stützle (2005), pp 151)

Theoretical analyses of optimisation Las Vegas algorithms and stochastic local search usually rely upon idealised assumptions to derive average, worst-case and asymptotic behaviours. Although there has been some recent advances in the theoretical application of run-time analyses to some ACO algorithms (Blum, 2004, Dorigo and Blum, 2005, Neumann et al., 2008, Neumann and Witt, 2009), their practical use is still somewhat limited and unrelated to the hybrid models as developed and discussed in this thesis. Consequently, any run-time analysis shall be implemented empirically.

6.2 Empirical Run-Time Analysis for Stochastic Local Search

The properties of stochastic local search and its direct relationship to optimisation Las Vegas algorithms as outlined in the previous section state that run-times and solution quality are random variables. In fact, the probability distribution associated with obtaining a solution for any instance is a bi-variate distribution based on the random variables associated with run-time and solution quality as indicated in Definitions 1 and 2. A formal definition for a run-time distribution (RTD) is given in Definition 3:

Definition 3 ***: Run-Time Distribution*** *Given an optimisation Las Vegas algorithm A' for an optimisation problem Π' and a soluble problem instance $\pi' \in \Pi'$, let $P_S(RT_{A',\pi'} \leq t, SQ_{A',\pi'} \leq q)$ denote the probability that A' applied to π' finds a solution of quality less than or equal to q in time less than or equal to t. The run-time distribution (RTD) of A' on π' is the probability distribution of the bi-variate random variable $(RT_{A',\pi'}, SQ_{A',\pi'})$, which is characterised by the the run-time distribution function $rtd : \mathbb{R}^+ \times \mathbb{R}^+ \mapsto [0,1]$ defined as $rtd(t,q) = P_S(RT_{A',\pi'} \leq t, SQ_{A',\pi'} \leq q)$.*

(Hoos and Stützle (2005), pp 159)

Although a RTD is a bi-variate distribution which could be graphically represented in a three-dimensional graph with orthogonal axes RT, SQ and P_S, it is usually common practise to work with either a fixed solution quality $SQ = q_f$ to obtain an empirical distribution of $P_S(t)$ or a fixed time parameter $RT = t_f$ to obtain a distribution of $P_S(q)$. Each empirical distribution is a collection of cumulative probabilities for a given instance and is used to determine the overall qualitative behaviour of an algorithm. A RTD can also be used to obtain performance statistical descriptors such as mean, standard deviation, median, inter-quartile range, etc. Thus, empirical RTDs can provide valuable information for the comparison of various different algorithms either for a particular instance and/or a group of test cases to determine any algorithmic dominance.

6.3 Deriving ACO and CE Empirical RTDs for the CFLP

Research into solving the CFLP using ACO and CE shows that all of the test instances in the OR-Library are solvable using a suitable metaheuristic. A detailed run-time analysis is required to determine which of the ACO algorithms together with the CE algorithm, of Caserta and Quiñonez Rico (2009), is the most dominant method. The first stage of this process is to determine what type of run-time analysis is required, the number of runs required per instance to generate the RTDs via cumulative probability distributions and which problem instances ought to be used for the analysis.

Research presented, in this thesis, on the hybridisation of ACO and an exact transportation problem solver indicates that all of the 37 basic test instances can be solved using a colony of five ants with $\mathcal{MM}$AS and HCF algorithms in an average time of less than 1.5 seconds per instance. Whereas CE, claims to solve these problems in less than 2.0 seconds per instance. However, there ought to be caution taken with these values as the number of runs per instance was limited to five with the median run-time being recorded for ACO, whilst the number of CE experiments per instance remains unknown. Average run-time statistics for the ACO application on the 12 larger test instances of less than 3.5 minutes per instance is even more spurious, as these experiments were aimed at indicating that ACO was able to solve these problems. Again the number of experiments conducted on the larger instances with the CE algorithm is unknown, although times of less than 2.0 minutes were reported.

There is some inherent ambiguity associated with these average run-times. However an important observation is that for any one of the test instances an optimal solution can be found either using an ACO or CE algorithm in a reasonably short run-time. This is advantageous when deriving empirical RTDs, because the solution quality can be fixed for optimality, i.e. 100% quality or 0% error. Thus, a RTD need only consider the probability distribution of the time taken to obtain an optimal solution for a given problem instance, within an acceptable experimental time window (e.g. algorithm terminates when either a maximum number of iterations is reached or a fixed time limit expires).

One of the issues associated with obtaining run-time distributions is that many experimental runs are required to construct the cumulative probability distributions. Hoos and Stützle (2005) discuss the implications of sample distribution size and use a basic sampling premise “the more the better” and use a sample size of 1000 individual experimental runs to construct their RTDs. So, using the average ACO run-times from the preliminary experiments for the 37 basic problems of 1.5 seconds, a RTD would take on average $1.5 \times 1000 \approx 25$ minutes of computational effort per instance to collect sample data. Using the same tactic for the larger problems an average time of 213 seconds would take approximately 59 hours per instance. The vast difference in these times suggest that the computational effort required to collect sample data for a RTD of a particular instance is more manageable from those problems belonging to the basic set of 37. This will also make algorithmic comparison for the various ACO $\mathcal{MM}$AS, HCF and CE algorithms a more manageable task.

6.3.1 Measuring RTDs

There are two methods that can be used to measure the data required to build a RTD for empirical analysis. The first is *computational run-time* that is associated with a computer based clock timer and is referred to as CPU time. Whilst the second is termed as *operation counts*, which reflects the number of important operations that can be used to determine or measure algorithmic performance. In a stochastic local search algorithm the count may refer to the total number of local search steps or evaluations made upon completion of a single run-time experiment. The use of operation counts is seen as the more flexible of the two methods as the RTDs obtained then become computational platform invariant, i.e. the RTD for an algorithm remains the same regardless of the computer it is executed on. The main disadvantage is that different algorithm designers may place emphasis on counting in different ways to one another and that the counts may also affected by different types of algorithmic hybridisation, e.g. how to count the use of an exact solver called from a library where access to the code is unavailable and then compare it to a user defined heuristic based upon approximation techniques.

Ultimately to compare different algorithms, it would be ideal to use CPU clock time as a measure whilst minimising the number of external variables that may affect performance such as using the same: computer, programming language, compiler with identical options, source callable library functions. The C++ code for the CE algorithm was available to download from the main author's web pages (http://iwi.econ.uni-hamburg.de/mcaserta/). Only the links to the problem instance data files and the COIN-OR solver had to be changed prior to compilation and program execution. The ACO algorithms were also coded in C++ and used the same COIN-OR solver. Since each of the algorithms could be compiled and run on the same computer, CPU-time was used as a performance measure. Although during the algorithmic development stages different computers had been used, all of the RTD experiments and associated sample data were generated for each of the algorithms on the same computer (Pentium 4 3.0GHz Linux PC with 2Gb of RAM, C++ compiled using the GNU g++ compiler using the -O option, COIN-OR version coin-linux-ix86-gcc4.0-02).

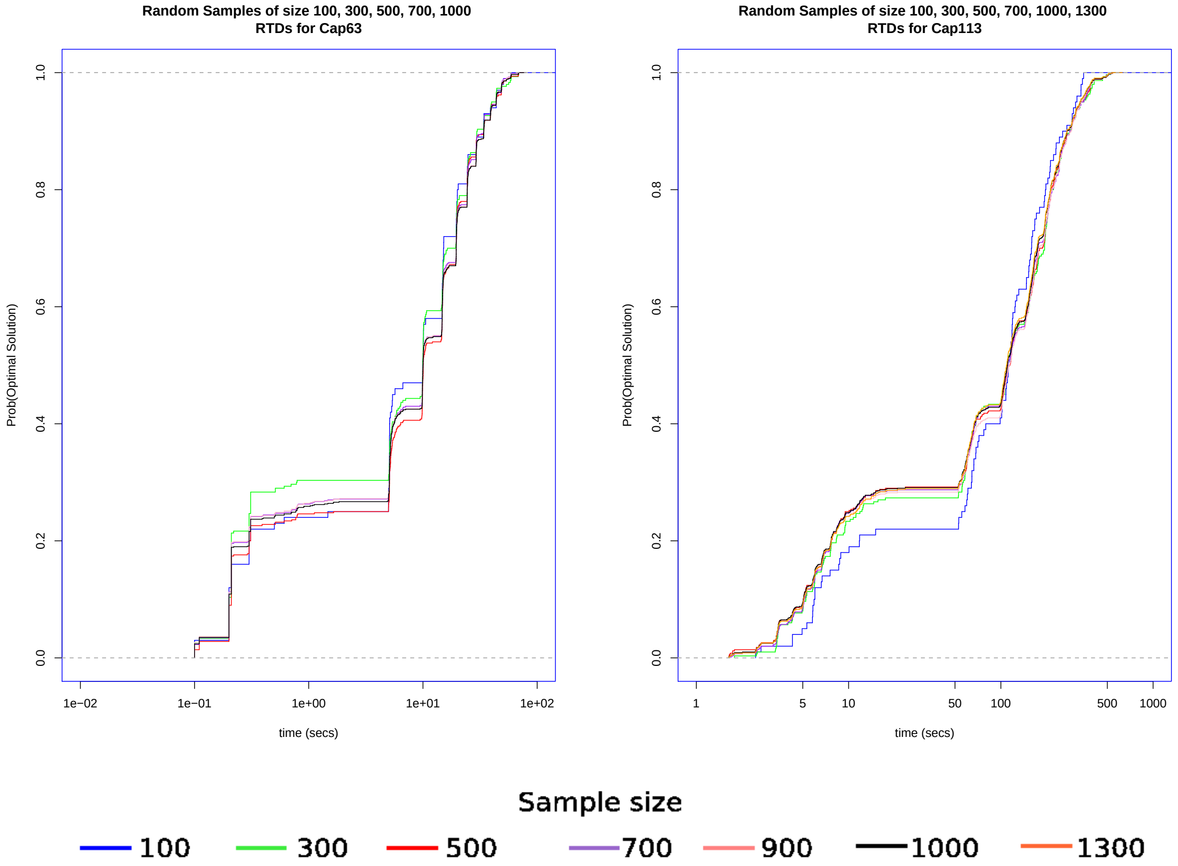
6.4 Qualitative Analysis of Empirical RTDs

The first stage of analysis was to select a problem instance from the set of 37 OR-Library test problems and construct a series of RTDs using the exact-hybrid MMAS algorithm described in chapter 5. Hoos and Stützle (2005) suggest that to obtain a reliable empirical RTD the sample size of the distribution, or number of runs for a particular instance, needs to be between 100 and 10000. Although sampling theory can be used to determine a sample size, such as the use of the *central limit theorem* for *normal* and *student* distributions, RTDs for stochastic local search algorithms size generally do not conform to standard assumptions and thus empirical sampling techniques are more appropriate.

A way to validate a selected sample size is to visually check and ensure that the RTD displays asymptotic behaviour as the sample size increases towards its chosen limit. This is to ensure that the run-time data collected for a problem instance provides an accurate representation of its run-time distribution. For the purpose of this study *cap63* was chosen as a typical instance to verify and validate the RTD sample size. RTD data for sample sizes of 100, 300, 500, 700 and 1000 runs were independently generated. Figure 6.1 clearly displays similar qualitative behaviour across all five sample sizes. Also, the graph demonstrates convergence to a common RTD as the sample size increases.

The whole process was then repeated using a second problem instance *cap113* and an extra set using a sample size of 1300 was also included. The extra set of 1300 run-times displayed a very similar behaviour and showed only minor differences when compared with a sample size of 1000, thus indicating asymptotic convergence. Hence, it is not worth the extra overall time-costs associated with generating 300 further run-time solutions when a sample size of 1000 is sufficient, i.e. a justifiable RTD sample size is 1000 run-time solutions.

Graphs of RTDs provide some very useful qualitative information about the probabilistic behaviour of the algorithm(s) that are being investigated. Stochastic local search algorithms such as ACO are adaptive search procedures that are able to move away from areas of stagnation, to avoid getting trapped at local minima (Dorigo and Stützle, 2004, Hoos and Stützle, 2005). Often evidence of this behaviour is demonstrated by the graphing of solution quality against run-time



Random Samples of size 100, 300, 500, 700, 1000
RTDs for Cap63

Random Samples of size 100, 300, 500, 700, 1000, 1300
RTDs for Cap113

Figure 6.1: Justification and Verification of Sample Size

for a single computational run of a particular instance (Arostegui et al., 2006). Although this approach may show evidence of the ability to move away from local minima or regions of stagnation, any inferences made relate only to that particular run and do not lend themselves to a generalisation of algorithmic behaviour. The empirical RTD for a given instance is an attempt at creating a general picture of algorithmic behaviour for that problem. Thus, across a series of test problems RTD graphical plots ought to provide evidence of an algorithm's adaptability as well as its suitability to solve those problems within some fixed time constraints (CPU-time and/or a maximum number of iterations).

The two graphs shown in Figure 6.1 were not only useful for setting the sample size, but they also displayed two of ACOs main attractive features, namely; adaptability and asymptotic convergence to a solution with a prescribed level of quality. Both graphs show a comparatively large flat region, this is evidence of algorithmic stagnation that occurs when the same facilities are being selected repeatedly. Fortunately, due to pheromone decay and local search the algorithm manages to overcome this situation and proceeds to find solutions that are optimal 100% of the time. The steepness of the RTD profile indicates the speed of convergence, where as the decreasing profile of the latter part of the graph displays characteristics of asymptotic behaviour. It is worthwhile restating that ACO may demonstrate incompleteness when trying to solve a problem. This occurs when the RTD profile falls short of a probability equal to one, or an optimal solution is not guaranteed 100% of the time within the fixed run-time constraints.

6.5 Comparative Qualitative Analysis

Assuming that a RTD can be effectively constructed using a sample size of 1000 for all 37 test instances, then a comparative graph displaying the qualitative behaviour of a set of algorithms can be simultaneously plotted for each test instance. This type of plot helps to identify any run-time issues associated with algorithmic stagnation, convergence rates, incompleteness and probabilistic dominance. Algorithmic stagnation often occurs when the algorithm becomes trapped at a local optima, this is observed in an RTD as a time-period when the probability of obtaining an optimal solution does not increase; e.g. in Figure 6.1 stagnation occurs in both *cap63* and *cap113* at times associated with a probability of obtaining an optimal solution of 0.3, which may relate to a general run-time characteristic of the algorithm that would be identifiable from further test instances. The convergence rates are observed by the steepness and shape of the RTD curve profile; e.g. *cap113* the rates of convergence prior and post stagnation are different, and suggests that the algorithm finds optimal solutions quicker once stagnation has occurred. Incompleteness is easily recognised from an RTD plot, as the distribution curve would fail to reach a probability of 1.0. This may occur either when an algorithm fails to move away from a region of stagnation or when an algorithm fails to converge quickly enough during the run-time. Probabilistic dominance of one algorithm over another is also easily identified from a RTD, as this occurs when the RTD curve profiles do not intersect with each other.

The following algorithms are considered for comparison: hybrid-exact $\mathcal{MM}$AS, HCF and CE. Both $\mathcal{MM}$AS, HCF algorithms are tested with *DROP-SWAP* and a binary one-flip local search procedures. The five algorithms are referred to as: MMAS, 1F-MMAS, HCF, 1F-HCF and CE. The RTDs for test instance were generated using a sample size of 1000 runs per instance, Figure 6.2 shows instances *cap63* and *cap113* and all of the RTDs plots are displayed in appendix B.

The graphs displayed in Figure 6.2 indicate that there may be some existence of algorithmic dominance for a given instance (1F-MMAS *Cap63*) but not for all instances. Closer inspection of both test instances reveals: 1F-MMAS, HCF and

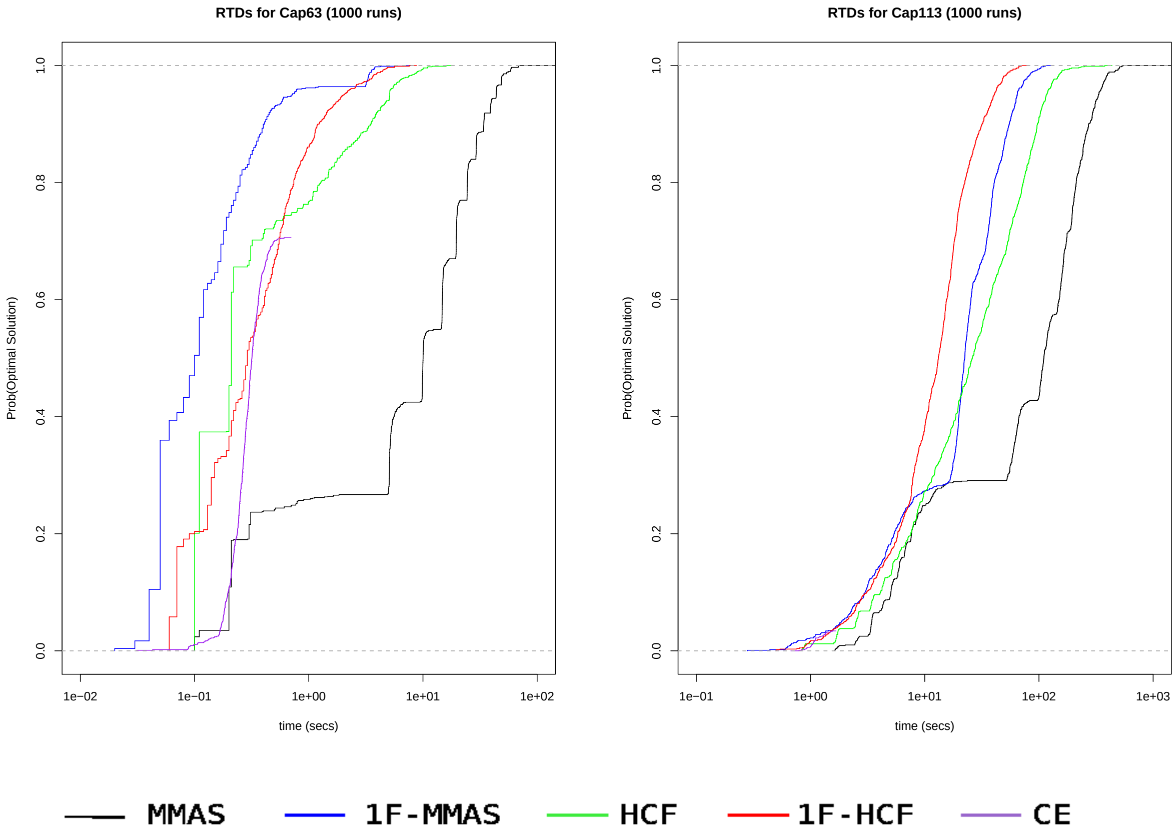


Figure 6.2: Comparative RTDs for two OR-Library Instances

	MMAS	1F-MMAS	HCF	1F-HCF	CE
Dominance	3	15	16	2	0
Completeness	37	36	37	36	0
Stagnation	19	3	7	2	3
Steepness	18	2	16	2	2

Table 6.1: Qualitative Summary Table

1F-HCF have probabilistic dominance over MMAS; there is no single probabilistic dominant algorithm, because the RTD profiles for 1F-MMAS, HCF and 1F-HCF intersect at various places; 1F-MMAS displays evidence of stagnation indicating that HCF and 1F-HCF are more reliable; profile for 1F-HCF looks more favourable than HCF (probabilistic dominance and steep smooth profile for *cap113*). Interestingly, although the CE profile for *cap63* is steep and smooth, it displays poor performance in obtaining a complete set of optimal solutions for both instances (66% and 3% success rates). Although, the RTDs for only two problems are displayed there is a clear indication that CE is less likely to be reliable at solving the CFLP than a hybrid ACO based algorithm.

A summary table of the five different qualitative performances for all 37 test instances is given in Table 6.1: Dominance is the number of times an algorithm appears to the left hand side of the plot; Completeness is the number of times an algorithm was 100% successful across the test instances; Stagnation refers to the number of times an algorithm appears to have the worst stagnation behaviour, but ignores any incompleteness that may be present; Steepness identifies the number of times an algorithm has the steepest profile regardless of stagnation and incompleteness attributes.

Although this analysis is very rudimentary, Table 6.1 clearly indicates issues of incompleteness and stagnation associated with the CE algorithm. Stagnation issues are also evident with the MMAS algorithm, yet it has the ability to solve the problems quickly if stagnation does not occur. A single probabilistic dominant algorithm does not emerge from the RTDs. However, there is some indication that 1F-MMAS and HCF may be the most reliable. A Chi-square test for independence also indicated that the outcomes of the four performance indicators are dependent upon the algorithm used. The expected counts for this test indicate that CE outcomes can be pooled, as the expected counts were all less than 5. Pooling 1F-HCF and CE outcomes leads to a rejection of a null hypothesis of independence, with the stagnation of MMAS being the most likely cause for rejection of the null hypothesis due to this having a standardised residual at the extreme end of the standard normal distribution.

6.6 Comparative Quantitative Analysis

Appropriate quantitative analyses are required to gain further statistical insights into the performance differences between the various algorithms. The profiles of RTDs are often a good starting point when comparing algorithmic performance, but comparative analyses also benefit from the inclusion of basic statistical descriptors. The stochastic nature of RTD profiles can display a great deal of variation in run-time data, due to effects of asymptotic behaviour and stagnation, so care needs to be taken when selecting what descriptive statistics to present. It would be unwise to present mean and standard deviation measures because these are affected by extreme data values that may be present in the tails of the RTDs. Also, standard differences of means hypothesis testing makes assumptions about the distributed data such as normality which are typically inappropriate for many RTD distributions. Analyses of RTDs lend themselves to quantile descriptors and non-parametric hypothesis testing.

This comparative study shall concentrate on the behaviour associated with the median run-times of RTDs and all of the analysis shall be conducted using the open source statistical programming package R. Initially, box-plots are presented to determine if any obvious algorithmic differences occur. Then a statistical bootstrapping technique is applied to display median sampling distributions, which is then followed up with comparisons of median confidence intervals.

6.6.1 RTD Median Run-Times

Standard box plots are often referred to as five-figure summary plots, which graphically display the minimum, lower-quartile, median, upper-quartile and maximum. There are some common useful adaptions to the basic box plot that display information such as outliers and extreme values. However, when comparing RTD data these basic plots may yield some useful information, but they may also lead to some misinterpretations. The likely cause for this is that the box plot is too basic and could benefit from some further adaption or annotation: confidence intervals about the median, variable width that could indicate relative distribution sizes or distribution density. Fortunately, these issues have been considered for some time and are readily available in R, (McGill et al., 1978). Figure 6.3 displays variable width notched box plots for *cap63* and *cap113*. The box width is relatively proportional to the number of data items in each category (number of optimal solutions found) and the maximum number of data items across each category (1000). The notch is a representation for a 95% confidence interval about the median. If the notches of boxes do not overlap then there is a significant difference between the medians. Thus the box plots for Cap63 indicate that 1F-MMAS has the shortest median run-time and because the notch does not overlap with any other notch it is significantly different to any of the other median run-times.

The *cap113* plot clearly displays that the CE algorithm has the significantly shortest median run-time. However its width is the very small compared to the others, which implies that any inference based on the CE median run-time is un-

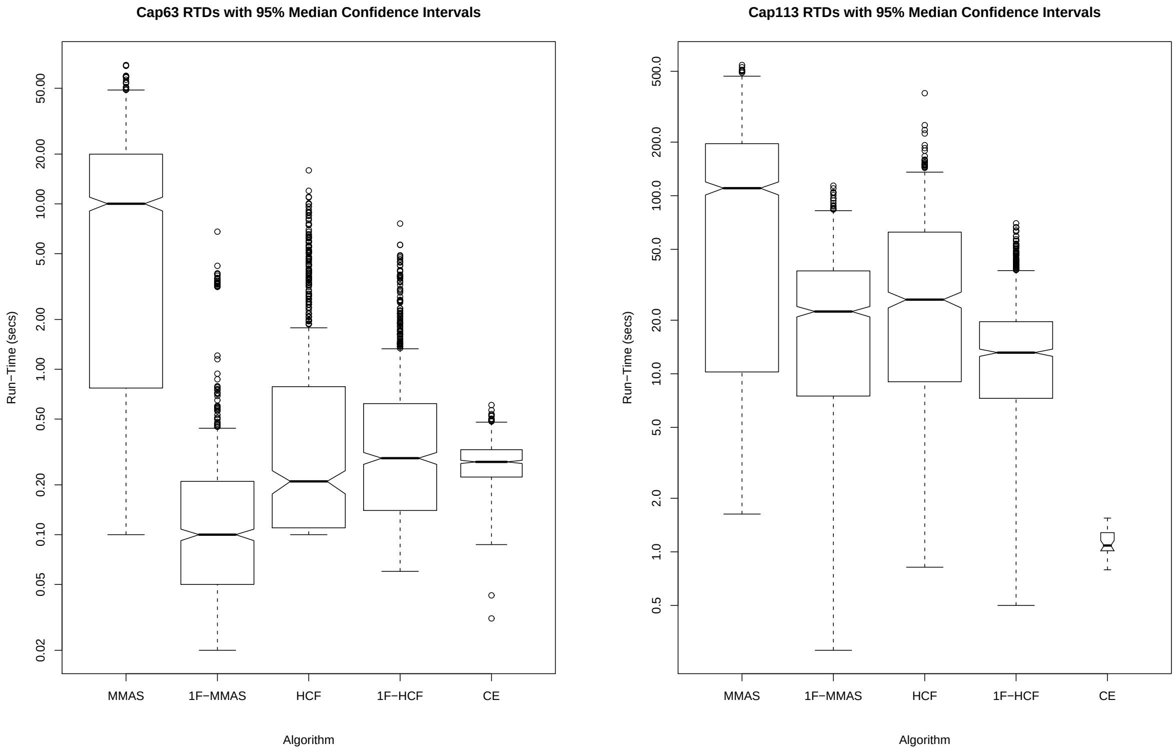


Figure 6.3: ACO and CE RTDs: Variable Width Notched Box Plots for *Cap63* and *Cap 113*

reliable as a comparative statistical descriptor. These plots provide a very useful diagnostic way of ranking algorithmic performance across an ensemble of problem instances, where ranking is associated with the box width, the median and 95% confidence interval. Appendix C displays the solution quality plots for all 37 instances and Table 6.2 shows the ranking of each algorithm.

The penultimate row of Table 6.2 lists the sums of ranks for each algorithm across the set of test problems. The final row displays the rank sum order, which supports the evidence from the qualitative analysis that the HCF algorithm to be the most reliable of the five algorithms tested.

6.6.2 Investigation of the Run-Time Median

One of the advantages of using the median run-time as a descriptor is that it corresponds to an average run-time that it will take to find the best solution. A disadvantage with this statistic is that it is derived from a sample of 1000 runs and may not accurately reflect the true population median measure. A large sample size for a RTD helps to reduce any significant sampling error, but too large a sample size would be impractical across all of the test instances. The asymptotic convergence of the RTDs displayed in Figure 6.1 justified the use of 1000 run-times to generate acceptable RTD probability profiles. These profiles suggest that the empirical approximation of a RTD would contain characteristics of its actual cumulative probability function, i.e. the empirical distribution is a reasonable representation. Thus, it is possible to use the RTD data as a sampling framework to generate a sampling distribution of the median. Furthermore, the sampling distribution of the median can then be used to obtain confidence intervals associated with the median.

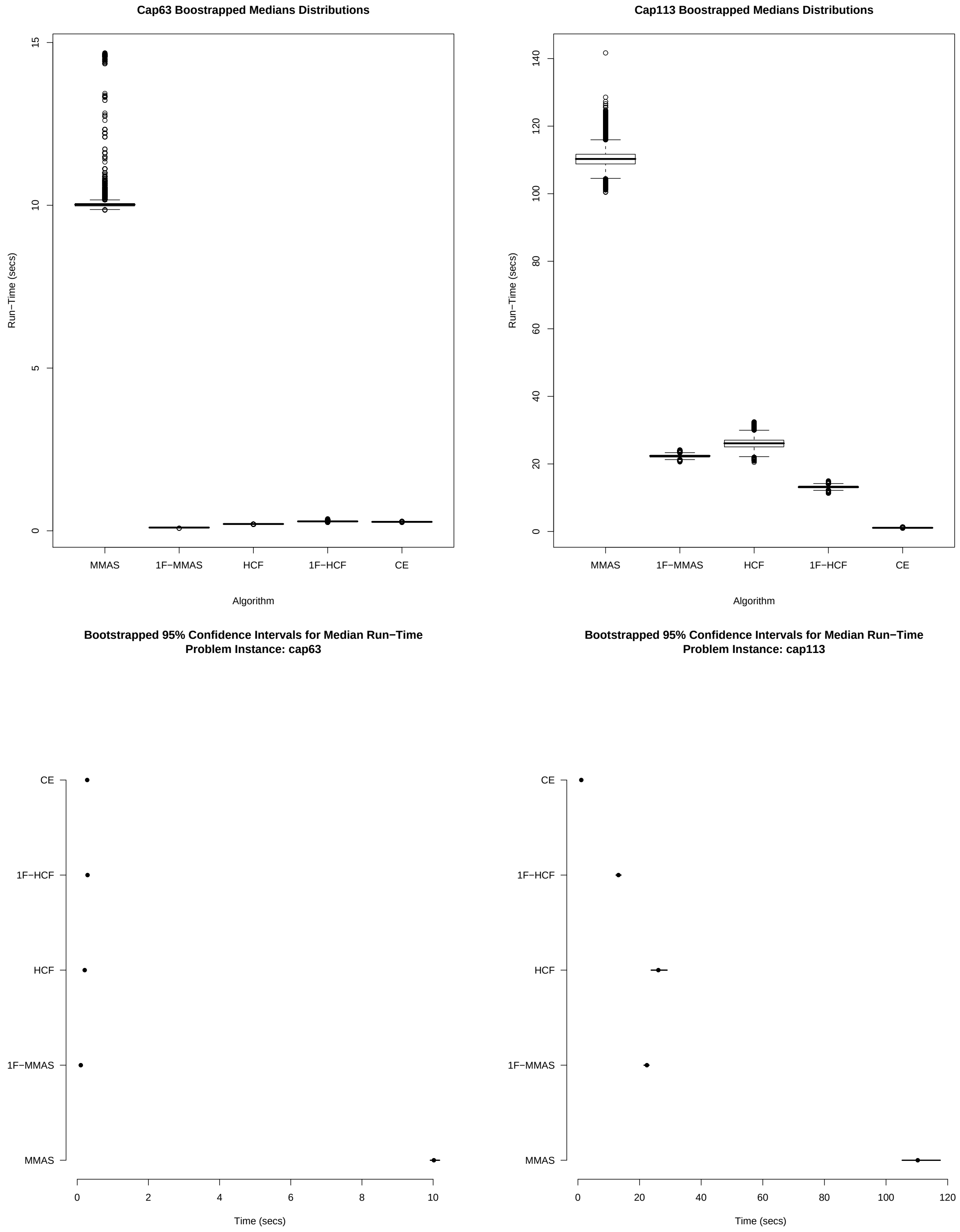
To construct a sampling distribution of the median for a particular problem instance, it is necessary to have an appropriate representative sampling framework

Prob	MMAS	1F-MMAS	HCF	1F-HCF	CE
cap41	3	1	3	2	5
cap42	3	1	4	2	5
cap43	3	1	3	2	5
cap44	4	1	3	2	5
cap51	3	2	3	1	5
cap61	3	1	4	2	5
cap62	4	1	2	3	5
cap63	4	1	2	3	5
cap64	1	4	1	3	5
cap71	3	1	3	2	5
cap72	3	1	3	2	5
cap73	4	1	3	2	5
cap74	2	1	2	4	5
cap81	4	1	2	3	5
cap82	4	1	3	2	5
cap83	3	1	2	4	5
cap84	2	3	1	4	5
cap91	2	4	1	3	5
cap92	2	3	1	4	5
cap93	2	4	1	3	5
cap94	1	4	1	3	5
cap101	4	1	3	2	5
cap102	1	4	1	3	5
cap103	3	4	1	2	5
cap104	1	4	2	3	5
cap111	4	3	2	1	5
cap112	2	4	1	3	5
cap113	5	2	3	1	4
cap114	1	4	2	3	5
cap121	1	3	2	4	5
cap122	2	3	1	4	5
cap123	2	3	1	4	5
cap124	2	3	1	4	5
cap131	3	2	1	4	5
cap132	2	3	1	4	5
cap133	3	4	1	2	5
cap134	1	4	2	3	5
Rank Sum	97	89	**73**	103	184
Rank	3	2	**1**	4	5

Table 6.2: RTDs: Variable Width Notched Box Plots Rankings

to select samples from. Each sample selected has a median value and a collective set of samples has a corresponding set of medians that form a sampling distribution. Bootstrap sampling adopts this technique to build empirical distributions which can then be used to obtain confidence intervals for a given statistical descriptor (Efron and DiCiccio, 1996). Thus a set of run-time distribution data can be used to derive confidence intervals for the median run-time, which is easily implemented in R. The main advantage of bootstrapping over traditional statistical methods is that no assumptions about the underlying sample distribution shape are necessary, whilst its disadvantage is that it is computationally intensive and often requires many samples to reduce sampling errors.

A rule of thumb for generating confidence intervals using boostrapping is to make the number of bootstrap samples as large as it needs to be by checking the stability of the 2.5% and 97.5% percentiles (Wood, 2005). Bootstrapped samples of 10,000 and 100,000 median run-times were generated from the median run-time distributions for each of the problem instances, *cap63* and *cap113*. The confidence intervals were calculated ten times for each problem using 10,000 and 100,000 samples. The bootstrapped sample of 100,000 median run-times gave a more reliable series of confidence intervals, and displayed a stability of two decimal places for both upper and lower percentiles. Box plots and confidence interval plots for the bootstrapped median sampling distributions for two test instances are given in Figure 6.4, with complete sets being presented in appendices D and E. The box plot boxes are very short, i.e. the data is mainly grouped around the median, which indicates that the sampling distributions are reliable. The plots also infer that 1F-MMAS has the significantly shortest median run-time for *Cap63* and 1F-HCF has the significantly shortest median run-time for *Cap113* (ignore CE due to small amount of data available in the run-time distribution). The 95% confidence intervals of median run-times for *Cap63* are difficult



Cap63 Boostrapped Medians Distributions

Cap113 Boostrapped Medians Distributions

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap63

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap113

Figure 6.4: Boostrapped Median Run-Time Sampling Distributions: Box Plots and 95% Confidence Intervals

to interpret as the median run-times distributions for four of the algorithms were very similar. However, the same plot for *Cap113* clearly shows the CE algorithm having the smallest confidence interval without crossing over any others. Since, the CE values are unreliable for this instance then the next best is the 1F-HCF algorithm as previously indicated.

These plots can be used simultaneously to create a ranking system, should pairs of confidence intervals overlap then they are treated as having equal rank. Table 6.3 shows the ranking of each algorithm per problem instance and Table 6.4 contains the associated 95% confidence intervals. Ranking supports HCF as having the most reliable median run-time, which is followed by the CE. However the median run-time sampling distributions were obtained from those runs that managed to find optimal solutions, i.e. the CE median run-time sampling distributions were derived from a smaller original data set. Although this may give an unfair advantage to the CE method for comparisons, HCF still outperformed it which indicates the advantage of using a HCF ACO based algorithm to solve the CFLP.

Prob	MMAS	1F-MMAS	HCF	1F-HCF	CE
cap41	3	1	3	2	5
cap42	3	1	4	2	5
cap43	3	1	3	2	5
cap44	4	1	3	2	5
cap51	3	2	4	1	5
cap61	3	1	4	2	5
cap62	4	1	2	3	5
cap63	5	1	2	4	3
cap64	2	5	1	4	3
cap71	3	1	3	2	5
cap72	3	1	3	2	5
cap73	4	1	3	2	5
cap74	2	1	2	4	5
cap81	5	1	2	3	4
cap82	5	1	4	2	3
cap83	4	2	3	5	1
cap84	3	4	1	5	2
cap91	3	5	1	4	2
cap92	3	4	1	5	2
cap93	3	5	1	4	2
cap94	2	5	3	4	1
cap101	2	3	5	4	1
cap102	2	5	1	4	3
cap103	4	5	2	3	1
cap104	1	5	2	3	4
cap111	5	4	3	2	1
cap112	3	5	2	4	1
cap113	5	3	4	2	1
cap114	2	5	3	4	1
cap121	2	4	3	5	1
cap122	3	4	2	5	1
cap123	3	4	2	5	1
cap124	3	4	2	5	1
cap131	5	3	2	4	1
cap132	3	4	2	5	1
cap133	4	5	2	3	1
cap134	1	5	2	4	3
Rank sum	118	113	**92**	126	101
Rank	4	3	**1**	5	2

Table 6.3: Boostrapped Median Sampling Distribution Rankings

	MMAS		1F-MMAS		HCF		1F-HCF		CE	
Prob	2.5%	97.5%	2.5%	97.5%	2.5%	97.5%	2.5%	97.5%	2.5%	97.5%
cap41	0.12	0.12	0.04	0.04	0.12	0.12	0.07	0.07	0.34	0.35
cap42	0.12	0.12	0.04	0.05	0.13	0.13	0.06	0.06	0.25	0.27
cap43	0.12	0.12	0.04	0.05	0.12	0.12	0.06	0.06	0.25	0.27
cap44	0.13	0.13	0.05	0.06	0.12	0.12	0.07	0.08	0.26	0.27
cap51	0.23	0.23	0.12	0.14	0.17	0.23	0.07	0.08	0.31	0.33
cap61	0.10	0.11	0.06	0.06	0.10	0.11	0.07	0.07	0.26	0.28
cap62	0.21	0.21	0.06	0.06	0.11	0.11	0.13	0.14	0.28	0.29
cap63	9.92	10.18	0.09	0.11	0.21	0.21	0.28	0.31	0.27	0.28
cap64	0.10	0.10	0.38	0.48	0.10	0.10	0.28	0.31	0.26	0.27
cap71	0.10	0.10	0.06	0.07	0.10	0.10	0.08	0.08	0.25	0.26
cap72	0.10	0.10	0.06	0.06	0.10	0.10	0.07	0.07	0.26	0.27
cap73	0.17	0.17	0.05	0.06	0.09	0.09	0.07	0.07	0.25	0.27
cap74	0.08	0.08	0.05	0.05	0.08	0.08	0.12	0.13	0.22	0.23
cap81	0.55	0.80	0.37	0.45	0.53	0.54	0.52	0.62	0.73	0.76
cap82	1.15	1.16	0.15	0.25	0.63	0.87	0.64	0.77	0.75	0.78
cap83	1.24	1.51	0.77	0.87	0.94	1.23	2.23	2.64	0.74	0.79
cap84	1.22	1.23	3.01	7.31	0.65	0.67	5.84	6.71	0.76	0.82
cap91	0.73	0.97	1.07	1.36	0.48	0.73	1.08	1.26	0.48	0.50
cap92	0.65	0.66	0.64	0.75	0.44	0.44	1.41	1.58	0.49	0.53
cap93	0.60	0.61	1.45	1.67	0.40	0.40	1.26	1.52	0.43	0.44
cap94	0.55	0.57	1.28	1.46	0.55	0.56	0.83	0.99	0.44	0.47
cap101	0.97	12.44	0.94	1.15	0.96	1.22	1.00	1.21	0.46	0.48
cap102	0.43	0.43	0.71	0.80	0.43	0.43	0.60	0.70	0.47	0.50
cap103	11.59	12.81	16.06	20.72	3.18	4.20	3.52	4.56	0.39	0.41
cap104	0.13	0.14	0.49	0.59	0.14	0.14	0.32	0.41	0.35	0.36
cap111	48.85	53.89	29.16	33.77	10.14	12.27	7.80	9.32	1.05	1.16
cap112	5.82	6.02	16.37	18.34	4.26	5.03	8.64	9.52	1.03	1.13
cap113	105.23	117.60	21.45	23.09	23.77	28.94	12.37	13.98	1.04	1.17
cap114	6.62	6.94	89.58	103.94	8.63	9.61	60.14	70.03	0.98	1.50
cap121	3.97	5.94	17.99	19.20	4.67	6.00	18.48	22.15	0.90	0.97
cap122	2.60	2.98	4.76	5.45	2.02	2.08	16.08	19.34	0.85	1.04
cap123	2.70	2.75	4.19	4.79	1.86	2.27	22.11	25.21	0.82	0.89
cap124	2.79	2.85	3.67	4.54	1.67	2.02	21.06	24.16	0.72	0.84
cap131	5.24	35.33	16.33	18.03	7.13	8.52	34.37	38.65	0.85	0.94
cap132	2.27	2.44	4.53	5.10	1.52	1.55	5.47	6.05	0.70	0.85
cap133	22.12	22.82	50.44	59.07	8.03	13.60	20.91	23.99	0.68	0.85
cap134	0.30	0.52	4.20	5.13	0.53	0.54	3.16	3.69	0.62	0.65

Table 6.4: Bootstrapped 95% Confidence Intervals

Chapter 7

Conclusions and Future Research Directions

The aims of this chapter are to provide a series of conclusions, directed by the results of research obtained during this study; generate a direct response to the general research question presented at the end of Chapter 2; give an indication of future research directions that emerge directly from this thesis.

The objectives of this chapter are to summarise three key phases of this thesis' study, which are related to the development and application of ACO for facility location and identify any contributions made to existing knowledge. The overall integration of these three phases shall provide the evidence required to support the acceptance or rejection of the research hypothesis, that was defined in Chapter 3. To assist making this decision a series of specific research questions that were designed to test the research hypothesis are revisited and answered.

The penultimate section of this chapter presents some recommendations for future research directions, which are initially aimed at the CFLP and then at a much broader context of capacitated facility location. Whereas the ultimate section clearly details this thesis' contributions to knowledge and how they will be of benefit to future researchers using stochastic local search and metaheuristics as solution techniques to combinatorial optimisation problems.

7.1 Phase One – Study Rationale

The first phase of the thesis was aimed at providing a rationale for this study, which was motivated by the ingenuity and captivating behaviour of ants in their quest for food. The first chapter described concepts of how ACO was derived from the behavioural observations of foraging ants and then developed into a metaheuristic for combinatorial optimisation. Although ACO was originally designed to solve discrete optimisation problems, Dorigo and Stützle (2004) stated that it could also be used for various types of optimisation such as mixed-integer problems, but did not give any examples. So, the development of an ACO algorithm which could solve the CFLP would contribute to the existing knowledge base of ACO related applications, specifically the solution of a previously untried mixed-integer problem.

The second chapter presented a review of academic research materials for the facility location, with an aim to identify if a dominant metaheuristic technique for solving the CFLP existed. ReVelle and Eislet (2005) and ReVelle et al. (2008) indicated that there was some *ambiguity* associated within published research associated with the CFLP and suggested that the problem may be more difficult to solve than previously thought. The review conducted for this study supports those claims of ambiguity. Not only was there evidence of inconsistencies with the results of test problems used by various heuristic techniques, but there was also a lack of continuity associated with the selection of test problems which was compounded by a desire to solve *large randomly generated instances*. Although commonly used metaheuristic techniques such as *Simulated Annealing, Tabu*

Search and Genetic Algorithms had been applied to the CFLP, none had managed to solve all of the OR-Library test instances. At the start of this study, there was no published research evidence available on the application of ACO to the mixed-integer form of the CFLP. Clearly, there was a gap in the existing knowledge base associated with identifying if a dominant metaheuristic for solving the CFLP existed.

The third chapter derived a research hypothesis to be tested by providing answers to a series of research questions and justified an empirical methodology as a means of experimental study. A series of run-time analyses were proposed as a suitable method for dealing with stochastic optimisation techniques, because run-time distributions were unlikely to conform to standard statistical distributions. In order to obtain statistical metrics for comparing algorithmic performance the sampling with replacement method of boot strapping was deemed most appropriate, as it reduces the chance of introducing inference errors.

This phase justified a rationale for study, as an attempt to determine if ACO was a suitable mechanism to solve the CFLP and how well it compared to other metaheuristic applications. In the quest for the advancement of existing knowledge, contributions in this phase were made by: recognising that a robust and consistent empirical design was necessary to determine algorithmic behaviour of metaheuristic heuristic applications to the CFLP, by using a previously untried probabilistic run-time distribution technique which would provide ample data for thorough qualitative and quantitative analyses. This approach was necessary to eradicate those ambiguities and shortcomings of previously published experimental designs and results. These new experimental design features would be augmented and tested by the development of several proposed ACO solution strategies, which would contribute to existing knowledge.

7.2 Phase Two – ACO Research Design and Development for the CFLP

The second phase of this study was aimed at designing and developing several ACO algorithms, tailored to the CFLP, with an objective of solving the OR-Library test problems. A design feature for an ACO implementation is that the problem must be represented as a graph or network. Various representations of the CFLP and their merits were discussed in Chapters 4 and 5.

The application of ACO with a bipartite graphical representation of the CFLP was investigated in Chapter 4. Two ACO algorithms were implemented, *Ant System* and *Max-Min Ant System*, without a local search procedure and neither found any optimal solutions. The *Max-Min Ant System* algorithm derived solutions of a better quality than *Ant System* as it gave smaller relative errors. However, it was concluded that the use of a bipartite solution construction graph with ACO was inappropriate and unlikely to produce high quality solutions for the CFLP. Two reasons for this are firstly, it is very difficult for ACO to provide partial solutions that require customer demand to be supplied by more than one facility and secondly, the solution graph does not sufficiently exploit the structure of the CFLP.

To exploit the structure of the CFLP a construction graph that used a hybrid approach to primarily select what facilities to locate using ACO was investigated in Chapter 5. Facilities selected by ACO define a transportation problem, that can either be approximated or solved exactly. A known issue with this approach is that transportation problems are needed to be solved at every step in the ACO solution construction phase and at each step of any local search procedure for every iteration. The local search strategies adopted at this stage were based on identifying facilities that could be *dropped* or *swapped* in order to improve feasible solutions obtained from the ACO construction phase. A novel local search initialisation was implemented that made use of the pheromone levels to determine the order in which the search was to be executed. This was found to improve computational performance when combined with the *DROP-SWAP* strategy and thus a worthy contribution. Published research on heuristic methods for the CFLP advocate the use of approximation techniques for embedded transportation problems, as exact solution method are considered to be computationally inefficient, see Bornstein and Azlan (1998), Bornstein and Campelo (2004) and Arostegui et al. (2006). Both approximate and exact techniques were investigated empirically. Surprisingly, those algorithms that used an embedded exact transportation solver were not only superior in run-time, they also managed to find optimal solutions for all of the OR-Library test instances; which had only been achieved by one other metaheuristic, (Caserta and Quiñonez Rico, 2009). Although this was an exciting observation that contradicts the issues of prolonged run-times, any interpretation of these results required some caution as only a handful of experiments per instance were performed. Consequently, to make any general conclusions a thorough empirical investigation was necessary. However, the results certainly demonstrated that combining ACO and an exact method in the way described resulted in an algorithmic design that was able to solve all of the OR-Library problems.

Chapter 5 introduced a HCF algorithm as an attempt to improve on the $\mathcal{MM}$AS and avoid algorithmic stagnation, that was observed during previous experiments. The main difference between HCF and $\mathcal{MM}$AS is in the pheromone update phase; HCF uses an entropic update based on the colony size and its performance, whereas $\mathcal{MM}$AS only uses a best ant solution update. Also, a different type of local search strategy (*binary-flip*) was implemented and experiments using a colony of five ants were conducted. Again all of the OR-Library test problems were solved for a handful of experiments on each test problem and the results were favoured towards the use of HCF, but further experimentation was required to give a clearer insight.

Although further experimentation was needed, the ability to solve these mixed-integer problems using ant algorithms was previously unknown and thus this research phase provides contributions to ACO, metaheuristics and facility location knowledge bases. Only one other metaheuristic has claimed to be able to solve all of these problems, namely the CE method by Caserta and Quiñonez Rico (2007, 2009). This research and development phase had clearly reached a point where a thorough empirical evaluation was needed to identify any general characteristics of algorithmic performance and to determine which, if any, was the dominant method.

7.3 Phase Three – Critical Evaluation

The final stage of this study, presented in Chapter 6, was aimed at conducting an extensive series of run-time analyses with an objective to evaluate any prowess of the derived ACO algorithms against a contemporary CE technique. As previously stated, this type of analysis for measuring metaheuristic performance for the CFLP is not evident within published research. Four ACO algorithms that were designed in the previous research study phase, combined with two different local search mechanisms, were investigated; $\mathcal{MM}$AS with *DROP-SWAP*, $\mathcal{MM}$AS with 1-Flip, HCF with *DROP-SWAP* and HCF with 1-Flip. The analyses of run-time probability distributions (RTDs) for ACO and CE were proposed during the first phase of research. Sample sizes for the RTDs were derived empirically by examining two problem instances using $\mathcal{MM}$AS algorithm with *DROP-SWAP*, see Figure 6.1, because these problems displayed algorithmic incompleteness with initial experimentation detailed in Table 5.4 of Chapter 5.

RTDs were collated using median run-times consisting of 1000 runs per problem instance. A rationale for the selection of 37 problems from the OR-Library was detailed in Section 6.3 and their RTDs, for each of the five algorithms, are given in Appendix B and Appendix C. These RTDs confirmed that all five algorithms are capable of deriving high quality or optimal solutions. However, the CE method consistently failed to find complete sets of optimal solutions across all of the 37 problems. Although, this could be explained by algorithmic incompleteness due to experimental time limits (maximum of 1000 iterations or 10 minutes), the CE method always met its convergence criteria before these time limits were reached. Clearly indicating that its convergence criteria was unreliable, which had a detrimental effect on the algorithm's ability to find optimal solutions.

The $\mathcal{MM}$AS and HCF algorithms that used the *DROP-SWAP* local search strategies successfully solved each of the problem instances, for all 1000 runs. However $\mathcal{MM}$AS had a tendency to suffer from stagnation (flat sections in a RTD), but demonstrated an adaptability by moving away from sticking points to eventually finding optimal solutions. Interestingly, the use of a 1-Flip local search appeared to enhance the performance of $\mathcal{MM}$AS more than the HCF. Also the use of 1-Flip local search failed to obtain optimal solutions, for both $\mathcal{MM}$AS and HCF, to one problem instance *cap114*. This can be explained by algorithmic incompleteness or stagnation, but is more likely to be caused by a weakness in the 1-Flip local search design. Analyses of qualitative run-time distribution profiles revealed that ACO is superior to its contemporary opponent. However, it is not too clear which of the derived ACO algorithms is the most dominant as intersecting profiles were observed.

To help identify if there were any significant statistical differences between the ACO variants a succession of quantitative statistical analyses were conducted on the RTD data sets and their results were presented. Sampling distributions of median run-times were generated using a bootstrapping technique, these were used to construct 95% confidence intervals and are presented in Table 6.4. Graphical summaries of bootstrapped median run-times and their confidence intervals are displayed in Appendix D and Appendix E. Although no probabilistic dominant ACO algorithm prevailed, rankings of the boostrapped confidence intervals indicated that the HCF algorithm, combined with a *DROP-SWAP* local search strategy, was statistically the most reliable and efficient.

The third phase of this research study presented a soundly justified collection of empirical analyses that had been previously unconsidered for the CFLP. The RTD profiles that were generated will provide future researchers with a valuable source of evidence in terms identifying characteristic behaviour of the developed ACO algorithms and exemplars of how to conduct qualitative and quantitative run-time analyses not only for facility location, but also across a variety of metaheuristic applications. The whole of this third phase is beneficial and makes worthwhile contributions to ACO, metaheuristics and facility location knowledge bases.

7.4 Testing the Research Hypothesis

The first phase of this study included a research methodology chapter which presented five research questions that needed to be answered to test the proposed research hypothesis: *The ACO algorithm is a useful metaheuristic for solving capacitated facility location problems.* The aims and objectives of this section are to provide answers to the five questions and thus make a decision on whether to accept or reject the research hypothesis.

1. What is a suitable representation for the CFLP within an ACO modelling framework?

   Answer: If ACO is restricted to determine the state of facility decision variables, then the structure of the CFLP can be exploited as it reduces to a transportation problem. ACO can then make moves on a construction graph consisting only of facility locations, where the edges or links of the graph fully connect one facility to all other facilities by single links, which represent the possible pathways an ant take from one facility to another. The pheromone model then associates pheromone levels with facility locations and ants are guided to facilities with higher pheromone levels via an edge or link. This is different to the standard pathway constructions that are presented by Dorigo and Stützle (2004) as pheromones are placed on links or edges. The hybridisation model allows pheromones to be influenced by not only selecting which facilities to locate but also the solutions to any underlying transportation problems. This technique allows for a more directed search procedure than those using a standard graphical representation for the CFLP, such as a bipartite graph.

Thus, a suitable graphical representation is to use a hybrid one that combines ACO to select facility locations with an exact method to assign customers to the selected facilities, as depicted in Figures 5.1 and 5.2.

2. How well do any of the derived solution techniques perform on test problems available from the OR-Library, (Beasley, 1990)?

   Answer: A hybrid $\mathcal{MM}$AS with an embedded approximate transportation problem solution technique was developed and tested on instances from the OR-Library. The results obtained were encouraging, but the solution errors were not as good as some of those in existing published research (Beasley, 1993, Bornstein and Azlan, 1998, Bornstein and Campelo, 2004). However, when the approximate transportation solver was replaced with an exact solver from the COIN-OR distribution, (Lougee-Heimer, 2003), a different picture emerged as solutions were found in faster times and they matched the known optimums.

Two ACO variants of $\mathcal{MM}$AS and HCF were implemented, where each was tested with two local search techniques (*DROP-SWAP* and *1-Flip*), all of the OR-Library test problems were solved. Furthermore, extensive run-time analyses were carried out on 37 of the test problems, which involved 148,000 experiments with run-time limits of 1000 iterations or 10 minutes. Both $\mathcal{MM}$AS and HCF failed to completely solve only one test instance within these run-time limits (*cap114*) with an average error of 0.2%, in both cases the *1-Flip* had been used.

It is clear that hybrid ACO algorithms that make use of an exact transportation problem solver are able to solve all of the OR-Library test problems. However, there is a correlation between run-time performance and the type of local search technique being used. Further run-time efficiency could also be achieved by restricting the usage of the exact transportation solver to only newly generated solutions rather than all solutions that would include repeated ones.

3. Is there a dominant ACO solution technique?

Answer: Observations of qualitative RTD graph profiles were used to help determine if there was a dominant ACO solution technique. The first stage was to check if there was any evidence of a probabilistic dominant algorithm. An algorithm is said to have probabilistic dominance over another algorithm if their run-time distributions do not intersect with each other, the dominant algorithm has a profile furthest to the left or closest to the vertical axis of the RTD graph. Should the profiles intersect with each other then dominance can be determined in terms of statistical significance, i.e. does one algorithm perform significantly better than another algorithm based upon a statistical measure such as mean or median run-times.

The main issue that arises when dealing RTDs is that these empirically derived distributions do not necessarily conform to assumptions that are required to perform classical parametric and non-parametric statistical significance testing. A way of overcoming this is to use statistical bootstrapping sampling techniques on the RTDs of each problem instance to determine confidence intervals for the median run-times. The sampling distributions for the median run-times can then be examined graphically and numerically to determine if any significant differences are present.

The RTD analyses revealed that at present there is no probabilistic dominant solution method. However, ranking confidence intervals of median run-times for four different ACO algorithms, using statistical boostrapping, indicated that HCF combined with a *DROP-SWAP* local search procedure gave the most efficient and reliable results.

4. How well does ACO compare to the successful CE solution method, (Caserta and Quiñonez Rico, 2009), across a range of test problems available from the OR Library?

Answer: A further series of 1000 experiments for each of the 37 OR-Library instances were conducted to allow for RTD comparisons with the corresponding ACO RTD data sets. During the experiments the CE reached its termination convergence criteria very quickly, yet a close inspection of the data collected revealed variations in the solution quality. The CE algorithm was able to find optimal solutions but not all of the time, which was not indicated in the published research of Caserta and Quiñonez Rico (2009). In fact the CE algorithm failed to reach a 100% optimal solution hit across all of the 37 instances. This behaviour was not due to run-time limitations. Thus, there appears to be an issue with the convergence criteria for this algorithm.

Clearly then, from the results obtained ACO is currently superior to the CE method. RTD analysis revealed although CE was able to find optimal solutions it was unreliable as it had tendency to converge to early, which resulted in poor solutions across all problem instances and many of the experiments that were conducted.

5. Does ACO provide a suitable framework for solving the CFLP?

Answer: **Yes**, the evidence acquired and accumulated during this study, particularly the contributions made from RTD and statistical analyses, strongly support the use of an hybridised ACO solution technique.

Research conducted and answers to the above questions in this Ph.D. study, clearly support the acceptance of the research hypothesis. Thus, *ACO is a useful metaheuristic for solving CFLPs*. Furthermore, as CE and ACO are the only metaheuristics currently known to be able to solve all of the OR-Library test problems and ACO outperforms CE, then ACO is currently the most promising (hybrid) metaheuristic solution method available for solving the CFLP. Consequently, the use of ACO as a solution platform for solving CFLPs is certainly a worthy proposal and provides a contribution to existing knowledge in the field of facility location.

7.5 Future Research Directions

This section provides a focus for future research directions which is intially aimed at the CFLP, before moving onto broader areas associated with the integration and practical application of facility location within other academic fields. Future research of the use of ACO to solve the CFLP could take several pathways:

- Algorithmic enhancement and run-time optimisation of the four ACO algorithms developed within this thesis, which would include:
  - A reduction in the number of transportation problems that need to be solved.
  - The use of facility selection lists to avoid re-evaluating previously derived transportation problems.
  - Recognise that efficiency improvements would not change the RTD profiles, significant run-time improvements would result in a left-shift of the RTD.
- Development of a swarm based HCF algorithm that addresses iterative solution construction issues for larger problems (100+ facilities and 1000+ customers).
  - Use a swarm (large colony) of ants.

  - Each ant selects a subset of facilities.
  - Evaluate the transportation problem for each ant.
  - Only apply local search to a small selection of the most promising ant solutions.

- Design an ACO process that simultaneously decides which facilities to include and not include in a solution.
  - Graphical representation would a chain involve facility binary state variables.
  - Use a swarm of ants.
  - Purpose of each ant is to select the solution state of a facility.
  - Only apply local search to a small selection of the most promising ant solutions.

- Hybridisation of ACO with Lagrangean heuristics or Tabu Search are also possible avenues for exploration, see Chen and Ting (2008) and Katagiri et al. (2009), Yoshikawa and Otani (2010).

The broader area of facility location and its integration within in what was traditionally separate areas of study has recently been brought to the attention of the academic community. Some of the main areas of focus are: facility location and supply chain management (Melo et al., 2009); facility location and vehicle routing (Salhi and Nagy, 2009); facility location and layout (Domschke and Krispin, 1997); facility location and network design (Drezner and Wesolowsky, 2003). All of these areas can be represented by a customer-facility network and thus could benefit from the application of ACO.

There are many real world business and industrial applications that can be solved using ACO including; transportion and logistics, vehicle routing, scheduling and data mining. AntOptima is an innovative commercial company that was set up in 2001 by members of the Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA) in Switzerland, which aims to provide solutions to industry by using contemporary artificial intelligence techniques and thus creating a link between theory and practice. The company provides services that utilise other concepts besides ACO, such as Tabu Search, Genetic Algorithms, simulation, Bayesian and Credal Networks. Its scientific team consists of academics who have all made significant contributions to artificial intelligence and include Prof. Marco Dorigo, the inventor of ACO and research director of IRIDIA at the Université Libre de Bruxelles, and Prof. Luca Maria Gambardella of IDSIA who has derived state of the art ACO algorithms for a variety of hard combinatorial optimisation problems.

7.6 Contribution to Knowledge

All three phases of this thesis provide their own individual contributions to existing knowledge bases in three areas of academic study; facility location, metaheuristics and ACO. Firstly, the area of capacitated facility location, by solving all of the OR-Library test problems a series of run-time distributions were produced that will be a valuable resource for future researchers. Secondly, metaheuristics utilising techniques that can simplify solution design by the integration of exact solution methods in a hybrid way is an area not to be ignored, especially as everyday computing power continues to increase. Thirdly, ACO is shown to be a very flexible metaheuristic that can be adapted to solving mixed integer problems using hybridisation techniques. A summary of this thesis' contributions to existing knowledge are:

- A clear rationale for the study is given by recognising a gap in the current knowledge base associated with metaheuristics and capacitated facility location.

- A previously untried ant hybrid scheme, that incorporates an exact method within it, and provides the ACO and facility location knowledge bases with a new technique that successfully solves all of the capacitated facility location test problems available in the OR-Library. This hybridisation of ACO in also inputs to the currently emergent academic field of hybrid metaheuristics and would have been impractical or infeasible to implement ten years ago.
- Run-time analyses for measuring metaheuristic performance for the capacitated location problem has not been considered in this way before. The RTD data revealed that ACO is superior to its contemporary opponent and is currently the most reliable metheuristic available to solve the CFLP.
- The most reliable ACO algorithm for solving the CFLP is a *HCF* implementation.

The successful implementation of the COIN-OR (Computational Infrastructure for Operations Research), (Lougee-Heimer, 2003), distribution package to solve transportation problems played a pivotal role in the research and development of this thesis. Without an effective and efficient exact transportation problem solver, the production of RTDs would have relied on more traditional approximation techniques and the integrity of any conclusions drawn from algorithmic behaviour would have been compromised. This software distribution is open-source and freely available to the operations research community and is real contender against its expensive commercial counterparts.

Finally, a trend in presenting algorithmic solution methods to the CFLP is to gloss over the problems available in the OR-Library as though they are trivial and quickly go on to consider and report on larger randomly generated instances.

What this research demonstrates is that in order to test a stochastic algorithm you must first ascertain its behaviour on standard problems before considering larger and potentially more difficult ones. A stringent set of run-times analyses can provide a great deal of information about an algorithm's behaviour and thus has the ability to help a researcher design and develop more reliable and efficient algorithms. Consequently, any future research into the use of ACO, or any other metheuristic, for capacitated facility location problems would benefit from the works undertaken during this study. A listing of research output associated with this study is given in Appendix A, all of which are available on request.

Bibliography

Adlakha, V. and K. Kowlaski (2004). A simple algorithm for the source-induced fixed-charge transportation problem. *J Opl Res Soc 55*(12), 1275–1280.

Agar, M. and S. Salhi (1998). Lagrangean heuristics applied to variety of large capacitated plant location problems. *J Opl Res Soc 49*(10), 1072–1084.

Ahuja, R., O. Ergun, J. Orlin, and A. Punnan (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics 123*(13), 75–102.

Ahuja, R., J. Orlin, S. Pallottino, M. Scaparra, and M. Scutellà (2004). A multi-exchange heuristic for the single-source capacitated facility location problem. *Mgmt Sci 50*(6), 749–760.

Al-khedhairi, A. (2008). Simulated annealing metaheuristic for solving p-median problem. *Int. J. Contemp. Math. Sciences 3*(28), 1357–1365.

Alp, O., E. Erkut, and Z. Drezner (2004). An efficient genetic algorithm for the p-median problem. *Annals of Operations Research 122*(1-4), 21–42.

Altlnel, I., E. Durmaz, N. Aras, and K. Özklsaclk (2009). A location–allocation heuristic for the capacitated multi-facility weber problem with probabilistic customer locations. *European Jounal of Operational Research 198*(3), 790–799.

Arostegui, M. J., S. Kadipasaogul, and B. Khumawala (2006). An empirical comparison of tabu search, simulated annealing and genetic algorithms for facilities location problems. *International Journal of Production Economics* (103), 742–754.

Aydin, M. and T. Fogarty (2004). A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems. *Journal of Heuristics 10*(3), 269–292.

Baker, B. (1982). Linear relaxation of the capacitated warehouse location problem. *J Opl Res Soc 33*, 475–479.

Balinski, M. (1965). Integer programming: Methods, uses, computation. *Mgmt Sci* (12), 253–276.

Balinski, M. (1966). *On Finding Integer Solutions to Linear Programs*. Mathematica. New Jersey: Princeton.

Barahona, F. and F. Chudak (2005). Near-optimal solutions to large scale facility location problems. *Discrete Optimization 2*(1), 35–50.

Barceló, J. and J. Casanovas (1984). A heuristic lagrangean algorithm for the capacitated plant location problem. *European Journal of Operational Research* (15), 212–226.

Barr, R., B. Golden, J. Kelly, M. Resende, and J. W. Stewart (1995). designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics* (1), 9–32.

Beasley, J. (1982). A note on solving large p-median problems. *European Jounal of Operational Research 21*(2), 270–273.

Beasley, J. (1988). An algorithm for soving large capacitated warehouse location problems. *European Journal of Operational Research* (33), 314–325.

Beasley, J. (1990). Or-library: Distributing test problems by electronic mail. In *Operations Research Proceedings*, Volume 41, pp. 1069–107. Springer.

Beasley, J. (1993). Lagrangean heuristcs for location problems. *Eur J Opl Res 65*, 383–399.

Beckers, R., J.-L. Deneubourg, and NewAuthor3 (1992). Trials and u-turns in the selection of a path by the ant lasius niger. *Journal of Theoretical Biology* (159), 397–415.

Bischoff, M. and K. Dächert (2007). Allocation search methods for a generalized class of location-allocation problems. *European Jounal of Operational Research*.

Blum, C. (2004, February). *Theoretical and Practical Aspects of Ant Colony Optimization*. Ph. D. thesis, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Blum, C., M. J. B. Aguliera, A. Roli, and M. Sampels (Eds.) (2008). *Hybrid Metaheuristics: An Emerging Approach to Optimization*. Studies in Computational Intelligence 114. Springer.

Blum, C. and M. Dorigo (2004). The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics - Part B 34*(2), 1161–1172.

Blum, C., A. Roli, and M. Dorigo (2001). HC–ACO: The hyper-cube framework for Ant Colony Optimization. In *Proceedings of MIC'2001 – Metaheuristics International Conference*, Volume 2, Porto, Portugal, pp. 399–403.

Bonabeau, E., M. Dorigo, and G. Theraulaz (1999). *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford University Press.

Bornstein, C. and H. Azlan (1998). The use of reduction tests and simulated annealing for the capacitated plant location problem. *Loc Sci 6*, 67–81.

Bornstein, C. and M. Campelo (2004). An add/drop procedure for the capacitated plant location problem. *Pesquisa Operacional 24*(1), 151–162.

Bramel, J. and D. Simchi-Levi (1995). A location-based heuristic for general routing problems. *Operations Research 43*, 649–660.

Bryman, A. and E. Bell (2007). *Business Research Methods* (2nd ed.). Oxford University Press.

Canovas, L., S. Garcia, M. Labbe, and A. Marin (2007). A strengthened formulation for the simple plant location problem with order. *Operations Research Letters 35*(2), 141–150.

Caserta, M. and E. Quiñonez Rico (2007, June). A cross entropy-based metaheuristic algorithm for large scale facility location problems. In *MIC - VII Metaheuristic International Conference*.

Caserta, M. and E. Quiñonez Rico (2009). A cross entropy-based metaheuristic algorithm for large-scale capacitated facility location problems. *J Opl Res Soc 60*(10), 1439–1448.

Chen, C. and C. Ting (2006). Applying multiple ant colony system to solve single source capacitated facility location problem. In M. Dorigo, L. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle (Eds.), *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006*, Volume 4150 of *Lecture Notes in Computer Science*, Berlin, Germany, pp. 508–509. Springer Verlag.

Chen, C. and C. Ting (2008). Combining lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem. *Transportation Research Part E 44*(6).

Christofides, N. and J. Beasley (1982). A tree search algorithm for the p-median problem. *European Jounal of Operational Research 10*(2), 196–204.

Christofides, N. and J. Beasley (1983). Extensions to a lagrangean relaxation approach for the capacitated warehouse location problem. *European Journal of Operational Research* (12), 19–28.

Church, R. and A. Murray (2008). *Business Site Selection, Location Analysis and GIS*. New York: John Wiley and Sons, Inc.

Colorni, A., M. Dorigo, and V. Maniezzo (1992). Distributed optimization by ant colonies. In *Proceedings of ECAL'91- First European Conference on Artificial Life*, pp. 134–142. Elsevier Publishing.

Correa, E., M. Steiner, A. Freitas, and C. Carnieri (2004). A genetic algorithm for solving a capacitated p-median problem. *Numerical Algorithms 35*(2-4), 373–388.

Cortinhal, M. and M. Captivo (2003). Upper and lower bounds for the single source capacitated location problem. *European Jounal of Operational Research* (151), 333–351.

Daskin, M. (1995). *Network and Discrete Location: Models, Algorithms and Applications*. New York: John Wiley and Sons, Inc.

Daskin, M. (2008). What you should know about location modelling. *Naval Res Logis* (55), 283–294.

Daskin, M. and S. Melkote (2001). Capacitated facility location/network design problems. *European Jounal of Operational Research* (129), 481–495.

Deneubourg, J.-L., S. Aron, S. Goss, and J. L. Pasteels (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behaviour* (3), 159–168.

Díaz, J. A. (2001). *Algorithmic Approaches for the Single Source Capacitated Location Problem*. Ph. D. thesis, Universitat Polytechnica de Catalunya, Barcelona, Spain.

Díaz, J. A. and E. Fernádez (2002). A branch and price algorithm for the single-source capacitated plant location problem. *J Opl Res Soc 53*(7), 728–740.

Domschke, W. and G. Krispin (1997). Location and layout planning. *OR Spectrum 19*, 181–194. 10.1007/BF01545586.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms [in Italian]*. Ph. D. thesis, Departimento di Elettronica, Politecnico di Milano, Milan.

Dorigo, M., M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. F. T. Winfield (2008). *Ant Colony Optimization and Swarm Intelligence, 6th International Conference, ANTS 2008*, Volume 5217 of *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag.

Dorigo, M. and C. Blum (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science* (344), 243–278.

Dorigo, M. and L. Gambardella (1997a). Ant colonies for the travelling salesman problem. *BioSystems 43*(2), 73–81.

Dorigo, M. and L. Gambardella (1997b). Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation 1*(1), 53–66.

Dorigo, M., L. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle (Eds.) (2006). *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006*, Volume 4150 of *Lecture Notes in Computer Science*. Berlin, Germany: Springer Verlag.

Dorigo, M. and K. Socha (2006, April). An introduction to ant colony optimization. Technical Report TR/IRIDIA/2006-010, Université Libre de Bruxelles.

Dorigo, M. and T. Stützle (2004). *Ant Colony Optimization*. Cambridge, MA: The MIT Press.

Dorigo, M., M. Zlochin, N. Meuleau, and M. Birattari (2002). Updating ACO pheromones using stochastic gradient ascent and cross-entropy methods. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. R. Raidl (Eds.), *Applications of Evolutionary Computing: EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTIM/EvoPLAN*, Volume 2279 of *Lecture Notes in Computer Science*, pp. 21–30. Berlin, Germany: Springer-Verlag.

Dréo, J., A. Pétrowski, P. Siarr, and E. Taillard (2006). *Metaheuristics for Hard Optimization*. Berlin, Germany: Springer-Verlag.

Drezner, Z. (Ed.) (1995). *Facility Location. A Survey of Applications and Methods*. New York: Springer.

Drezner, Z., K. Klamroth, A. Schobel, and G. Wesolowsky (2001). *The Weber Problem*, pp. 1–36. Berlin, Germany: Springer-Verlag.

Drezner, Z. and G. O. Wesolowsky (2003). Network design: selection and design of links and facility location. *Transportation Research Part A: Policy and Practice 37*(3), 241–256.

Efron, B. and T. DiCiccio (1996). Boostrap confidence intervals. *Statistical Science 11*(3), 189–228.

Erlenkotter, D. (1978). A dual-based procedure for uncapacitated facility location. *Operations Research* (26), 992–1009.

Fathali, J. (2006). A genetic algorithm for the p-median problem with pos/neg weights. *Applied mathematics and Computation 183*(2), 1017–1083.

Fathali, J., H. Kakhki, and R. Burkard (2006). An ant colony algorithm for the pos/neg weighted p-median problem. *Central European Journal of Operations Research 14*(3), 229–246.

Filho, V. and R. Galváo (1998). A tabu search heuristic for the concentrator location problem. *Location Science* (6), 189–209.

Fleszar, K. and K. Hindi (2008). An effective vns for the capacitated p-median problem. *European Jounal of Operational Research 191*(3), 612–622.

França, P., N. M. Sosa, and V. Pureza (2006). An adaptive tabu search algorithm for the capacitated clustering problem. *International Transactions in Operational Research 6*(6), 665–678.

Gambardella, L., E. Taillard, and G. Agazzi (1999). *MACS-VRPTW: A Multiple Ant Colony System For Vehicle Routing Problems With Time Windows*, pp. 63–76. McGraw-Hill.

Garey, M. R. and D. S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Ghoseiri, K. and S. Ghannadpour (2009). An efficient heuristic method for capacitated p-median problem. *International Journal of Management Science and Engineering Science 4*(1), 72–80.

Ghosh, D. (2003). Neighborhood search heuristics for the uncapacitated facility location problem. *European Journal of Operational Research* (150), 150–162.

Goldberg, A. (1997). An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms* (22), 1–29.

Goldengorin, B., D. Ghosh, and G. Sierksma (2004). Branch and peg algorithms for the simple plant location problem. *Computers and Operations Research 31*(2), 241–255.

Guignard, M. (1988). A lagrangean dual ascent method for simple plant location problems. *European Jounal of Operational Research 35*, 193–200.

Guner, A. and M. Sevkli (2008). A discrete particle swarm optimization algorithm for uncapacitated facility location problem. *Journal of Artificial Evoloution and Applications 2008*(Article ID 861512), 9 pages.

Hakimi, S. (1964). Optimum locations of switching centres and the absolute centres and medians of a graph. *Operations Research* (12), 450–459.

Hakimi, S. (1965). Optimum locations of switching centres in a communications network and some graph related theoretical problems. *Operations Research* (13), 462–475.

Hillier, F. and G. Lieberman (2005). *Introduction to Operations Research* (8th ed.). McGraw-Hill.

Hindi, K. and K. Pieńkosz (1999). Efficient solution of large single-source, capacitated plant location problems. *J Opl Res Soc 50*(3), 268–274.

Hoefer, M. (2003). Experimental comparision of heuristic and approximation algorithms for uncapacitated facility location. In *Proceedings of the Second International Workshop on Experimental and Efficient Algorithms (WEA)*, Volume 2647 of *Lecture Notes in Computer Science*, Berlin, Germany, pp. 165–178. Springer-Verlag.

Hölldobler, B. and E. Wilson (1994). *Journey to the Ants*. A story of scientific exploration. Cambridge, Massachusetts: The Belknap Press.

Holmberg, K., D. Ronnqvist, and D. Yuan (1999). An exact algorithm for the capacitated facility location problem with single sourcing. *Eur J Opl Res 113*, 544–559.

Hoos, H. and T. Stützle (2005). *Stochastic Local Search Foundations and Applications*. San Francisco, CA 94111: Morgan Kaufman.

Hotelling, H. (1929). Stability in competition. *Ecomonic Journal* (39), 41–57.

Jaramillo, J., J. Bhadur, and R. Batta (2002). On the use of genetic algorithms to solve location problems. *Computers and Operations Research* (29), 761–779.

JI, J.-Z., H.-X. ZHANG, R.-B. HU, and C.-N. LIU (2009). A bayesian network learning algorithm based on independence test and ant colony optimization. *Acta Automatica Sinica 35*(3), 281–288.

Jourdan, L., M. Basseur, and E.-G. Talbi (2009). Hybridizing exact methods and metaheuristics: A taxomony. *European Jounal of Operational Research* (199), 620–629.

Kariv, O. and S. Hakimi (1979a). An algorithmic approach to network location problems, part i: The p-centres. *SIAM Journal of Applied Mathematics* (37), 513–538.

Kariv, O. and S. Hakimi (1979b). An algorithmic approach to network location problems, part ii: The p-median. *SIAM Journal of Applied Mathematics* (37), 539–560.

Karup, J. and P. Pruzan (1983). The simple plant location problem: Survey and synthesis. *European Jounal of Operational Research 12*, 36–81.

Katagiri, H., T. Hayashida, I. Nishizaki, and J. Ishimatsu (2009). A hybrid algorithm based on tabu search and ant colony optimization for minimum spanning tree problems. In V. Torra, Y. Narukawa, and M. Inuiguchi (Eds.), *Modeling Decisions for Artificial Intelligence*, Volume 5861 of *Lecture Notes in Computer Science*, pp. 315–326. Springer Berlin.

Kaveh, A. and S. Shojaee (2008). Optimal domain decomposition via p-median methodology using aco and hybrid acga. *Finite Elements in Analysis and Design 44*(8), 505–512.

Kirca, Ö. and A. Satir (1990). A heuristic for obtaining an initial solution for the transportation problem. *J Opl Res Soc 41*(9), 865–867.

Klose, A. and A. Drexl (2004). Facility location models for distribution system design. *European Jounal of Operational Research*.

Krishnaswamy, K. N., A. I. Sivakumar, and M. Mathirajan (2009). *Management Research Methodology: Integration of Methods and Techniques* (3rd ed.). Palgrave.

Kuehn, A. and M. Hamburger (1963). A heuristic program for locating warehouses. *Mgmt Sci* (9), 643–666.

Kumweang, K. and R. Kawtummachai (2005). Solving a sscflp in a supply chain with aco. *Suranaree J Sci Technol 12*(1), 28–38.

Lawler, E. (1963). The quadratic assignment problem. *Mgmt Sci* (9), 586–599.

Levanova, T. and M. Loresh (2004). Algorithms of ant system and simulated annealing for the p-median problem. *Automation and Remote Control 65*(3), 431–438.

Levanova, T. and M. Loresh (2006). Ant colony optimization algorithm for the capacitated plant location problem. In *12th IFAC Symposium on Information Control Problems in Manufacturing - INCOM 2006*, Volume 3, pp. 423–428.

Lorena, L. and E. Senne (2003). Local search heuristics for capacitated p-median problems. *Networks & Spatial Economics 3*(4), 407–419.

Lorena, L. and E. Senne (2004). A column generation approach to capacitated p-median problems. *Computers and Operations Research 31*(6), 863–876.

Lougee-Heimer, R. (2003). The common optimization interface for operations research. *IBM Journal of Research and Development 47*(1), 57–66.

Lourenço, H. and D. Serra (2002). Adaptive search heuristics for the generalized assignment problem. *Math & Soft Comp 9*, 209–234.

Love, R., J. Morris, and G. Wesolowsky (1988). *Facilities Location: Models and Methods*. New York: North Holland.

Lu, Z., N. Bostel, and P. Dejax (2005). *Simple Plant Location Problem with Reverse Flows*, Volume 94 of *Applied Optimization*, pp. 151–166. Berlin, Germany: Springer-Verlag.

Mathirajan, M. and B. Meenakshi (2004). Experimental analysis of some variants of vogel's approximation method. *Asia-Pacific Journal of Operational Research 21*(4), 447–462.

McGill, R., J. Tukey, and W. Larsen (1978). Variations of box plots. *The American Statistician 32*(1), 12–16.

Melo, M., S. Nickel, and F. S. da Gama (2009). Facility location and supply chain management - a review. *European Journal of Operational Research 196*(2), 401–412.

Michel, L. and P. Hentenryck (2004). A simple tabu search for warehouse location. *European Jounal of Operational Research 157*(3), 576–591.

Mirchandani, P. and R. Francis (Eds.) (1990). *Discrete Location Theory*. New York: John Wiley and Sons, Inc.

Mladenovic, N., J. Brimberg, P. Hansen, and J. Moreno-Perez (2007, June). The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research 127*(3), 927–939.

Montemanni, R., L. Gambardella, A. Rizzoli, and A. Donati (2005). Ant colony system a dynamic vehicle routing problem. *Journal of Combinatorial Optimization* (10), 327–343.

Neumann, F., D. Sudholt, and C. Witt (2008). Rigorous analyses for the combination of ant colony optimization and local search. In *Ant Colony Optimization and Swarm Intelligence, 6th International Conference, ANTS 2008*, Volume 5217 of *Lecture Notes in Computer Science*, pp. 132–143.

Neumann, F. and C. Witt (2009). Runtime analysis of a simple ant colony optimization algorithm. *Algorithmica 54*(2), 243–255.

Olivetti, F., F. V. Zuben, and L. N. de Castro (2005). Max min ant system and capacitated p-medians: Extensions and improved solutions. *Informatica 29*, 163–171.

Osman, I. and S. Ahmadi (2007). Guided construction search metaheuristics for the capacitated p-median problem with single source constraint. *J Opl Res Soc* (58), 100–114.

Owen, S. and M. Daskin (1998). Strategic facility location: A review. *European Jounal of Operational Research* (111), 423–447.

Pang, C.-Y., W. Hu, X. Li, and B.-Q. Hu (2009, July). Apply local clustering method to improve the running speed of ant colony optimization.

Reese, J. (2006). Methods for solving the p-median problem: An annotated bibliography. *Networks 48*(3), 125–142.

Resende, M. and R. Werneck (2004). A hybrid heuristic for the p-median problem. *Journal of Heuristics 10*(1), 59–88.

Resende, M. and R. Werneck (2006). A hybrid multistart heuristic for the uncapacitated facility location problem. *European Jounal of Operational Research 174*(1), 54–68.

ReVelle, C. (1997). A perspective on location science. *Location Science 5*(1), 3–13.

ReVelle, C. and H. Eislet (2005). Location analysis: A synthesis and survey. *European Jounal of Operational Research* (165), 1–19.

ReVelle, C., H. Eislet, and M. Daskin (2008). A bibliography for some fundamental problem categories in discrete location science. *European Jounal of Operational Research* (184), 817–848.

Rolland, E., D. Schilling, and J. Current (1997). An efficient tabu search procedure for the p-median problem. *European Jounal of Operational Research 96*(2), 329–342.

Rönnqvist, M., S. Tragantalerngsak, and J. Holt (1999). A repeated matching heuristic for the single-sourced capacitated facility location problem. *European Journal of Operational Research* (116), 51–68.

Rubinstein, R. (1997). Optimization of computer simulation models with rare events. *European Jounal of Operational Research* (99), 89–112.

Rubinstein, R. (1999). The simulated entropy method for combinatorial and continuos optimization. *Methodology and Computing in Applied Probability* (2), 127–190.

Rubinstein, R. (2001). *Combinatorial Optimization, Cross-Entropy, Ants and Rare Events*, pp. 304–358. Stochastic optimization: Algorithms and Applications. Kluwer.

Rubinstein, R. (2002). The cross-entropy method and rare-events for maximal cut and bipartition problems. *ACM Transactions on Modelling and Computer Simulation 12*(1), 27–53.

Rubinstein, R. and D. Krose (2004). *The Cross-Entropy Method: a Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*. Springer-Verlag.

Sa, G. (1969). Branch and bound and approximate solutions to the capacitated plant location problem. *Operations Research* (17), 1005–1016.

Salhi, S. (2002). Defining tabu list size and aspiration criterion within tabu search methods. *Computers and Operations Research* (29), 67–86.

Salhi, S. and R. Atkinson (1995). Subdrop: A modified drop heuristic for location problems. *Location Science 3*(4), 267–273.

Salhi, S. and G. Nagy (2009). Local improvement in planar facility location using vehicle routing. *Annals of Operations Research 167*, 287–296. 10.1007/s10479-007-0223-z.

Saunders, M., P. Lewis, and A. Thornhill (2007). *Research Methods for Business Students* (4th ed.). Prentice Hall.

Scheuerer, S. and R. Wendolsky (2006). A scatter search heuristic for the capacitated clustering problem. *European Jounal of Operational Research 169*(2), 533–547.

Sevkli, M. and A. Guner (2006). A continuous particle swarm optimization algorithm for uncapacitated facility location problem. In M. Dorigo, L. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle (Eds.), *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006*, Volume 4150 of *Lecture Notes in Computer Science*, Berlin, Germany, pp. 316–323. Springer Verlag.

Smith, H., G. Laporte, and P. Harper (2009). Locational analysis: Highlights of growth to maturity. *J Opl Res Soc 60*(1), S140–S148.

Sörensen, K. (2008, May). Investigation of practical, robust and flexible decisions for facility location problems using tabu search and simulation. *J Opl Res Soc 59*(5), 624–636.

Sridharan, R. (1993). A lagrangean heuristic for the capacitated plant location problem. *European Journal of Operational Research* (66), 305–312.

Sridharan, R. (1995). Invited review: The capaciated plant location problem. *Eur J Opl Res 87*, 203–213.

Stützle, T. (1999). *Local Search Algorithms for Combinatorial Problems: Analysis, Improvements and New Applications*, Volume 220. Germany, Infix: Sankt Augustin.

Stützle, T. and H. Hoos (1997). The *max-min* ant system and local search for the travelling salesman problem. In S. Voss, S. Martello, I. Osman, and C. Roucairol (Eds.), *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, Piscataway, NJ, pp. 309–314. IEEE Press.

Stützle, T. and H. Hoos (2000). The *max-min* ant system. *Fut Gen Com Sys 16*(8), 889–914.

Sun, M. (2006). Solving the uncapacitated facility location problem using tabu search. *Computers and Operations Research 33*(9), 2563–2589.

Taha, H. (2006). *Operations Research: An Introduction* (8th ed.). Prentice Hall.

Tarrent, F. and D. Bridge (2005). When ants attack: Ant algorithms for constraint satisfaction problems. *Art Int Rev 24*, 455–476.

Theraulaz, G. and E. Bonabeau (1999). A brief history of stigmergy. *Artificial Life 5*, 97–116.

Venables, H., H. Chen, and A. Moscardini (2005). The fixed charge capacitated location problem – an ant colony optimization approach. In *Tenth International Symposium on Locational Decisions – Abstracts*, pp. 247–248.

Venables, H. and A. Moscardini (2006). An adaptive search heuristic for the capacitated fixed charge facility location problem. In M. Dorigo, L. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle (Eds.), *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006*, Volume 4150 of *Lecture Notes in Computer Science*, Berlin, Germany, pp. 348–355. Springer Verlag.

Venables, H. and A. Moscardini (2007a). An ant based heuristic for the capacitated fixed charge location problem. Unpublished working paper.

Venables, H. and A. Moscardini (2007b). The fixed charge capacitated location problem: – an ant based solution procedure. Conference Paper. 22nd European Conference on Operational Research, available on request.

Venables, H. and A. Moscardini (2008). Ant based heuristics for the capcitated fixed charge location problem. In M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. F. T. Winfield (Eds.), *Ant Colony Optimization and Swarm Intelligence, 6th International Conference, ANTS 2008*, Volume 5217 of *Lecture Notes in Computer Science*, pp. 234–242.

Venables, H. and A. Moscardini (2010, July). Evaluation of anant based hybrid metaheuristic used to solve the capacitated facility location problem. Conference Paper. 24th European Conference on Operational Research, available on request.

Weber, A. (1909). Uber den standort der industrien.

Wood, M. (2005). Bootstrapped confidence intervals as an approach to statistical inference. *Organizational Research Methods 8*(4), 454–470.

Xu, Y., M.-H. Lim, Y.-S. Ong, and J. Tang (2006). A ga-aco-local search hybrid algorithm for solving quadratic assignment problem. In M. Cattolico (Ed.), *GECCO*, pp. 599–606. ACM.

Yoshikawa, M. and K. Otani (2010). Ant colony optimization routing algorithm with tabu search. In *Proceedings of The International MultiConference of Engineers and Computer Scientists 2010*, Volume III, pp. 2104–2107.

Zanjirani, F. and M. Hekmmatfar (Eds.) (2009). *Facility Location: Concepts,* *Models. Algorithms and Case Studies*. Contributions to Management Science. Berlin, Germany: Springer-Verlag.

Appendix A

Research Output

1. A conference paper was presented at ISOLDEX, Seville, Spain, Venables et al. (2005).
2. A presentation, research poster and research publication were outcomes of an international conference on Ant Conoly Optimisation at the Universit Libre de Bruxelles, Brussels, Belgium, Venables and Moscardini (2006).
3. A working paper on Max-Min Ant System with the use of approximation of transportation problems was completed as part of a research deliverable, Venables and Moscardini (2007a).
4. A conference paper on the use of Ant Colony System, was prepared and presented at an international conference EURO XXII at the University of Economics Prague, Czech Republic, Venables and Moscardini (2007b).
5. A presentation, research poster and research publication were outcomes of an international conference on Ant Conoly Optimisation at the Universit Libre de Bruxelles, Brussels, Belgium, Venables and Moscardini (2008).
6. A conference paper on the evaluation of ACO as a solution technique for the CFLP, was prepared and presented at an international conference EURO

XXIV at at the Faculty of Sciences of the University of Lisbon, FCUL, Venables and Moscardini (2010).

Appendix B

Algorithmic Solution Quality: Empirical Run-Time Distributions

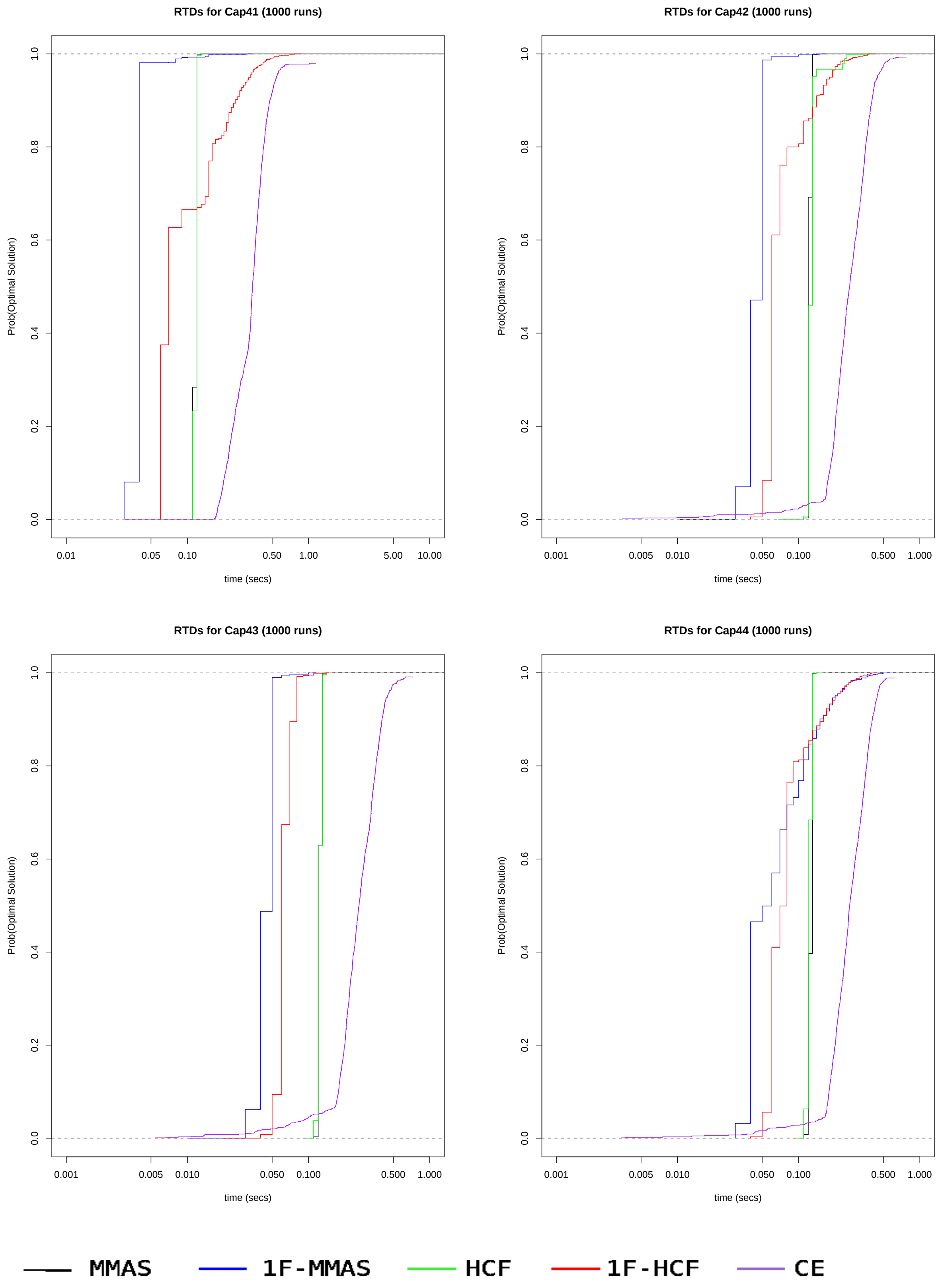

RTDs for Cap41 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
0.01
0.05
0.10
0.50
1.00
5.00
10.00
time (secs)
RTDs for Cap42 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
0.001
0.005
0.010
0.050
0.100
0.500
1.000
time (secs)
RTDs for Cap43 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
0.001
0.005
0.010
0.050
0.100
0.500
1.000
time (secs)
RTDs for Cap44 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
0.001
0.005
0.010
0.050
0.100
0.500
1.000
time (secs)
MMAS
1F-MMAS
HCF
1F-HCF
CE

Figure B.1

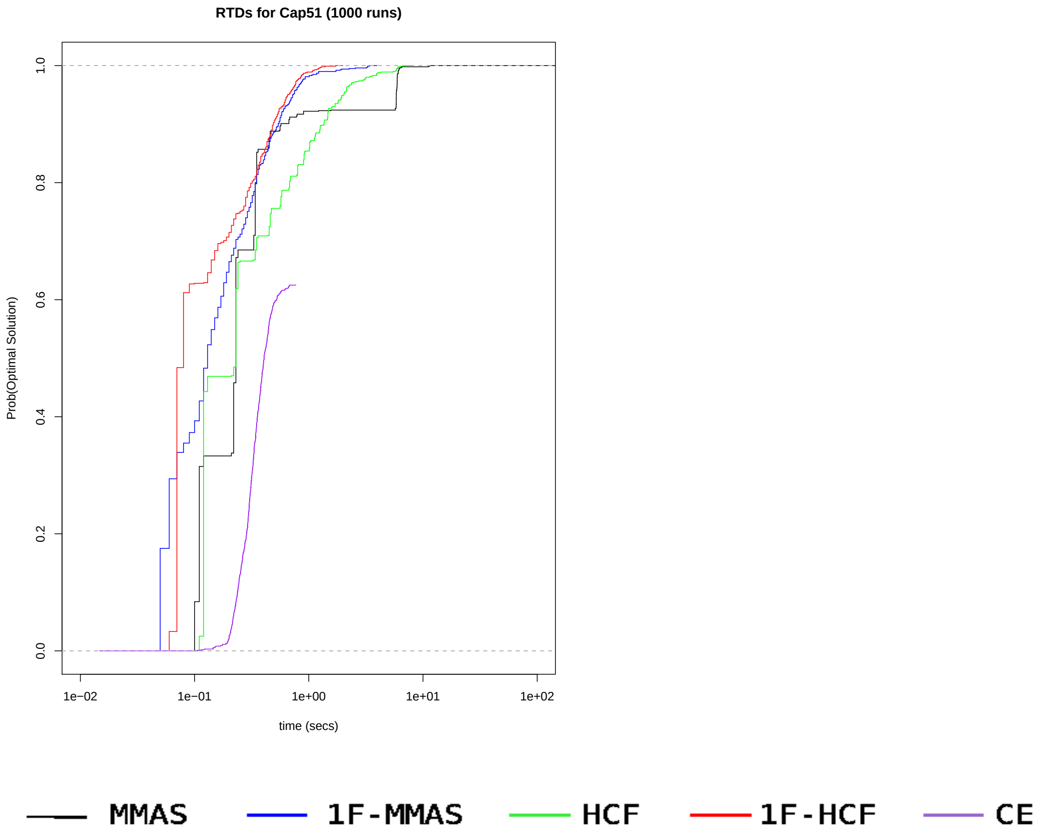

RTDs for Cap51 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
1e−02
1e−01
1e+00
1e+01
1e+02
time (secs)
MMAS
1F-MMAS
HCF
1F-HCF
CE

Figure B.2

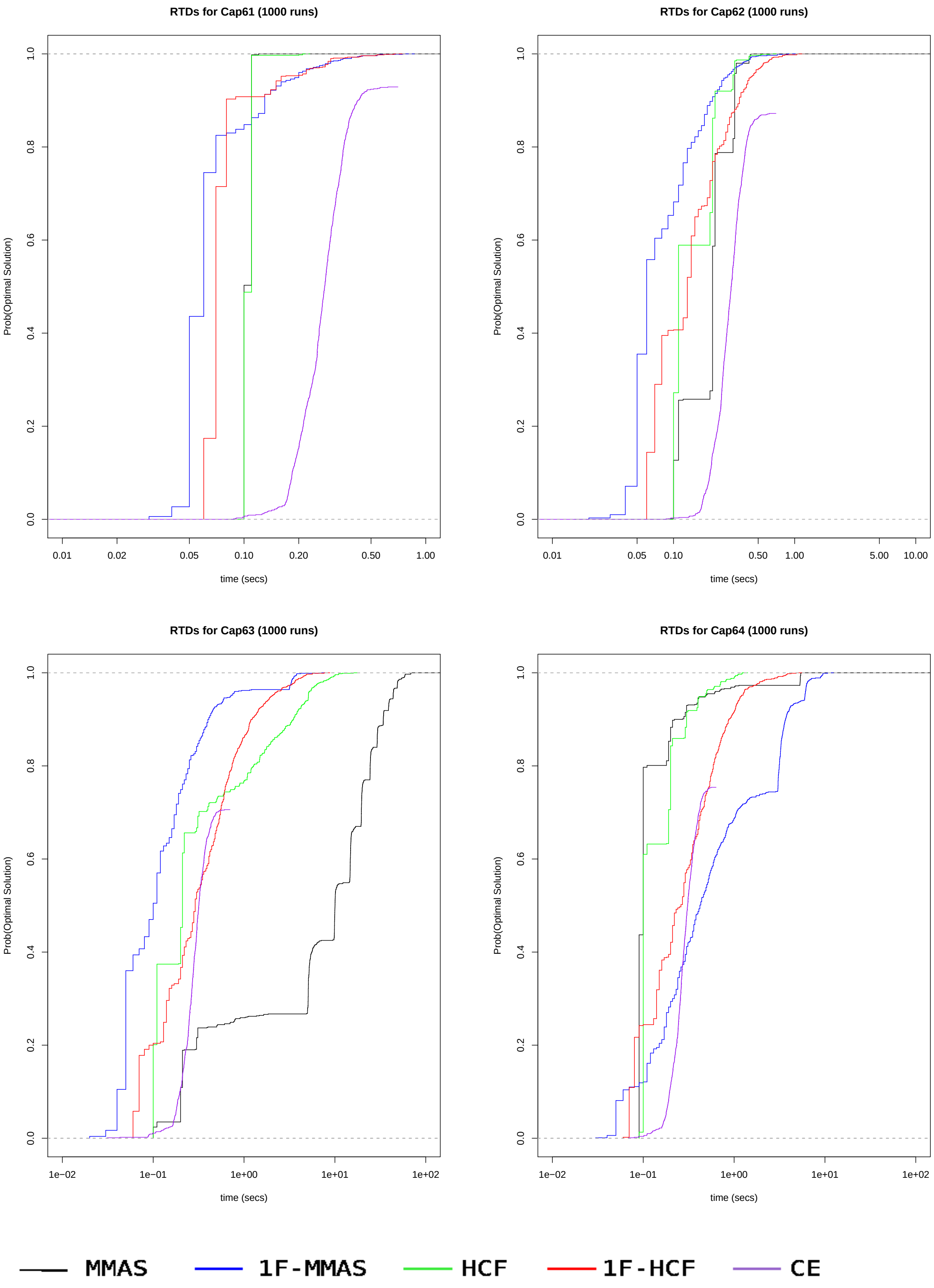

RTDs for Cap61 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
0.01
0.02
0.05
0.10
0.20
0.50
1.00
time (secs)
RTDs for Cap62 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
0.01
0.05
0.10
0.50
1.00
5.00
10.00
time (secs)
RTDs for Cap63 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
1e−02
1e−01
1e+00
1e+01
1e+02
time (secs)
RTDs for Cap64 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
1e−02
1e−01
1e+00
1e+01
1e+02
time (secs)
MMAS
1F-MMAS
HCF
1F-HCF
CE

Figure B.3

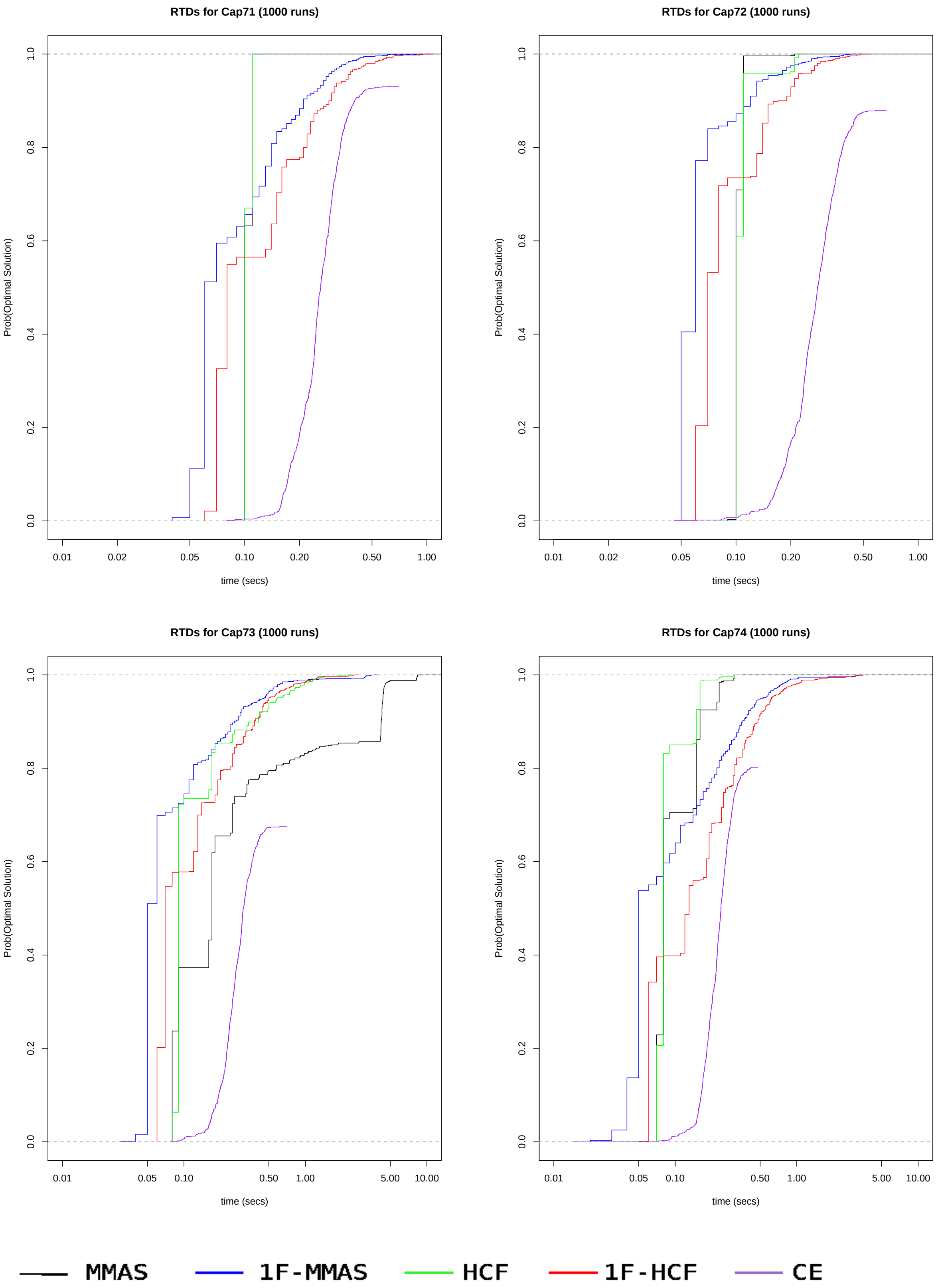

RTDs for Cap71 (1000 runs)
RTDs for Cap72 (1000 runs)
RTDs for Cap73 (1000 runs)
RTDs for Cap74 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
time (secs)
0.01
0.02
0.05
0.10
0.20
0.50
1.00
5.00
10.00
MMAS
1F-MMAS
HCF
1F-HCF
CE

Figure B.4

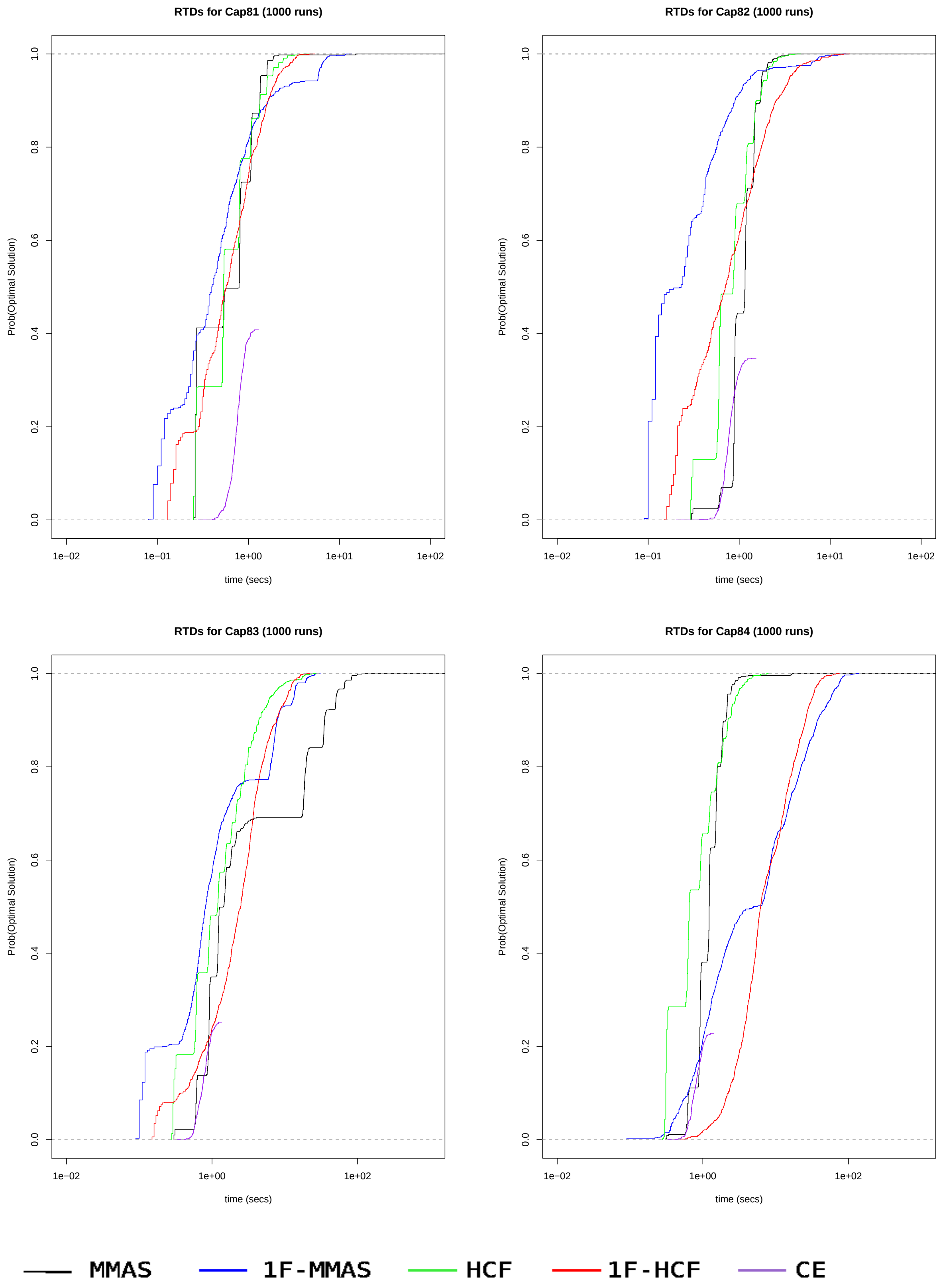

RTDs for Cap81 (1000 runs)
RTDs for Cap82 (1000 runs)
RTDs for Cap83 (1000 runs)
RTDs for Cap84 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
1e−02
1e−01
1e+00
1e+01
1e+02
time (secs)
MMAS
1F-MMAS
HCF
1F-HCF
CE

Figure B.5

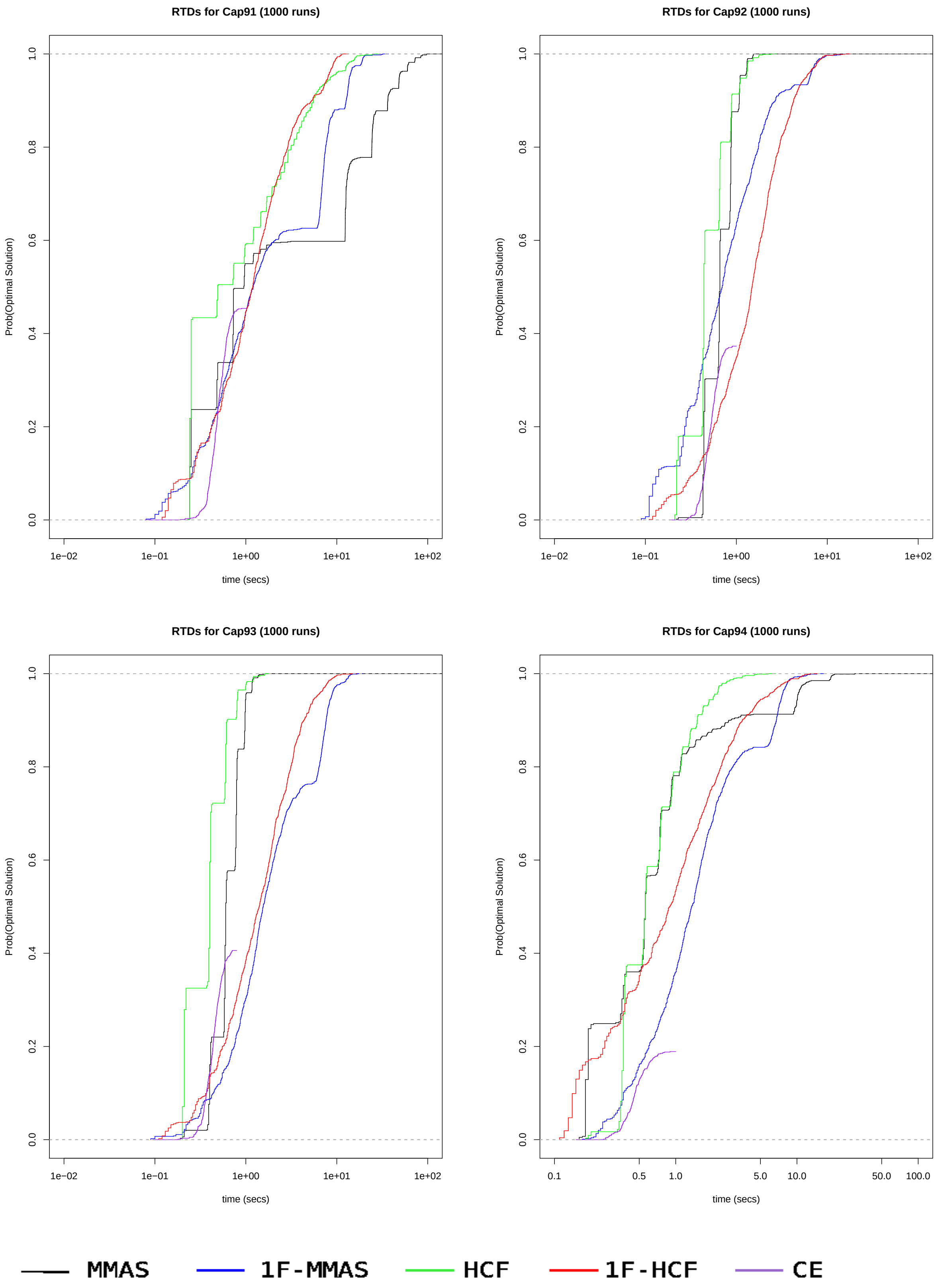
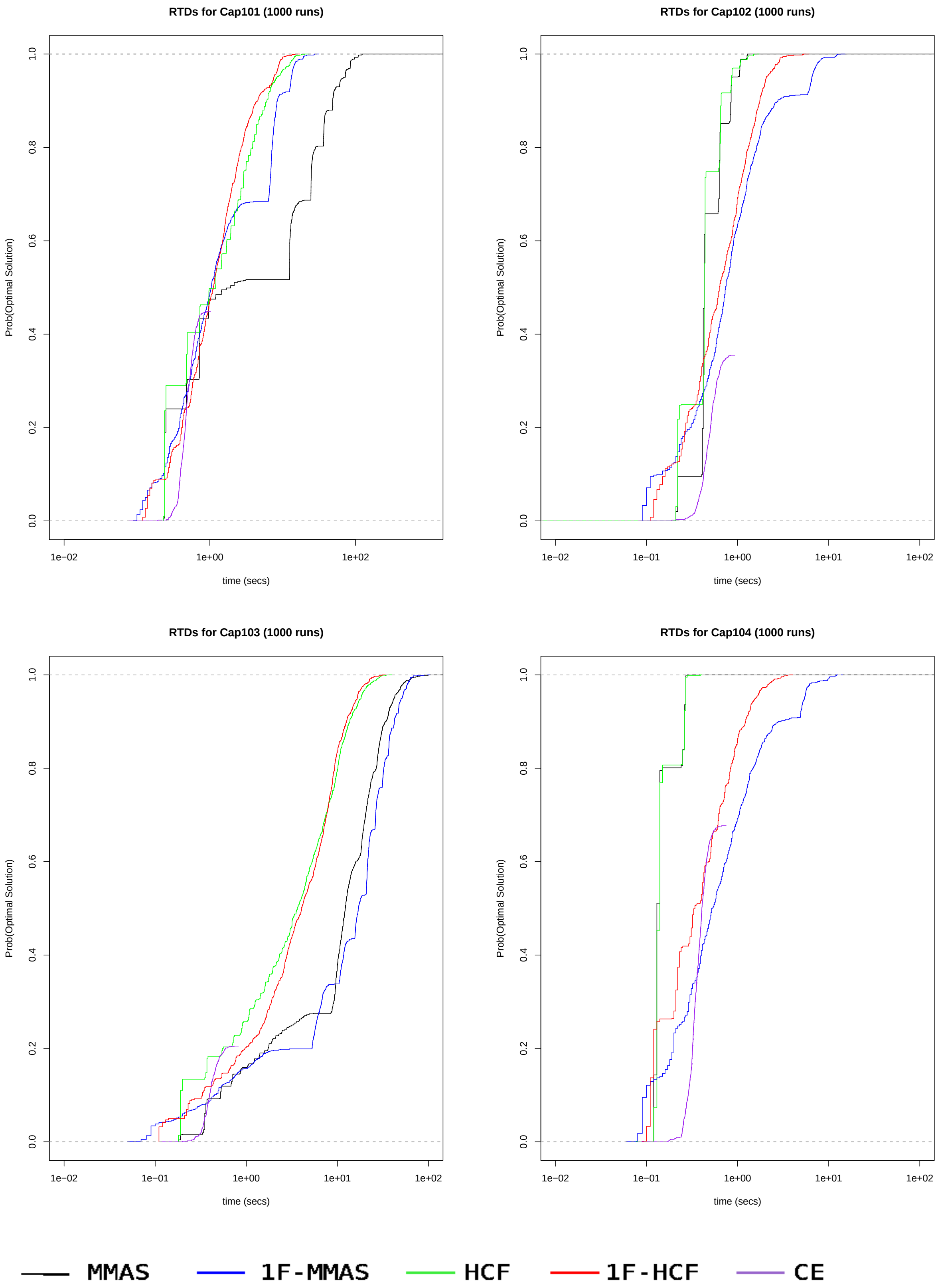

RTDs for Cap91 (1000 runs)
RTDs for Cap92 (1000 runs)
RTDs for Cap93 (1000 runs)
RTDs for Cap94 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
time (secs)
1e−02
1e−01
1e+00
1e+01
1e+02
0.1
0.5
1.0
5.0
10.0
50.0
100.0
MMAS
1F-MMAS
HCF
1F-HCF
CE

Figure B.6



RTDs for Cap101 (1000 runs)

RTDs for Cap102 (1000 runs)

RTDs for Cap103 (1000 runs)

RTDs for Cap104 (1000 runs)

Figure B.7

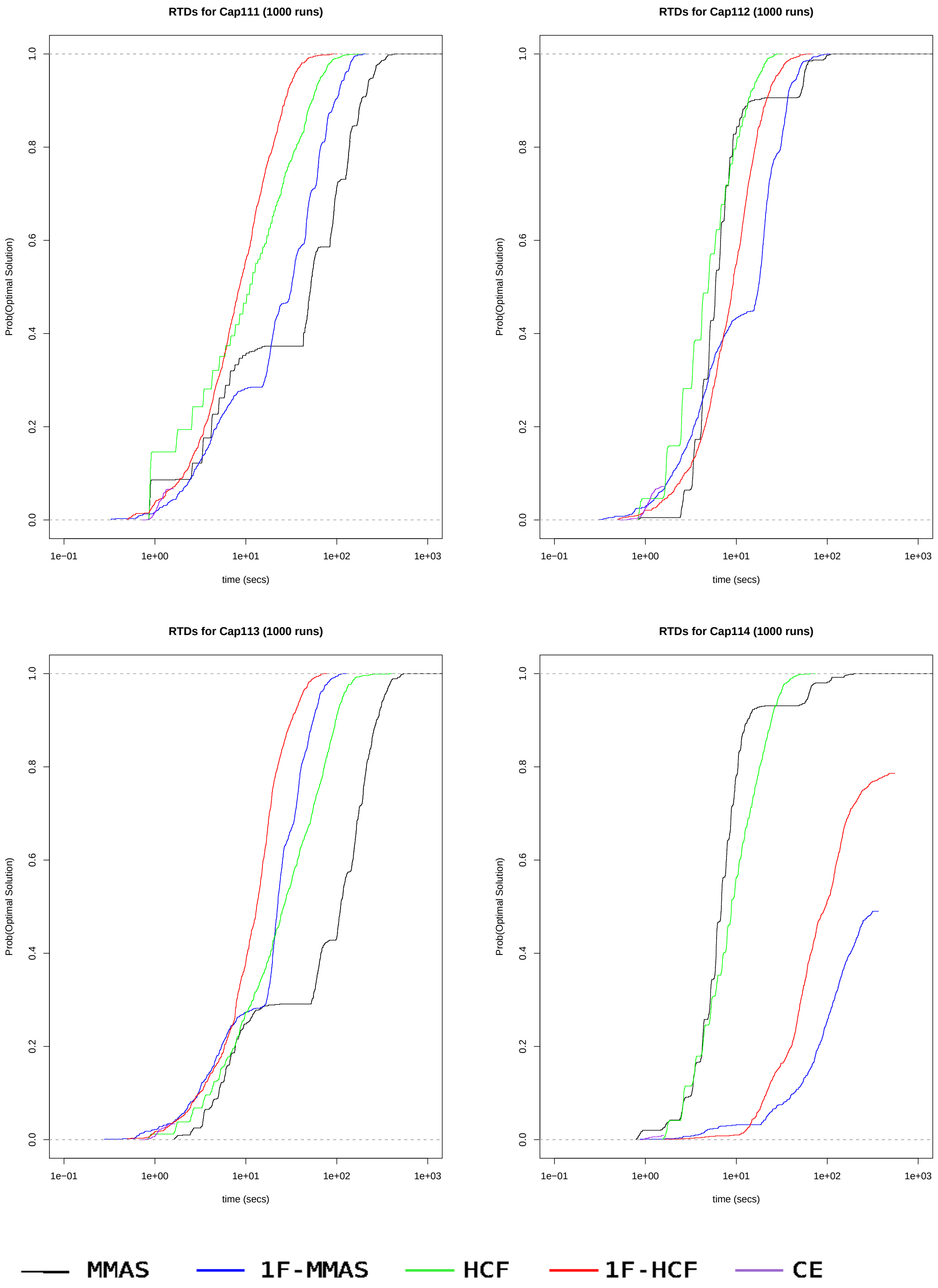

RTDs for Cap111 (1000 runs)
RTDs for Cap112 (1000 runs)
RTDs for Cap113 (1000 runs)
RTDs for Cap114 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
1e−01
1e+00
1e+01
1e+02
1e+03
time (secs)
MMAS
1F-MMAS
HCF
1F-HCF
CE

Figure B.8

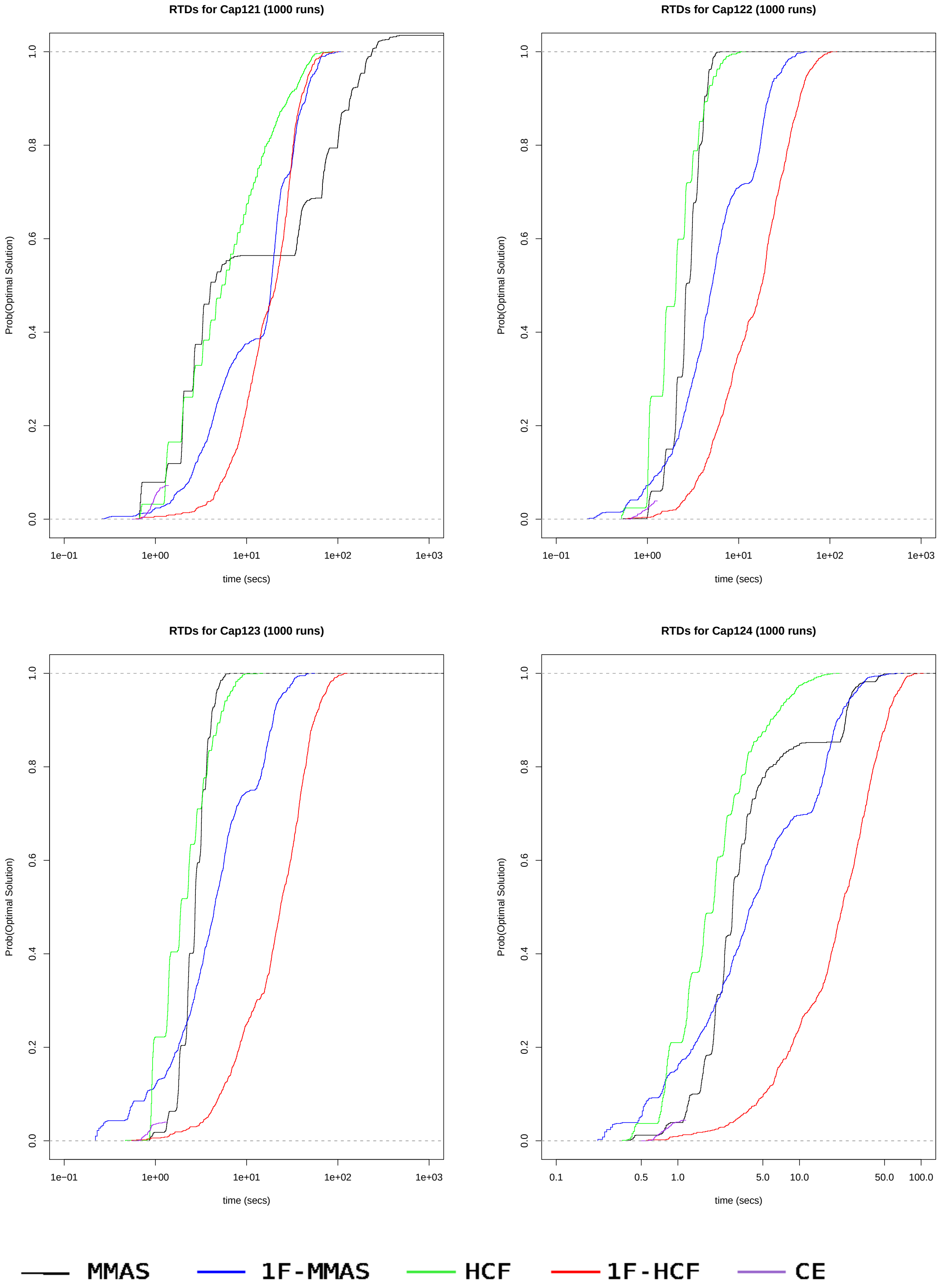

RTDs for Cap121 (1000 runs)
RTDs for Cap122 (1000 runs)
RTDs for Cap123 (1000 runs)
RTDs for Cap124 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
time (secs)
1e−01
1e+00
1e+01
1e+02
1e+03
0.1
0.5
1.0
5.0
10.0
50.0
100.0
MMAS
1F-MMAS
HCF
1F-HCF
CE

Figure B.9

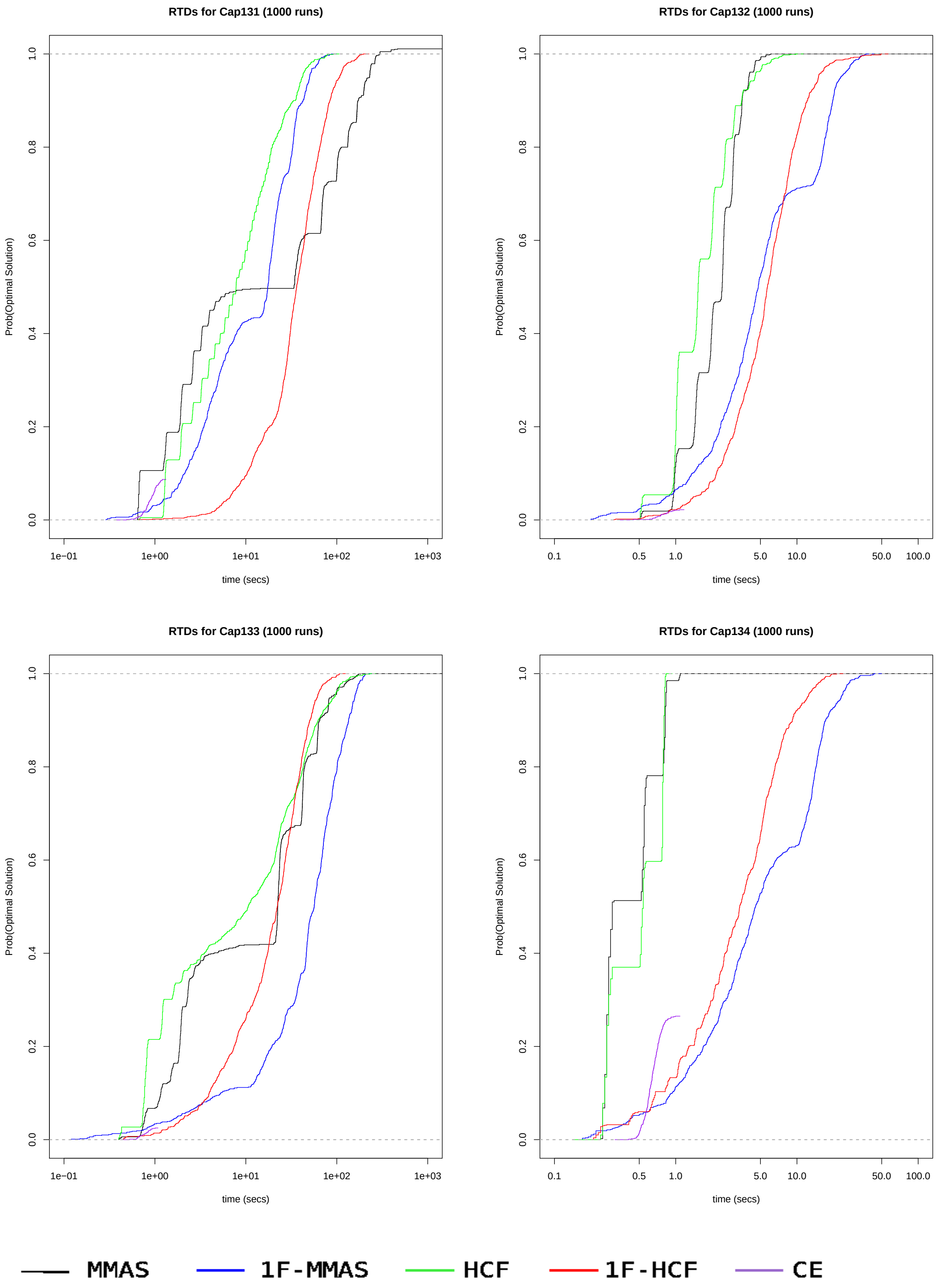

RTDs for Cap131 (1000 runs)
RTDs for Cap132 (1000 runs)
RTDs for Cap133 (1000 runs)
RTDs for Cap134 (1000 runs)
Prob(Optimal Solution)
0.0
0.2
0.4
0.6
0.8
1.0
1e−01
1e+00
1e+01
1e+02
1e+03
0.1
0.5
1.0
5.0
10.0
50.0
100.0
time (secs)
MMAS
1F-MMAS
HCF
1F-HCF
CE

Figure B.10

Appendix C

Algorithmic Solution Quality: Graphical Descriptive Summaries of Run-Time Distributions

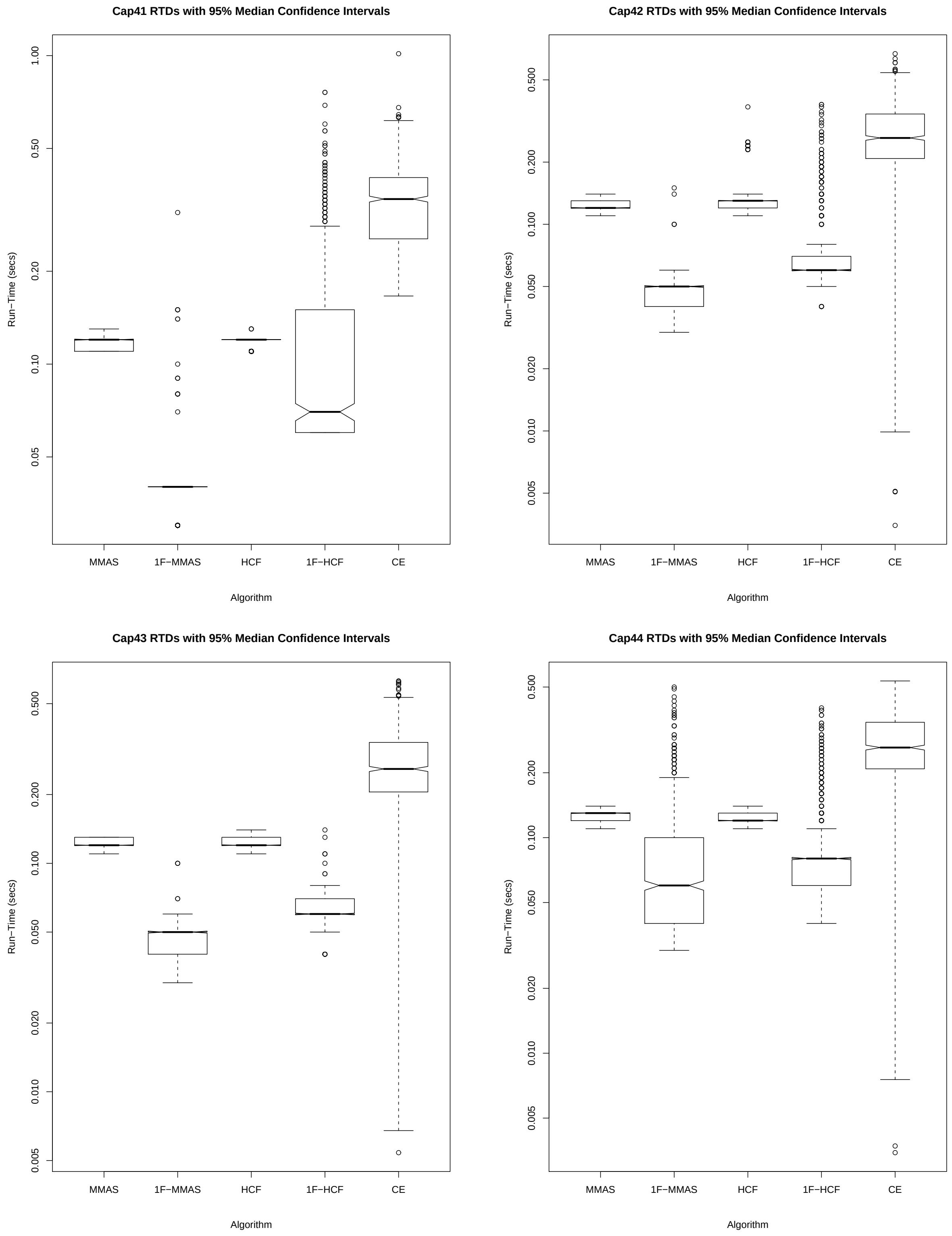

Cap41 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
1.00
0.50
0.20
0.10
0.05
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap42 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.500
0.200
0.100
0.050
0.020
0.010
0.005
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap43 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.500
0.200
0.100
0.050
0.020
0.010
0.005
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap44 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.500
0.200
0.100
0.050
0.020
0.010
0.005
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure C.1

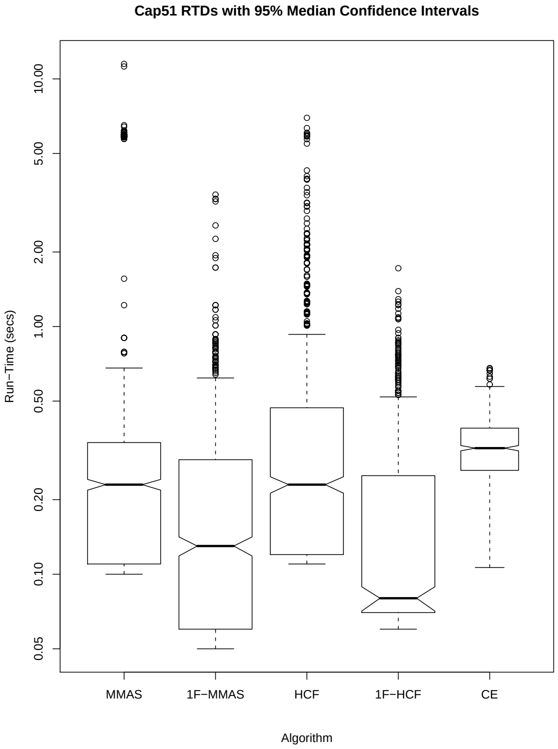

Cap51 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.05
0.10
0.20
0.50
1.00
2.00
5.00
10.00
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure C.2

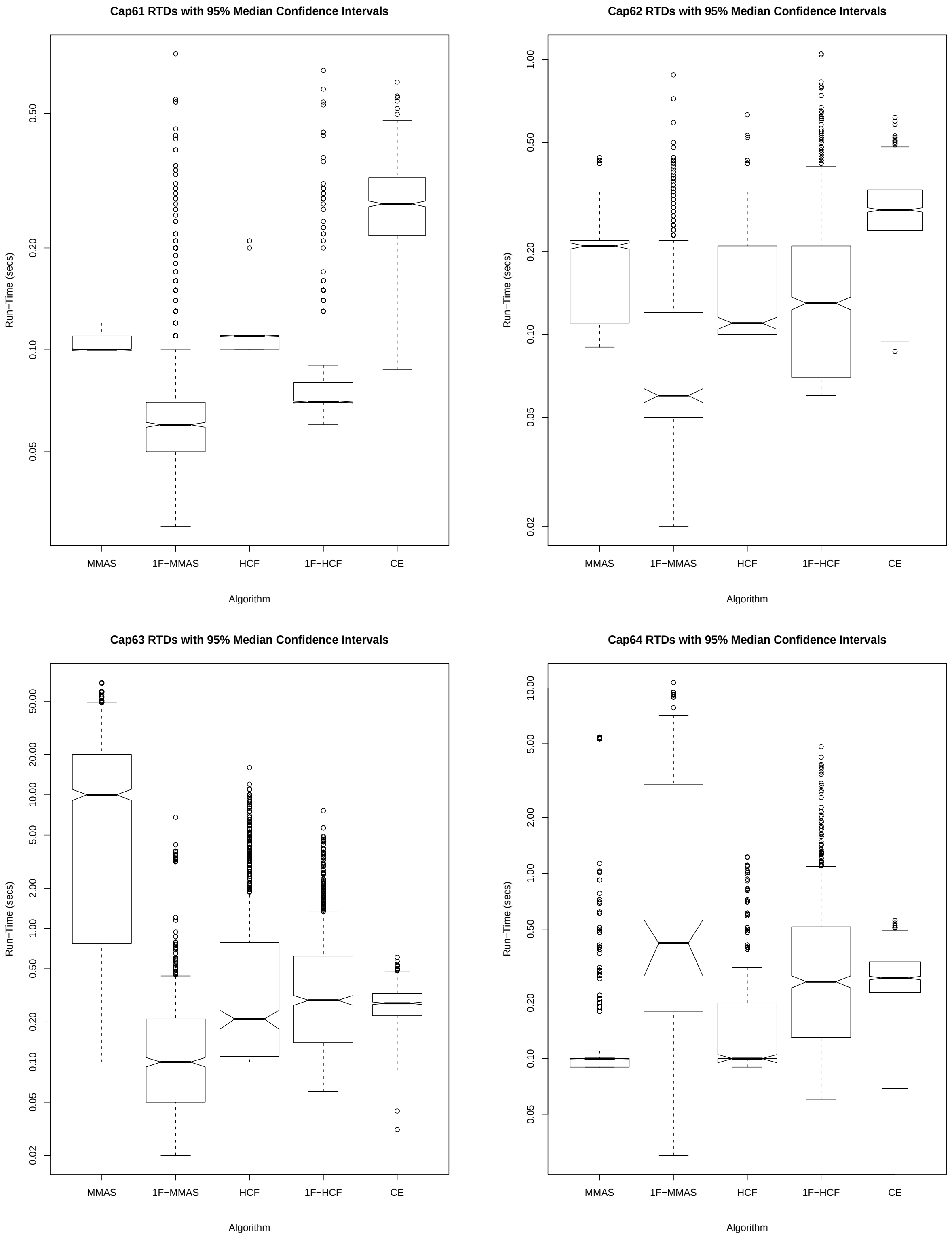

Cap61 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.05
0.10
0.20
0.50
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap62 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.02
0.05
0.10
0.20
0.50
1.00
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap63 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.02
0.05
0.10
0.20
0.50
1.00
2.00
5.00
10.00
20.00
50.00
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap64 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.05
0.10
0.20
0.50
1.00
2.00
5.00
10.00
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure C.3

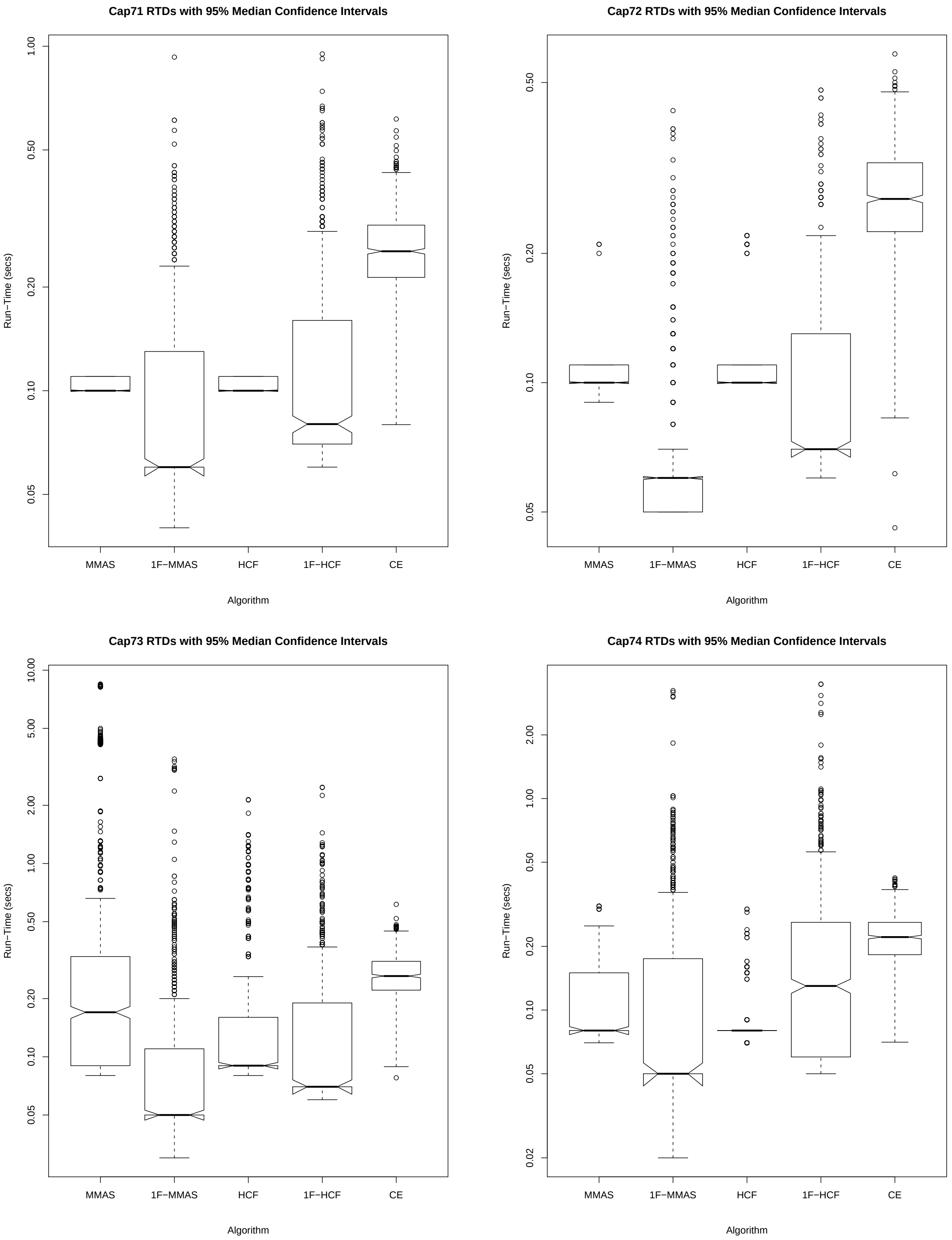

Cap71 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.05
0.10
0.20
0.50
1.00
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap72 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.05
0.10
0.20
0.50
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap73 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.05
0.10
0.20
0.50
1.00
2.00
5.00
10.00
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap74 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.02
0.05
0.10
0.20
0.50
1.00
2.00
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure C.4

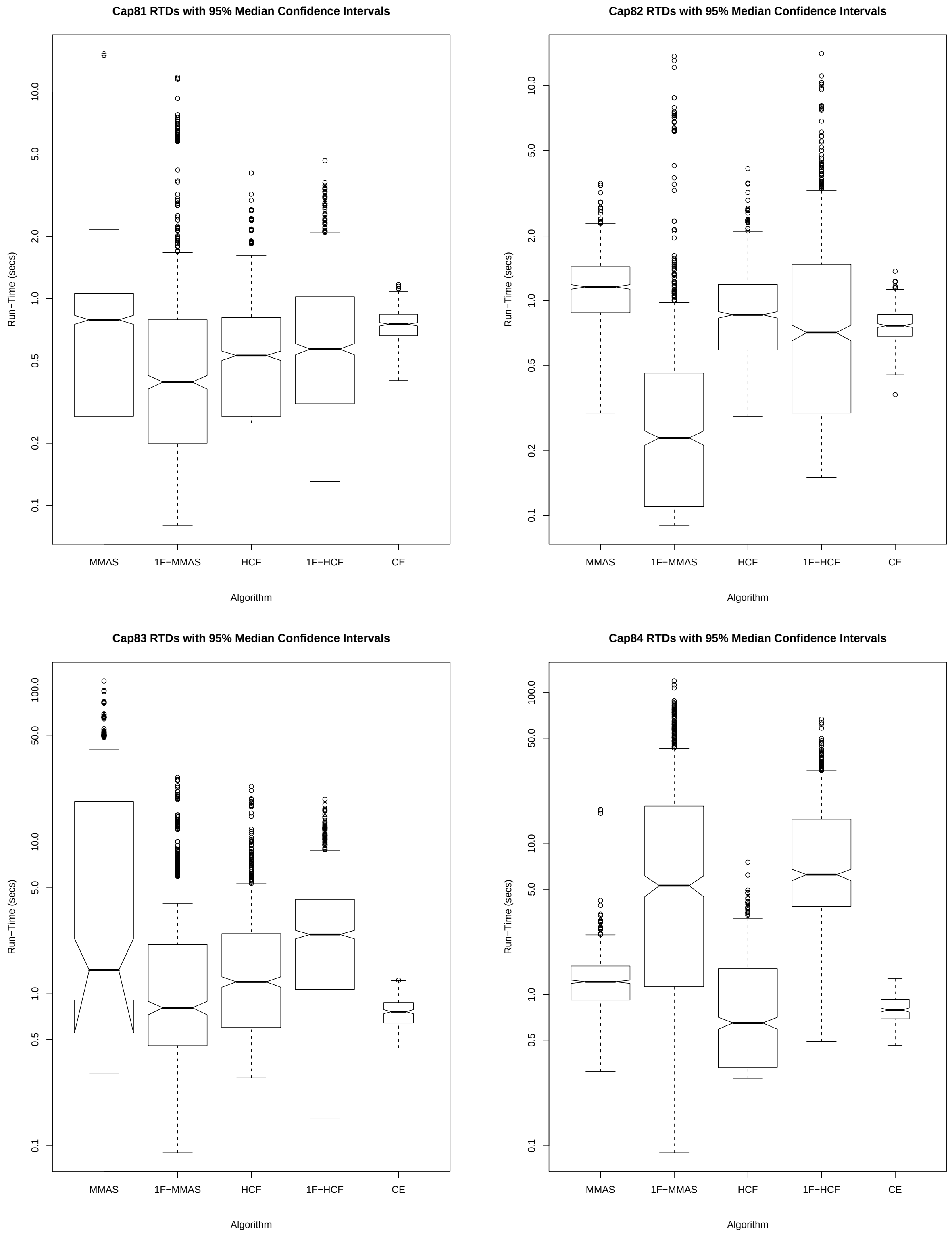

Cap81 RTDs with 95% Median Confidence Intervals
Cap82 RTDs with 95% Median Confidence Intervals
Cap83 RTDs with 95% Median Confidence Intervals
Cap84 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
Algorithm
MMAS
1F−MMAS
HCF
1F−HCF
CE
0.1
0.2
0.5
1.0
2.0
5.0
10.0
50.0
100.0

Figure C.5

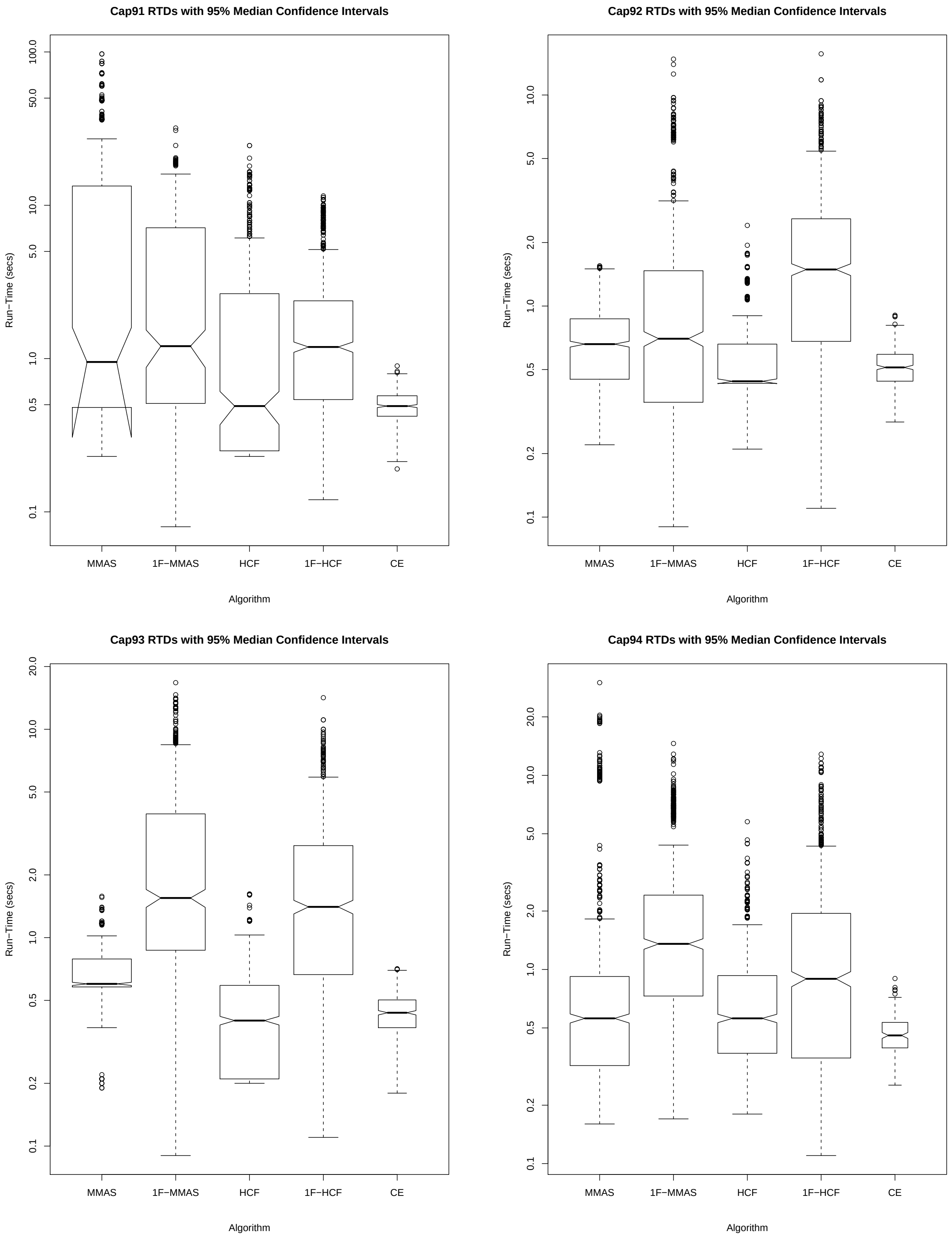

Cap91 RTDs with 95% Median Confidence Intervals
Cap92 RTDs with 95% Median Confidence Intervals
Cap93 RTDs with 95% Median Confidence Intervals
Cap94 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
Algorithm
MMAS
1F−MMAS
HCF
1F−HCF
CE
100.0
50.0
20.0
10.0
5.0
2.0
1.0
0.5
0.2
0.1

Figure C.6

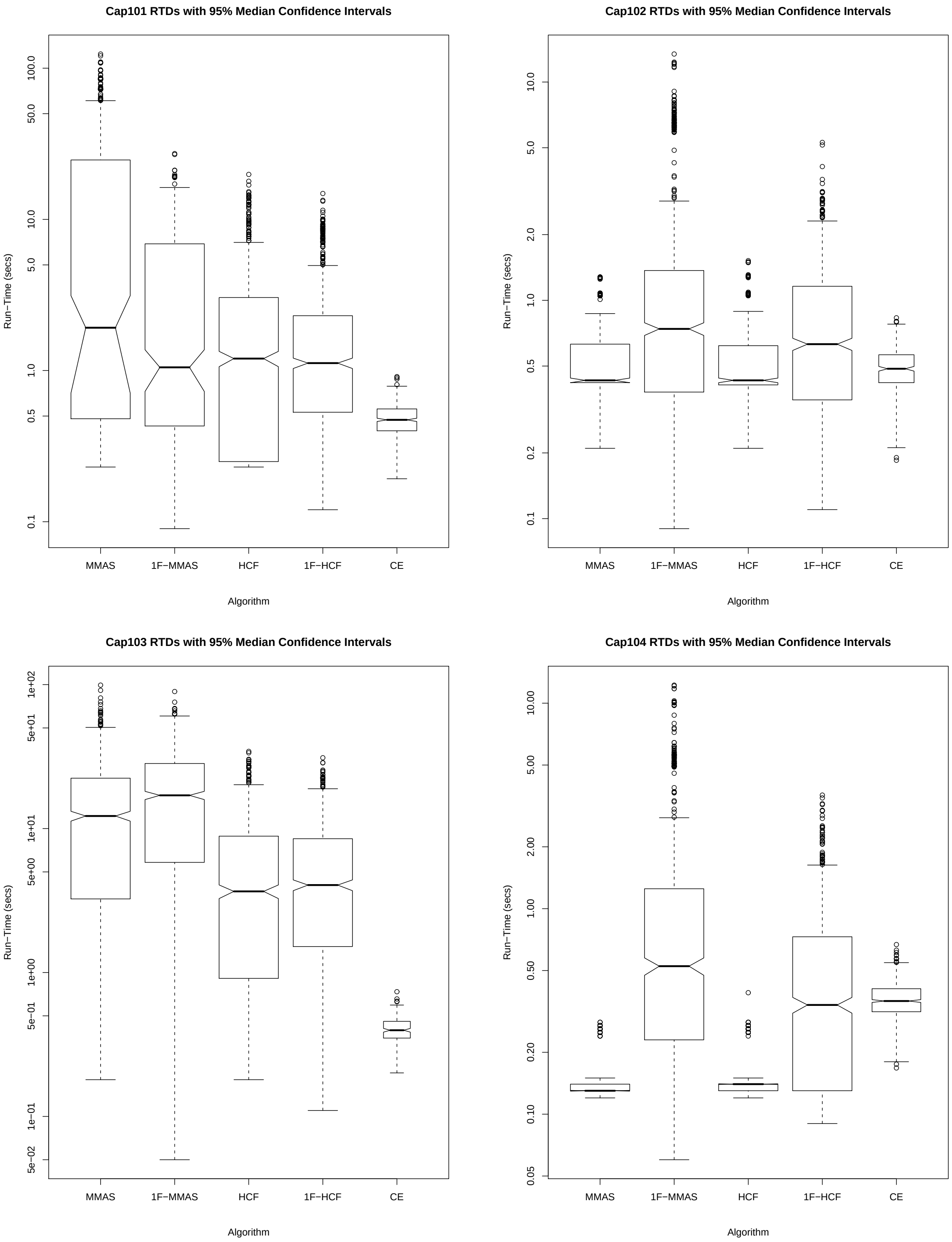

Cap101 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.1
0.5
1.0
5.0
10.0
50.0
100.0
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap102 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.1
0.2
0.5
1.0
2.0
5.0
10.0
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap103 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
5e−02
1e−01
5e−01
1e+00
5e+00
1e+01
5e+01
1e+02
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap104 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.05
0.10
0.20
0.50
1.00
2.00
5.00
10.00
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure C.7

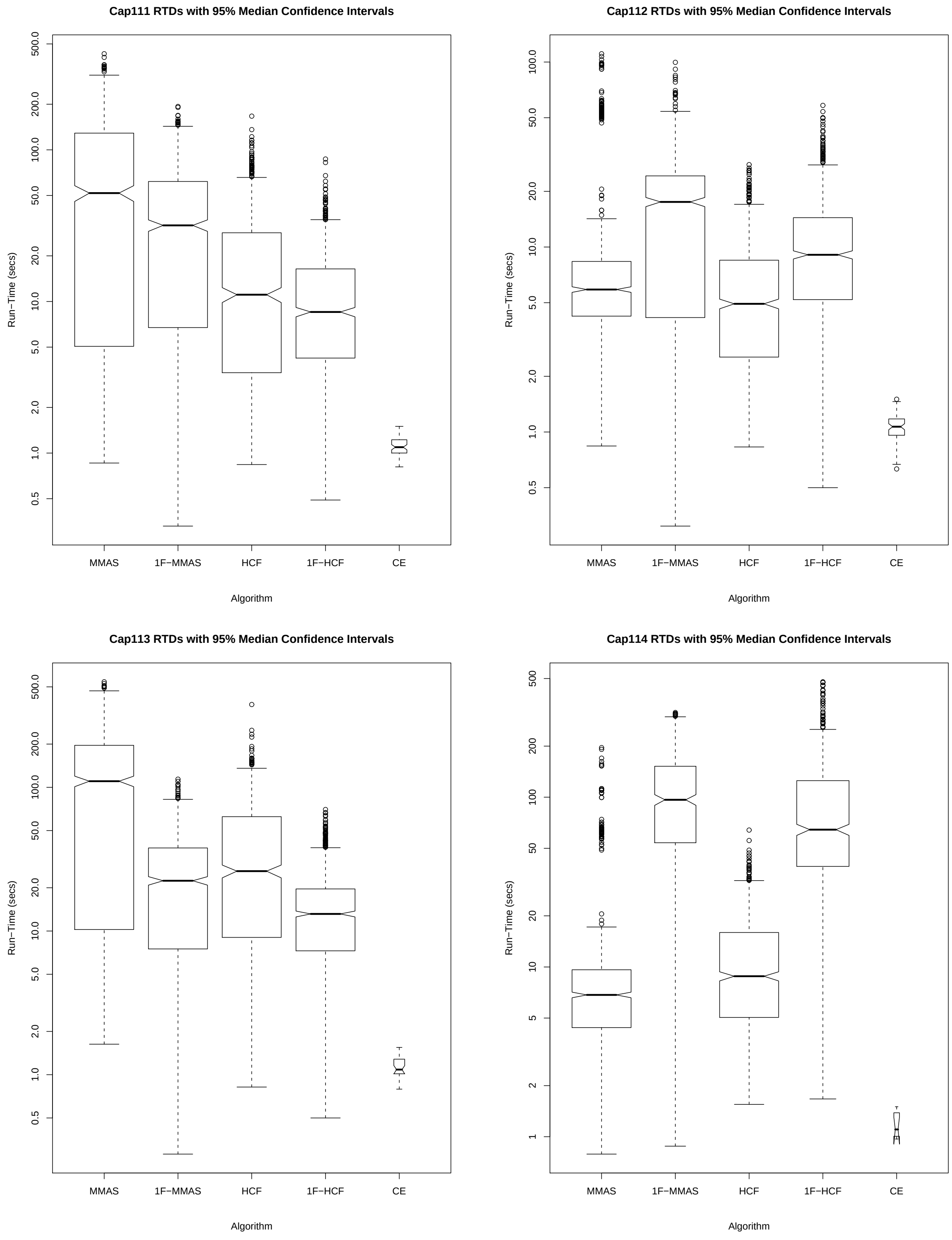

Cap111 RTDs with 95% Median Confidence Intervals
Cap112 RTDs with 95% Median Confidence Intervals
Cap113 RTDs with 95% Median Confidence Intervals
Cap114 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.5
1.0
2.0
5.0
10.0
20.0
50.0
100.0
200.0
500.0
1
2
5
10
20
50
100
200
500
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure C.8

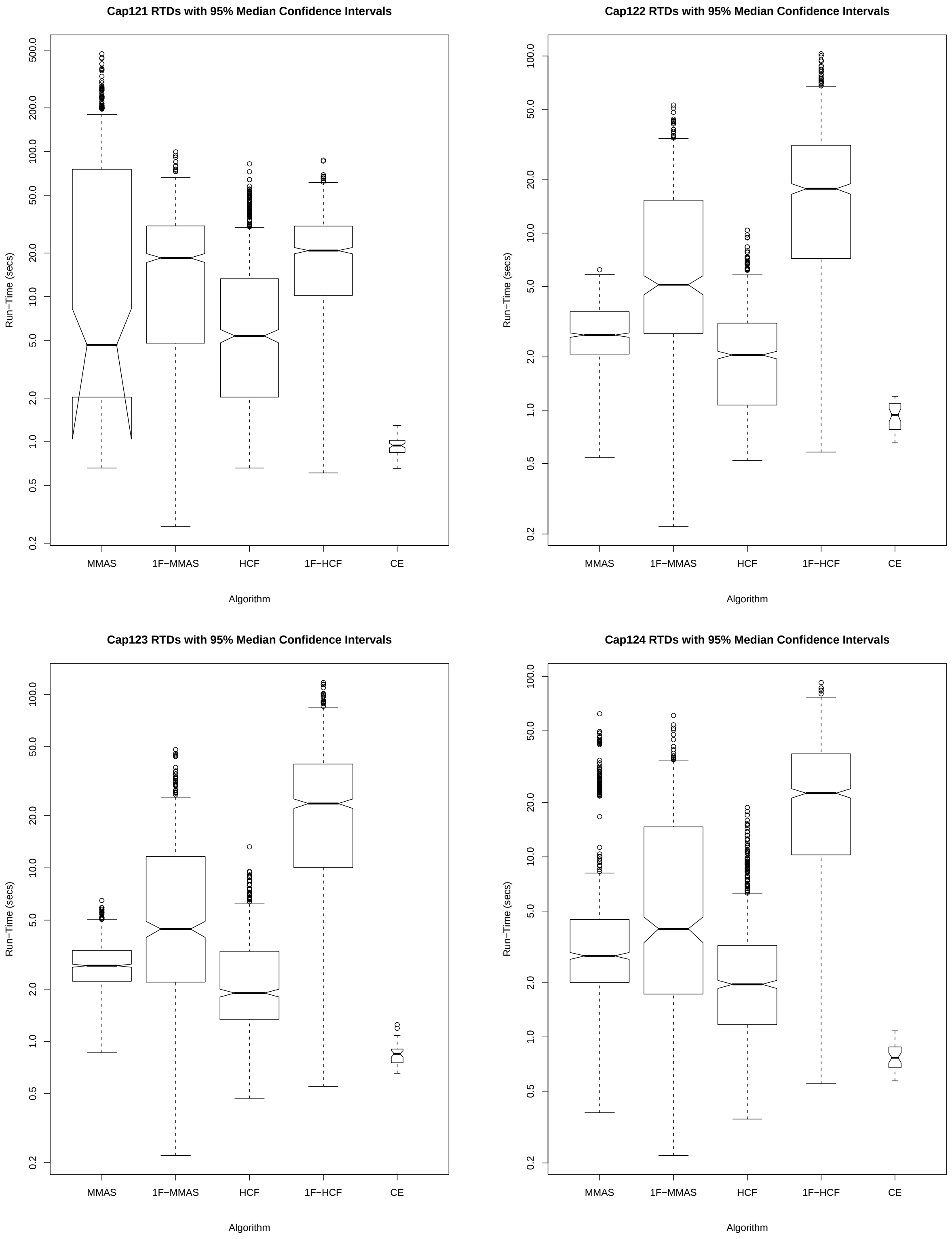

Cap121 RTDs with 95% Median Confidence Intervals
Cap122 RTDs with 95% Median Confidence Intervals
Cap123 RTDs with 95% Median Confidence Intervals
Cap124 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
0.2
0.5
1.0
2.0
5.0
10.0
20.0
50.0
100.0
200.0
500.0
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure C.9

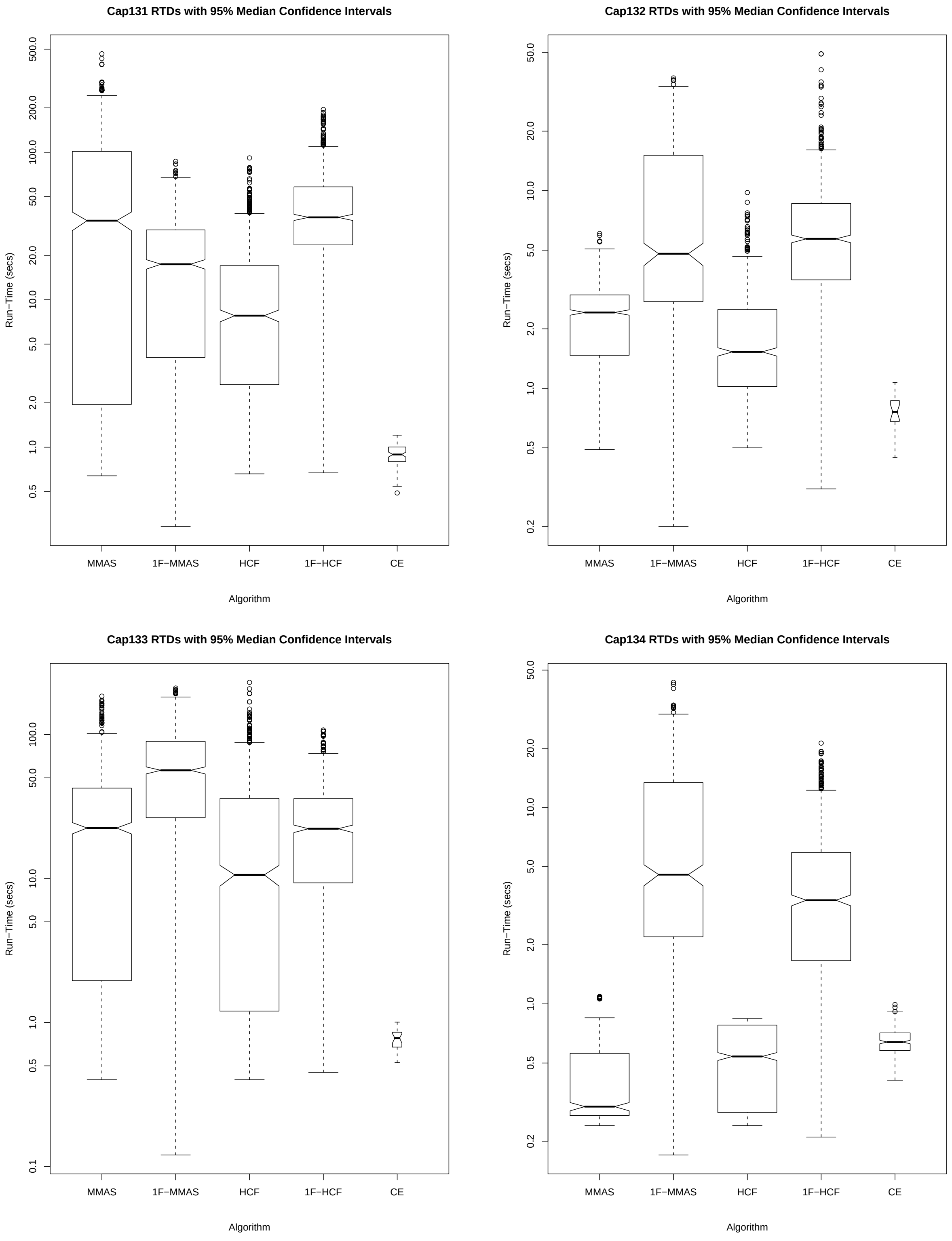

Cap131 RTDs with 95% Median Confidence Intervals
Cap132 RTDs with 95% Median Confidence Intervals
Cap133 RTDs with 95% Median Confidence Intervals
Cap134 RTDs with 95% Median Confidence Intervals
Run−Time (secs)
Algorithm
MMAS
1F−MMAS
HCF
1F−HCF
CE
0.1
0.2
0.5
1.0
2.0
5.0
10.0
20.0
50.0
100.0
200.0
500.0

Figure C.10

Appendix D

Algorithmic Solution Quality: Graphical Descriptive Summaries of Bootstrapped Median Distributions

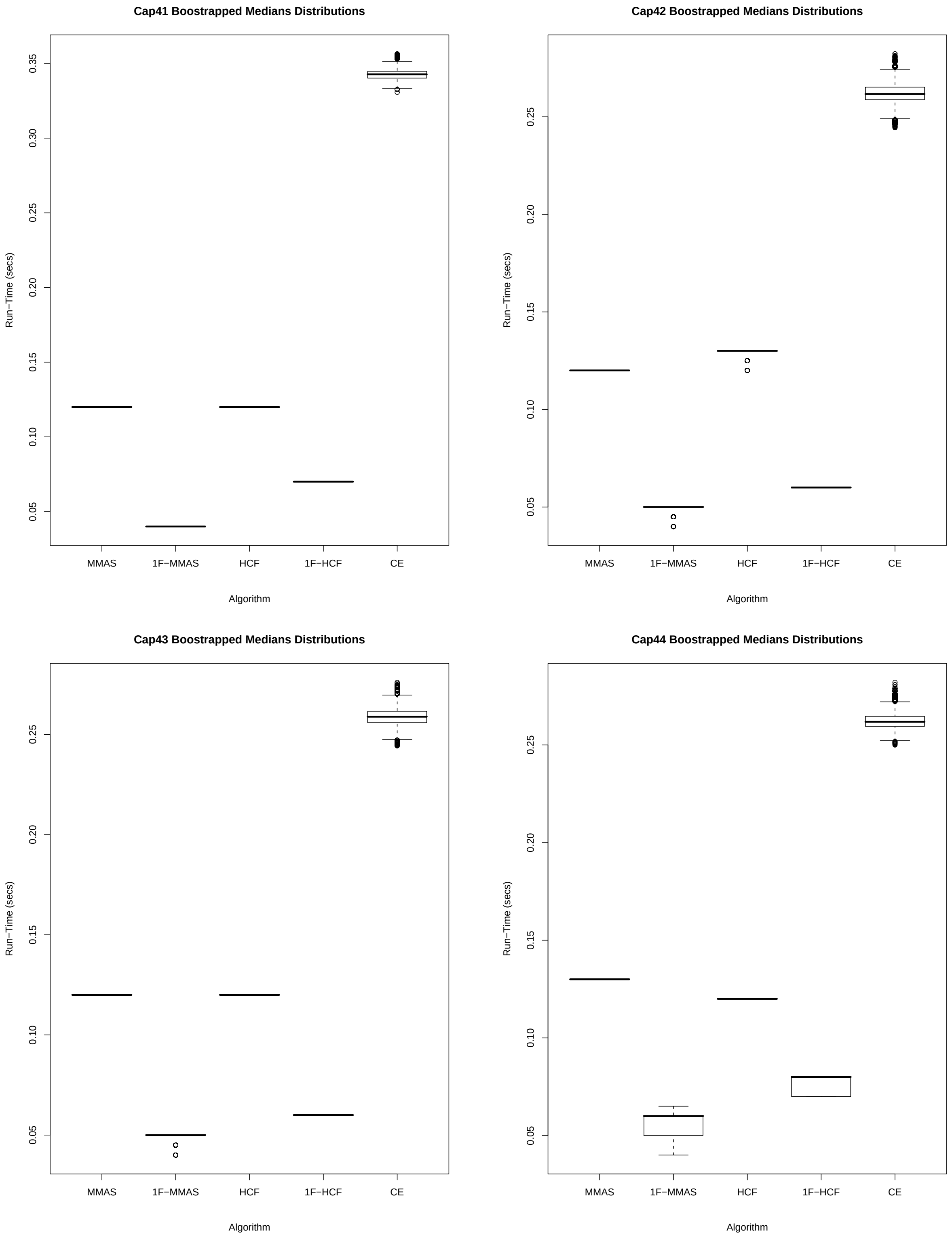

Cap41 Boostrapped Medians Distributions
Run−Time (secs)
0.05
0.10
0.15
0.20
0.25
0.30
0.35
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap42 Boostrapped Medians Distributions
Run−Time (secs)
0.05
0.10
0.15
0.20
0.25
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap43 Boostrapped Medians Distributions
Run−Time (secs)
0.05
0.10
0.15
0.20
0.25
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap44 Boostrapped Medians Distributions
Run−Time (secs)
0.05
0.10
0.15
0.20
0.25
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure D.1

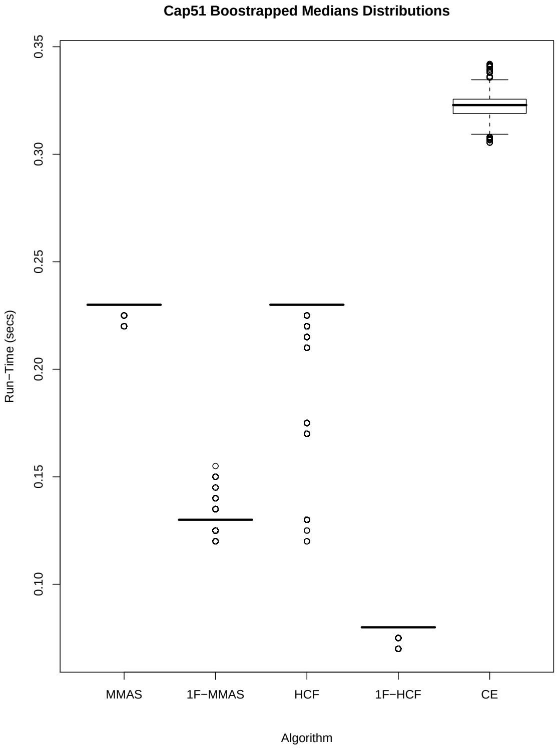

Cap51 Boostrapped Medians Distributions
Run−Time (secs)
0.10
0.15
0.20
0.25
0.30
0.35
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure D.2

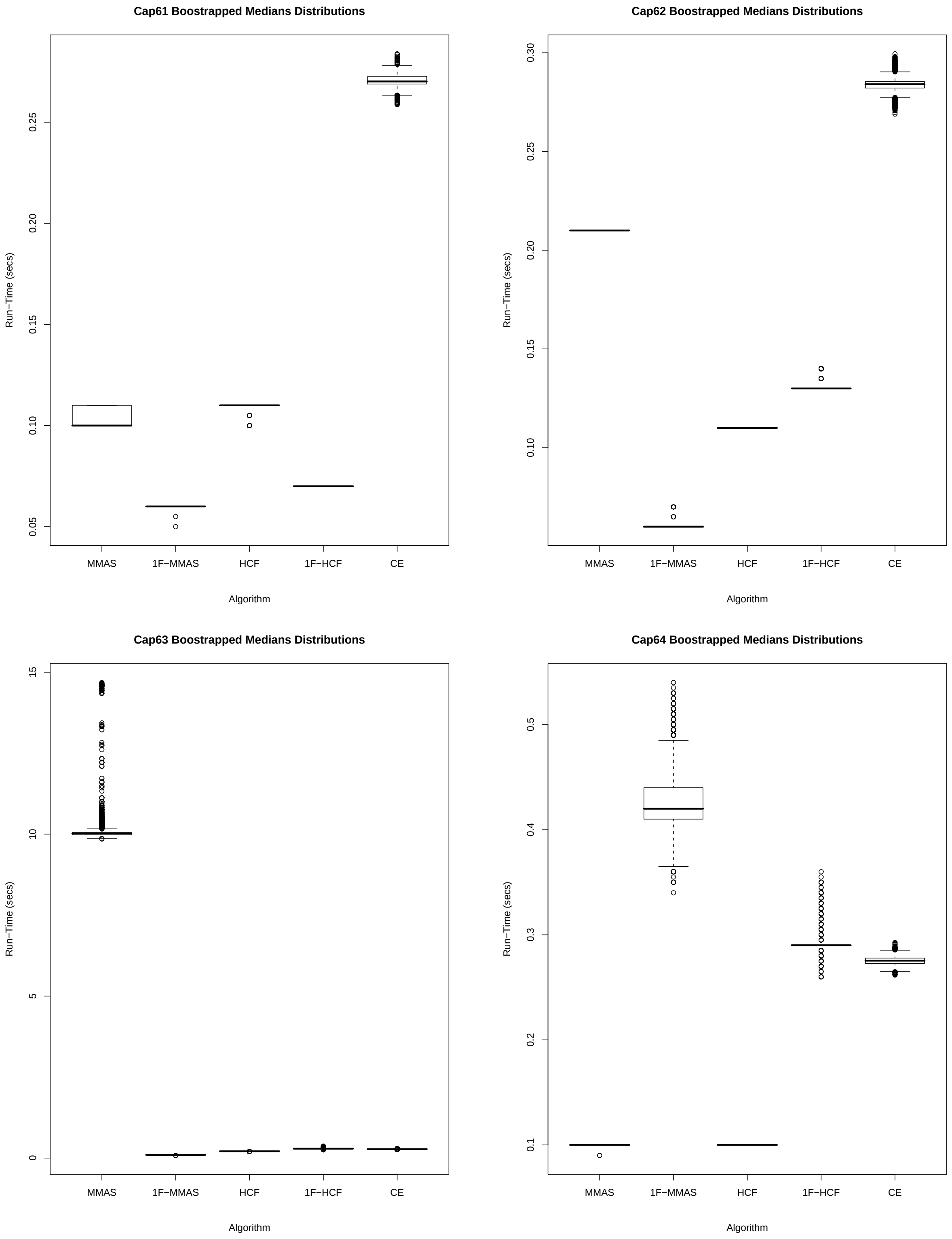

Cap61 Boostrapped Medians Distributions
Run−Time (secs)
0.05
0.10
0.15
0.20
0.25
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap62 Boostrapped Medians Distributions
Run−Time (secs)
0.10
0.15
0.20
0.25
0.30
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap63 Boostrapped Medians Distributions
Run−Time (secs)
0
5
10
15
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap64 Boostrapped Medians Distributions
Run−Time (secs)
0.1
0.2
0.3
0.4
0.5
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure D.3

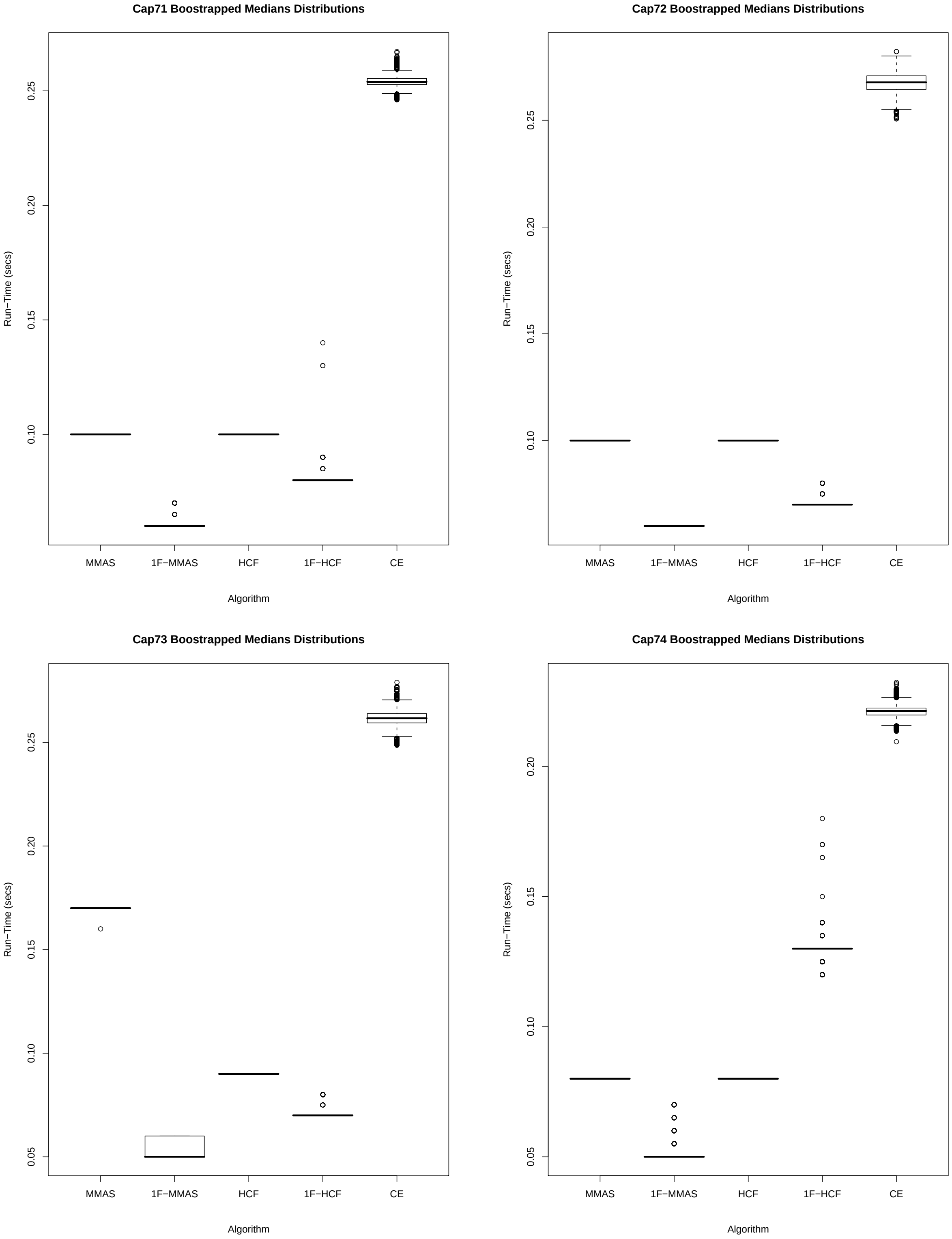

Cap71 Boostrapped Medians Distributions
Run−Time (secs)
0.10
0.15
0.20
0.25
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap72 Boostrapped Medians Distributions
Run−Time (secs)
0.10
0.15
0.20
0.25
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap73 Boostrapped Medians Distributions
Run−Time (secs)
0.05
0.10
0.15
0.20
0.25
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap74 Boostrapped Medians Distributions
Run−Time (secs)
0.05
0.10
0.15
0.20
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure D.4

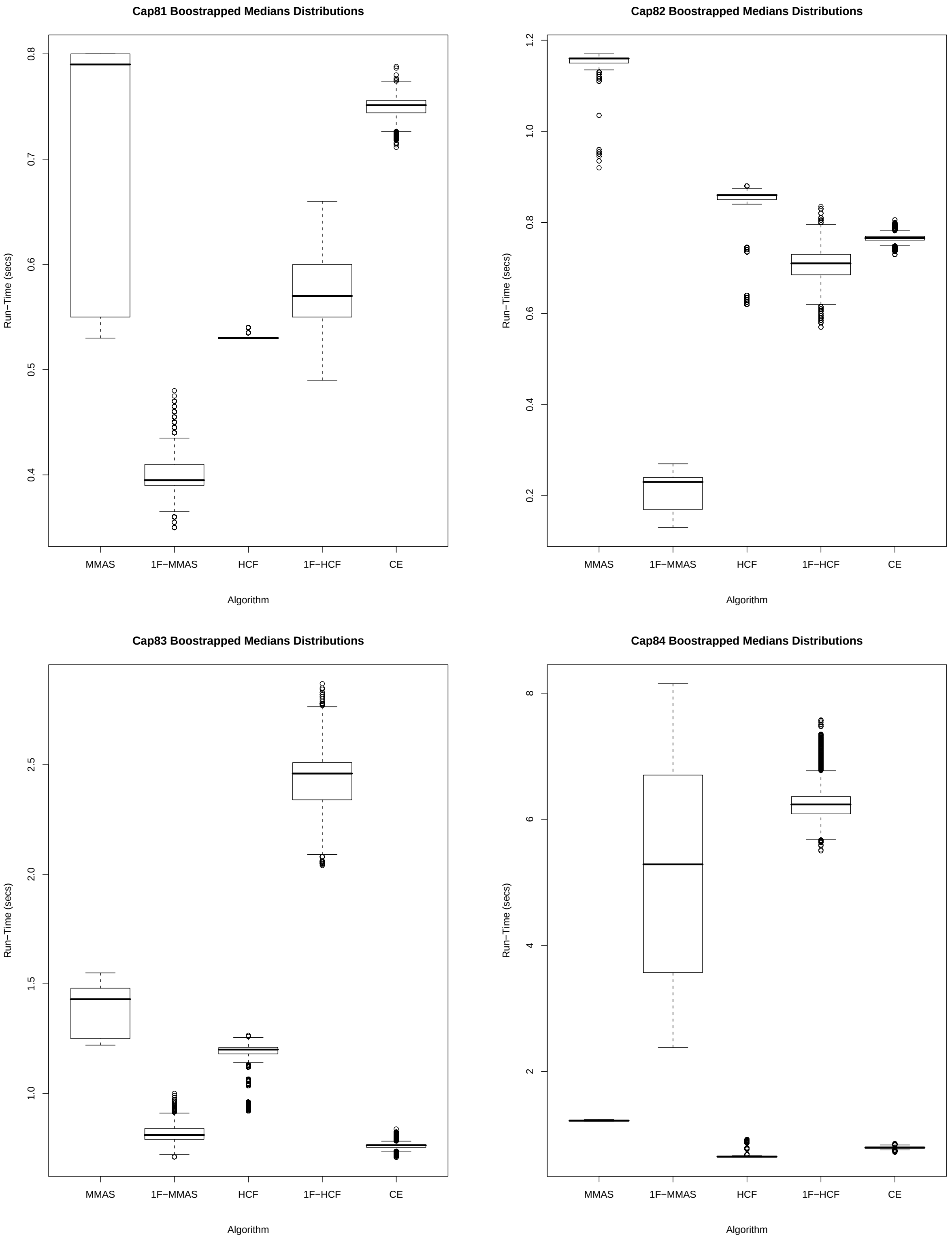
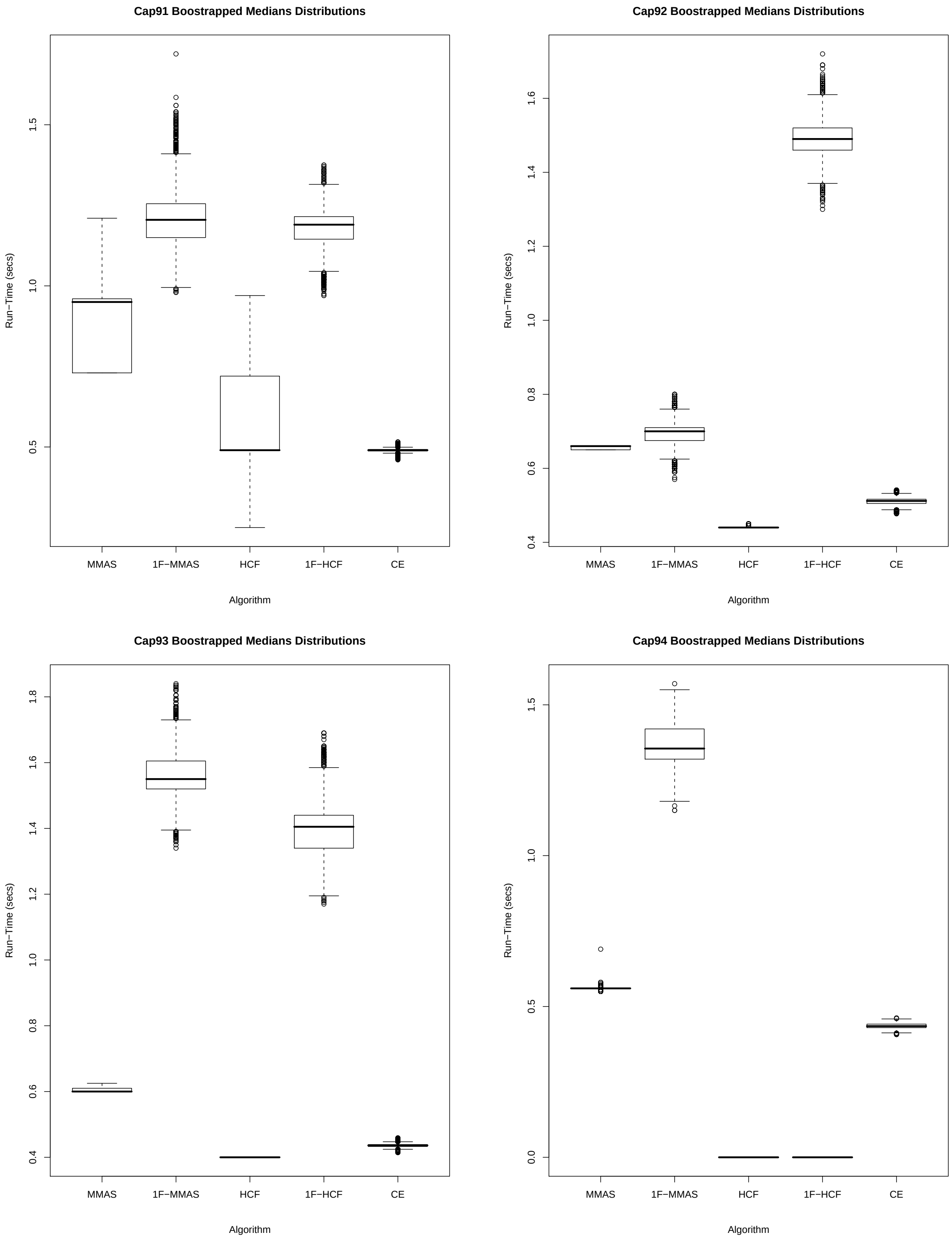

Cap81 Boostrapped Medians Distributions
Run−Time (secs)
0.4
0.5
0.6
0.7
0.8
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap82 Boostrapped Medians Distributions
Run−Time (secs)
0.2
0.4
0.6
0.8
1.0
1.2
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap83 Boostrapped Medians Distributions
Run−Time (secs)
1.0
1.5
2.0
2.5
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap84 Boostrapped Medians Distributions
Run−Time (secs)
2
4
6
8
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure D.5


Cap91 Boostrapped Medians Distributions

Cap92 Boostrapped Medians Distributions

Cap93 Boostrapped Medians Distributions

Cap94 Boostrapped Medians Distributions

Figure D.6

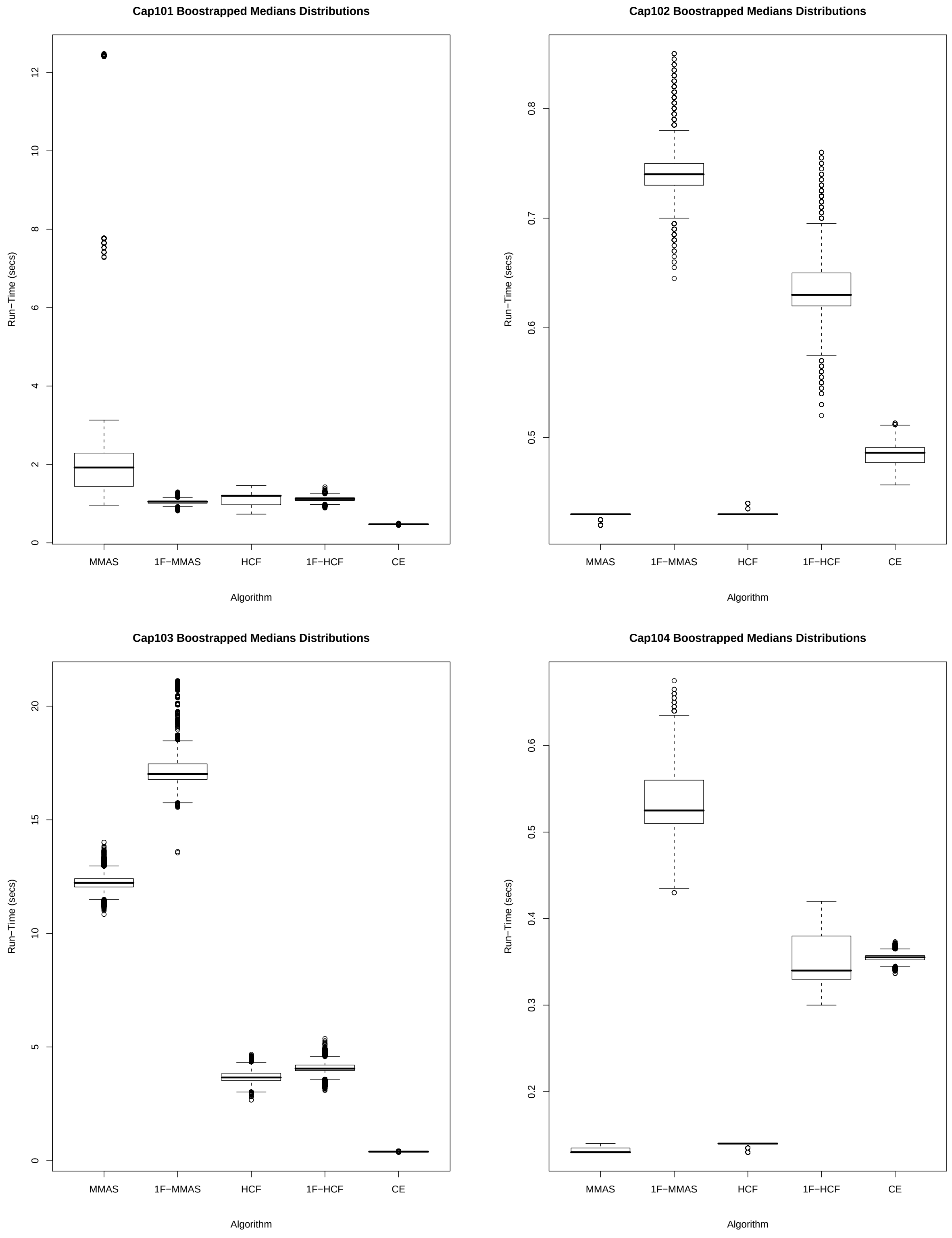

Cap101 Boostrapped Medians Distributions
Run−Time (secs)
0
2
4
6
8
10
12
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap102 Boostrapped Medians Distributions
Run−Time (secs)
0.5
0.6
0.7
0.8
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap103 Boostrapped Medians Distributions
Run−Time (secs)
0
5
10
15
20
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap104 Boostrapped Medians Distributions
Run−Time (secs)
0.2
0.3
0.4
0.5
0.6
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure D.7

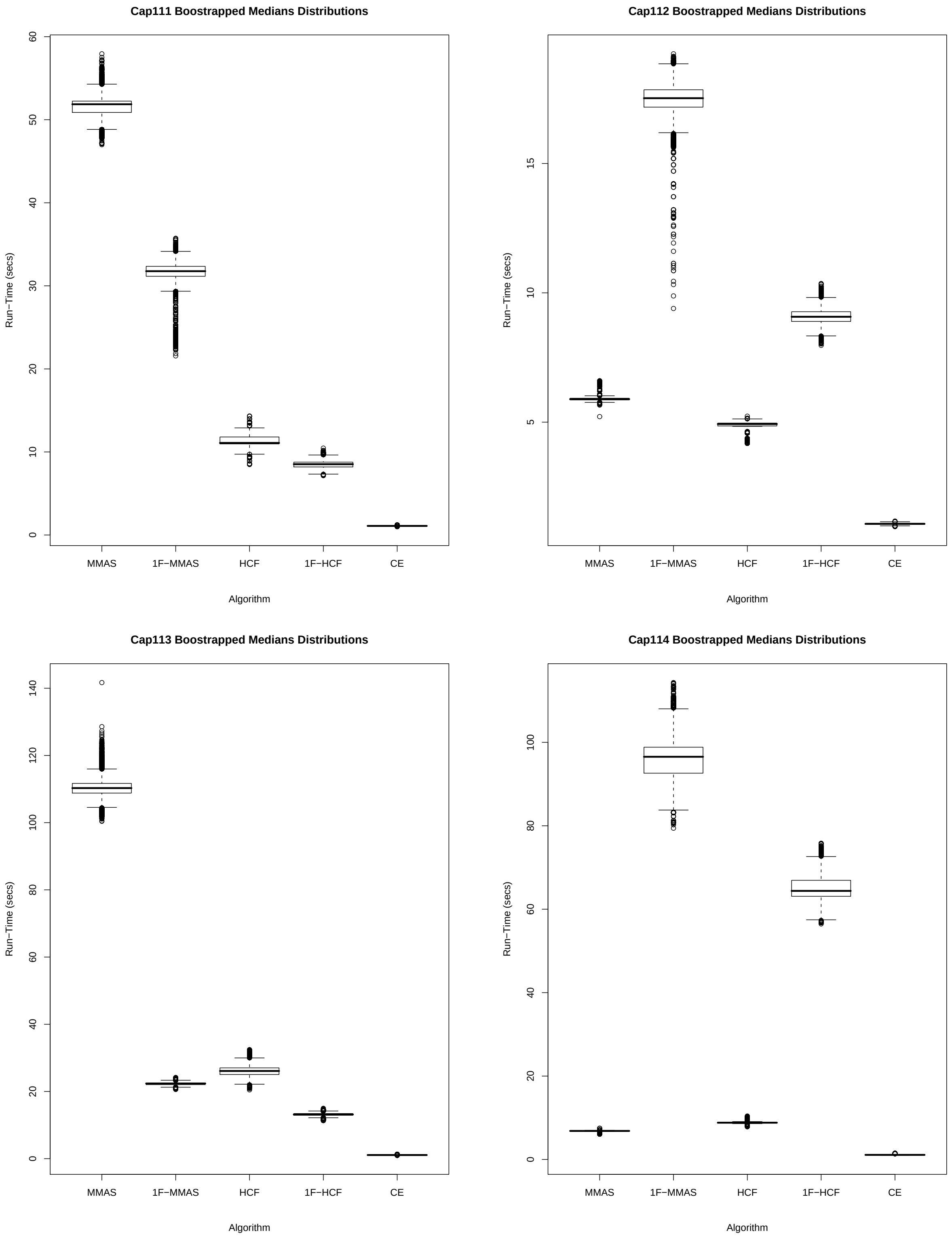

Cap111 Boostrapped Medians Distributions
Run−Time (secs)
0
10
20
30
40
50
60
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap112 Boostrapped Medians Distributions
Run−Time (secs)
5
10
15
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap113 Boostrapped Medians Distributions
Run−Time (secs)
0
20
40
60
80
100
120
140
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap114 Boostrapped Medians Distributions
Run−Time (secs)
0
20
40
60
80
100
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure D.8

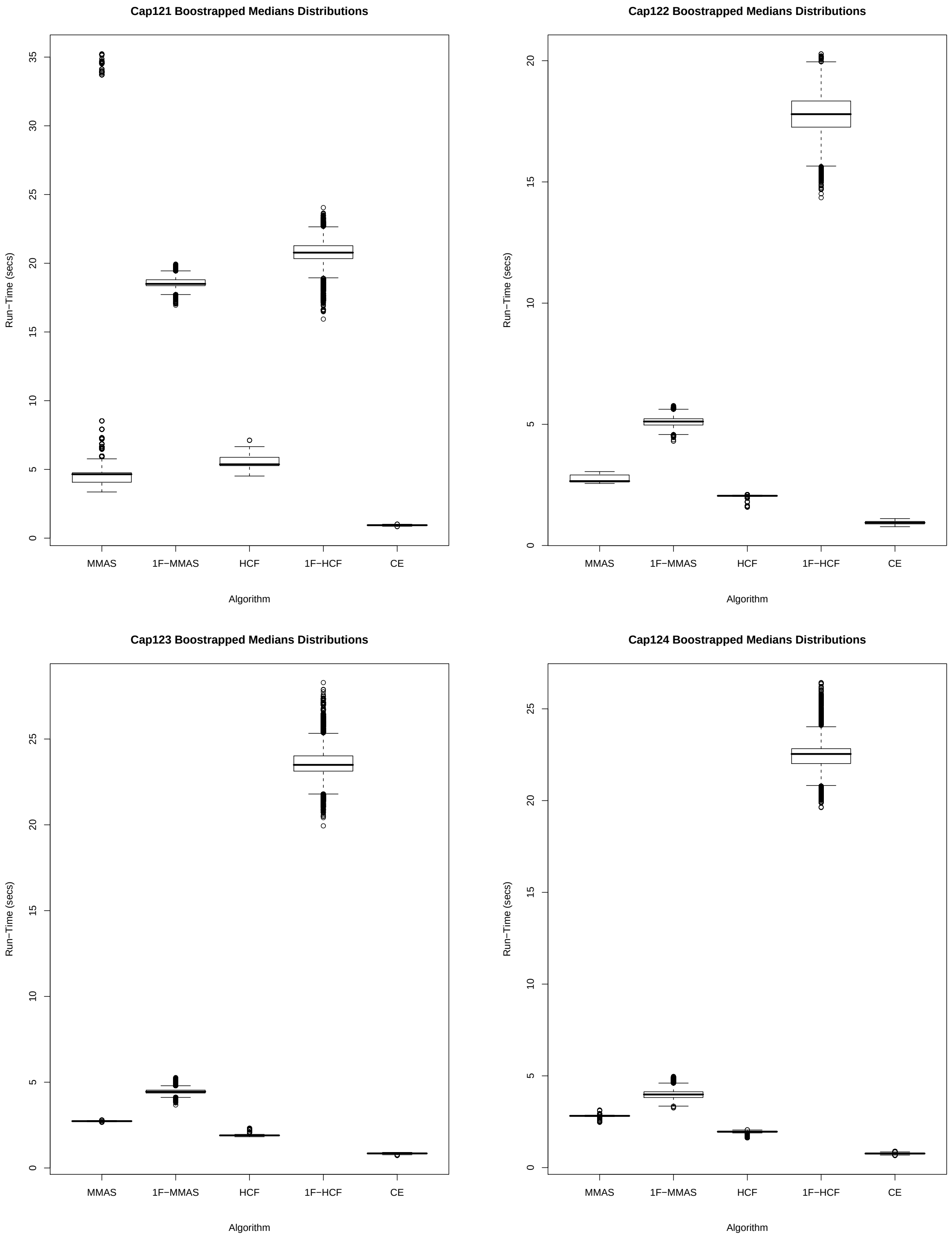

Cap121 Boostrapped Medians Distributions
Cap122 Boostrapped Medians Distributions
Cap123 Boostrapped Medians Distributions
Cap124 Boostrapped Medians Distributions
Run−Time (secs)
Algorithm
MMAS
1F−MMAS
HCF
1F−HCF
CE
0
5
10
15
20
25
30
35

Figure D.9

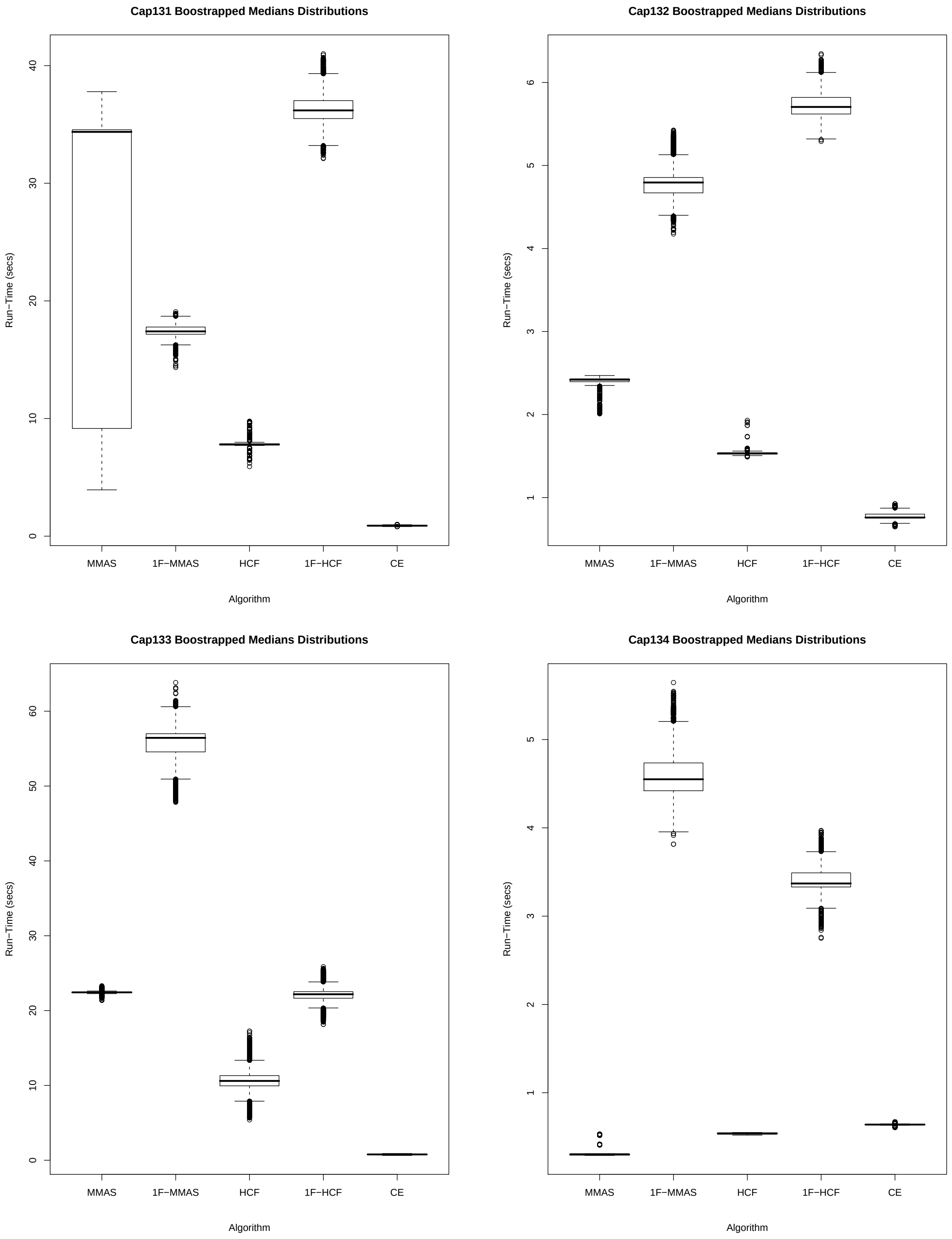

Cap131 Boostrapped Medians Distributions
Run−Time (secs)
0
10
20
30
40
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap132 Boostrapped Medians Distributions
Run−Time (secs)
1
2
3
4
5
6
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap133 Boostrapped Medians Distributions
Run−Time (secs)
0
10
20
30
40
50
60
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm
Cap134 Boostrapped Medians Distributions
Run−Time (secs)
1
2
3
4
5
MMAS
1F−MMAS
HCF
1F−HCF
CE
Algorithm

Figure D.10

Appendix E

Algorithmic Solution Quality: Bootstrapped 95% Confidence Intervals for Median Run-Times

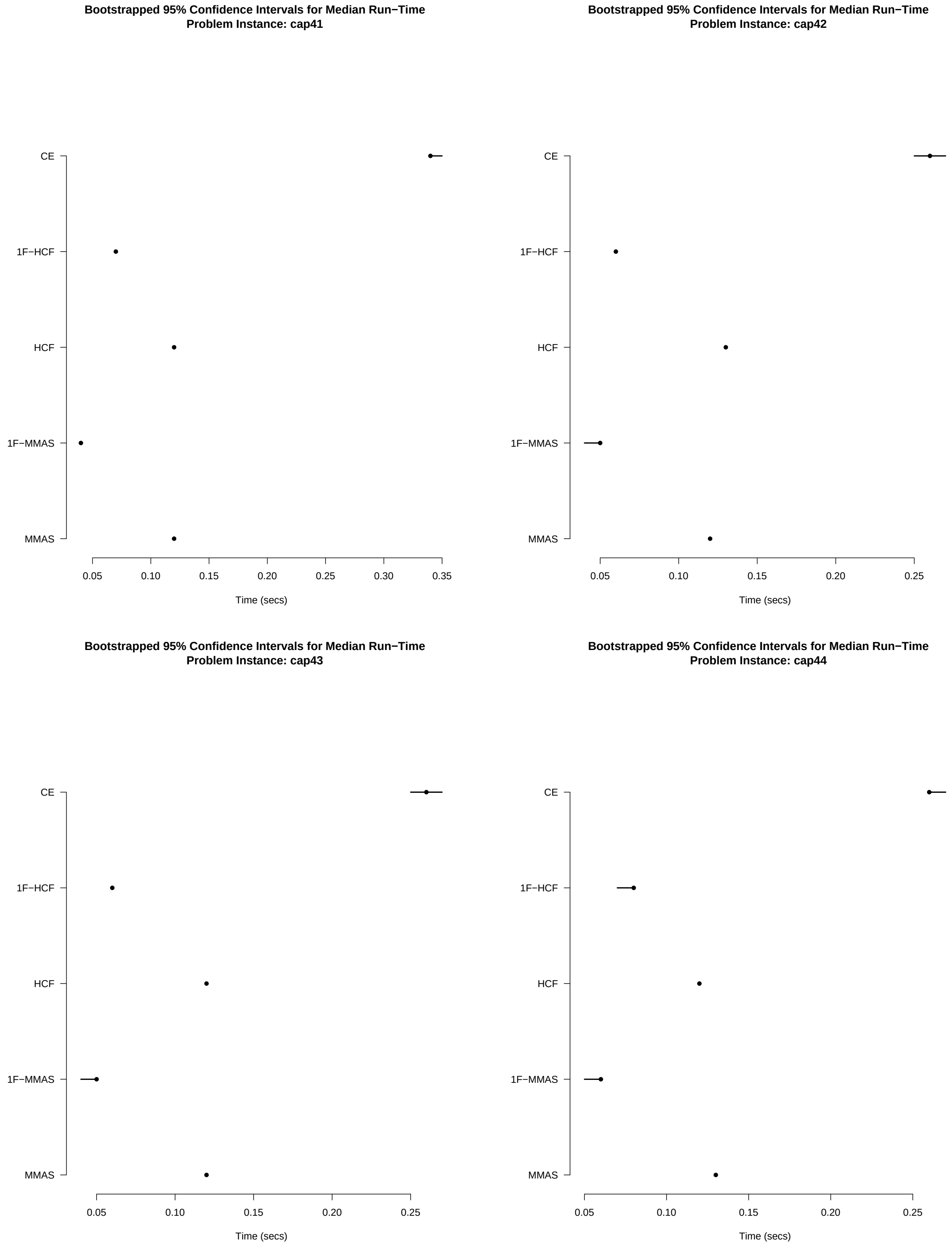

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap41
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap42
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.05
0.10
0.15
0.20
0.25
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap43
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.05
0.10
0.15
0.20
0.25
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap44
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.05
0.10
0.15
0.20
0.25
Time (secs)

Figure E.1

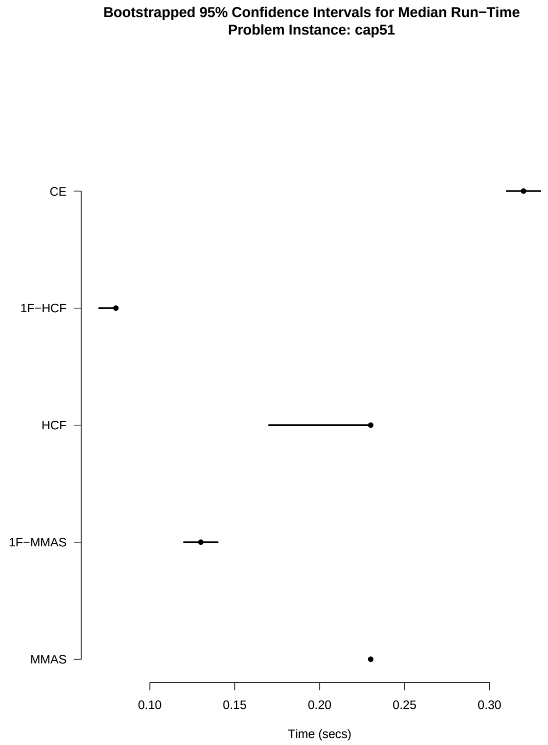
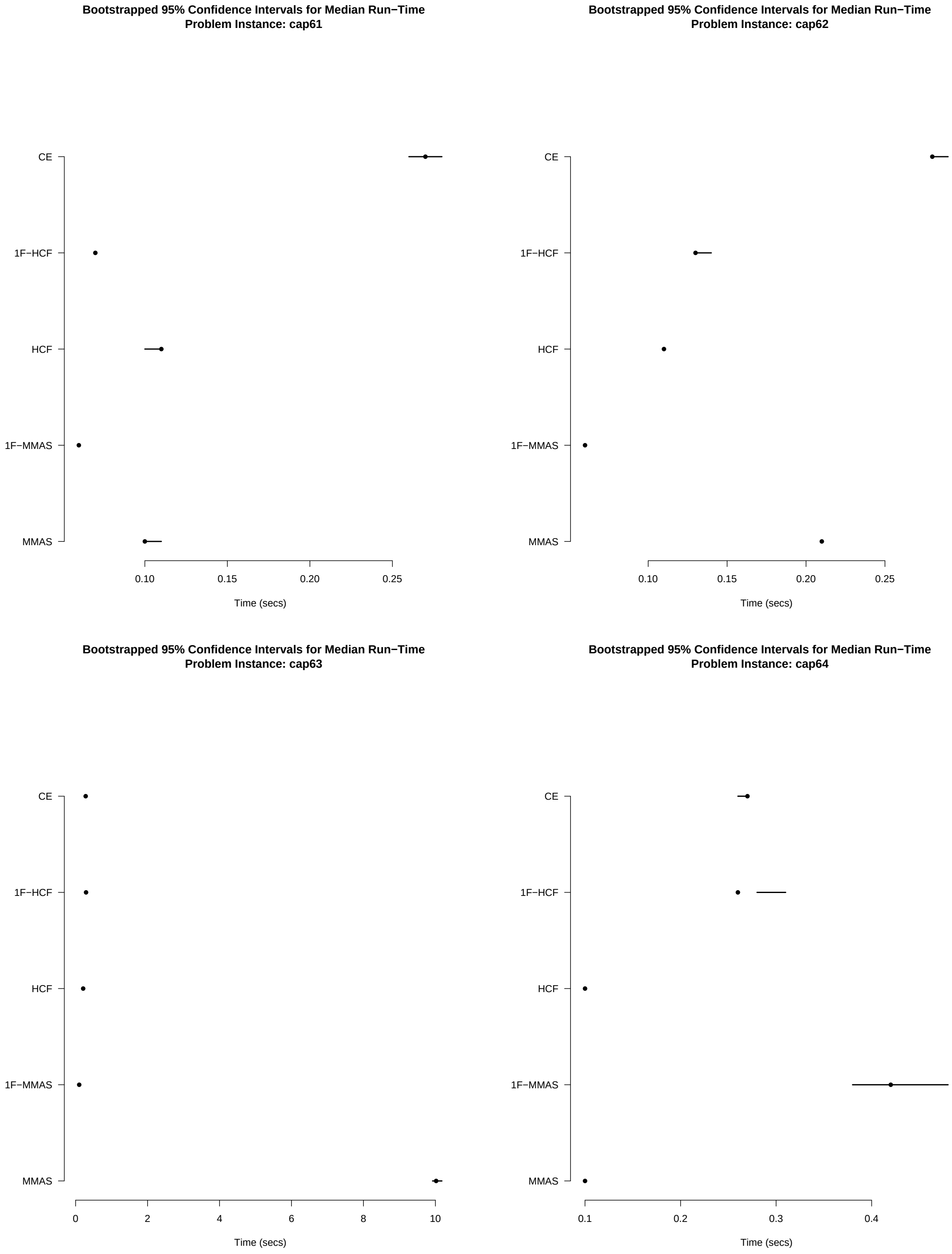

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap51
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.10
0.15
0.20
0.25
0.30
Time (secs)

Figure E.2



Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap61

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap62

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap63

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap64

Figure E.3

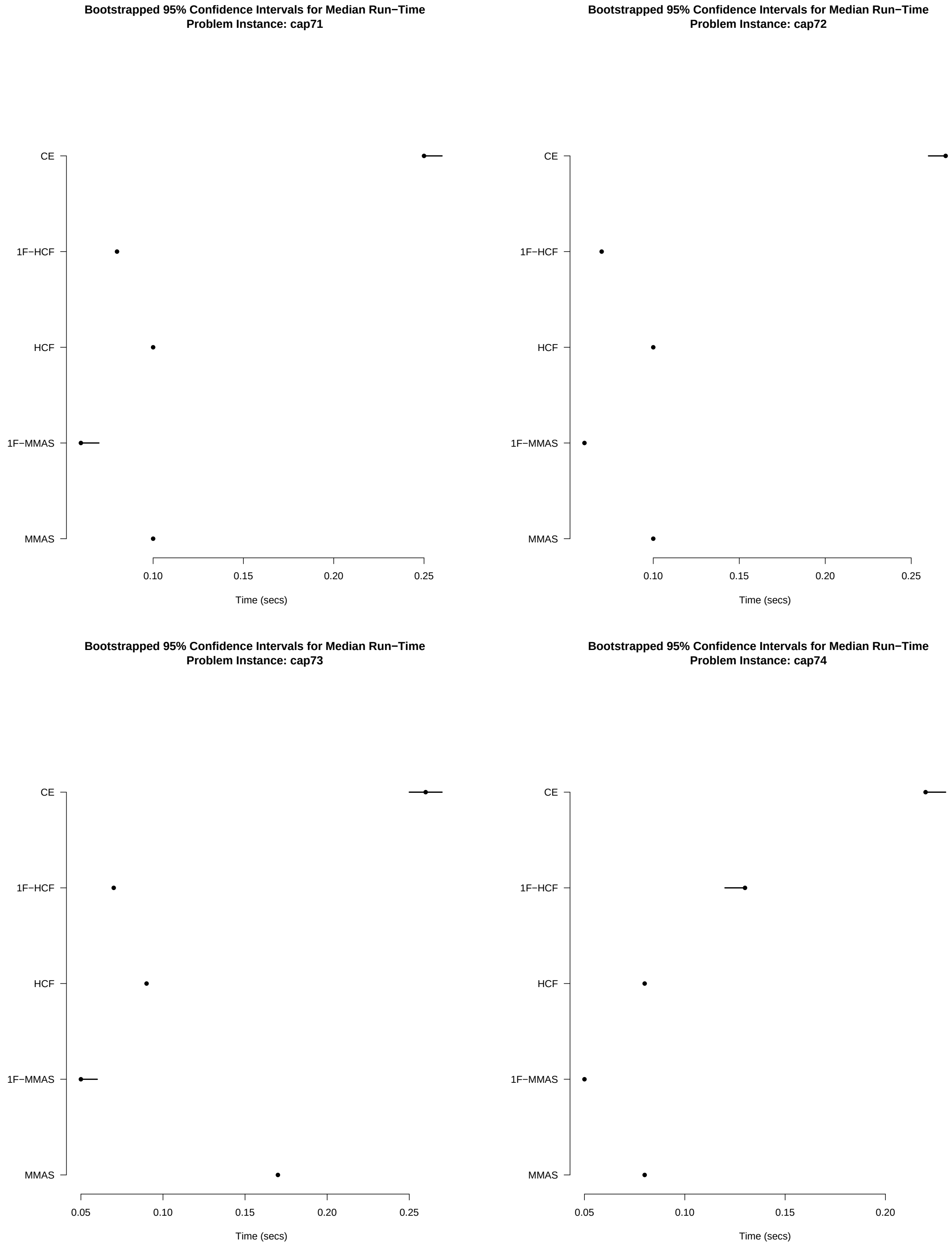

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap71
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.10
0.15
0.20
0.25
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap72
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.10
0.15
0.20
0.25
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap73
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.05
0.10
0.15
0.20
0.25
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap74
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.05
0.10
0.15
0.20
Time (secs)

Figure E.4

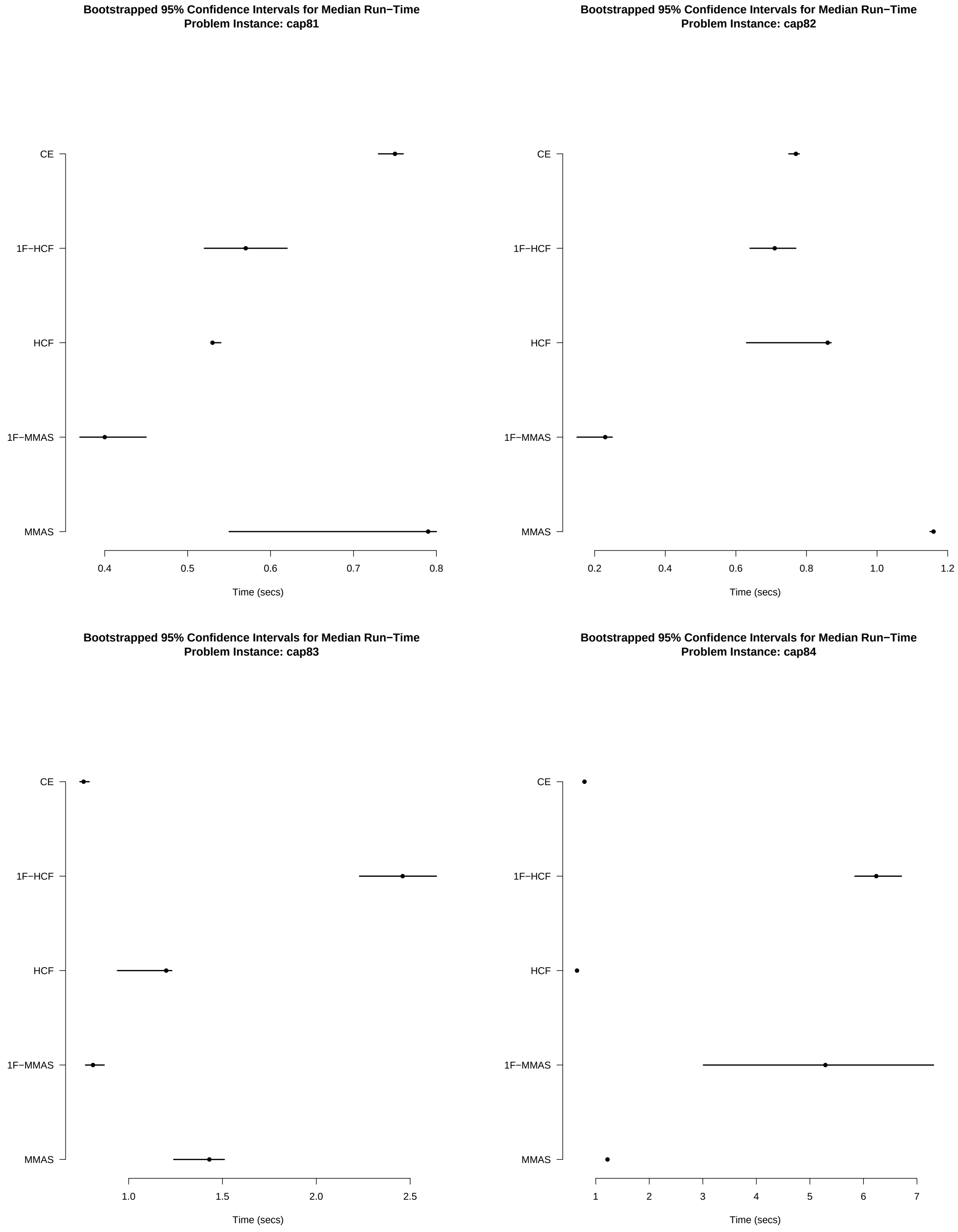

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap81
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.4
0.5
0.6
0.7
0.8
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap82
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.2
0.4
0.6
0.8
1.0
1.2
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap83
CE
1F−HCF
HCF
1F−MMAS
MMAS
1.0
1.5
2.0
2.5
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap84
CE
1F−HCF
HCF
1F−MMAS
MMAS
1
2
3
4
5
6
7
Time (secs)

Figure E.5

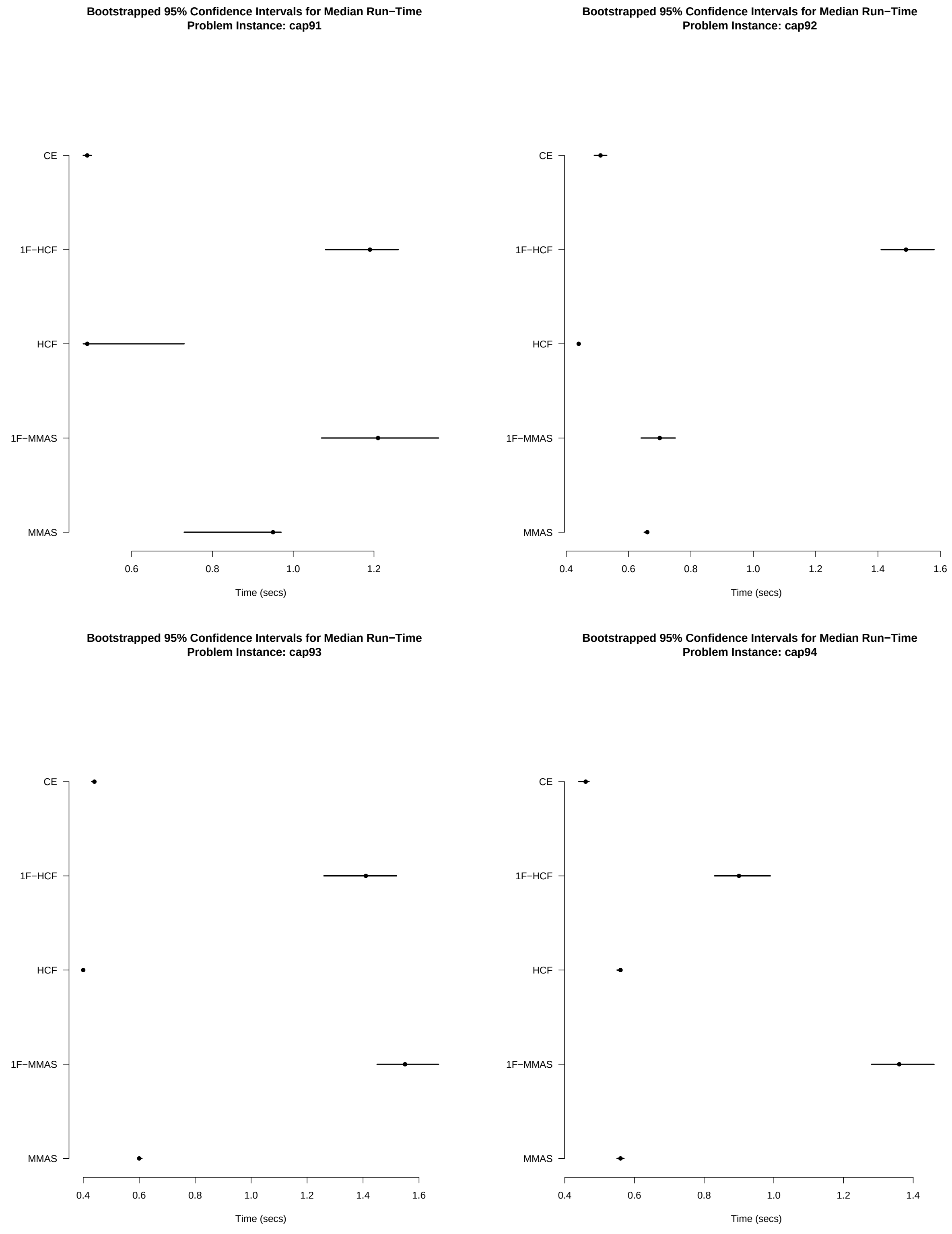
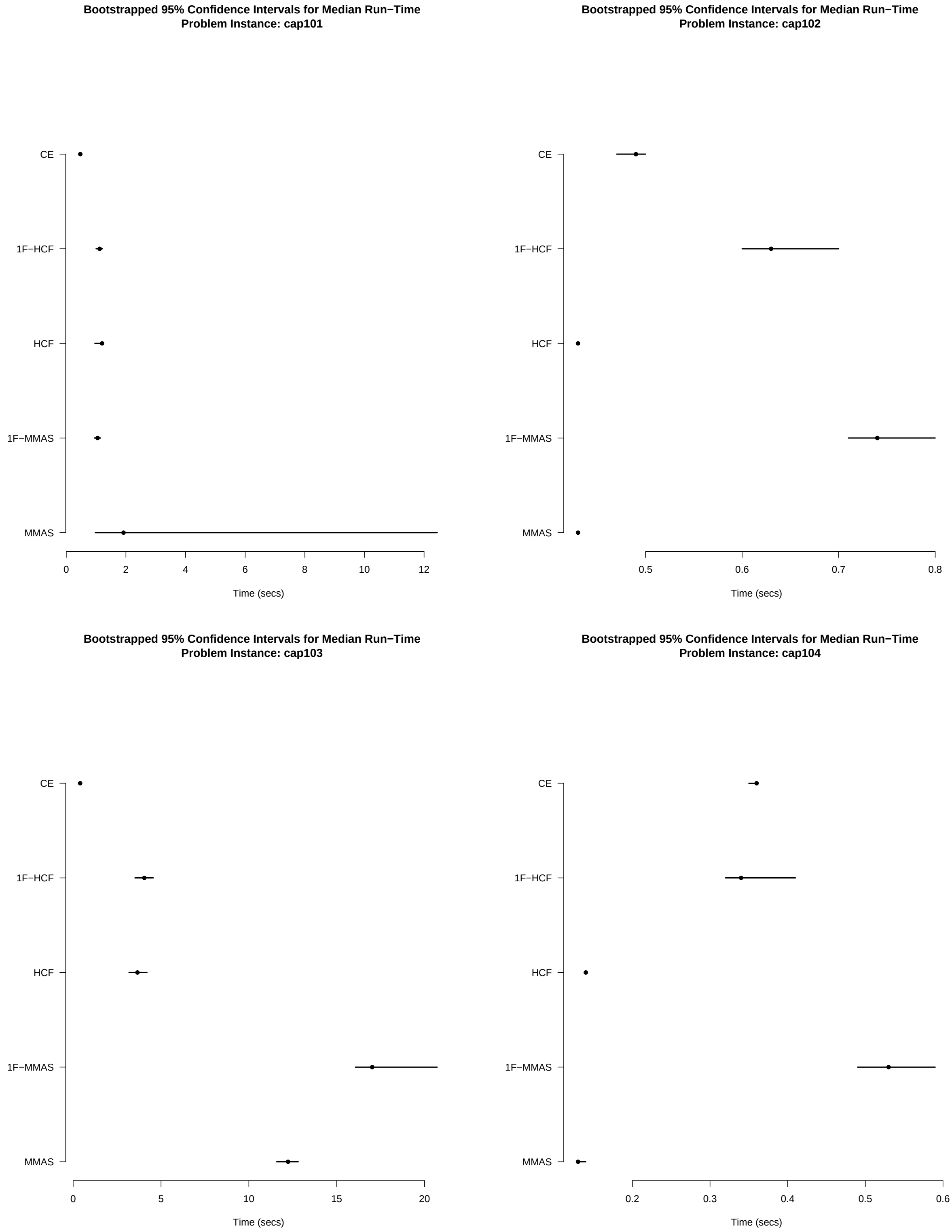

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap91
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.6
0.8
1.0
1.2
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap92
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap93
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap94
CE
1F−HCF
HCF
1F−MMAS
MMAS
0.4
0.6
0.8
1.0
1.2
1.4
Time (secs)

Figure E.6



Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap101

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap102

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap103

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap104

Figure E.7

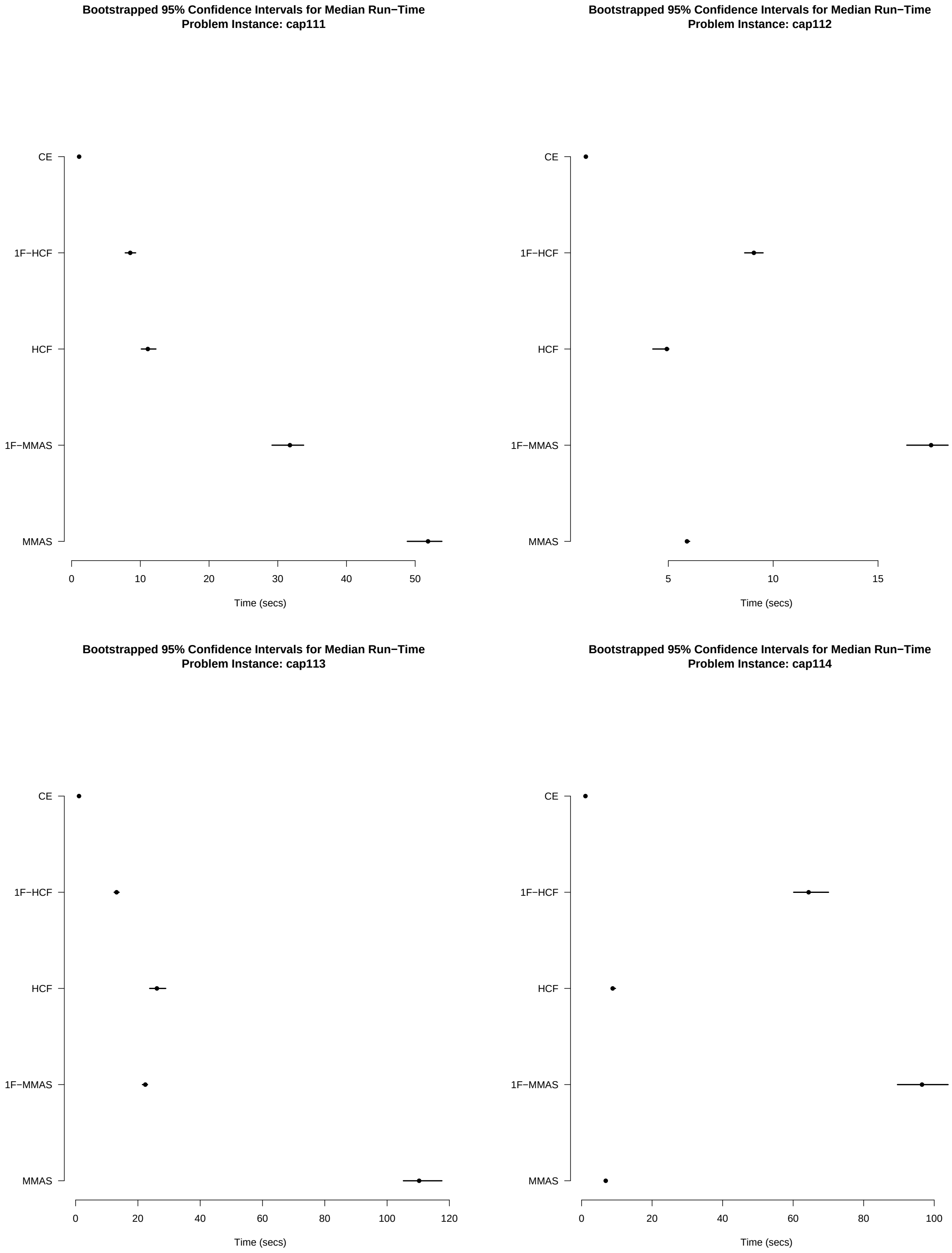
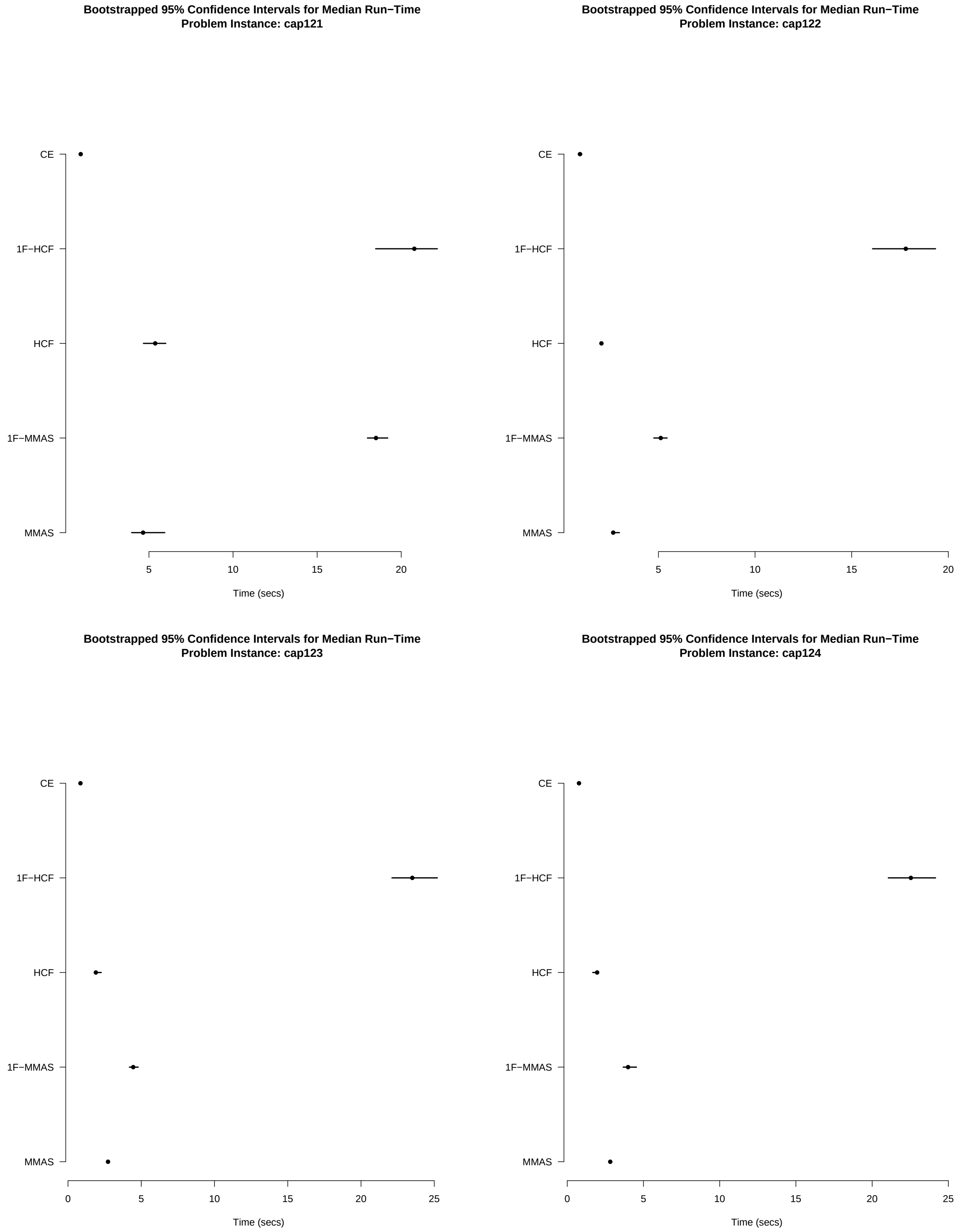

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap111
CE
1F−HCF
HCF
1F−MMAS
MMAS
0
10
20
30
40
50
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap112
CE
1F−HCF
HCF
1F−MMAS
MMAS
5
10
15
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap113
CE
1F−HCF
HCF
1F−MMAS
MMAS
0
20
40
60
80
100
120
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap114
CE
1F−HCF
HCF
1F−MMAS
MMAS
0
20
40
60
80
100
Time (secs)

Figure E.8


Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap121

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap122

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap123

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap124

Figure E.9

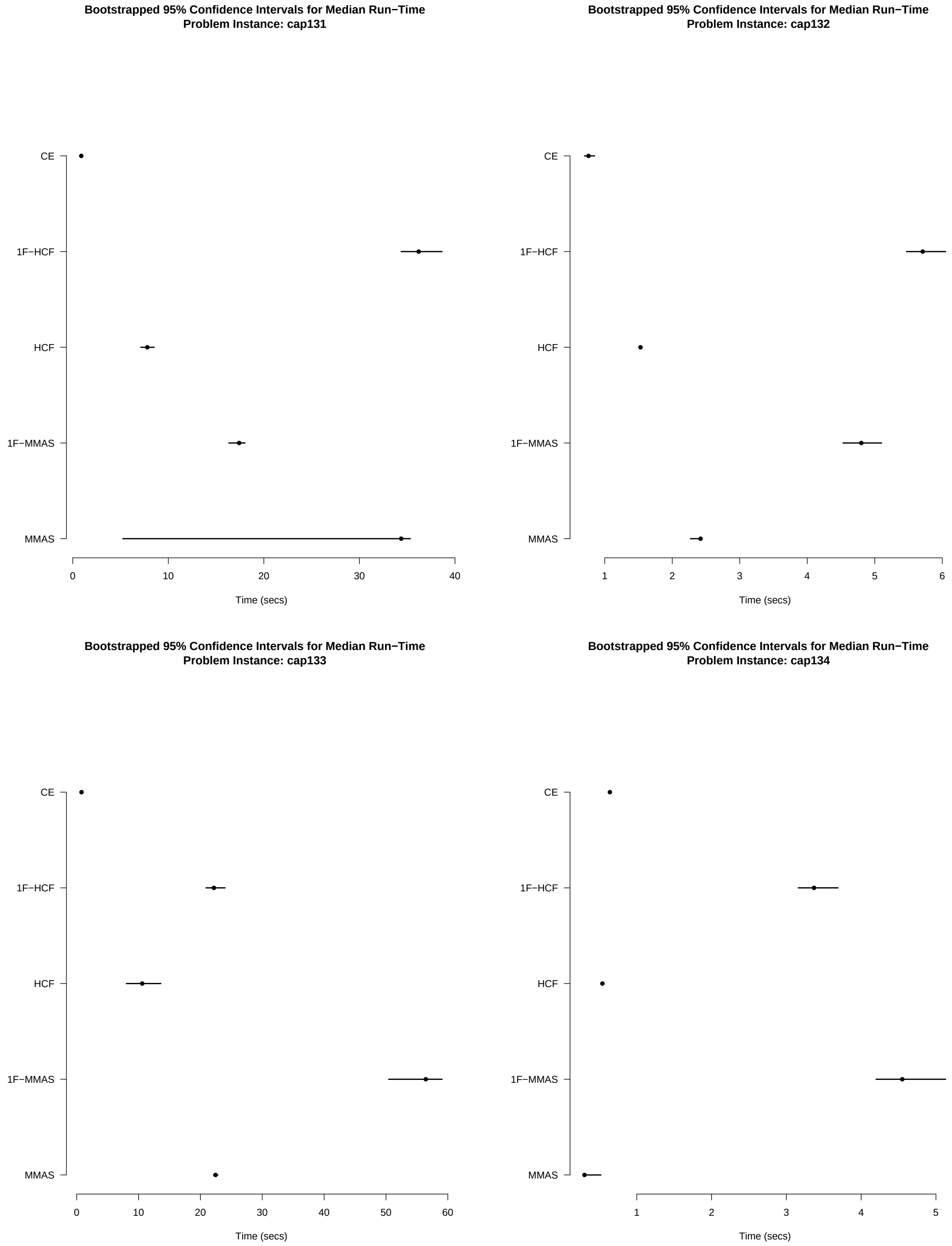

Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap131
CE
1F−HCF
HCF
1F−MMAS
MMAS
0
10
20
30
40
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap132
CE
1F−HCF
HCF
1F−MMAS
MMAS
1
2
3
4
5
6
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap133
CE
1F−HCF
HCF
1F−MMAS
MMAS
0
10
20
30
40
50
60
Time (secs)
Bootstrapped 95% Confidence Intervals for Median Run−Time
Problem Instance: cap134
CE
1F−HCF
HCF
1F−MMAS
MMAS
1
2
3
4
5
Time (secs)

Figure E.10