ISSN 1744-1986

Technical Report N° 2007/ 20

META-OPTIMISATION OF MIGRATION TOPOLOGIES IN MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

J Erlank

29 September, 2007

Department of Computing
Faculty of Mathematics, Computing and Technology
The Open University

Walton Hall, Milton Keynes, MK7 6AA
United Kingdom

http://computing.open.ac.uk

META-OPTIMISATION OF MIGRATION TOPOLOGIES IN MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

A dissertation submitted in partial

fulfilment of the requirements for

the Open University's Master of Science Degree in

Computing for Commerce and Industry.

JONATHAN ERLANK

(T6085951)

MARCH 2008

WORD COUNT (CHAPTERS): 17195

Preface

Acknowledgements

I would like to acknowledge the assistance given by my supervisor Dr. Lyndon Lee during this dissertation.

I would also like to acknowledge the assistance from my employer Credit Suisse for allowing study leave to complete this work and allowing use of computing resources.

Table of contents

List of Figures

List of tables

Abstract

Multi-objective evolutionary algorithms use a population-based approach and the principles of evolution to find Pareto-optimal solutions to problems with multiple competing objectives.

Unlike single-objective problems, in which there is often only one optimal solution (for example a global minimum for a given function), a multi-objective problem will have a set of *Pareto-optimal* solutions, all of which are equally optimal in the sense that they represent different trade-offs between competing objectives; and therefore no one solution can be said to be the 'best'.

Evolutionary algorithms (whether single- or multi-objective) frequently model solutions to a problem using a population of *chromosomes*, each of which represents a single encoded solution. These chromosomes are then subjected to repeated *evolutionary operators* to allow the population to iterate towards one or more optimal solutions. Such operators typically include *crossover* (sharing genetic material between chromosomes), *mutation* (allowing variation within a chromosome) and *selection* (choosing 'good' chromosomes from which to create the next generation).

Evolutionary algorithms are also frequently implemented in parallel, and a common paradigm is to use a number of distributed sub-populations executing on different machines. In this so-called island paradigm, the *migration operator* is introduced. In a similar manner to biological evolution, two isolated sub-populations might diverge genetically. Migration is a process that occasionally allows genetic material to move between sub-populations. Research shows that this process may have a beneficial impact on the *genetic diversity* of both sub-populations, thus improving the overall quality of the results. The way in which genetic material moves between sub-populations is dictated by the links between the sub-populations (termed the *migration topology*). In addition to the benefits, there is also a *cost* to migration: solutions must be distributed over a network or between running processes.

One class of problem that can be solved using an evolutionary algorithm is that of *network optimisation* – in particular the problem of finding an optimal topology of a network given certain criteria. Such problems can involve single or multiple objectives, depending on how the problem is defined.

In this research I combine the concepts of network optimisation, migration topologies, diversity and cost; and examine the effectiveness of a multi-objective evolutionary algorithm in finding a Pareto-optimal migration topology *for itself*. In this way, an identical multi-objective algorithm is applied at both the problem domain-level and the meta-level. That is, the multi-objective algorithm searches for Pareto-optimal migration topologies for a selected domain problem which is itself being solved by a parallel distributed version of the same algorithm. To the best of my knowledge, this is a novel application of the same algorithm in this way; and demonstrates the generality of such algorithms.

The primary hypothesis is therefore that it is possible to find such Pareto-optimal topologies using the same algorithm at two levels. The results obtained support this hypothesis.

1. Introduction

1.1. Background

Multi-objective evolutionary algorithms (MOEAs) use a population-based approach and the principles of evolution to find Pareto-optimal solutions to problems with multiple competing objectives.[1] Pareto-optimal solutions are ones in which these competing objectives are traded off against each other, and no solution can be said to be overall better than any other. The main concepts are illustrated in Figure 1 (reproduced from Abraham and Jain (2005)).

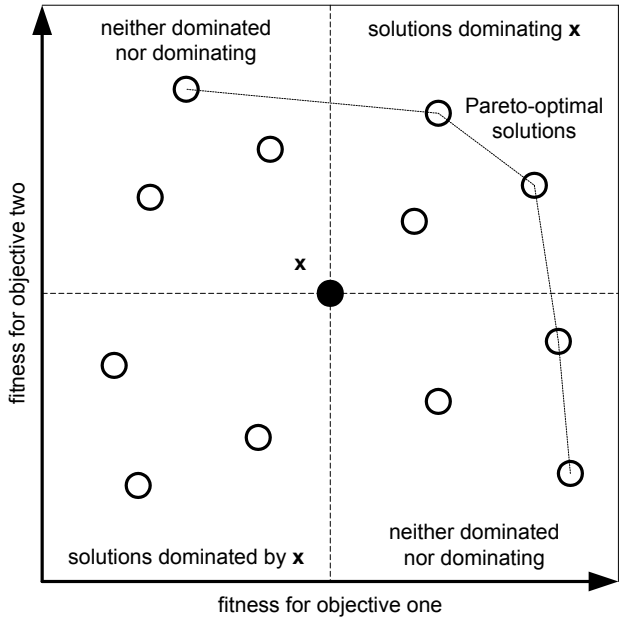

Figure 1 Pareto-optimal solutions and concept of dominance

In Figure 1, each solution has a fitness value for two objectives. With respect to solution **x**, the top right quadrant shows solutions that *dominate* **x**, because they have better fitness values for both objectives. Similarly, solutions in the bottom left quadrant are *dominated by* **x**. Solutions in the top left and bottom right quadrants are neither dominated by nor dominate **x** since they are better than **x** in one objective but worse in another. Finally, there is a set of Pareto-optimal

[1] Paraphrased from Deb (2001) Ch. 1

solutions, which contains the solutions that are not dominated by any other solution. (See Abraham and Jain for a more detailed explanation).

MOEAs are algorithms that search for Pareto-optimal solutions to a problem with two or more objectives. Like single-objective algorithms, they typically present many opportunities for parallelisation and we can therefore identify parallel distributed multi-objective evolution algorithms (PDMOEAs) as a important subclass of MOEAs.

PDMOEAs commonly divide a population of solutions into a number of distributed sub-populations, with periodic migration of some individuals between the sub-populations to help maintain diversity in the overall population. Migration plays the same role as in biological evolution, by allowing favourable genes to spread throughout the population. The method of migration in PDMOEAs varies from algorithm to algorithm and in most cases is chosen on a trial and error basis to achieve acceptable behaviour. The arrangement of the distributed sub-populations and the set of migration links between them is generally described as the *topology* of the PDMOEA.

This research investigates the use of a multi-objective evolutionary approach to obtaining an effective PDMOEA topology in which the cost of migration is to be minimised, and the overall diversity of the final population is to be maximised. High diversity is valuable because it may allow the algorithm to avoid becoming trapped in local optima; and low cost is valuable because it reduces the resources required by the algorithm. These two objectives allow the problem to be formulated as a multi-objective problem with Pareto-optimal solutions. This extends previous work on using evolutionary approaches to optimise other aspects of evolutionary algorithms and represents an application of an evolutionary algorithm at two levels: the *domain-level* and the *meta-level*.

The *domain-level* problem is the problem that is being solved by a PDMOEA, and it may be any problem with multiple competing objectives (for example, an engineering problem in which the weight of a component must be minimised while strength must be maximised).

The *meta-level* problem is specifically the problem of finding an optimal migration topology for the PDMOEA which is operating at the domain-level.

The remainder of Chapter 1 has three parts. The first part looks briefly at the general structure of a PDMOEA. The second section looks at prior applications of MOEAs in solving network design problems and optimising graph topologies, and discusses how the same techniques can be applied to optimising the topology of a PDMOEA. Finally, the last section draws these strands together and elaborates on the research question outlined above.

1.2. Parallel distributed MOEAs

Figure 2 illustrates the typical approach in a PDMOEA (adapted from Lin et al. (2002)). Although there exist numerous other approaches (discussed in Chapter 2), the multi-population approach is the most popular.

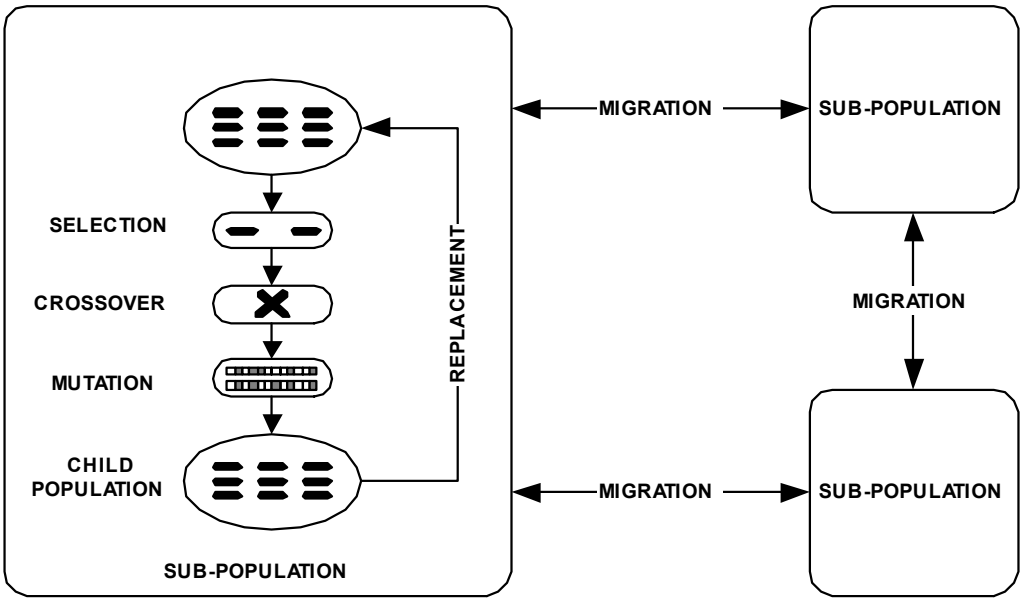

Figure 2 General operation of distributed evolutionary algorithms

In this approach, the selection, crossover and mutation operations occur within each sub-population. Selection is the process of choosing solutions with a high fitness value, crossover is the process of sharing genetic material between solutions, and mutation is the process of introducing variation into the population. Unlike a single-population approach, however, an

additional evolutionary operation is used: migration. Individuals are chosen for migration according to a predefined migration policy, and transferred between sub-populations. Depending on the nature of the algorithm and the chosen parameters, this may take place synchronously or asynchronously, and at various intervals. The goal is either to increase diversity or to improve convergence rates. Without migration, the population will simply consist of independent sub-populations which may be exploring the same region of the search space (thus wasting resources), or individual populations may become trapped in local optima and stagnate (thus potentially not finding a good set of solutions). In either case, the benefits of parallelisation will have been reduced or lost.

Note that the number of sub-populations, the links between them and the migration parameters may vary widely: that is, there are many migration topologies and many migration strategies.

1.3. Network and graph optimisation with MOEAs

This section discusses the application of MOEAs to network design problems and graph optimisation problems; and suggests how a similar approach can be applied to finding the Pareto-optimal topologies for a PDMOEA.

1.3.1. Network design problems

Network design problems appear often as real-world examples in the literature on both single-objective and multi-objective evolution. Solving such a problem generally means finding a network layout that delivers some desirable characteristic like high resilience, high throughput, low cost, efficient routing and so on.

Recent research (Banerjee and Kumar (2007)) describes the application of MOEAs to network topology design problems. Their work is summarised here as a representative example of the general approach. Their discussion is given in terms of the network model, the objectives and constraints, and the algorithm details. Further examples are discussed in the literature review in Chapter 2.

The design parameters for the real-world network model consist of

- the total number of nodes N
- a distance matrix D giving the physical distance between nodes
- a traffic matrix T providing the expected peak network traffic between pairs of nodes
- the types of network equipment (including amplifier equipment, which the authors note is an important part of real networks), and link costs
- the traffic delay model (Poisson or self-similar)

The objectives are to minimise the average packet delivery delay, and minimise the network cost. The average packet delivery delay is calculated according to the traffic model. The cost is given as the sum of the costs of the nodes, links and amplifier units.

Additional constraints are placed on the solutions: a flow constraint (ensuring that link capacities are not exceeded) and a reliability constraint (ensuring that a node failure does not cause the network to become unconnected).

The algorithm uses a set of chromosomes, each of which is a constant length bit string, and which represents a possible network topology. The chromosomes consist of two parts – the first part encodes the details of the network equipment at each node, and the second part encodes the details of the links between the nodes. As an indication, and using the formulae in Banerjee and Kumar (2007), the chromosome size for a network with 8 different types of network equipment and 36 nodes (the largest network simulated by the authors) can be calculated as 108 bits for the network equipment and 630 bits for the links, giving a total size of 738 bits. This shows that even relatively large networks can be reliably simulated.

The fitness of a chromosome is evaluated using Pareto-ranking. Pareto-ranking is simply a way of ordering solutions such that the rank of each solution is equal to one more than the number of solutions dominating it. A solution A is dominated by another solution B when B is better in at least one objective, while not being worse in any other objective. In other words, B is unambiguously a better solution. The fitness is calculated directly as

$$Fitness = \frac{1}{Rank^2}$$

The above objectives, constraints and encoding were used in the Pareto Converging Genetic Algorithm (PCGA), and applied to real data sets comprising networks of U.S., European and Chinese cities. Note that the simulation of each network (in order to evaluate the objective and hence determine the fitness) involved the use of other algorithms (notably Dijkstra's shortest path algorithm).

As mentioned, the above research is a good example of the approach taken in the literature to optimising networks using evolutionary algorithms, and shows how such NP-hard problems can be solved relatively well with an evolutionary approach. This leads us on to the more general view of such problems described next.

1.3.2. Graph optimisation problems

The work of Banerjee and Kumar is just one example of a class of network design problems. Such problems can be viewed as a specialised subset of a large set of abstract graph optimisation problems.

In general, it is intuitive (and obvious) to model such problems by using the mathematical abstraction of a graph consisting of vertices and edges. This has a number of advantages – graphs are very general, there is a mature body of knowledge surrounding them and they are both easy to understand and powerful. In the network design problem above, we can readily see that the second part of the chromosome is a direct encoding of a matrix of edges. Note that since the links are bi-directional, the graph is undirected – other problems may require a directed graph model.

We can picture a slightly more abstract view of these problems as illustrated in *Figure 3*. In this diagram, it is important to note that we are dealing with two distinct algorithms. The first algorithm shown schematically on the top left is attempting to find an optimal graph for the overall problem (or in the case of a multi-objective problem, a Pareto-optimal set). The second algorithm shown schematically on the right is a problem-domain specific algorithm that simulates

the network defined by a particular graph in order the determine the fitness value for the graph and hence drive the first evolutionary algorithm. In the network design example in section 1.3.1, the equivalents would be PCGA, and the network cost and delay calculating algorithms respectively.

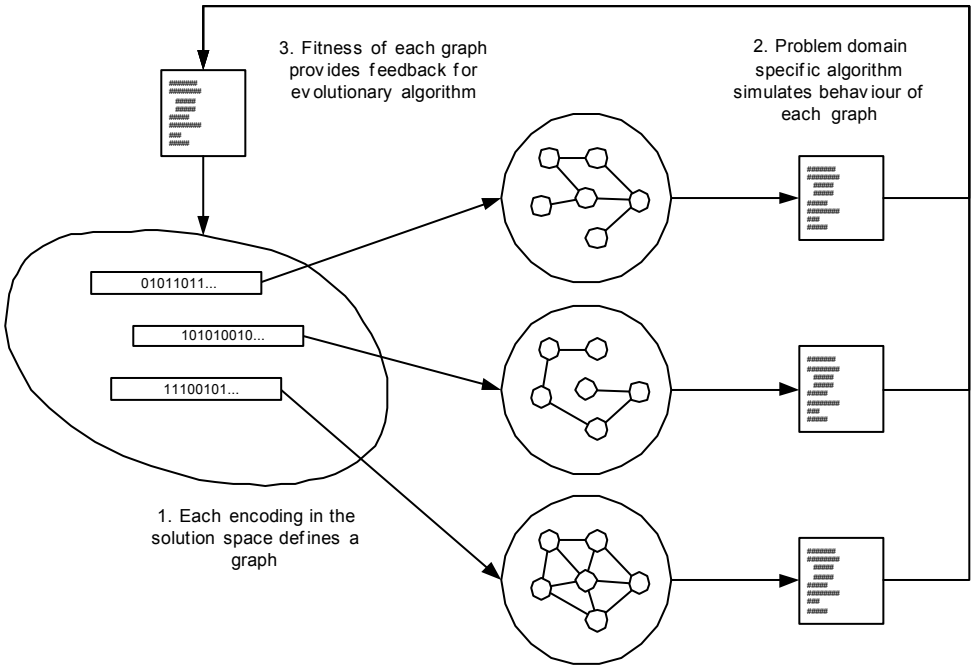

Figure 3 General graph evolution

The two algorithms are coupled by the encoded graphs and the fitness function. The problem domain algorithm interprets and simulates each encoded graph accordingly. This simulation may vary widely in complexity. The results of the simulation must allow a valid fitness function to drive the evolution of the population of graphs. We can envisage further coupling at this level if we allow domain information to further direct the evolution (for example by supplying information about which aspects of the graph might need modifying). In the network example above, the multi-objective nature of the algorithm means that the Pareto-ranking determines the fitness directly from the objective values.

Although the description above is general in the sense that the algorithm on the left may be a single- or a multi-objective algorithm; for the rest of this dissertation I will assume that it is a multi-objective algorithm.

1.3.3. Application to PDMOEA topology

The network problem and more general graph problem described above are, respectively, specialised and abstract views of the same problem. Both views are helpful when considering how to apply the same approach to a PDMOEA topology. The abstract view allows us to generalise the approach and apply it to other problems, while the specialised view gives us confidence that MOEAs have been successfully applied to similar real world problems.

As discussed above, PDMOEAs are often implemented by dividing one population into a number of distributed sub-populations, in which solutions in one sub-population might periodically migrate to other sub-populations. The entire population can therefore be described in terms of the topology (the way that sub-populations are connected and their relative size) and the migration parameters (the rate of migration and the number of individuals migrated in each iteration).

This leads directly to the possibility of using a similar approach to the one used by Banerjee and Kumar to find a Pareto-optimal set of topologies for a PDMOEA. This is shown conceptually in *Figure 4*. Unlike the network model described above, the problem domain algorithm *is itself a (parallel distributed) MOEA*, which operates using the population topology described by the individuals in the first population. For this reason, the first algorithm is described as a meta-level algorithm.

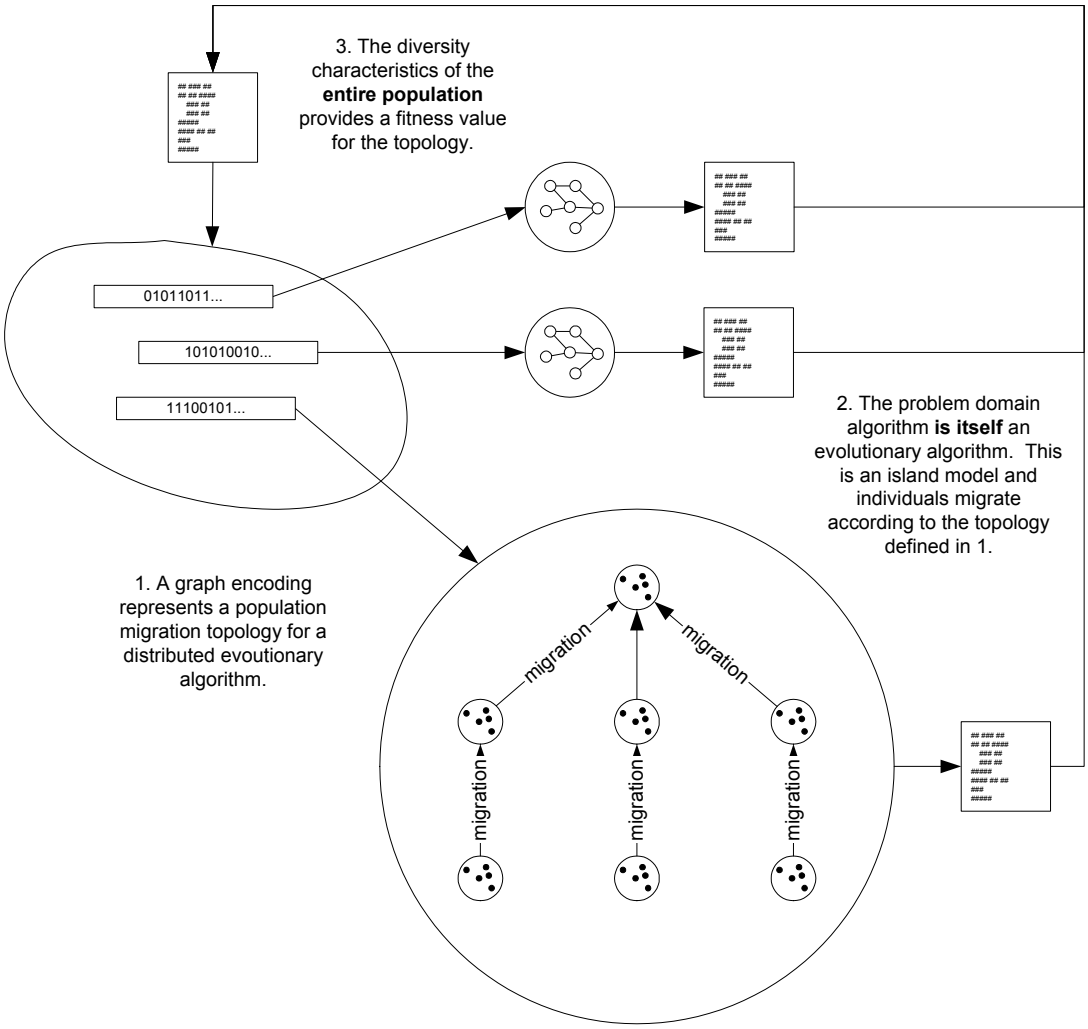

Figure 4 Meta-level evolution of PDMOEA topologies

Following Banerjee and Kumar, we can systematically encode the migration links and the associated parameters of a PDMOEA using some binary string. All such strings represent a solution space that can be searched to find Pareto-optimal population topologies. Using cost and diversity as example objectives, the Pareto-optimal set would contain solutions that minimise the cost of migration (perhaps measured by summing the number of individuals migrating per population epoch) and maximise diversity of the solutions (perhaps measured by calculating the average distance between solutions in the total population for each epoch[2]).

[2] An epoch is a number of generations

The population topology is modelled as a graph. Each node in the graph represents a sub-population. Each edge represents a migration route between two sub-populations. There are many possible ways of encoding and interpreting the graph and thereby simulating the PDMOEA behaviour. Some of these consideration are outlined here, and will be elaborated on in the remaining research.

Edges in the graph represent migration paths. If we assume a directed graph (that is, individuals can migrate from sub-population P_a to P_b, but not necessarily from P_b to P_a, then we would need a minimum of K^2 bits to encode the set of possible edges for K sub-populations. We can go further, however, and assign weights to the edges. The weight assigned to an edge indicates the number of individuals which migrate on each iteration. The exact mechanism for selecting individuals for migration between sub-populations has been the subject of some research (described in Chapter 2), but a common method is to select individuals using a *best-sent worst-replaced* strategy (Power et al. (2005)). Using a fixed number of bits N to encode the size of the set of migrating individuals M_i from sub-population P_i, we have the following formula (where e and q are used for readability).

$$0 \le e < 2^N$$
$$0 \le q \le |P_i|$$
$$|M_i| = q\frac{e}{2^N}$$

For nodes in the graph (i.e. sub-populations) we can vary the size of the sub-populations. Using variable sized sub-populations might lead to interesting results at the expense of a larger genome and more run-time complexity. For example, if we choose N bits to encode the size of a sub-population, a minimum sub-population size of m and a size increment of v, then the sub-population size $|P_i|$ would vary as follows (where d is used for readability):

$$0 \le d < 2^N$$
$$m \le |P_i| \le m + dv$$

With this approach the search space is accordingly larger.

The detailed examples above are representative of a possible approach, but are not exhaustive. However, we can now see how the topology of a PDMOEA population can easily be encoded in a form that can be subjected to a standard MOEA. This leads to the research question described in the following section.

1.4. Research question

This research tests experimentally whether it is possible to find a migration topology that improves the diversity of solutions obtained by a PDMOEA operating on a domain-level problem, by using a MOEA at a meta-level.

For fair comparison, the results are compared with topologies commonly used in the literature: ring, fully connected, unconnected and hypercube. The term *static topology* is used to describe these pre-selected topologies, as opposed to dynamically evolved topologies.

Obtaining a diverse set of solutions in the Pareto-front is important in multi-objective algorithms. By measuring diversity across an entire population made up of distributed sub-populations we can obtain one objective for the meta-level algorithm.

From the discussion above, and the literature review in Chapter 2, we can also see that the cost of migration (which is a factor in distributed algorithms in general) is not considered directly in most research. By establishing a measure for the cost of migration, we obtain another objective for the meta-level algorithm.

The literature review in Chapter 2 discusses how MOEAs can evolve network structures (i.e. showing application to real world problems). This approach can be readily adapted to find good migration topologies for PDMOEAs.

The research will thus show not only that finding migration topologies using an MOEA is generally possible; but also that migration topologies do influence diversity. This will be evident from the different diversity scores obtained comparing the results with hand-chosen topologies.

The following section discusses the contribution that this research will make to the general body of knowledge in this area.

1.4.1. Contribution to knowledge

This dissertation represents a building block in the area of self-adapting evolutionary algorithms by using an evolutionary algorithm to essentially optimise its own behaviour when running as a parallel distributed algorithm. The problem of optimising a parallel distributed topology in such a way is a novel application of such an algorithm. By allowing the cost of migration to influence the topology, we can assess the cost-benefit of a particular migration strategy in a more sophisticated way than simply measuring the speedup of a particular algorithm.

Most studies have relied on static topologies for parallel distributed algorithms (for example, ring, torus and hypercube topologies) which are arbitrarily pre-selected, or chosen on the basis of a small number of experiments. Although adaptive techniques have been used to modify algorithm parameters on the fly, to the best of my knowledge no attempts have been made to evolve a migration topology directly.

It is important to be clear that the focus of this research is on the application of a meta-level evolutionary algorithm to find good topologies for a parallel distributed evolutionary algorithm. The choice of domain-level problem is secondary to the meta-level problem. The actual empirical content of the research focuses on demonstrating that there is a set of Pareto-optimal solutions to the meta-level problem (i.e. the migration topology) that are at least as good as any that can be pre-selected.

In particular, this work can be placed in the context of the roadmap outlined by Lin et al. (2002), which is discussed briefly above and elaborated on in Chapter 2. It also suggests a number of promising further research directions which are discussed in the final chapter.

1.5. Objectives

This section describes the specific objectives necessary to support the research aims described in section 1.4. The techniques used to meet these objectives are described in Chapter 3.

1.5.1. Identify a suitable algorithm

The first step is to identify a suitable algorithm *A* that can be used as both a meta-level *(A_{meta})* and domain *(A_{domain})* algorithm. This requires examining the various classes of algorithm and any necessary characteristics, and considering what modifications might be required to support both the necessary meta-level and domain-level implementations. Note that it is not required that the meta-level and domain-level algorithms be the same, but this simply reduces development time and demonstrates the generality of the results.

1.5.2. Design a graph encoding

The second objective is to design an encoding format *E* for a MOEA topology along the lines suggested in the problem description above. This format needs to allow the migration topology and associated parameters to vary under crossover and mutation operators; and also allow the migration cost to be measured (since cost is one of the necessary objectives).

1.5.3. Select an appropriate test problem

A sample test problem *T* is necessary for the domain-level simulations. This needs to be one in which the diversity of the solutions can be readily tested (since the diversity is the second of the necessary objectives).

1.5.4. Select a set of static topologies

For fair comparison, a small set of static or well-defined topologies previously used in the literature is required. This is partly based on the literature and partly common sense.

1.5.5. Simulate the meta-level and domain-level algorithms

This is the main experimental work. A population of topologies is encoded using E and used as the population for a MOEA (the meta-level algorithm A_{meta}). Each topology is used to define the migration topology for a PDMOEA (the domain-level algorithm A_{domain}). Each PDMOEA is then simulated using the test problem T to evaluate the diversity of the resulting population. The diversity and associated cost of migration is then used to drive the evolution of the population at the MOEA level (the meta-level).

1.5.6. Comparison of simulated results

Since the primary research method involves simulation of the meta-level and domain level algorithms, development of a Java application is necessary. The full details of this part of the research is given in chapter 3.

1.6. Summary

This chapter introduced the background and the goals of the research.

- Multi-objective problems were discussed, and the concepts of Pareto-optimality and dominance were introduced.
- Key concepts for evolutionary algorithms that solve such problems were discussed.
- The application of MOEAs to network design problems was explained and related to the problem of finding migration topologies for a PDMOEA.
- The core idea of simultaneously using a multi-objective algorithm at a meta-level and domain-level was introduced. This led to the statement of the research question, and specific objectives to support the research.

The following chapter presents an in-depth literature review to explore this background in more detail.

2. Literature review

This chapter presents an overview of previous research in the field of parallel MOEAs. The following threads underlie the research behind this dissertation, and each is considered in turn.

Firstly, the general approaches to parallelisation of evolutionary algorithms and MOEAs in particular are discussed. Different parallel paradigms and algorithms are covered, as well as the importance of migration and the costs associated with it. The role of diversity and convergence is considered. This section provides the general background and core concepts for the problem outlined in Chapter 1.

Secondly, previous work on self-adaptive approaches to evolutionary algorithms are discussed. This provides substance to the notion of using an adaptive approach to parameterise an evolutionary algorithm, especially with regard to the migration topology.

Finally, the use of MOEAs to optimise network design problems is discussed, which illustrates how similar problems have been solved in other fields.

A short summary at the end reiterates the key points.

2.1. Parallelisation of multi-objective evolutionary algorithms

Genetic algorithms are frequently classed as examples of 'embarrassingly parallel' problems. This means that they are generally very easy to partition among large numbers of processors, although much depends on both the method used to parallelise the algorithm and the problem domain itself.

For parallelisation of algorithms in general, Amdahl's law (Wikipedia, accessed 21-May-2007) places an upper bound on the speedup that can be achieved by applying multiple processors to a problem. This applies regardless of the problem domain or specific details of the algorithm. Despite this limitation, the nature of MOEAs means that a significant speedup can be obtained fairly easily. Numerous attempts to parallelise MOEAs can be found in the literature, and the high level principles and motivations are described below.

2.1.1. Paradigms

This section discusses different paradigms used in parallelisation of genetic algorithms, especially multi-objective ones.

Parallel genetic or evolutionary algorithms have been used for optimisation problems for many years and Cantu-Paz (1997) is a detailed survey of the research into parallel genetic algorithms in general. However, multi-objective algorithms have peculiarities of their own that demand special treatment (Lopez Jaimes and Coello Coello, 2005). Van Veldhuizen et al. (2002) give a general discussion of the issues encountered when parallelizing MOEAs in particular. All three of these publications use essentially the same classification of algorithms into three paradigms: master-slave, island and diffusion. The first two paradigms receive the most attention in the literature, and much of the following discussion focuses on the island paradigm.

The *island paradigm* partitions the population of solutions into sub-populations, each of which is processed independently. Usually some method of sharing individuals between populations is used, either to maintain genetic diversity or address convergence issues; and migration techniques have been the subject of some research, not least with respect to addressing the cost of communication (Cantu-Paz, 1997).

The *master-slave paradigm* dedicates at least one processor to dispatching work (usually objective function evaluation) to a set of slave processors and subsequently collating and processing results. In this model, it seems likely that the master processor will either be idle for at least some of the time or will be a bottleneck.

The *diffusion paradigm* (or fine-grained parallelism) involves arranging solutions according to a chosen topology in which the evolutionary operators (EVOPs) will operate only within the local neighbourhood. This approach is suitable for implementation on massively parallel machines (Cantu-Paz, 1997) and also carries high communication costs (Van Veldhuizen et al., 2002).

Van Veldhuizen et al. also present a task decomposition of a MOEA, which identifies fitness evaluation as being readily parallelised, and recombination, mutation and selection as serial tasks.

However, it is not readily apparent why these latter tasks cannot also be parallelised, and Cantu-Paz refers to earlier work by Robertson (1987) in which selection, mating and crossover are parallelised.

With a few exceptions, most of the literature is relatively unsystematic and empirical in terms of approaches to parallelizing MOEAs. Some researchers have provided a generalised overview of parallel genetic algorithms (e.g. Alba and Tomassini (2002)) or generalised population structures (Mehnen et al. (2004)).

When considering any MOEA algorithm it is easy to identify parts of the algorithm that are intrinsically serial. For example, the Non-Dominated Sorting Genetic Algorithm (NSGA-II) described by Deb et al. (2000) requires a non-dominated sort of the overall population (i.e. the union of the archived solutions and the parent population) in order to create a child population. It is not easy to see how this process can be parallelised. If we consider this part of the NSGA-II algorithm as serial, then we have our upper bound on the speedup that can be achieved for NSGA-II in general. Intuitively it seems likely that similar constraints would apply to most generational algorithms, since each generation would require some serial work to collate results or otherwise consider a population as a whole.

Alba and Tomassini (2002) provide a detailed survey and taxonomy of parallel genetic algorithms, although they do not discuss multi-objective algorithms. They identify a broad family of algorithms that uses *panmictic* populations. In these, any individual can potentially mate with any other. This is distinct from the set of algorithms that use a structured population in which mating is distributed. Panmictic algorithms can be further divided along a spectrum from *generational* to *steady-state* algorithms. At one extreme, generational algorithms replace the entire population at each iteration of the algorithm, while steady-state algorithms replace a single individual at each iteration. Clearly there is a continuum between these extremes.

The island paradigm has received the most attention, with many different approaches being taken to partitioning either the decision or solution space, and to migration strategies between islands. In some cases, the distinction made by Van Veldhuizen between the island and master-slave

paradigm is blurred. For example, Tanimura et al. (2003) suggest a grid approach which uses a large connected grid of heterogeneous machines (e.g. the Internet) each processing a sub-population. This approach still requires the coordination of an agent (i.e. as in a master-slave paradigm). The grid approach presents difficulties on a number of grounds. It requires an agent to distribute work to the nodes in the grid, which may become a communication or processing bottleneck. The agent also requires special logic to coordinate work between nodes with different capabilities.

Hiroyasu et al. (1999b) describe an algorithm that uses the island paradigm, but introduces an extra sharing step. For each iteration of the algorithm, the sharing step gathers the populations from each island into a single population and removes individual solutions according to how densely they are clustered in the solution space. The purpose of this is to maintain diversity over the entire population, but the authors note that the sharing step takes a lot of time. The authors do not provide detailed analysis of what proportion of time is spent in each phase of the algorithm. This could be described as a crude attempt to maintain diversity by migrating all individuals at each step.

Another method of partitioning the search space into multiple islands is given by Lopez Jaimes and Coello Coello (2005). They divide the search space into a hierarchy of islands according to the numerical precision[3] of the decision variables. However, the authors note a number of difficulties with this arrangement which are not described here, except to mention that they require changes to the algorithm to ensure that all islands are contributing efficiently to the solution. In other words, ensuring that the parallel implementation is efficient can impose additional overhead to deal with complexities arising from the island paradigm itself.

From the above we can see that a range of different techniques is encountered in parallelisation of genetic algorithms. What should also be clear is that by far the most popular and effective

[3] The authors use the term resolution rather than precision.

approach is the island model in which multiple independent sub-populations evolve in isolation, usually with periodic migration between the sub-populations. This leads us to a discussion of migration.

2.1.2. Migration and associated costs

Most of the common approaches to parallelisation using the island paradigm must pay attention to the migration strategy. Alba and Tomassini (2002) have pointed out that the population structure may have a more significant impact on the efficacy of the algorithms than the parallelisation itself. They note the interesting result that the use of a structured population in a genetic algorithm may actually lead to superior numerical performance. This can occur regardless of whether the algorithm is actually executed on a parallel architecture or not.

In other words, the distribution of the overall population is what makes the algorithm effective, not the fact that the sub-populations are evolving in parallel. This suggests that a naïve approach to parallelisation will not necessarily be the best approach, especially if excessive migration takes place, since that will carry a cost, but provide no benefit.

Many existing attempts to parallelise MOEAs must go to some lengths to manage the overhead of a distributed processing model. It is easy to see how a problem domain in which computing the fitness evaluation functions is relatively expensive will be an example of such a problem. This is borne out by Xu et al. (2005) in parallelisation of genetic algorithms for spectroscopic analysis of X-rays using a master-slave model. The authors used a version of NSGA-II modified for parallel execution. In this case evaluation of the spectral model is relatively expensive and they observed a speedup of approximately 100 when using 140 processors.

Alba and Tomassini (2002) further categorise structured population algorithms into distributed and cellular algorithms. These correspond to coarse-grained and fine-grained classifications given elsewhere. The authors discuss this classification at length. They describe some results concerning population sizing, topologies and migration rates. The authors emphasise the need to make optimal choices of topology to balance communication and computation, and comment on

the high diversity usually obtained by distributed algorithms. Once again, we can see how the choice of topology has an impact on the results obtained.

In some cases the island paradigm must take special steps to avoid multiple islands carrying out redundant processing. For example, the self-adjusting approach described by Zhu and Leung (2002) allows islands to coordinate the search in such a way that activity in one island will influence the search direction in neighbouring islands. This seems to be a rather mechanical way of avoiding the problems of overlapping search spaces, and will add to the overhead of the algorithm. While this does not correspond directly to migration, it will carry a similar overhead, because information must be exchanged between different islands in the same way as migration of individuals.

The island paradigm may also be inefficient even when the Pareto-optimal front is partitioned into regions, each of which is assigned to a processor. To ensure efficiency, this requires *a priori* knowledge of the shape of the front (Lopez Jaimes and Coello Coello, 2005). If the partitioning is not optimal, then much processing time could be wasted. In principle, a distributed algorithm that intrinsically maximises diversity with optimal migration parameters would be a better approach than relying on a (possibly inaccurate) partitioning of the search space.

An approach which demonstrates a crude form of migration is given in Streichert et al. (2005). In this case a number of island populations are gathered and redistributed at regular intervals. This represents an extreme approach, in which diversity is maximised in the overall population at the expense of migrating all individuals in each generation.

Hiroyasu et al. (2000) compare the performance of a fine-grained and a coarse-grained algorithm on a parallel system consisting of 16 PCs. The key factor in such a system is that the network communication between nodes is relatively slow. They conclude that the fine-grained paradigm is not suited to this kind of architecture because the network costs outweigh any parallel performance improvement.

From the above we can see that migration plays a beneficial role in the efficacy of a PDMOEA, but that there is an associated overhead that should be taken into account. In the context of this research, we can see that the cost of migration can easily be taken into account as an objective in a meta-level MOEA which optimises the migration topology.

2.1.3. Diversity and convergence

It should be apparent from the above discussion that high diversity and efficient convergence are both desirable in MOEAs. Some specific research is discussed in more detail here.

The problem of how diversity should be measured is important, and is essential if we want to use a diversity measure in the operation of a meta-level MOEA. Hiroyasu et al. (2005) use a concept of *cover rate* to measure the diversity of the solutions in the *objective space*. The objective space is the n-dimensional space in which each dimension corresponds to one of the objective functions – for example, the objective space for a problem with two objectives is a simple Cartesian plane.

Hiroyasu et al. divide the objective space into a number of divisions N between the minimum and maximum values for the objective function and simply count the number of divisions M that contain an individual. The ratio $M:N$ is the cover rate, with values close to 1 indicating high diversity. This measure is simple, but can be criticised on two grounds: the minimum and maximum may not necessarily be known in advance; and the measure does not account for crowding (that is, solutions that are close together), since the measure makes no distinction between a division containing many solutions (highly crowded) versus a division containing only one solution (uncrowded).

Other diversity measures can be found in the workings of individual algorithms. A *crowding distance* measure is used in NSGA-II (Deb et al., 2000), which eliminates individual solutions according to how close they are to other solutions, using the size of an enclosing box in the objective space. Other approaches in the literature use measures based on the average Euclidean distance between individuals.

Berntsson and Tang (2003) present a convergence model, which explores the impact of different migration parameters and topologies. They focused on the use of unreliable distributed networks in which nodes may fail; and investigated the impact on convergence accordingly. Using the Internet as an example, latency and bandwidth problems make it desirable to reduce the data transfer between populations; and for reliability reasons, suggest that a fully connected topology would be best. However, we would expect that a fully connected topology would increase data transmission costs, so there is a trade-off between these two goals. This again suggests that the necessary convergence and diversity characteristics come at the cost of migration; and that it should be possible to find an optimal topology with a multi-objective algorithm.

It should be clear from the above that the topology or population structure used in parallel genetic algorithms influences the convergence and diversity of solutions. We can now consider how self-adaptive techniques have been used to adapt migration parameters.

2.2. Self-adaptive techniques and adaptation of migration parameters

Lin et al. (2002) present an overview of self-adaptive multi-population genetic algorithms (MGAs). Although they do not discuss multi-objective algorithms, their discussion is equally relevant to single- and multi-objective algorithms. They note that the performance and behaviour of MGAs is "heavily affected by an appropriate choice of parameters, such as connection topology, migration method, population number, migration interval etc.". They suggest that self-adaptation is an effective scheme for parameter setting for genetic algorithms, and categorise such adaptive issues into three categories: population structures, genetic operations and operation rates. Finally, they describe a lattice of algorithm classes, each cell of which represents those algorithms characterised by the adaptation of some of the adaptive categories. For example, one class in the lattice represents those algorithms in which the population structure is adapted. This class is further subdivided according to whether the sub-population size, number, distribution or topology is adapted.

Lin et al. thus present a roadmap for further research into self-adaptive algorithms. This provides a context within which the proposed research question described above can be placed. Although the use of a meta-level algorithm to evolve population topologies does not fall exactly into their categorisation, the principle remains the same: to use adaptive techniques to modify the behaviour of a genetic algorithm.

In a different paper by two of the same authors, Lin et al. (2004) describe experiments in adapting migration parameters. In particular, they note the sensitivity of MGAs to migration parameters, and cite previous experiments in which migrating too frequently or too infrequently degraded the performance of the algorithm. They therefore devise a scheme in which the migration rate and interval are adapted, and find that this approach competes with hand chosen parameters.

Hiroyasu et al. (1999a) discuss the use of a randomised migration rate for distributed genetic algorithms in general; while Power et al. (2005) discuss a technique whereby the diversity of the population is taken into account when selecting individuals for migration. In this method, migration is triggered by a low population diversity score. While this is not precisely a self-adaptive approach, it does illustrate again the idea that diversity and migration are closely related. The population topology in their experiments used a fully connected topology, which on the face of it seems like an arbitrary restriction.

Berntsson and Tang (2003) discuss convergence for asynchronous PDMOEAs. Their experiments focus on investigating parameter relationships to help define a migration policy for Internet based parallel genetic algorithms. In the same way that diversity is affected by migration parameters, so too is convergence; with migration topology, rate and intervals affecting the results. Perhaps unsurprisingly, they find that the migration induced selection differential is largest in a fully connected topology.

The relevance of these results is to show that migration parameters and characteristics have an impact on the performance and quality of the results obtained by multi-population evolutionary algorithms. This underpins the research proposal in this dissertation, which is to apply adaptive techniques to finding population topologies for PDMOEAs. In particular, the finding of

Berntsson and Tang above give us good reason to think that we will find a trade-off between communication costs and the diversity of the final solutions.

2.3. Evolutionary algorithms applied to network/graph optimisation problems

This section considers the application of MOEAs to optimising network design problems, and relates back directly to the problem outlined in Chapter 1.

Pullan (2002) describes a multi-objective approach for optimising network survivability. A given network with a fixed set of nodes and edges has a fitness value assigned based on quantifying the impact on survivability of removing an individual node from the network. The survivability measure is proposed by the same author in Pullan (2002). The survivability S of the network is the sum of the individual survivability measures for each node ΣS_i. The problem becomes a multi-objective one when the variability var(S) is also considered. Maximizing S and minimizing var(S) will define a Pareto-optimal set in which there is a trade-off between survivability and the significance of any particular node.

Roy et al. (2002) describe the application of the NSGA algorithm to the problem of optimising QoS (Quality of Service) routing within a network for wireless multicasting. They choose three objectives to establish non-domination relationships between solutions: (i) maximising the probability of meeting delay constraints, (ii) maximising the probability of a path in the multicast tree providing the required bandwidth and (iii) maximising the residual bandwidth after the necessary bandwidth has been allocated to the multicast tree. The authors use the NSGA algorithm (Srinivas and Deb, 1994) and find a set of multicast paths using a depth-first search. The network under simulation does not vary once the algorithm has started: the problem is to find the multicast routes; and the chromosome used in the population is an encoding of the multicast paths in the network, not the network itself.

Banerjee and Sharan (2004) discuss a single-objective approach to optimising wavelength allocation to paths in optical networks. They present a solution encoding using the k-shortest

paths for selected pairs of nodes. This approach is again using a static network definition but searching for paths within the network.

Banerjee and Kumar (2007) describe a multi-objective solution to minimise the cost and average delay of an Internet-like network subject to flow and reliability constraints. This work has already been discussed in more detail in Chapter 1.

Finally, as an aside, it is worth mentioning the work in Mehnen et al. (2004). They propose an elegant way of describing a population structure using hypergraphs. This provides us with a generalised way of modelling any population structure. A hypergraph, like a graph, consists of vertices and edges, except that instead of connecting just two edges; a hypergraph edge may connect any number of nodes. Using such a model, any arbitrary population structure can be described. For example, using edges to partition the mating pool, a diffusion or cellular algorithm would include edges linking only a few neighbouring individuals; whereas a panmictic population would include one edge containing all nodes. It is easy to see that more than one such graph might be used to represent different aspects of the population. This has relevance in the same way as the traditional graph representation discussed in Chapter 1: it is relatively easy to encode a hypergraph using a bit string, and hence use these bit strings as chromosomes in a population of hypergraphs for an evolutionary algorithm. However, the complexity seems high, and hypergraphs will be left as possible future research directions.

2.4. Summary

This chapter has discussed some of the key aspects related to the research described in this dissertation.

- MOEAs lend themselves to parallelisation, and PDMOEAs are an important class of algorithm.
- The island paradigm is commonly used in PDMOEAs, especially ones that try to make use of cheap networked resources such as PC clusters or heterogeneous nodes on the Internet.

- The population structure in any PDMOEA dictates the effectiveness of the algorithm.
- The cost of migration in the island paradigm is not always taken directly into account when assessing an algorithm.
- The diversity of solutions obtained by PDMOEAs can depend on the migration topology. Good diversity is an important goal for a PDMOEA.
- Self-adaptive techniques can be used successfully to modify the behaviour of a multi-population genetic algorithms to improve the results. Previous research has suggested that evolving migration topology is an open area of research. The application of this principle to PDMOEAs is a natural step.
- MOEAs have been successfully applied to network design problems and the same approach can be applied to optimising PDMOEA topologies.

The above points, taken in turn, justify the research: to use a meta-level multi-objective evolutionary algorithm to search for optimal network topologies for a PDMOEA, using migration cost and a diversity measure as objective functions.

3. Research methods

This research is founded on some key conceptual insights. The generality of multi-objective algorithms is readily apparent from reviewing the literature (Chapter 2). It is conceptually appealing to use the same algorithm at two levels, and in such a way that only the problem specific aspects need to be different. Specifically, the mutation, crossover and selection operators are identical, as well as the multi-objective specific operators (non-dominated sorting and crowding operators). Only the encoding and interpretation of the solutions must be different: in the case of the meta-level algorithm, the encoded solutions are interpreted as graphs; while in the case of the domain-level algorithm they are interpreted as numeric values.

The research method includes both analytical and empirical work. Since the research question aims to show that it is possible to use an MOEA to evolve a viable topology for a PDMOEA, the research method is primarily intended to provide evidence that the approach will work, and hence a working software implementation and adequately analysed results is sufficient to show that this is the case. Chapter 4 presents the results from the experimental work and Chapter 5 presents the analysis.

The method is similar to that used by previous researchers. Many of the papers reviewed in Chapter 2 follow a fairly typical pattern of (a) proposing and explaining the details of an algorithm (b) presenting empirical data gathered from an implementation of the algorithm to support claims. Banerjee and Kumar (2007) is a robust example of this approach.

Alternative, and somewhat more analytical approaches can be found in Alberto and Mateo (2004), and Arabas and Kozdrowski (2001). In the former, the authors provide some theoretical study of the correctness and complexity for graph representations of multiple populations. It is possible that this could lead to more formalised analysis of the role of migration in distributed populations. This however, would require a high level of abstraction and detailed analysis, and it is not clear that there is sufficient theoretical grounding in the literature for such an attempt. In the latter study, the authors focus on the details of the problem domain and provide a fairly robust

mathematical analysis of it. Since this research does not require such a heavy dependence on the details of the problem domain, such a level of analysis is not appropriate in this case.

The approach for this research is thus similar to other research, with Banerjee and Kumar (2007) being a typical example.

3.1. Analytical component

This section describes the general principles applied during the literature review and in construction of the experimental component of this research. Section 3.2 describes the experimental component in detail.

The necessary analysis fell into these key areas.

(a) Selection of a multi-objective evolutionary algorithm that is suitable for use at a domain-level and a meta-level. This required examination of taxonomies of algorithms and some additional work to extend these in a suitable way. A good choice of algorithm is one for which there already exists a workable multi-population version documented in the literature. It is not inevitable, but it is likely be a generational algorithm, since non-generational algorithms are likely to be complex to implement in a parallel version (see Borges and Barbosa (2000) for an example of a non-generational algorithm). Most importantly, it should be easy to adapt to work with a dynamic topology, since this is the main driver of the research.

(b) Design of an encoding for the meta-level population. This required a review of work done in network and graph optimisation problems. The examples given in Chapter 1 are typical; other examples were reviewed, although not exhaustively since the encoding is quite straightforward.

(c) Design and selection of the appropriate objective functions to drive the meta-level fitness. For measuring the cost of a topology, network design problems provide inspiration, since

cost is frequently optimised for in such problems. An objective function for diversity is related to the choice in (a) above.

(d) Selection of static topologies for comparison. This key analysis allows fair comparison of the evolved solutions against a static solution. This was simply a case of choosing topologies that had been used in prior research for other PDMOEAs.

3.2. Experimental method

This section describes the development and prototyping necessary to meet the objectives of the research.

3.2.1. Selected algorithm

The algorithm chosen for this research must have certain properties. The general taxonomy of algorithms given by van Veldhuizen et al. (2002) is used in the following discussion.

The algorithms should be a population based algorithm. This is for two reasons: firstly, it is much more practical to implement a population model in a distributed parallel mode than a fine-grained algorithm. Secondly, it is harder[4] to model migration in a diffusion based parallel model. A population based algorithm presents distinct sets of solutions that can be distributed across different machines (often simply a case of assigning one population to one machine), whereas a fine-grained model requires more work to allocate solutions to processors and possibly more book-keeping in the implementation. For similar reasons, having discrete populations provides convenient boundaries across which migration can take place. A diffusion based parallel algorithm does not present such clear boundaries.

The algorithm should have clearly defined generational boundaries. This is to allow analysis of the population diversity at discrete points. Algorithms without such generations will require

[4] Albeit not impossible – see Mehnen et al. (2004) for ideas.

sampling of diversity at various arbitrary times, leading to more complex analysis or more assumptions when interpreting the results.

The selected algorithm meeting the above criteria is the NSGA-II algorithm described by Deb (2001) and paraphrased here.

Step 1. Set $t = 0$ and create an initial population P_t.

Step 2. Create an offspring population Q_t from P_t using selection, crossover and mutation operators

Step 3. Combine the populations P_t and Q_t to create population R_t.

Step 4. Generate the set F containing all non-dominated fronts in R_t (a non-dominated front is a set of solutions that are mutually non-dominating – see Deb (2001) for a full definition).

Step 5. Create a new population P_{t+1} consisting of solutions in each non-dominated front F_1, F_2, …, F_n in turn, up to the limiting population size. In the case where the last front F_n will cause the size of P_{t+1} to exceed the population size limit, apply a crowding sort to eliminate solutions that are close together so that the maximum diversity is preserved.

Step 6. Set $t = t + 1$

Step 7. Repeat from 2 for a given number of generations.

For step 5, the population size of P is limited, and the set F will contain twice as many solutions as the limit. To deal with this problem, the algorithm selects only a subset of solutions from the last front F_n to ensure that the size of P_{t+1} remains fixed. The subset to include from F_n is determined by using a crowding-distance operator to select widely spaced solutions. The crowding sort first calculates a measure for each solution indicating how close it is to neighbouring solutions; and then sorts the set of solutions in descending order. The necessary number of solutions is then chosen in order, thus eliminating the most densely packed solutions.

Step 5 also gives the algorithm its name: this is the non-dominated sort to produce the set F.

The terminating condition is a pre-defined number of generations. In practice, the number of generations is determined by trial and error to allow the algorithm enough time to converge to a solution set. A more sophisticated approach might be to measure how much the population is improving in each generation, and stop when no further improvements can be detected. This might lead to a minor research question: how can this be achieved, and is it worth doing? For the purposes of this research, a fixed number of generations was set by experimentation (see section 4.1).

3.2.2. Encoding population topologies

As described in section 1.3.3, a population topology can readily be encoded as a simple graph. The literature has a number of examples of how this can be done: for the purposes of this research, the model for the topology is as follows.

The encoding of the graph of migration topology is determined by

- whether the graph is directed
- the number of sub-populations
- whether the edges are weighted, the number of bits to use for each weight and how this should be interpreted
- whether the sub-population size should be allowed to vary

The choices for each of these options is described below.

Firstly, the graph is directed. The direction of each link indicates the direction of migration. That is to say, a link from sub-population A to sub-population B in the graph means that a portion of the solutions in A will migrate to B, but there will be no corresponding migration from B to A. This differs from a typical real-world network design problem in that links are typically bi-directional in a network. The reason for this design choice is based on the notion that diversity will be more likely to be preserved across the population if the blending between populations is not symmetric. Assessing whether this is true falls out of the scope of this research (however, see future work in section 6.2).

The number of sub-populations *K* is arbitrarily fixed at 8 and the size of each sub-population is set to 16. This is felt to be sufficient to allow migration to have a real effect, while keeping the chromosome size and population sizes manageable.

The edges are weighted, and N=4 bits are used for each weight. This gives 2^4 distinct values which range from 0 to 15; and these weights determine which proportion of the population migrates across this link. The proportion of the population that is eligible to migrate is arbitrarily set to half the sub-population size. Using the formula in section 1.3.3, we have

$$|M_i| = \frac{1}{2} \cdot 16 \cdot \frac{e}{2^4 - 1}$$
$$0 \le e \le 2^4 - 1$$

Simply put, the number of individuals migrating at each migration step varies from 0 to 8. Different population sizes would result in different numbers of migrants.

Since the graph is directed, we have (K-1) individual links between the K sub-populations, giving us NK(K-1) = 4·8·7 = 224 bits in the genome.

3.2.3. Meta-level fitness functions (diversity and cost)

Diversity (spread)

The diversity metric used is taken directly from Deb (2001) p. 328 and repeated here:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} |d_i - \bar{d}|}{\sum_{m=1}^{M} d_m^e + |Q|\bar{d}}$$

where Q is the population, d_i is a distance measure between neighbouring solutions and d^e_m is a distance measure for the extreme solutions.

This equation can be illustrated as shown in Figure 5 for two objectives. The five filled circles are solutions, and the arrowed lines between them show the distance values for both the extreme solutions and the intermediate solutions. The equation above will have a zero value when the two

extreme solutions are at the minimum values for the corresponding objective functions; and the distance between each consecutive pair of solutions is equal to the mean distance.

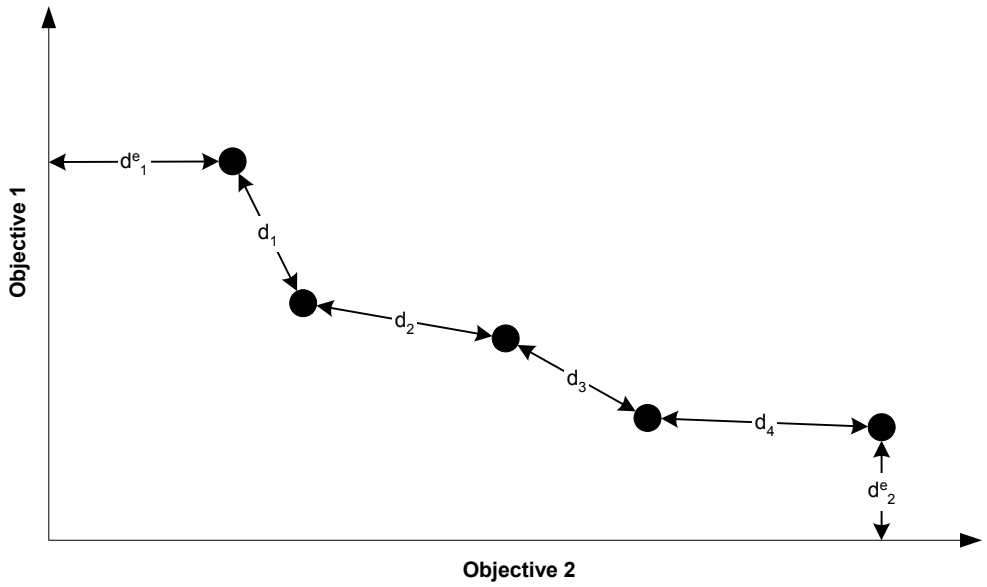

Figure 5 Illustration of the spread calculation

For convenience, the term *spread* will be used to refer to specific values of the diversity function. The spread value at the domain-level is used directly to calculate fitness values for each meta-level solution: that is, migration topologies that minimise the value for Δ at the domain-level are those migration topologies that will be selected by the meta-level algorithm.

It is important to be clear on how this metric is interpreted. Figure 6 illustrates the following discussion.

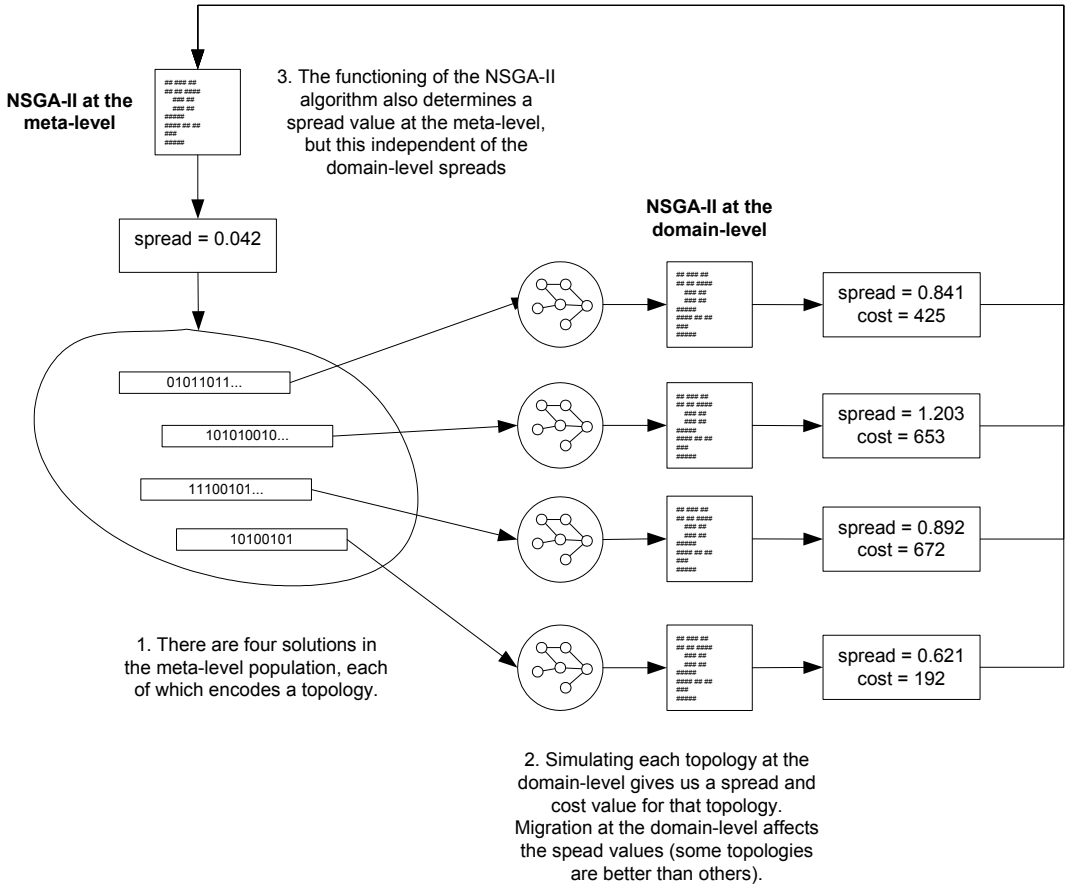

Figure 6 Spread values at the domain- and meta-level

The spread metric is an intrinsic part of the NSGA-II algorithm, and must be interpreted differently depending on whether we are considering the meta-level or the domain-level algorithm. For simplicity, assume that we have only one non-dominated front at any time. At the meta-level, there will be only one value for Δ , since there is only one population of solutions. (In this case, it will indicate how well distributed the solutions in the meta-level population are). However, each of these meta-level solutions encodes a topology used by a PDMOEA which is solving a domain-level problem, and each of these PDMOEAs has an associated Δ value at the domain-level. It is these domain-level values that determine the fitness function to use at the meta-level. In short, the value of Δ at the meta-level serves only to support the functioning of the NSGA-II algorithm; whereas the values of Δ for solutions at the domain-level are much more

significant, since they directly couple the domain-level and meta-level together via the meta-level fitness function.

To reiterate, the spread value for a population represents the distribution of solutions along the first non-dominated front (recall that this is the set of all solutions such that none of the solutions dominates any other). Two examples are shown in the graphs below. Each graph shows one non-dominated front. The axes are the two objective functions for the problem.

The solutions in Figure 7 are not well distributed according to the spread value, since the distance between consecutive solutions differs significantly from the mean distance between solutions; and hence the spread has a relatively high value of 0.607. The solutions in Figure 8 however, have a much lower spread value (0.0408) because they are evenly spaced along the non-dominated front.

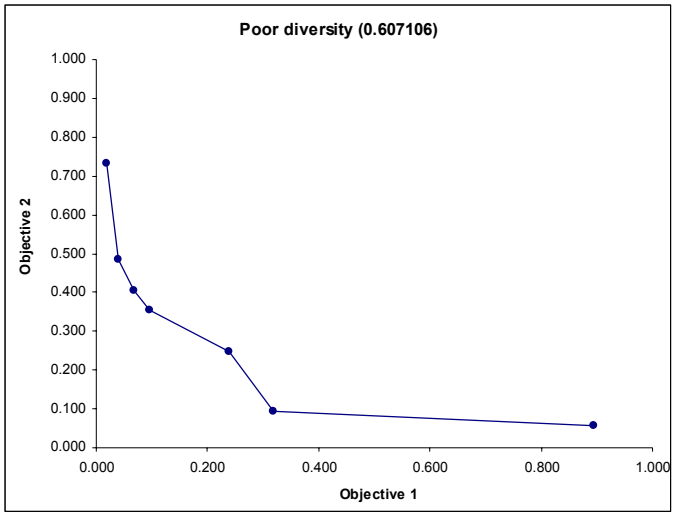

Figure 7 Graph of solutions exhibiting a poor distribution

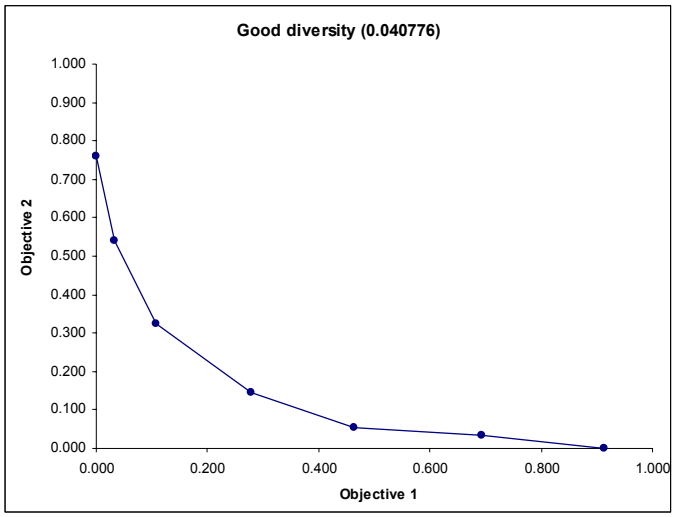

Figure 8 Graph of solutions exhibiting a good distribution

Note that these examples are contrived only in order to clearly illustrate the differences between different non-dominated fronts and they do not correspond to actual experimental results, which will be discussed later. In fact, the experimental results produce spread values that are higher (i.e. worse) than the contrived values in the figures above – see section 4.3.

Cost

The cost of a topology is simply calculated from the sum of the weights in the graph. This is directly proportional to the total number of individuals migrated in each simulation run. This means that topologies that have a high degree of connectivity will generally have a higher cost, since there will be more links in the graph. Similarly, if individual weights are relatively high, then the cost metric will increase accordingly.

For example, if there are eight nodes arranged in a ring, and one directed link from each node to its immediate neighbour in a clockwise direction, then there are eight links. If each link has a weight of 15, then the cost for this topology is 120. See Figure 9

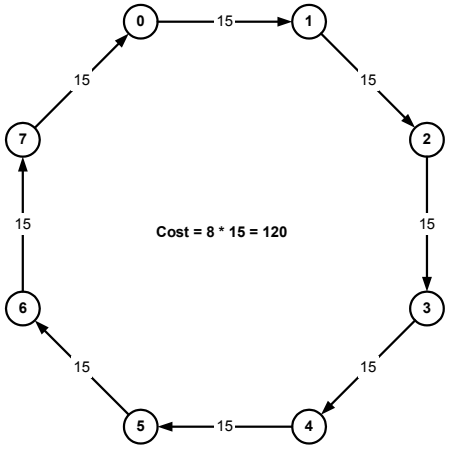

Figure 9 Cost for a simple ring topology

3.2.4. Migration model

Migration between sub-populations at the domain-level is implemented by extending the NSGA-II algorithm described in 3.2.1.

Two changes are required.

Firstly, following each generation of the entire set of sub-populations, a set of migrant populations must be generated, each representing the set of solutions migrating into the corresponding sub-population. This is where the population topology is taken into account. This is described as step 7 below.

Secondly, each generation of each sub-population must incorporate a (possibly empty) set of incoming migrant solutions into the sub-population. This is done by extending step 3 to include the migrant solution into the combined population prior to the non-dominated sort.

The modified algorithm is given below. Note that each iteration of the algorithm will be executed up to step 7 for **all** sub-populations (shown as P^i for the *i*-th sub-population). At this point, the meta-level algorithm determines the migrant populations by considering the population topology.

Step 1. Set $t_i = 0$ and create an initial population P^i_t for each sub-population *i*. Set M^i_t to the empty set.

Step 2. Create an offspring population Q^i_t from P^i_t for each sub-population *i* using selection, crossover and mutation operators

Step 3. Combine the populations P^i_t and Q^i_t and M^i_t to create population R^i_t.

Step 4. Generate the set F^i containing all non-dominated fronts in R^i_t.

Step 5. Create a new population P^i_{t+1} consisting of solutions in each non-dominated front F^i_n in turn, up to the limiting population size.

Step 6. Set $t_i = t_i + 1$

Step 7. Following the above process for **every** sub-population *i*, determine M^i_{t+1} by selecting solutions from $\{P^n_t\}$ according to the migration topology.

Step 8. Repeat from step 2 for a given number of generations.

By including the migrated solutions in the combined set before the non-dominated sort in step 3, the algorithm will simply apply the same selection criteria to a larger set; and therefore any improved solutions from the migrated set will be immediately incorporated into the overall population by step 5.

3.2.5. Test problems

The test problem to be executed by the domain-level algorithm is not the primary focus for the research. Consequently, the initial experiments were done with the Min-Ex problem described in Deb (2001). This has two objective functions which are to be minimized subject to constraints:

$$Min-Ex: \begin{cases} f_1(x) = x_1 \\ f_2(x) = \dfrac{1+x_2}{x_1} \\ 0.1 \le x_1 \le 1 \\ 0 \le x_2 \le 5 \end{cases}$$

3.2.6. Analysis of static topologies

The details of the static topologies that are used for comparison are described briefly here. The tables below contain the matrices of weights for the links in the graph. Obviously the graph is not reflexive – it is not meaningful to migrate from a population to itself.

Fully connected

Each sub-population can provide migrants to every other sub-population. The number of migrants at each step is the maximum allowed under the constraints described above. The matrix of weights is as follows. Note that since 4 bits are used for encoding weights, the maximum value for each weight is 2^4-1, i.e. 15. Zero values are shown as blank for readability.

Table 1 Weight matrix for fully-connected topology

	1	2	3	4	5	6	7	8
1		15	15	15	15	15	15	15
2	15		15	15	15	15	15	15
3	15	15		15	15	15	15	15
4	15	15	15		15	15	15	15
5	15	15	15	15		15	15	15
6	15	15	15	15	15		15	15
7	15	15	15	15	15	15		15
8	15	15	15	15	15	15	15	

Ring

Each sub-population can provide migrants to the next sub-population, arranged in a ring. Again, the maximum number of migrants is used in each case. The migration is unidirectional.

Table 2 Weight matrix for ring topology

	1	2	3	4	5	6	7	8
1		15						
2			15					
3				15				
4					15			
5						15		
6							15	
7								15
8	15							

Hypercube

Each sub-population can provide migrants to the adjacent sub-populations, arranged in a 2x2x2 cube.

Table 3 Weight matrix for hypercube topology

	1	2	3	4	5	6	7	8
1		15	15		15			
2	15			15		15		
3	15			15			15	
4		15	15					15
5	15					15	15	
6		15			15			15
7			15		15			15
8				15		15	15	

Unconnected

This is the trivial case where all weights are zero.

3.2.7. Implementation details

This section briefly discusses some implementation details. The implementation language is Java; and each sub-population is simulated in a multi-population algorithm using a separate object structure within the same Java Virtual Machine (JVM). The meta-level algorithm executes periodically according to the algorithm outlined in 3.2.4 to migrate individuals from one population to another. Note that this is simulating a parallel algorithm on a serial machine. No attempt has been made to build a real distributed algorithm, so many of the problems associated with distributed algorithms fall away without affecting the results. For example, there is no need for a separate communications layer, since all data will be present in the same JVM.

3.3. Data collection and analysis

Data is collected to measure whether the population of topologies actually iterates to a better solution. The data collection is built into the experimental application, so that a database of results is built up as the program executes.

The following data is collected:

- the encoded topology for every solution in the meta-level population
- the objective function values for each topology for each generation of the meta-level population

Having the above data for each generation of the meta-level algorithm provides information to ascertain (a) whether the meta-level algorithm actually iterates toward better topologies, and (b) whether the obtained results compare favourably with the selected static topologies.

Given the nature of the research question, it will not require complex analysis to establish these two facts, and these will be sufficient to show that the algorithm functions correctly.

3.3.1. Simulation parameters

Each experimental run of the meta-level algorithm requires parameters to control both the meta-level algorithm and the domain-level algorithm. These are discussed briefly here.

G_{dom} is the number of generations in each domain-level algorithm. This value influences how repeatable the experiments are. A low value means that the domain-level algorithm has not converged to a stable spread value; which means that the fitness function feeding into the meta-level algorithm is not much better than a random value.

G_{meta} is the number of generations for the meta-level algorithm, simulating a population of topologies.

S_{meta} is the number of simulations required for each encoded topology in the meta-level population for each generation of the meta-level algorithm. The number of simulations must be sufficient to allow calculation of a stable mean value for the spread (which in turn drives the meta-level algorithm).

Since each topology in the meta-level population describes a number of independent domain-level algorithms N, we can see that the overall number of domain-level generations that must be executed for every meta-level generation will be the product $G_{dom} \,.\, S_{meta} \,.\, N$

3.3.2. Statistical significance

Since there is no deterministic way to find the spread value for a topology (the spread for any topology must be estimated experimentally by simulating N domain-level algorithms for G_{dom} generations at least once), we need some way of establishing whether any two spread values are statistically different. This is done using a t-test (Wikipedia, accessed 05-Jan-2008) on two sample sets of size S_{meta} to check whether the mean of the samples is different within a certain confidence interval.

3.4. Summary

This chapter has presented the details of the research method.

- Similar research in the literature was discussed
- The details of the selected algorithm, the method for encoding graphs, and the calculation of spread and cost values were explained in detail.
- The domain-level test problem was given.
- Changes to the algorithm to support migration were explained.
- Some well defined static topologies were described.
- The methods to be used to collect and analyse data during simulation were explained, and the use of statistical mean testing was introduced as a way to ensure robustness of results.

The following chapter describes the results of the experimental work.

4. Results

The results of the experimental work is presented in this chapter. Some initial results on the behaviour of the algorithm with respect to calculating spread values are discussed first. Then the results of simulating known static topologies and evolving dynamic topologies are given. Some conclusions are drawn, but full analysis is reserved for Chapter 5.

4.1. Determining spread values

This section describes a result from simulating the domain-level algorithm for a single non-distributed population (i.e. without introducing any migration or meta-level complexity). As discussed in section 3.3.1, in order to obtain sensible results, there is a minimum number of generations required to allow the domain-level population to reach a stable spread value.

From trial and error testing, I observed that the characteristics of the NSGA-II algorithm seem to cause the solutions in the population to rapidly settle into a distribution that does not change appreciably after the first few generations. This is shown for a number of simulation runs in Figure 10 below. The graph shows the spread of the domain-level population for two separate simulations of an NSGA-II population. Each population is run for 100 generations (although the pattern seems to continue for a much larger number of generations).

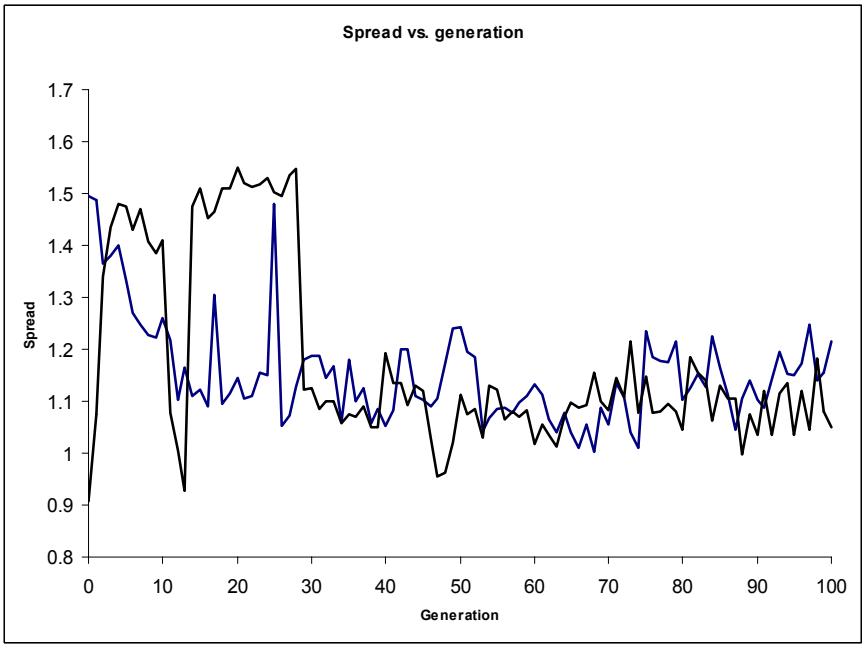

Figure 10 Spread for two populations

Note that the spread for each generation seems to converge rapidly to a particular value. Only a few generations are required to achieve a stable value (about 30). Once this point is reached, the spread seems to oscillate around a value that is not driven by the simulation parameters; but rather driven by the underlying pseudo-random sequence used to simulate the population. In other words, the choice of random seed has more influence over the resulting diversity than initially expected. After the spread stabilises, the population does not become more diverse over time. This points to a limitation in the NSGA-II algorithm such that once it reaches a certain level of diversity, it does not continue to find steadily more diverse solutions. This is discussed in the analysis in Chapter 5.

4.2. Static topologies

This section presents the results of simulating the diversity values computed from statically chosen topologies described in 3.2.6

4.2.1. Experimental results

The following table shows the spread and cost values for the chosen static topologies. The spread values given are the mean values calculated over 500 simulations of the domain-level algorithm, with 50 generations per simulation. The variance of the spread is also calculated from the same 500 samples.

Table 4 Mean spread for static topologies

Topology	Cost (fixed)	Mean spread	Variance
Fully connected	840	1.249	0.0147
Unconnected	0	0.995	0.0203
Ring	120	1.105	0.0184
Hypercube	360	1.187	0.0164

4.2.2. Statistical significance

A glance at the mean spreads given in Table 4 shows that the unconnected topology produces the most diverse results, since it has the lowest spread value, indicating relatively well spread solutions; while the fully connected topology has the highest spread value, indicating relatively poorly distributed solutions.

As described in section 3.3.2, a t-test can be used on the full data sets to evaluate whether these results are statistically significant. The t-values for comparing each pair of topologies are:

Table 5 t-values for static topologies

	Hypercube	Ring	Unconnected
Connected	7.864	17.650	30.355
Unconnected	22.422	12.553	
Ring	9.787		

Since we have 500 samples for each topology, we have (2n - 2) = 998 degrees of freedom, and the corresponding probability P(T <= t) for a two-tailed test is:

Table 6 Probability of identical mean spreads

	Hypercube	Ring	Unconnected
Connected	9.59154E-15	6.91487E-61	6.6969E-144
Unconnected	1.70132E-90	1.15661E-33	
Ring	1.16051E-21		

These values are very low, so we can be confident (statistically) that the mean spread values for the chosen static topologies are different. We can therefore conclude that the topology used by the meta-level migration algorithm definitely has an influence on the diversity of the solutions obtained by the underlying distributed domain algorithm. This is an important result and will be discussed further in Chapter 5.

4.3. Evolved topologies

This section discusses the primary experimental work: simulating a meta-level evolutionary algorithm to search for good migration topologies. The meta-level algorithm was run for 500 generations, and the starting and ending populations are shown. The repeatability of the tests is discussed. The best evolved results are then shown for comparison on the same chart as the static topologies described in section 4.2.

4.3.1. Initial population

The initial population is shown in Table 7 and Figure 11 below. The population has been initialised with a random set of topologies (each point in the chart represents one topology). Each topology has been simulated a number of times in order to estimate the initial spread value (recall that the cost of the graph is fixed by the weights and can be readily calculated).

The solutions are shown as a single set, and each of the four non-dominated fronts is highlighted by linking the points in that front. In Table 7, the best spread and cost values are highlighted in bold italic.

Table 7 Initial population data

Solution	Front	Spread	Cost
1	1	***1.191***	395
2	1	1.200	***360***
3	2	1.192	411
4	2	1.204	390
5	2	1.211	382
6	2	1.217	378
7	2	1.226	377
8	3	1.202	417
9	3	1.209	394
10	3	1.212	389
11	3	1.220	381
12	4	1.205	451
13	4	1.208	429
14	4	1.209	413
15	4	1.213	402
16	4	1.214	393

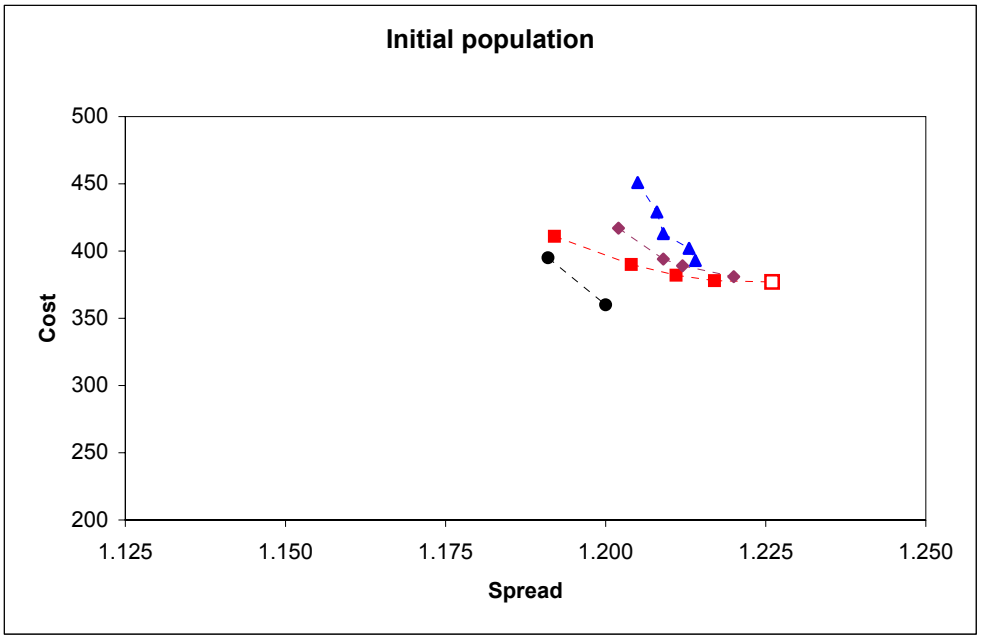

Figure 11 Initial population

One of the solutions displayed in the graph (solution 7 in the table and indicated in the chart as a hollow square) has been chosen for further illustration. This solution has a spread of 1.226 and a cost of 377. The matrix of migration weights encoded by this solution is given in Table 8. Note the random nature of the weights (also note that they sum to 377 – i.e. the cost of this solution; and that they range from 0 to 15 – see section 3.2.6). Each solution in the population shown above represents one such matrix.

Table 8 Example initial topology

	1	2	3	4	5	6	7	8
1		13	9	13	13	11	8	4
2	8		1	13		3	7	7
3	3	8		1	4	5	7	5
4		7			2	8	5	13
5	10	7	12				13	3
6	5	3	5	2	4		11	10
7	7	9	12	14	7	12		2
8	10	5	15	15	2	3	1	

4.3.2. Population after 500 generations

Table 9 and Figure 12 below show the population after simulating the meta-level algorithm for 500 generations. Note that the scale in the chart is identical to the scale in Figure 11. In this case, there are three non-dominated fronts in the population. The solutions with the best values for spread and cost respectively are highlighted in bold italic (solutions 1 and 6 respectively). Note that by definition these must be in the first non-dominated front.

Table 9 Population data after 500 generations

Solution	Front	Spread	Cost
1	1	***1.131***	394
2	1	1.135	393
3	1	1.143	391
4	1	1.148	315
5	1	1.151	252
6	1	1.163	***232***
7	2	1.139	412
8	2	1.151	340
9	2	1.152	262
10	2	1.170	239
11	2	1.194	233
12	3	1.155	289
13	3	1.164	273
14	3	1.168	263
15	3	1.181	252
16	3	1.184	240

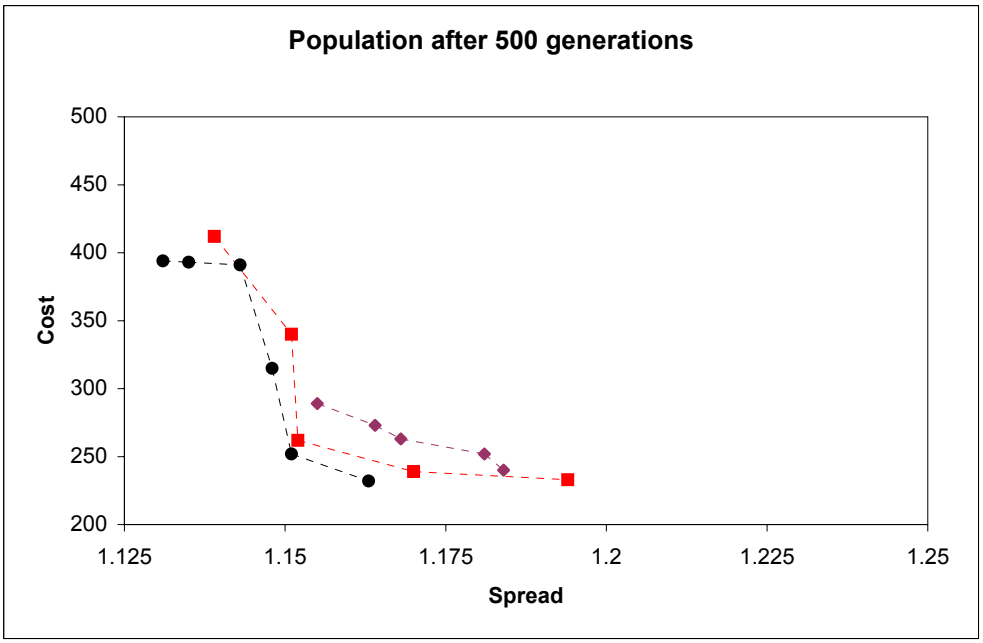

Figure 12 Population after 500 generations

In Figure 12, the solutions are clustered more toward the bottom left hand section of the graph, whereas in the initial population they are clustered more randomly in the top right hand section. This is expected, since the meta-level algorithm is searching for solutions that both minimise cost and minimise the spread value. The solutions shown are thus better in both these objectives than the solutions in the initial population.

There are three non-dominated fronts, compared to the four in the initial population and the solutions are more evenly distributed. This is typical of the results from NSGA-II, since the selection operator uses the non-dominated ranking – that is, solutions in the first non-dominated front are preferred to those in the second and subsequent fronts (and so on); and therefore the algorithm has a tendency to reduce the number of fronts.

Note that the spread values range from 1.131 to 1.194. This is a relatively narrow range (especially when considering the spread values of the static topologies described in section 4.2). The cost values, however, range from 232 to 412 – covering a proportionally larger range of values for this objective function (since the cost is an integer value between 0 and 840 – see section 3.2.3). From this we can speculate that the cost is "easier" to optimise than the spread; or

that there are more ways to improve cost without worsening the spread than vice versa. More discussion is given in Chapter 5.

4.3.3. Repeatability and significance of results

Any of the solutions in the initial and final populations shown above can be extracted and simulated independently. This is necessary (or at least prudent), since it allows us to check that the distributed domain-level algorithm has not produced random results.

Suppose we choose the solution that has the lowest spread value from the first non-dominated front in the final population (solution 1 in Table 9). This solution has a spread value of 1.131 and a cost of 394 and it is Pareto-optimal because it is not dominated by any other solutions. We can run the distributed domain-level algorithm in isolation using this topology to verify that we get a similar spread (i.e. about 1.131). Provided the domain-level parameters are kept consistent with the parameters used in the main simulation, the results (that is, the measured spread) should also be consistent.

Two solutions from the final population were selected for this treatment. The results are shown in Table 10.

Table 10 Repeated test runs of selected final solutions

Repeated run	Best spread (solution 1)	Best cost (solution 6)
1	1.142	1.184
2	1.149	1.166
3	1.145	1.172
4	1.147	1.181
5	1.150	1.178
6	1.144	1.185
7	1.147	1.157
8	1.155	1.171
9	1.144	1.179
10	1.138	1.167
Mean	1.146	1.174

Note that the values obtained are not quite the same as the full simulation results. Solution 1 has a mean spread of 1.146 in the repeated test runs, but a mean spread of 1.131 in the simulation runs. However, from the repeated test runs, it still appears that solution 1 is producing better spread

values than solution 6 and therefore that the topology represented by that solution is producing more diverse solutions but with a higher cost (this is exactly the trade-off that we are trying to demonstrate experimentally).

We can attach a confidence factor to these results by conducting a t-test to see whether the mean spread values for these two solutions are actually different (see section 3.3.2). Note that we should not select the data from one experimental run for each solution from the table above, since that might introduce selection bias (which run would we pick?). Instead, the data for all the runs was pooled to give one large data set with 5000 samples (500 generations for each of the 10 simulations).

The results of the t-test are given in Table 11. This gives us a very low probability that the solutions are actually the same, and we can be confident that the algorithm is finding different solutions; despite the fact that the repeated test runs do not produce exactly the same values as the full population simulation.

Table 11 t-test for final solutions

	Solution 1	Solution 6
Mean	1.146	1.174
Variance	0.0206	0.0178
Observations	5000	
t-value	10.119	
P(T <= t)	5.9574E-24	

Finally, we can also compare the best initial solution with the best final solution. This gives us the data in Table 12 and shows that the final solution with the best spread is significantly different from the initial solution with the best spread. Again, we can be confident that we are seeing real results.

Table 12 t-test for best initial and best final solution

	Best spread (final)	Best spread (initial)
Mean	1.146	1.209
Variance	0.0206	0.0172
Observations	5000	
t-value	22.837	
P(T <= t)	7.20E-113	

4.3.4. Comparison with static topologies

The chart below combines the first non-dominated front from the final simulated population (described in section 4.3.2) and the static topologies (as given in Table 4). Note that the scale is different from the previous two charts.

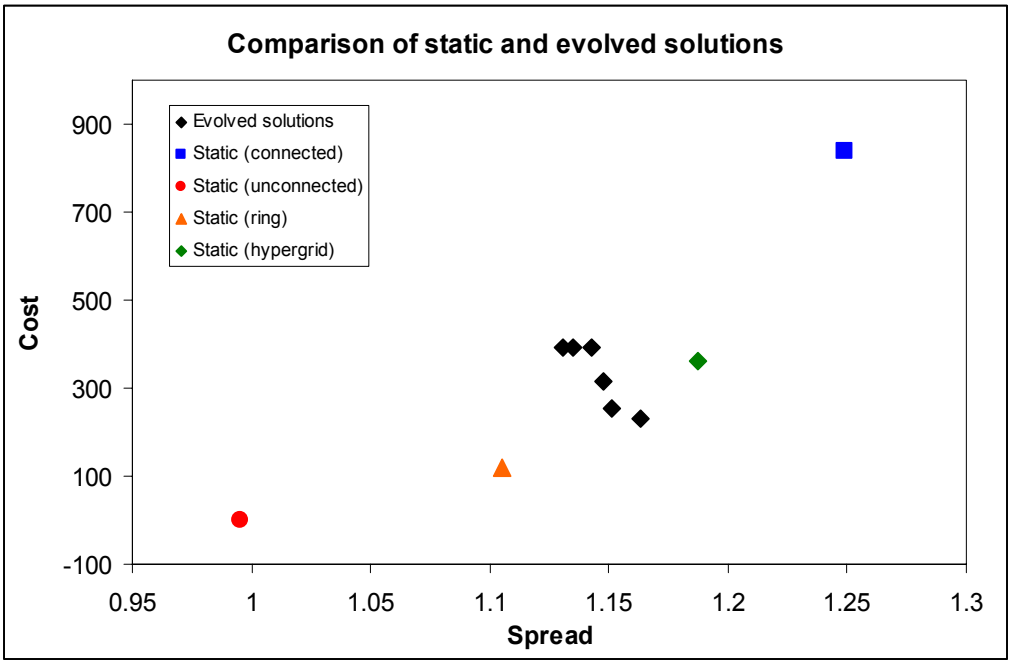

Figure 13 Comparison of static and evolved solutions

From this chart we can clearly see that the evolved solutions are intermediate between the hand chosen solutions. For example, the connected graph has the highest cost, and produces high spread values – hence it is both expensive, and does not produce very diverse solutions. By

comparison, the ring topology produces more diverse solutions than the evolved topologies and with a lower cost.

As discussed earlier, the fact that the unconnected topology turns out to have both the lowest cost and the best spread values will mean that it dominates all other possible solutions. The implications of this are discussed in Chapter 5 and section 5.1.2 in particular.

4.4. Summary

This chapter has presented the key experimental results and briefly considered the implications.

- The spread values for a single random topology were presented, showing how the NSGA-II algorithm converges to a stable spread value over a number of generations.
- The results of simulating the chosen static topologies were presented. Statistical significance was explored using a t-test and shows that the topologies are producing different mean spreads with a high degree of confidence.
- The results of simulating a meta-level population of topologies was given, including an examination of the repeatability of the results. More t-tests showed that the results were significant.
- Finally, the static and dynamically evolved solutions were compared.

The following chapter analyses the results in more detail, and draws the main conclusions.

5. Analysis

This chapter analyses the results from Chapter 4 in more detail and in light of the original objectives. Some of the limitations in the experimental approach are discussed, and some key conclusions are drawn.

5.1. Comparison of evolved and static topologies

5.1.1. Best evolved solutions and the trade-off between cost and diversity

In section 4.3.2 we saw that the evolution at the meta-level produced better solutions over time. That is, the migration topologies in the final population of the meta-level simulation had lower cost and mean spread values than the initial random solutions in the first population. Using a t-test to evaluate the probability that the solutions were actually distinct showed two things. Firstly, the best final solutions after 500 generations are statistically different from a typical solution in the initial population with a high confidence level. Secondly, the best final solutions are also statistically different from each other with a similarly high confidence level. This means that the meta-level evolution is finding topologies that improve the respective cost and diversity objectives while trading off between them – in other words, the effect is real and not due to chance.

It is also clear that the cost is trivially easy to improve without much affecting the spread. What this suggests is that a low-cost migration strategy may be essentially the same as a high-cost strategy with respect to the obtained spread. When we consider the entire population of possible topologies, we would find many graphs that are essentially the same in terms of the migrations that occur during simulation, but which have widely varying costs. Exploring this in more depth is suggested as a future research direction.

5.1.2. Comparison with static topologies

In Figure 13 we can see the evolved meta-level solutions and the known static topologies plotted on the same graph. It is clear that some static topologies are better than the evolved solutions, while some are worse.

The most significant observation is that the unconnected graph produced the best spread. Since this topology has a cost of zero and the best spread (of known solutions), it will dominate all other solutions that we might find. If the unconnected graph had produced the *worst* spread, this would not have been a problem, since there would be other graphs with a higher cost but a lower spread. This might seem to be a major problem, since it means that no evolved topology can be better than the unconnected graph, and therefore no amount of simulation will find a better solution. Moreover, it means that any use of hand chosen topologies is equally irrelevant – which calls into question whether migration is even valuable at all.

However, this result does not mean that the essential approach is invalid. There are two possible explanations for this result, both related to the experimental method. Firstly, the method used to determine the mean spread values may unfairly benefit the unconnected graph. Secondly, the method used for migration might have a similar bias.

Mean spread function for topologies

In retrospect, the method chosen to compute the mean spread for each topology was somewhat naïve. As described in section 3.2.3, the mean spread directly drives the fitness of solutions at the meta-level. For a given migration topology, a PDMOEA was simulated using that topology (that is, a number of domain-level populations were executed for a given number of generations and with periodic migration). At the end of the simulation, all the solutions from each domain-level population were aggregated into one large population. A crowding sort (see section 3.2.1) was applied to this population to obtain the most diverse solutions, and the associated spread was assigned as the value of the spread objective function for that topology. For example, if there are

8 sub-populations each of size 40, then the aggregate population has 320 solutions; and the crowding sort reduces this back down to 40 solutions from which a spread value is calculated.

This approach appears to favour unconnected graphs, because the individual domain-level sub-populations will evolve independently until the end of the simulation and this will allow divergence in each population. There is no migration between sub-populations during the simulation, and therefore no sharing of genetic material. By contrast, a fully connected graph would allow frequent sharing of genetic material and this would hade a different effect on diversity. By applying the crowding sort to the aggregate population, we are effectively selecting the most diverse set of solutions. This may have a distorting effect that is particularly noticeable for unconnected graphs, since in this case the individual sub-populations have evolved for the longest period in isolation.

A different approach might be to measure the mean spread across all individual domain-level populations and avoid aggregating the sub-populations at the end of the simulation. This would measure the effect of migration at the domain-level for each sub-population, rather than measuring the diversity of the aggregated final population; and possibly lead to different results. In particular, since unconnected topologies do not have any migration, this approach would show more strongly whether the individual populations in more connected graphs were benefiting from migration. This is suggested as a future research direction.

Effectiveness of meta-level migration

The method used to select solutions for migration was not necessarily the best either. Solutions were chosen randomly from the domain-level populations. Lin et al. (2004) and Power et al. (2005) showed that it can be better to select individuals for migration by taking into account the impact on the diversity of the resulting population. For example, it might be better to include the best N solutions in a migrant population, rather than a random selection. Since this previous work was concerned with single-objective optimisation, some modifications are likely to be required in applying the same approach to a multi-objective solution. For example, it might be necessary to

evaluate combinations of solutions to determine which subset might be most likely to improve the spread in the destination population.

5.2. Reflections on experimental approach

5.2.1. Suitability of NSGA-II

The NSGA-II algorithm was selected for reasons discussed in 3.2.1. This section considers three characteristics of that algorithm and discusses whether it was a good choice.

Crowding metric and stable spreads

The first observed characteristic of NSGA-II is that the spread obtained by a single simulation of the algorithm at the domain-level tends to settle into a stable value fairly quickly (described in section 4.1).

This effect is likely to be due to the selection method used in the algorithm. The crowding distance selection (see section 3.2.1) means that solutions in the non-dominated front that are relatively close to other solutions will be eliminated. However, in order to further improve the spread, these are possibly the very solutions that should be retained, since they may be better positioned to increase the spread value. For example, if we had a perfect distribution (with a spread of zero), then we would see each solution spaced equally along the non-dominated front – that is, the distance between consecutive solutions would be equal to the mean distance. Each solution is thus in a position that we might describe as *perfect* – it is exactly in the right place to contribute to a zero spread. Now consider a typical population of solutions with a non-zero spread. Some of these solutions might be in perfect positions, and some might not. The crowding sort will eliminate solutions according to whether they are close to other solutions, but will not take into account whether the solutions are in perfect positions or not; and hence we may eliminate solutions that are in positions that we might be better off retaining.

In addition, the extreme solutions bounding the non-dominated front will be problematic. The non-extreme solutions will be approximately evenly spaced between the extreme solutions; but

the distance between the extreme solutions and the minimum values of each objective function may still contribute significantly to the spread (see equation in section 3.2.3). In practice therefore, to improve the spread significantly in any generation, the extreme solutions would need to move closer to their respective minima and simultaneously we would need all the other solutions in that generation to be suitably redistributed. In NSGA-II, this would only happen if the mutation and crossover operators produced the appropriate solutions. There is a very low probability of such changes all happening at the same time – and hence the algorithm stabilises at a relatively poor distribution.

This is noted as a possible enhancement to the algorithm in the conclusions below.

NSGA-II at the meta-level

The second observed characteristic of NSGA-II is that selection at the meta-level may not be as effective as at the domain-level. As described in section 3.2.4, the NSGA-II algorithm was modified to allow migration between domain-level populations. Apart from this change, the meta-level and domain-level algorithms are identical.

The crowding distance operator at the meta-level probably has a more arbitrary effect than at the domain-level. Recall that the crowding distance operator allows selective elimination of solutions to keep the population size constant. For the domain-level problem, the objective values are given by the following functions (see section 3.2.5 for the full definition):

$$f_1(x) = x_1$$
$$f_2(x) = \frac{1 + x_2}{x_1}$$

These objective functions f_1 and f_2 are continuous (x_1 is constrained not to be zero), so we can easily see that Pareto-optimal solutions at the domain-level will fall onto a continuous curve, both for the objective functions as well as the variables x_1 and x_2. At the domain-level, closely spaced solutions in the variable space x_n are closely spaced in the objective space f_n as well.

However, at the meta-level, there is no such smooth distribution of solutions, because the equivalent objective function for spread can only be determined by experiment (we must simulate the associated topology); and the encoded topologies do not fall into any continuous space either (for example, we cannot plot a set of topologies directly on a Cartesian plane – what would the coordinates be?). This means that, unlike the domain-level solutions, two solutions at the meta-level may be close together in the objective space (that is, they may have similar spreads and similar costs), but we cannot say anything about how the underlying topologies are related – the topologies may be completely different.

This means that the crowding distance operator at the meta-level may be no different from random selection. By only considering the spread and cost values for each meta-level solution, there is no way to tell whether the solutions that are being retained are better than those that are eliminated – and hence the argument that the crowding distance operator is more arbitrary at the meta-level than at the domain-level.

An alternative solution would require a more sophisticated graph-aware selection operator that takes into account the nature of the underlying topology. See section 5.2.2.

Rate of meta-level evolution

The final point regarding NSGA-II is the rate at which the algorithm operated at the meta-level. In section 4.1 we can see that at the domain level, the algorithm tends to converge to a stable state fairly quickly and that future improvements were quite slow to emerge (if they emerged at all). We can be sure that there *are* better solutions, because we have also seen that the unconnected graph is better than the evolved solutions – hence the algorithm ought to eventually find it. At the meta-level, the number of simulations required meant a large amount of computing time was necessary to adequately simulate the meta-level population. Run times of up to 36 hours on an 8-CPU machine were possible, although the rapid stabilisation of the algorithm meant that after a few hours, relatively little progress was made.

The conclusion here is that the algorithm is probably not finding iteratively better solutions very well. New graphs are generated with crossover and mutation from the existing solutions; but almost all the time they are not actually improvements on the existing solutions.

A number of changes could be made to the algorithm to improve this. The fitness functions could be extended to include a better method of calculating fitness of each topology, rather than the simple simulation approach taken here. Alternatively, at the meta-level, a more specialised method of encoding graphs might allow more scope for the evolutionary operators to select better solutions and also for the generative function to produce solutions that improve the convergence of the algorithm at the meta-level. This is discussed in the following section.

5.2.2. Evolutionary graph optimisation

Evolutionary graph optimisation can be seen as a specialised area for evolutionary algorithms; and multi-objective graph optimisation is even more specialised. Some examples of this in the literature have been considered in this research (Alberto and Mateo (2004), and Arabas and Kozdrowski (2001)). Below are some observations from this research on special considerations that might be necessary for this kind of work.

Graph encoding effectiveness

The encoding of graphs will have an influence on the effectiveness on the evolutionary algorithm. Encoding the number of graphs involving N nodes gives us at least N(N-1) possibly encodings; and if weighted links are used, this number increases significantly. This leads to a large solution space. However, previous research, (for example Banerjee and Kumar (2007)) shows that even a large solution space like this can be feasibly searched with an evolutionary algorithm.

It is easy to see that many graphs may in fact be logically identical because of the large degree of symmetry in the solution space. In other words, there are many graphs which have a ring topology, and the only difference between them is the ordering of the nodes in the ring. Any evolutionary algorithm searching this space might be wasting a great deal of time on evaluating essentially identical solutions.

It follows that it is possible that different methods of encoding graphs will enhance the efficiency of the algorithm. For example, a 'generating' approach might be taken, which could specify graphs in a more parsimonious form – e.g. by encoding all ring topologies using an identical representation, in which the encoding 'generates' a canonical ring topology.

A different encoding method might also lead to better evolutionary operators. In particular, the crossover and mutation operations may need modification. In the methods used in this research, a matrix of weights is the essential operand, and is treated as a sequence of bits in the crossover and mutation operators. This might be modified – it might make more sense to modify a graph by using more explicit changes, for example, atomic operations to add or remove links, or to change the corresponding weights by a small increment. This would lead to a more systematic search, since each solution gives rise to solutions that are more 'similar' than a simple bit-wise mutation or crossover.

We can conclude that the approach taken in this research and elsewhere in the literature could therefore be improved upon. This is noted in the conclusions as scope for further research.

Detailed graph analysis

Following from the above, there is a large body of mathematical graph theory that might be applied to enhance the analysis of the evolved graphs. For example, the degree of connectedness in a graph might be used as an objective function for an algorithm.

More sophisticated metrics for graphs might need more specialised evolutionary graph algorithms which direct the search algorithm in a more efficient way. We can speculate that this could lead to significant improvements in the performance of the algorithm, for similar reasons to those in the previous section: the search might be more systematic.

5.3. Summary

The above discussion analysed some of the key results and related them to the original objectives.

- We can find improved migration topologies using a MOEA and justify the results using statistical tests.
- The evolved solutions are better than some static topologies while worse than others. There are experimental reasons for this that suggest further research.
- The NSGA-II algorithm has some characteristics that make it less effective at the meta-level than at the domain-level. This again suggests further research.
- The application of MOEAs to the area of graph optimisation requires some specialisation to achieve the best results.

The final chapter contains conclusions for the overall research.

6. Conclusions

This chapter presents conclusions following from the analysis in chapter 5.

6.1. Final conclusions

The research aim was to test experimentally whether it was possible to find migration topologies for a parallel distributed MOEA (a domain-level use of the algorithm) by using an MOEA again at a meta-level to evolve the topologies. The cost of migration and the diversity of solutions obtained in the domain-level were used as fitness functions at the meta-level.

The following key conclusions can be justified:

- It is possible to evolve such topologies in a manner consistent with previous research (section 4.3). The primary objective of the research has been met.
- The same multi-objective algorithm can be used at both a domain-level and at a meta-level with only the necessary modifications to support migration (sections 3.2.1 and 3.2.4). This shows the generality of such algorithms, and shows how the problem of finding an optimal network configuration is similar to finding a migration topology – thus linking different fields of research.
- The evolved solutions can be shown to be statistically different both from each other and from the original random solutions within a high level of confidence (section 4.3.3). This shows the use of robust statistical techniques to justify experimental conclusions.
- The evolved solutions are not better than *all* static solutions and this reveals some limitations in the experimental method – in particular, the approach taken tends to favour unconnected graphs (section 4.3.4). This also raises questions about the role of migration in distributed evolutionary algorithms in general. It is not as obvious as it might seem that an algorithm that uses migration is necessarily going to produce more diverse solutions than one that doesn't.

- The results can be seen in the context of previously suggested roadmaps for research in the field of parallel distributed MOEAs (Chapter 2). In particular, the idea of using an evolutionary approach to optimise aspects of an evolutionary algorithm has been suggested by Lin et al. (2002).

The results above support the hypothesis that migration topologies can be evolved for multi-objective algorithms by using the same algorithm at a domain-level and a meta-level. This is an extension of work by other researchers (Banerjee and Kumar (2007), and others) which uses a multi-objective algorithm to evolve domain specific graphs. However, to the best of my knowledge, this research is the first example of using an algorithm to evolve characteristics of its own migration strategy.

The generality of the results is apparent: the abstract nature of the research follows from the fact that a MOEA can be applied to any problem in which multiple objectives can be traded off against one another. The specific problem investigated in this research happened to be the application of the algorithm to its own behaviour in a distributed topology, but with little modification other similar abstract problems can be modelled.

Within the scope of this research, we can be confident that we have seen a concrete effect in finding migration topologies using an MOEA. However, the results when compared to static topologies show that some static topologies are still better than the evolved ones. Some of the reasons for this are apparent in the research method, and suggestions are given below for how this may be improved. In particular, the chosen methods for evaluating spreads tend to favour certain topologies – addressing this problem is likely to improve the results.

The overall conclusion, however, is that the research objective has been met; and that this work represents a contribution to knowledge in the area of multi-objective evolutionary algorithm research and in graph optimisation in particular.

6.2. Future research

The following further research areas might be pursued.

- Testing other meta-level objective functions
- Investigations into better migration strategies
- Testing more sophisticated graph encoding techniques and devising more graph-oriented algorithms

6.2.1. Other meta-level objective functions

The focus in this research has been on using the cost and diversity of the resulting multi-objective solutions in the meta-level algorithm. Other metrics may be chosen, in particular the convergence characteristics of an algorithm. In this case, there may or may not be a trade-off between the cost, diversity and convergence.

One possible research topic would be to investigate this using similar robust statistical methods as applied in this research. Coello Coello (2005) points out that there are no generally accepted definitions of convergence and diversity (among others) in the research community. The approach taken in this research has been to use a convenient measure of diversity that is closely related to the selected algorithm (NSGA-II). A more generalised measure of diversity could be applied that produces better, different or more general results.

One hypothesis is:

- There is a trade-off between diversity and convergence for MOEAs and rapid convergence may come at the expense of poor diversity. Can this be shown experimentally?

6.2.2. Better migration strategies

The migration strategy used in this research does not necessarily lead to improving diversity. Previous research (Power et al., 2005)) has investigated some aspects of this, and further research could be done into the methods used to select individuals for migration.

Another (perhaps controversial) observation is that migration in PDMOEAs may in fact have a detrimental effect by consuming resources but delivering no benefit. In some of the literature, some benefits of distributed parallelisation are obvious (it is easy to achieve a speedup), but some other benefits are somewhat taken for granted. For example, Hiroyasu et al. (1999a) compare a randomised migration rate with a fixed migration rate – but do not consider whether migration as a whole is beneficial.

Other modifications could be made to NSGA-II to allow for more directed migration which may have a greater effect on the resulting diversity. Coupled with this, the method chosen to assess the spread of a population could be improved.

In summary, we might have the following hypotheses:

- There is no statistical difference between random migration and *best-sent worst-replaced* migration for MOEAs.
- Can a migration strategy be coupled directly to the fitness function to promote higher diversity for a particular algorithm (say NSGA-II)?

6.2.3. Graph optimisation using evolutionary algorithms

The area of graph optimisation in general using evolutionary algorithms deserves further research. The methods used to generate graphs using crossover and mutation operators could be compared. One such research question might be to establish whether a particular method of encoding a graph is any more effective than another method by comparing the convergence characteristics of simulating using each method.

Finally, following from the above, more research could be done into the use of multi-objective algorithms in graph optimisation in general. More sophisticated and mathematically oriented approaches could be taken.

The last hypothesis is then:

- A parsimonious graph representation would allows an evolutionary algorithm to avoid redundant evaluations of graphs that are isomorphic and significantly reduce simulation time.

References

Abraham, A. and Jain, L. (2005) 'Evolutionary Multiobjective Optimization', in Abraham, A., Jain, L. and Goldberg, R. (editors), *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, pp. 1-6, Springer, USA, 2005.

Alba, E. and Tomassini, M. (2002) 'Parallelism and Evolutionary Algorithms', *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, October 2002.

Alberto, I. and Mateo, P.M. (2004) 'Representation and management of MOEA populations based on graphs', *European Journal of Operational Research*, 2004, issue 159 (2004), pp. 52–65.

Amdahl's law (n.d.) [online] *Wikipedia*, http://en.wikipedia.org/wiki/Amdahl%27s_law [Accessed 21-May-2007].

Arabas, J. and Kozdrowski, S. (2001) 'Applying an Evolutionary Algorithm to Telecommunication Network Design', *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, August 2001.

Banerjee, N. and Sharan, S. (2004) 'A Evolutionary Algorithm for Solving the Single Objective Static Routing and Wavelength Assignment Problem in WDM Networks', *Proceedings of International Conference on Intelligent Sensing and Information Processing*, 2004, vol. 1, pp. 13-18.

Banerjee, N. and Kumar, R. (2007) 'Multiobjective Network Design for Realistic Traffic Models', *GECCO'07*.

Borges, C.C.H. and Barbosa, H.J.C. (2000) 'A Non-generational Genetic Algorithm for Multiobjective Optimization', *2000 Congress on Evolutionary Computation*, vol. 1, pp. 172-179, San Diego, California, July 2000.

Cantú-Paz, E. (1997) 'A Survey of Parallel Genetic Algorithms', *Technical Report IlliGAL 97003,* University of Illinois at Urbana-Champaign, 1997.

Coello Coello, C.A. (2005) 'Recent trends in EMO', in Abraham, A., Jain, L. and Goldberg, R. (editors), *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, pp. 1-6, Springer, USA, 2005.

Deb, K. (2001) *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, Chichester, UK, 2001, ISBN 0-471-87339-X.

Deb, K., Agrawal, S., Pratab, A. and Meyarivan, T. (2000) 'A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II', *KanGAL report 200001*, Indian Institute of Technology, Kanpur, India, 2000.

Embarrassingly parallel (n. d.) [online] *Wikipedia* http://en.wikipedia.org/wiki/Embarrassingly_parallel [Accessed 21-May-2007].

Hiroyasu, T., Miki, M., Negami, M. (1999a) 'Distributed genetic algorithms with randomized migration rate', *IEEE International Conference on Systems, Man, and Cybernetics, 1999*, Tokyo, Japan, vol. 1, pp. 689-694.

Hiroyasu, T., Miki, M. and Watanabe, S. (1999b) 'Distributed genetic algorithms with a new sharing approach in multiobjective optimization problems', *Proceedings of the 1999 Congress on Evolutionary Computation,* 1999*,* Washington DC, USA, vol. 1, 6-9 July 1999, pp. 69-76.

Hiroyasu, T., Miki, M. and Tanimura, Y. (2000) 'The differences of parallel efficiency between the two models of parallel genetic algorithms on PC cluster systems', *Proceedings of the Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region,* May 2000, vol. 2, pp. 945-948.

Hiroyasu, T., Miki, M., Kamiura, J., Watanabe, S. and Hiroyasu, H. (2005) 'MOGADES: Multi-Objective Genetic Algorithm with Distributed Environment Scheme', in Abraham, A., Jain, L. and Goldberg, R. (editors), *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, pp. 201-227, Springer, USA, 2005.

Lin, W.Y., Hong, T.P. and Liu, S.M. (2004) 'On adapting migration parameters for multi-population genetic algorithms', in *IEEE International Conference on Systems, Man and Cybernetics 2004*, 10-13 October 2004, vol. 6, pp. 5731-5735.

Lin, W.Y., Lee W.Y. and Hong, T.P. (2002) 'On self-adaptive multi-population genetic algorithms', in *IEEE International Conference on Systems, Man and Cybernetics*, 6-9 October 2002, vol. 6.

Lopez Jaimes, A. and Coello Coello, C.A. (2005) 'MRMOGA: parallel evolutionary multiobjective optimization using multiple resolutions', *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, Edinburgh, Scotland, 2-5 September 2005, vol. 3, pp 2294-2301.

Mehnen, J., Michelitsch, T., Schmitt, K. and Kohlen, T. (2004) 'pMOHypEA: Parallel Evolutionary Multiobjective Optimization using Hypergraphs', University of Dortmund, Department of Machining Technology, ISF, Germany.

Power, D., Ryan, C. and Azad, R. (2005) 'Promoting diversity using migration strategies in distributed genetic algorithms', *The 2005 IEEE Congress on Evolutionary Computation*, 2-5 September 2005, vol. 2, pp. 1831-1838.

Pullan, W. (2002) 'Optimising Multiple Aspects of Network Survivability', in *Congress on Evolutionary Computation (CEC'2002)*, vol. 1, pp. 115-120, Piscataway, New Jersey, May 2002.

Roy, A., Banerjee, N. and Das, S.K. (2002) 'An efficient multi-objective QoS-routing algorithm for wireless multicasting', in *Vehicular Technology Conference, 2002 (VTC Spring 2002)*, 6-9 May 2002, vol. 3, pp. 1160-1164.

Srinivas, N. and Deb, K. (1994) 'Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms', in *Journal of Evolutionary Computation*, vol. 2, no. 3, pp. 221-248.

Streichert, F., Ulmer, H. and Zell, A. (2005) 'Parallelisation of Multi-Objective Evolutionary Algorithms Using Clustering Algorithms', *Fourth International Conference on Evolutionary Multi-Criterion Optimization*, Guanajuato, Mexico, 2005, in *Lecture Notes in Computer Science*, vol. 3210, Lecture Notes in Computer Science, Springer-Verlag, pp. 92-107.

Student's t-test (n.d.) [online] *Wikipedia*, http://en.wikipedia.org/wiki/T_test [Accessed 05-Jan-2008].

Tanimura, Y., Hiroyasu, T. and Miki, M. (2003) 'Discussion on searching capability of distributed genetic algorithm on the grid', *Proceedings of the 2003 Congress on Evolutionary Computation 2003*, 8-12 December 2003, Canberra, Australia, vol. 2, pp. 1086-1094.

Van Veldhuizen, D.A., Zydallis, J.B. and Lamont, G.B. (2002) 'Issues in parallelizing multiobjective evolutionary algorithms for real world applications', *Proceedings of the 2002 ACM symposium on Applied computing SAC '02*, Madrid, Spain, ACM Press, pp. 595-602.

Xu, K., Louis, S.J. and Mancini, R.C. (2005) 'A Scalable Parallel Genetic Algorithm for X-ray Spectroscopic Analysis', *GECCO'05,* June 25-29 2005, Washington D.C. USA.

Index