Chapter 7

CONCLUSIONS

This thesis reviewed algorithms aimed at solving dynamic optimisation problems. New DE based algorithms for dynamic environments, which were shown to be improvements over previous DE-based algorithms, were presented. The principal findings of this thesis are summarised in this chapter. Section 7.1 gives an overview of the core content of each of the chapters. The primary objective of this study was to improve on current DE-based algorithms for optimising dynamic environments. Section 7.2 explains how the primary and the sub-objectives were achieved. Future work arising from this study is described in Section 7.3.

7.1 Summary

Chapter 2 briefly described the concepts of optimisation and optimisation algorithms. Genotypic diversity and measures of diversity were outlined. The chapter reviewed DE and approaches to self-adapting control parameters. Dynamic optimisation environments were described and it was concluded that the three major considerations when classifying dynamic environments are: the fitness landscape composition, the change types, and the change pervasiveness (refer to Section 2.5.2). The chapter argued that the three factors that influence the ability of an optimisation algorithm to optimise a dynamic environment are: the hardness of the fitness landscape, the frequency of changes, and the severity of changes.

The moving peaks benchmark (MPB) and the generalised dynamic benchmark gen-

erator (GDBG) were critically discussed. An extension to the MPB, which allows the investigation of environments in which the number of optima fluctuates over time, was presented. The discussion concluded that both benchmarks are appropriate for investigating the effect of varying the change period and number of dimensions. The MPB is best suited to investigating the effect of varying the change severity and number of optima, while the GDBG is best suited to investigating the effect of different change types and underlying functions. Chapter 2 concluded with a description of performance measures for dynamic environments. The offline error was identified as the most appropriate performance measure for use in this thesis.

Chapter 3 reviewed algorithms aimed at optimisation in dynamic environments. The two main problems associated with using DE in dynamic environments, namely loss of diversity and outdated information, were described. A literature survey of related work identified three broad categories of approaches that can be used to categorise strategies for adapting algorithms for dynamic optimisation: increasing diversity, using memory, and employing parallel searching. Seven specific strategies were identified that are commonly used in dynamic optimisation algorithms. These strategies are: change control parameters, custom individuals, refresh information, sentinels, selection, history, and multiple populations. Algorithms aimed at solving dynamic optimisation problems were discussed in terms of their base algorithms: GA, PSO, DE, and other algorithms. The strategies that are used by each of the algorithms were indicated. Two DE-based algorithms, DynDE and *jDE* were discussed in detail. Section 3.4.1 showed that the subcomponents of DynDE allow it to maintain much higher diversity than the normal DE algorithm.

Chapters 4 to 6 presented new algorithms. These are tabulated in Table 7.1, along with their respective advantages and disadvantages to aid the discussion in the rest of this section.

Chapter 4 commenced with a description of the exclusion threshold approach that is used in this thesis in the absence of knowledge of the number of optima in the environment. The appropriate number of individuals per sub-population and the appropriate number of Brownian individuals were experimentally determined for DynDE. The results of the experiments showed that using a small sub-population size gives better results in the majority of investigated cases. While this result may seem counterintuitive, it can be

Table 7.1: Summary of novel algorithms

Description	Advantages	Disadvantages
CPE is aimed at lowering the error faster after changes occur. Only the best performing sub-population is evolved at a time.	Performed better than DynDE in the majority of experimental environments.	Did not perform better than DynDE at high change period in low dimensions.
RMC evaluates the midpoint value between the best individuals in sub-populations flagged for exclusion. Sub-populations are not reinitialised if they are located on separate optima.	Better results than those of DynDE were achieved in low dimensions.	RMC generally had a very small impact on the algorithm's performance.
CDE is the combination of CPE and RMC.	Performed better than DynDE in the majority of experimental environments. CDE also resulted in significant improvements over its predecessor algorithms. The general applicability of CDE was illustrated by the incorporation of its sub-components into *jDE*.	Did not perform better than DynDE at high change period in low dimensions.
DynPopDE is an extension of CDE which adapts the number of sub-populations during the optimisation process. DynPopDE is aimed at situations where the number of optima is unknown or fluctuating.	DynPopDE performed better than DynDE and CDE on MPB instances with various settings for the number of optima. Improvements were also found when the number of optima fluctuates.	DynPopDE did not scale well to other functions, and was generally inferior to CDE.
jSA2Ran is an adaptation of the approach of Brest *et al.* [2009] to self-adapt the scale and crossover factors.	jSA2Ran generally performed better than CDE over a wide range of benchmark instances. jSA2Ran has fewer parameters to tune than CDE.	The magnitude of improvements over CDE was relatively small.
SABrNorRes self-adapts the Brownian radius.	SABrNorRes performed better more often on the experimental environments than CDE. Large improvements were found at high change periods and low dimensions. SABrNorRes has fewer parameters to tune than CDE.	SABrNorRes proved to be inferior to CDE in 100 dimensions.
SACDE is the combination of jSA2Ran and SABrNorRes.	SACDE generally performed better than CDE over a wide range of benchmark instances. SACDE has fewer parameters to set than its predecessor algorithms.	SACDE did not perform better than CDE more often than SABrNorRes. SACDE was inferior to CDE in 100 dimensions.
SADynPopDE was created by incorporating the self-adaptive components of SACDE into DynPopDE.	SADynPopDE performed better than DynPopDE on environments where the number of optima is fluctuating. DynPopDE has fewer parameters to set than its predecessor algorithms.	SADynPopDE was inferior to DynPopDE on various settings of the number of optima on the MPB. SADynPopDE did not scale well to other functions, and was generally inferior to SACDE.

explained by the fact that smaller sub-population sizes allow the optimisation algorithms to perform more generations between changes in the environment. The benefit of greater diversity that can be achieved by using larger sub-population sizes is thus offset by the loss of generations that could have been used to converge to optima.

DynDE was extended by incorporating two novel approaches to form the competi-

tive population evaluation (CPE) and reinitialisation midpoint check (RMC) algorithms. CPE utilises a custom performance value to evolve selectively only one sub-population at a time. This results in optima being discovered in sequence rather than in parallel. The global optimum is thus discovered earlier, which reduces the average error of the algorithm. DynDE employs exclusion, a technique which reinitialises a sub-population when it converges to an optimum occupied by another sub-population. This dual convergence to a single optimum is detected by determining whether the best individuals of each sub-population are located within a threshold distance of each other. A disadvantage of the exclusion approach is that some optima are at times located within the exclusion threshold of another, and sub-populations should consequently not be reinitialised. This weakness is partially remedied by RMC's checking if a valley exists between the best individuals in each sub-population. CPE and RMC are combined to form competitive differential evolution (CDE).

CPE, RMC and CDE were evaluated on a wide range of benchmark instances in Chapter 4. Scalability studies showed that the offline errors of DynDE, CPE, RMC and CDE increase when changes in the environment become more frequent or more severe, and when the number of dimensions is increased. It was also shown that the peak function can have a considerable influence on the performance of dynamic optimisation algorithms.

Experimental evidence was presented that shows that RMC yields minor improvements, localised in low dimensional problems, over DynDE. This observation was experimentally explained by showing that situations where optima are located within the exclusion threshold of each other are much more likely to occur in low dimensional rather than in high dimensional cases.

CPE was found to perform better than DynDE in a wide range of experiments, but especially in high dimensions. No benefit of using CPE over DynDE was found on problems where changes occur very infrequently. This result is expected since CPE improves DynDE by locating better optima earlier, after changes in the environment. In the absence of frequent changes in the environment, the benefit of locating better optima early is reduced.

CDE was shown to be a slight improvement over its constituent parts, CPE and RMC, but was found to behave very similar to CPE in all cases when high numbers of dimensions were used. An analysis of the convergence profiles of DynDE and CDE found that CDE's

current error reduces faster than that of DynDE after changes in the environment. The diversity profile of CDE was found to be very similar to that of DynDE, but the average sub-population diversity of CDE was found to decrease at a faster rate than that of DynDE at the commencement of the optimisation process.

CDE and DynDE were compared in terms of average lowest error found just before changes in the environment to illustrate that CDE does not merely exploit the offline error performance measure. The experimental results showed that CDE performed better more often, in general, than DynDE on the benchmark environments. The improvements of CDE over DynDE were more pronounced at low change periods and high dimensions.

CDE was shown to compare favourably to the reported results on the MPB of other state-of-the-art algorithms. The general applicability of the competitive population evaluation and reinitialisation midpoint check approaches were illustrated by their incorporation into *jDE*. The adapted *jDE* algorithm was shown to outperform the normal *jDE* algorithm in the majority of the experimental environments.

Chapter 5 presented and motivated the dynamic population DE (DynPopDE) which was created by adapting CDE for dynamic problems in which the number of optima is unknown or fluctuates over time. DynPopDE dynamically introduces a new sub-population when stagnation occurs for current sub-populations, and removes sub-populations that are not converging to new optima in the environment.

Experiments with DynDE and CDE on problems with large numbers of optima showed that it is not an effective strategy to use the same number of sub-populations as the number of optima. Considerably better results were found when a constant number of 10 sub-populations was used in these algorithms.

A scalability study found that increasing the number of optima in a dynamic environment generally results in higher offline errors for DynDE, CDE and DynPopDE. CDE was found to perform considerably better than DynDE on problems where the number of optima is unknown. DynPopDE in turn performed better than CDE when larger numbers of optima are present. However, the improvement was more pronounced in low dimensional cases. An analysis of DynPopDE's convergence profiles concluded that the reason for DynPopDE's sub-optimal performance in high dimensions is that the population spawning and removal processes are ineffective in high dimensions. This causes DynPopDE to

be unable to distinguish between environments with large numbers of optima and environments with small numbers of optima. DynPopDE thus creates either too many or too few sub-populations in high dimensions. DynPopDE was shown to compare favourably with the reported results of other algorithms focusing on unknown numbers of optima.

Dynamic optimisation problems where the number of optima fluctuate over time have not been widely investigated by other researchers. To address this gap in the research, DynDE, CDE and DynPopDE were studied using an extension of the MPB that simulates these problems. Comparisons between DynDE, CDE and DynPopDE on problems where the number of optima fluctuates over time showed that CDE performed considerably better than DynDE. DynPopDE was found to be marginally more effective than CDE, but considerable improvements of DynPopDE over CDE were found when low change periods were used.

DynPopDE's process of spawning and removing sub-populations was validated by incorporating the spawning and removal process used in MPSO2 into CDE. The performance of the resulting algorithm, MPSO2CDE, was compared to that of DynPopDE. DynPopDE performed better than MPSO2CDE in the majority of the environments that were investigated.

A major weakness of DynPopDE was identified when evaluating its performance using various underlying functions. DynPopDE proved to be inferior to CDE on the majority of the functions of the GDBG. The GDBG benchmark instances all have different numbers of optima. If DynPopDE was truly effective at adjusting the number of sub-populations to suit the environment, it should have performed better than CDE on the GDBG environments. The set of environments on which DynPopDE can be used effectively is thus severely limited due to the strong function-dependence of DynPopDE.

Chapter 6 investigated the incorporation of self-adaptive control parameters into CDE. Strategies were considered to self-adapt the scale and crossover factors, as well as the Brownian radius. Three existing approaches were considered to self-adapt the scale and crossover factors: SDE of Omran *et al.* [2005a], Barebones DE of Omran *et al.* [2009], and the approach of Brest *et al.* [2006][Brest *et al.*, 2009] which was also used in the successful *jDE* algorithm. Since CDE uses DE/best/2/bin while both the approaches of Omran *et al.* [2005a] and Brest *et al.* [2006] use DE/rand/1/bin, these two approaches

were also investigated using DE/best/2/bin. Further alterations that were investigated included resetting the scale and crossover factors to their initial values when a change in the environment occurs, and selecting the initial values from a normal distribution.

An experimental comparison found that the approach of Brest *et al.* [2009], but in conjunction with DE/best/2/bin and with initial values selected from a normal distribution, is the most effective algorithm for incorporation into CDE. This algorithm was referred to as jSA2Ran. A comparison to CDE showed that jSA2Ran performed better on the experimental environments more often than CDE, and that improvements were more pronounced at high change periods.

Four approaches were investigated for self-adapting the Brownian radius. All the approaches involved randomly selecting the Brownian radius from a distribution controlled by the average of a history of successful Brownian radius values. The initial value which controlled the Brownian radius was set to the radius of the first sub-population that was evaluated. Two distributions were tested, i.e. Gaussian and Cauchy. Additionally, the effect of resetting the Brownian radius after changes in the environment, was investigated. An experimental comparison found that SABrNorRes, the approach that utilises a normal distribution with resetting after changes, performed the best. SABrNorRes was experimentally compared to CDE. The results showed that SABrNorRes generally performed better than CDE, especially at high change periods. SABrNorRes performed better than DynDE on environments with a high change period where DynDE generally performed better than CDE. However, it was found that SABrNorRes was inferior to CDE in the majority of 100 dimensional experiments.

SABrNorRes and jSA2Ran were combined to from SACDE. SACDE performed better more often on the benchmark environments than CDE. A scalability study found that SACDE and SABrNorRes generally exhibited very similar scaling behaviour. Despite this fact, SABrNorRes performed better than CDE more often than SACDE. SACDE does, however, have the advantage of having fewer parameters than SABrNorRes.

An investigation into the convergence profiles of the new algorithms found that the reason why the self-adaptive algorithms performed better than CDE at high change periods is that the current errors of the self-adaptive algorithms reached lower values than those of CDE. The self-adaptive algorithms thus not only have the advantage of a fast reduction

in current error after a change in the environment, but also in the low current error value that is achieved.

The diversity profiles of the self-adaptive algorithms showed that jSA2Ran's diversity did not meaningfully differ from that of CDE, but that SABrNorRes and SACDE exhibited different behaviour. The overall diversity and average sub-population diversity of SABrNorRes and SACDE sharply increased after changes in the environment. The average sub-population diversity of SABrNorRes and SACDE was generally higher than that of CDE.

Incorporating the self-adaptive approach used in SACDE into DynPopDE to form SADynPopDE did not result in a more effective algorithm on problems where the number of optima is unknown, although SADynPopDE did perform slightly better than DynPopDE over a range of functions. Large improvements of SADynPopDE over DynPopDE were found on environments where the number of optima fluctuates over time. SACDE also showed considerable improvements over CDE on these environments, but was inferior to SADynPopDE. A comparison to other algorithms found that SACDE compares favourably to state-of-the-art algorithms aimed at dynamic environments.

7.2 Achievement of Objectives

The objectives of this study were outlined in Section 1.2. The primary objective of this thesis was to create improved DE-based algorithms for dynamic environments. Eight new algorithms were created and were shown to yield better results than existing DE-based algorithms. The sub-objectives were achieved as follows:

- Existing algorithms aimed at solving dynamic optimisation problems, specifically focusing on algorithms based on differential evolution, were identified and reviewed.
- Three broad categories of approaches to solve dynamic optimisation problems were found in the literature. Seven general strategies used to achieve the three approaches were identified.
- DynDE was extended to form CDE and DynPopDE, and hybridised with *jDE* to form SACDE and SADynPopDE.

- Dynamic optimisation problem environments that have not been thoroughly investigated by other researchers, namely environments in which the number of optima fluctuate over time, were investigated. Scalability studies were also conducted using larger experimental sets than those generally used by other researchers.
- The use of adaptive control parameters to reduce the number of parameters that must be manually tuned was investigated. The number of sub-populations is adapted in DynPopDE, while SACDE self-adapts the DE scale and crossover factors and the Brownian radius.
- Extensive scalability studies were performed on existing and newly created algorithms.
- The performances of the new algorithms were compared to the reported results of other algorithms.

All the objectives of this study were achieved.

7.3 Future Work

The components that make up CDE, namely competing populations and reinitialisation midpoint check, do not depend on any DE specific behaviour. Future studies could investigate the general applicability of these components on PSO and GA-based algorithms for dynamic environments. Section 3.3 highlighted the many similarities of DE, PSO and GA-based approaches to solving DOPs, for example, the use of multiple populations or swarms. Hybrid algorithms can be formed by incorporating facets of the approaches presented in this thesis into existing PSO and GA-based algorithms. The resulting hybrid algorithms may be an improvement on current algorithms.

The approach to spawning and removing sub-populations, which was introduced in Chapter 5, was found to be beneficial on specific environments, but did not scale well to different functions. This limits the current applicability of DynPopDE. Future work can include the improvement of DynPopDE to extend its usefulness to more environments. DynPopDE does have the advantage of not having to tune the number of sub-populations

parameter. Research that results in an improved technique of spawning and removing sub-populations may enable DynPopDE to be as effective as CDE over a wide range of problems, while still retaining DynPopDE's advantages.

jDE is currently one of the state-of-the-art DE algorithms aimed at dynamic environments. SACDE was shown generally to outperform *jDE* on a wide range of dynamic environments. However *jDE* did perform better than SACDE on several benchmark instances. The self-adaptive scale and crossover component of *jDE* was incorporated into SACDE and showed improved results. Future work could include further hybridisations of the two algorithms which could result in further improvements. For example, the aging metaphor used by *jDE*, which results in increased diversity, could potentially improve the performance of SACDE on DOPs with large change periods by preventing the algorithm from stagnating.

The scope of this thesis was delineated to exclude environments in which the number of dimensions changes. The algorithms developed in this study are consequently not suited to environments with a dynamic number of dimensions. Future work could include the application of DE to problems with dynamically changing constraints. The resulting algorithms would be more generally applicable than CDE, DynPopDE, SACDE and SADynPopDE.