Optimising the Flow of Experiments to a Robot Scientist With Multi-Objective Evolutionary Algorithms

Emma Byrne
Department of Computer Science
University of Wales, Aberystwyth
elb@aber.ac.uk

ABSTRACT

A Robot Scientist is a physically implemented system that applies artificial intelligence to autonomously discover new knowledge through cycles of scientific experimentation. Additionally, our Robot Scientist is able to execute experiments that have been requested by human biologists. There arises a multi-objective problem in the selection of batches of trials to be run together on the robot hardware. We describe the use of the jMetal framework to assess the suitability of a number of multi-objective metaheuristics to optimise the flow of experiments run on a Robot Scientist. Experiments are selected in batches, chosen in order to maximise the information gain and minimise the use of resources. The evolutionary multi-objective algorithms evaluated here perform well in finding solutions to this problem, either finding a long, fairly efficient Pareto optimal front, or a shorter, highly efficient Pareto optimal front.

Categories and Subject Descriptors

J.3 [**Computer Applications**]: Life and Medical Science—*Biology and Genetics*; I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*

General Terms

Algorithms, Performance

Keywords

Evolutionary multi-objective optimisation, Robot Scientists, Experiment flow optimisation, Yeast genomics, Evaluation, Performance

1. INTRODUCTION

A Robot Scientist is a physically implemented robotic system that applies techniques from artificial intelligence to execute cycles of automatic scientific experimentation. Our recently built Robot Scientist[1] is one of the most advanced laboratory automation systems in existence. The advances that distinguish our Robot Scientist are its AI software, the design of the hardware and the complete autonomy of the knowledge discovery process [3].

A Robot Scientist has hardware that permits it to carry out experiments (see Figure 1). These experiments may have been devised by the Robot Scientist itself or they may have been requested by human biologists. For details of how our Robot Scientist autonomously devises and executes cycles of scientific discovery, see Section 2.

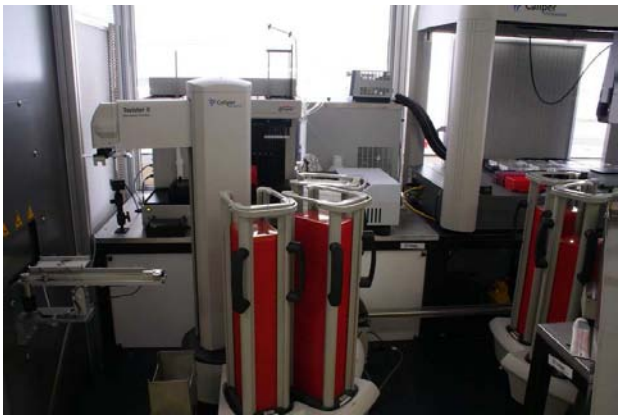

Figure 1: The hardware system of our Robot Scientist. Visible here are one of the robot arms (twisters). The automated freezer library in which yeast strains are stored is on the left and the chemical dispenser is on the right. Several other components, including plate readers and incubators, are not shown.

One of the tasks any Robot Scientist must perform is to select batches of experiments to be run together from a database. In batching these experiments, there are multiple objectives to consider such as: maximising information gain, minimising the use of resources and ensuring that the batch is able to be run according to the physical constraints of the robot hardware. This problem has all the characteristics of a multi-objective problem. We wish to optimise along several orthogonal dimensions simultaneously. In addition, these dimensions are epistatic in that the values in each dimension are dependent on, but not a function of, the values

[1]http://www.aber.ac.uk/compsci/Research/bio/robotsci/

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

in the other dimensions. We call the problem of generating a Pareto optimal set of such batches the experiment flow optimisation problem.

A multi-objective evolutionary approach is an appropriate solution to our problem. There are several state of the art metaheuristics that have been suggested as potential approaches to solving multi-objective problems. By their nature multi-objective problems admit of several solutions, all of which may be Pareto optimal[2]. Several existing, state of the art metaheuristics have been shown to give good results on abstract multi-objective problems. These metaheuristics include: the multi-objective genetic algorithms SPEA2 [11], NSGA-II [1] and PAES [4]; the particle swarm optimiser (PSO) OMOPSO [9] and the scatter search method AbYSS [6].

The jMetal framework, described in detail in [2] implements these metaheuristics, alongside libraries of operators for selection, mutation and crossover. We have used this framework to evaluate existing metaheuristics against a new, real world problem: experiment flow optimisation for a Robot Scientist.

1.1 Important Terminology

In this paper the term *experiment* always refers to a biological experiment carried out on a Robot Scientist. A *trial* always refers to a set of such experiments. A *run* refers to a single execution of a metaheuristic applied to the experiment flow optimisation problem for a Robot Scientist.

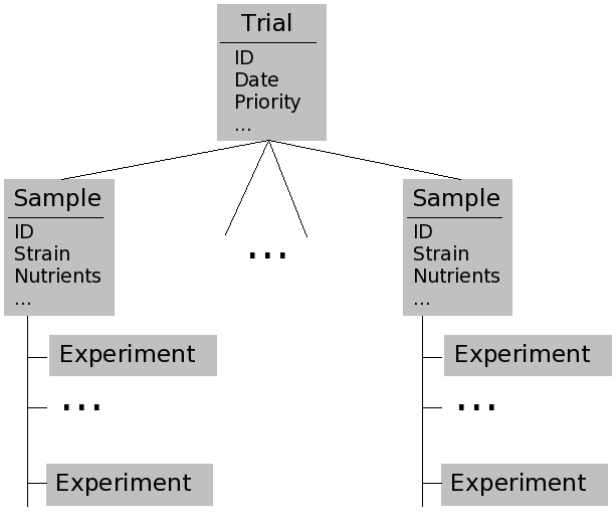

Figure 2: A simplified model of the relationship between Trials, Samples and Experiments.

Figure 2 shows the relationship between the terms that describe the biological experiments that our Robot Scientist performs. Trials are sets of work that we wish the robot to carry out. Each trial consists of a number of *samples*, usually one sample of interest and several controls. A knockout strain is a strain of yeast that has had one or more genes removed. Wild type yeast is yeast that has had no genes removed. A nutrient is a chemical that is necessary for the yeast's growth. Each sample is a pairing of a knockout or wild type strain of yeast and one or more nutrients. Each experiment is one of many replicates of a sample. Each experiment is individually grown in a single well of a 96 well plate.

[2]A solution is Pareto optimal with respect to a set of solutions if and only if the solution is not outperformed on at any objective

2. THE ROBOT SCIENTIST

Our Robot Scientist is capable of autonomously designing and initiating >1,000 new strain/nutrient experiments each day, with each experiment lasting up to 4 days, using over 50 different yeast strains. Not only are the experiments automated and performed by laboratory robots, but the hypotheses are generated by computer. Our Robot Scientist uses artificial intelligence to derive hypotheses concerning the functions of genes in yeast and to devise experiments to test these hypotheses. Experiments are designed by intelligent software and executed on the robot hardware. The results are analysed automatically and are fed back into the next round of hypothesis formation and experimentation [3].

Our Robot Scientist has both an intelligent (software) architecture and a lab automation (hardware) architecture. Figure 3 shows how these two elements work together. Grey rectangles represent the functions of the AI features of our Robot Scientist; the three dimensional box represents the functions that are the responsibility of our Robot Scientist's hardware; cylinders represent data storage and the oval represents submissions of trials from human biologists. All trials, whether human or robot generated, are submitted via a Grid interface. We make the robot hardware, along with a number of our Robot Scientist's services, available to other institutions as an Equipment Grid, via this Grid Interface.

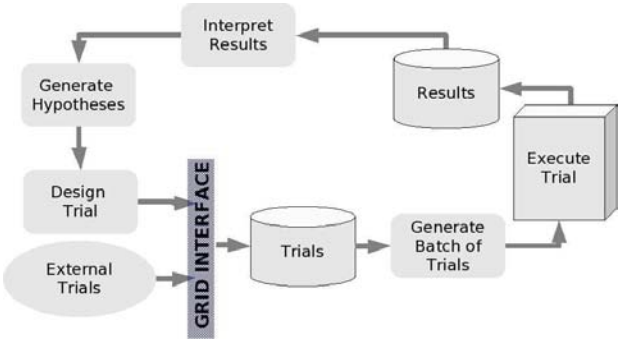

Figure 3: The cycles of experimentation carried out by our Robot Scientist.

Our Robot Scientist autonomously discovers new biological knowledge about the enzymatic function of genes in *Saccharomyces cerevisiae* (beer yeast). Our Robot Scientist's intelligent hypothesis generator uses abductive logic programming, a repository of background knowledge about yeast metabolism and experimental results to determine the function of individual genes or sets of genes. The experiments are carried out on the robot hardware and the results fed back to the hypothesis generator. Each experiment consists of a small amount of a strain of yeast that is either wild type or that has had one or more genes knocked out. This is grown in a cocktail of nutrients. By comparing the growth of knockout and wildtype yeast, with and without nutrients, it is possible to abduce the enzymes that the missing gene(s) code for.

Our Robot Scientist's hypothesis generator uses abductive logic programming to derive sets of plausible functions for a given gene in *S. cerevisiae* based on previous experimental data and background knowledge about the organism. Specifically, it determines which enzymes a given gene may code for. If there is only one plausible function for this gene then it is tentatively accepted as new knowledge. If there are several plausible functions, these are sent to an experiment generator. This experiment generator uses background knowledge about *S. cerevisiae* and about experiments, to design a trial (a set of connected experiments) that will lead to the greatest information gain, in terms of how many of these hypotheses it may rule out, at the lowest cost. The chosen trial is then submitted via a Grid interface to a database of queued trials. Human biologists may also submit trials via the same Grid interface.

Trials are run on the robot hardware. The hardware consists of a number of lab automation tools that work together to store, dispense, incubate and observe the samples that make up each trial. A diagram of the robot hardware is available in [10]. Growth of the yeast samples is monitored using an optical density (OD) reader that measures the attenuation of light through a single well of a 96 well plate, which contains an individual experiment. These measurements are returned to our Robot Scientist via a database, and are accessed by a module that interprets these results to determine whether significant growth has occurred in the samples. It is these results that are then fed in to the next round of hypothesis generation.

On submission to the database, trials are given a time stamp and an ID. We need to send the trials in the database to the robot hardware in batches. It is here that the experiment flow optimisation problem occurs.

3. EXPERIMENT FLOW OPTIMISATION

The batch selection process must autonomously select sets of experiments that can be run together and that optimise the trade-off between information discovery and resource use. We need the batch maker to be able to optimise the batch of trials it selects on several dimensions. The experiment flow optimisation problem requires that we:

- maximise the number of trials that will be run in this batch
- minimise the amount of resources (e.g. 96 well plates) that will be used
- favour those trials that have been flagged as high priority
- favour those trials that have been queued the longest, by maximising the amount of time that the trials in the selected batch have been waiting
- constrain the number of chemicals to the maximum number of dispensing reservoirs on the liquid dispenser - this is a hard constraint
- minimise the amount of time that samples remain out of the freezer

For ease of analysis, we only consider the number of trials, number of plates and amount of wait time in this paper. Restricting analysis to three dimensions simplifies the evaluation of the metaheuristics. We are attempting to maximise the number of trials present in the batch, minimise the number of plates used and maximise the amount of queue time for the trials in the batch, which is equivalent to trying to run the oldest trials first.

3.1 Using evolutionary multi-objective optimisation

There are multiple features of the selected batches of trials that must be optimised and the features that must be optimised are incommensurable. For example, the length of time that experiments have been queued and the number of trials to be executed are measured in different units and there is no non-arbitrary conversion between the two.

The features to be optimised are epistatic, that is, the value of one feature is dependent on one or more of the others. For example, the number of plates required to run the trials is dependent on (but not a function of) the number of trials to be run in the batch. We wish to maximise the number of trials in a batch whilst simultaneously minimising the number of plates. We have two options: i) to combine these different dimensions into one single fitness measure or ii) to develop a set of Pareto optimal solutions. Option i) requires some formula that reconciles these various incommensurable, yet epistatic features into a single function. There is no meaningful way of doing so - we cannot determine *a priori* how many plates it is "worth" expending in order to run an extra n trials.

We favour multiple answers that may be better on different dimensions. For example we may wish to give the robot the option to select from among the Pareto optimal batches the one that uses the fewest plates for example, or the one that exhausts the most queuing time. Finally, if we extend the range of work that we require the robot scientist to perform, by adding new hardware for example, we need to be able to add extra dimensions to the optimisation task. Adding extra dimensions is simply a matter of augmenting the fitness function in order to derive the necessary values.

The landscape of the experiment flow optimisation problem is rugged. This is a result of the epistatic nature of this problem. Every step in a given dimension may also result in a change of value in all other dimensions. For this reason, an evolutionary approach, which maintains a diverse population of solutions, and which explores the space in all dimensions simultaneously, is ideal for our needs.

The features of the experiment batching problem make it particularly suited to an evolutionary multi-objective optimisation approach. Firstly, we have multiple features of the solution that we wish to optimise simultaneously. Evolutionary multi-objective optimisation methods permit us to evolve a Pareto optimal front. This allows us to search for solutions (batches) that are fit in all dimensions of fitness and excel in one or more dimensions. It also results in a *set* of solutions rather than a single solution. One solution can be chosen from amongst this population.

3.2 Evolutionary multi-objective optimisation in other experimental design tasks

There are a number of other scientific discovery tasks that have successfully exploited the evolutionary multi-objective optimisation approach. For example, [8] describes using NSGA-II in the domain of photo-chemistry, in order to pre-

dict the characteristics of molecules in excited- and ground-state. In [7], the MOGA PESA-II is successfully used to optimise the settings of gas chromatography and time-of-flight mass spectrometry instrumentation used in biological experiments.

However, our approach is unlike these. The range of experiments it is possible to carry out on our Robot Scientist is determined by the large number of strain and nutrient combinations that it is possible to select. For each strain of yeast, it is possible to carry out in the order of 10^{16} experiments consisting of unique combinations of available nutrients. Our Robot Scientist relies on abductive logic programming and extensive domain knowledge to select the most useful trials from this large space. The need for multi-objective optimisation arises not at the point of experimental design but at the point of deciding which trials should be run in one batch.

4. METAHEURISTICS

jMetal [2] is a Java-based framework that contains a suite of metaheuristics and the generic operators from which these are constructed. The framework was designed to facilitate the design of new multi-objective metaheuristics. However, for this evaluation, the suite was used to evaluate the performance of existing metaheuristics on our real world, multi-objective optimisation problem.

The metaheuristics available in jMetal were chosen as they are among the state of the art in evolutionary multi-objective optimisation. Importantly, the approaches we have chosen are well documented. The jMetal documentation [2] demonstrates that the implementations in the framework are correct with respect to their reference implementations.

This paper will examine the performance of three MOGAs: PAES, NSGA-II and SPEA2, a hybrid scatter search approach,pae AbYSS, and a multi-objective particle swarm optimiser, OMOPSO. These have been chosen because of the good performance they have demonstrated in recent studies, and also because they cover several diverse evolutionary methods. A brief review of these methods is presented here, in order to assist the reader in understanding the results of the evaluation.

4.1 PAES

PAES stands for the Pareto Archived Evolution Strategy [4]. The version implemented in jMetal is (1+1)PAES, a gradualist approach that eschews crossover, using mutation only to explore the space. PAES has an aggressive approach to discarding dominated solutions - if the parent dominates the child, the child is rejected; if the child dominates the parent, the parent is rejected; if parent and child mutually non-dominate, and the child dominates solutions in the archive, the child becomes the new parent and all solutions in the archive dominated by the child are removed; finally, if parent and child mutually non-dominate, and the child dominates no solution in the archive, the one in the most crowded space (with respect to the archive) is rejected with the other becoming, or remaining, the parent. In [4], Knowles and Corne demonstrate that this leads to performance with relatively low time complexity compared to other, contemporary methods. Additionally, they demonstrate that PAES generates a population that has a good approximation to the Pareto front for, among others, the routing problem.

4.2 NSGA-II

NSGA-II [1] is a variant of NSGA (Non-dominated Sorting Genetic Algorithm) that addresses the computational complexity of earlier approaches by adopting a fast non-dominated sorting approach that enables elitism and that also reduces the computational complexity of the algorithm. This approach is described in detail in [1] but briefly it consists of iteratively searching for non-dominated fronts and uses a novel book-keeping strategy that significantly reduces the number of comparisons to be made in this search. A crowding operator is used to maintain population diversity, rather than the sharing operator used in the original NSGA. This eliminates the need to set a share parameter σ_{share} *a priori*. Crowding also has lower computational complexity than sharing. NSGA-II also introduces a constraint handling approach that gives inferior rank to individuals in proportion according to the degree to which they violate constraints.

4.3 SPEA2

SPEA2 [11] builds on SPEA (Strength Pareto Evolutionary Algorithm), but takes into account some recent improvements in evolutionary multi-objective optimisation methods, notably in NSGA-II and PESA. SPEA2 improves on the original SPEA algorithm in that it: has an improved fitness assignment scheme; incorporates a density estimation technique based on the k-nearest neighbour metric and uses an archive truncation method that maintains boundary solutions.

In the original SPEA metaheuristic the strength of a solution was calculated with respect to dominance of members of the *archive* only. This led to poor ranking where, for example, the archive contained only one individual. To address this problem SPEA2 maintains an archive of constant size and measures strength with respect to both the archive and the working population. The SPEA2 fitness measure incorporates a raw fitness value (the sum of the strength measures of the solutions that dominate an individual) and the distance metric, which penalises solutions that are close to other individuals.

The fixed archive size demands strategies to address both the situation where the number of non-dominated solutions is smaller than the archive size and the situation where the number of non-dominated solutions exceeds the archive size. In the former case, the best individuals are selected from the dominated members of the working population in order to bring the archive up to size. In the second case a truncation operation is carried out. This truncation operator exploits the k-nearest neighbour metric in order to remove members of the archive that are in the most crowded areas of the front. Using the k-nearest neighbours metric ensures that the solutions at the outer limits of the Pareto front are maintained in the archive.

4.4 OMOPSO

OMOPSO is a Multi-Objective Particle Swarm Optimiser (MOPSO) that has been shown to obtain superior results when contrasted with other MOPSOs [9]. It develops the standard Particle Swarm Optimisation (PSO) approach of creating a population of individuals (particles) that move about the search space in an attempt to emulate the performance (by mimicking the velocity) of the best performing particles. In order to achieve multi-objective optimisation, MOPSOs identify a *set* of leader particles on the

Pareto front, rather than a single leader in the case of PSOs. OMOPSO uses the methods of Pareto dominance and crowding in order to select the set of leading particles, and incorporates three types of mutation (no mutation, uniform mutation and non-uniform mutation) each applied to a subset of the particles in the swarm, in order to ensure that exploring and exploiting behaviours are applied to the search space. Finally, in order to limit the size of the solution archive, OMOPSO uses ϵ-dominance to divide the search space into a grid, and restrict the population to a single non-dominated solution in each grid block.

4.5 AbYSS

AbYSS is an Archive-based hYbrid Scatter Search approach to solving multi-objective problems [6]. It is based on the standard scatter search techniques of deterministically combining members of a reference set in order to generate new solutions to a given problem. Scatter Search methods conventionally exclude stochastic methods but AbYSS incorporates stochastic operators in order to better address multi-objective problems. AbYSS combines operators used by three MOGAs: the (1+1)PAES method is used to carry out local search as the systematic improvement method. The constrained dominance method developed in NSGA-II is used as a niching measure in determining the fitness of solutions thus generated. The SPEA2 fitness measure is used when selecting individuals to enter the archive.

AbYSS generates two reference sets, R_1 and R_2. In R_1 are stored the solutions that are considered best, via the SPEA2 fitness measure. R_2 contains those individuals with the best distance values, that is, those that are furthest from their neighbours. For each pair of individuals in R_1 and for each pair of individuals from R_2, pair-wise crossover is carried out to generate new solution subsets. In [6], linear crossover was tried, as is standard for the scatter search approach, but SBX (simulated binary crossover) proved to be more effective. Using SBX requires relaxing the scatter search restriction on using stochastic search measures, but was considered by Nebro *et al* in [6] to be justified by the better performance that results.

5. APPLYING JMETAL TO EXPERIMENT FLOW OPTIMISATION

To simplify the analysis of results, the decision was taken to use simulated data whose characteristics were known, rather than using real data. This data was generated using our Robot Scientist's trials database structure as a template. Java's built in pseudo-random number generator, together with knowledge of the distribution of strains, chemicals and replicates typically used in trials, was used to create individual records. The data therefore has the same structure as, and is of a similar distribution to, the data that would be submitted to our Robot Scientist. Sets of 100 trials were generated using a Java application that generates the sort of data we might retrieve from the trials database. 100 trials is at the upper limit of the number of trials we might expect to be waiting to be batched at any one time. A set of 100 trials consists of 400 samples and around 20,000 individual experiments. A trial is described by:

- Trial ID
- the date on which it was submitted
- the priority of the trial (low or high), and
- the samples used in the experiment

Each sample record contains details of:

- the knockout strain of yeast that is being grown
- the nutrients in which it will be grown, and
- how many copies (replicates) of this sample are required

Each replicate is an individual experiment that will be grown in an individual well of a 96 well plate.

The representation of the decision variable is an array of bits. This array is the same length as the set of trials from which the batch must be selected. In the case of these simulations, the array is 100 bits long and each bit represents one of the 100 trials from which a batch must be selected. A 1 in position n indicates that trial n should be included in the batch, whilst and 0 in position n indicates that trial n should be excluded from the batch.

A fitness function was written to evaluate batches of trials. There are three objectives in this function: to maximise the number of trials to be run in the batch; to minimise the number of plates used in the batch and to maximise the amount of waiting time that will be exhausted by running these trials. The decision variable determines which trials will be in a batch. The fitness function returns three values for a given batch: the sum of the number of trials; the sum of the number of plates these trials will require and the sum of the amount of time that these trials have waited.

This fitness function was used with five of the metaheuristics provided with jMetal: Abyss, OMOPSO, NSGA-II, PAES and SPEA2. Each of these metaheuristics was run 20 times against the set of trial data. The function values (trials, plates and time waiting) and the variable (the array of bits that determines which of the trials are in the batch) of the resulting populations were stored.

Where there is an option to set these variables, the maximum, initial and archive population sizes are set to 100. For each of the metaheuristcs the number of evaluations was 25,000 except for OMOPSO. OMOPSO has a parameter for the maximum number of iterations, which was set at 250, as the time taken to run 25,000 iterations was excessive.

For consistency's sake the single point crossover (probability: 0.9) and polynomial mutation (probability: 0.01) operators were used in each run. In future work we will look at the effect of tailoring these operators using, *inter alia*, the operators available in the jMetal library.

6. RESULTS

At this stage, we restrict ourselves to a qualitative analysis of the performance of these metaheuristics. Figures 4 to 7 show the comparative performance of the metaheuristics we have applied to the experiment flow optimisation problem. In future work we intend to carry out quantitative comparisons of these metaheuristics, via binary quantitative comparisons, such as Two Set Coverage and Two Set Hypervolume (examples of these methods are available in [9]).

Each set of points is the set of results found in 20 runs over the set of 100 trials. The more closely these points coalesce around a line, the more reliably the metaheuristic returns

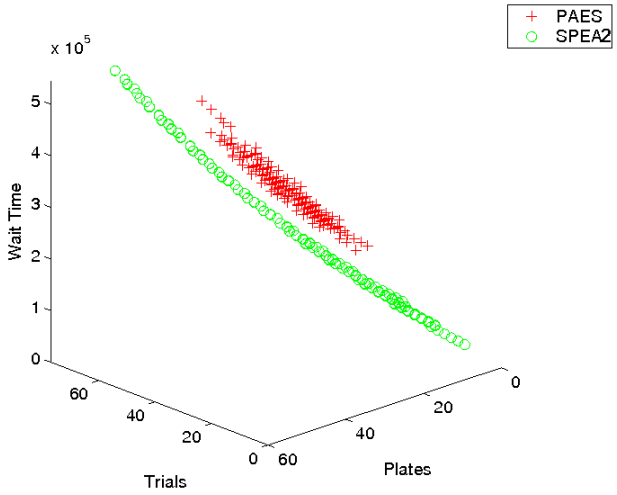

Figure 4: SPEA2 and PAES performance: 20 runs on data for 100 trials. The Pareto fronts discovered by the PAES metaheuristic contain individuals that dominate those discovered by SPEA2, but SPEA2 explores more of the Pareto front

the same front on each run. The more "rightward" the line, the more optimal the front, that is: the more trials, fewer plates and most exhausted queue time each batch contains. The longer the length of the line from the minimum to the maximum values for each axis, the greater the spread of results discovered in the population.

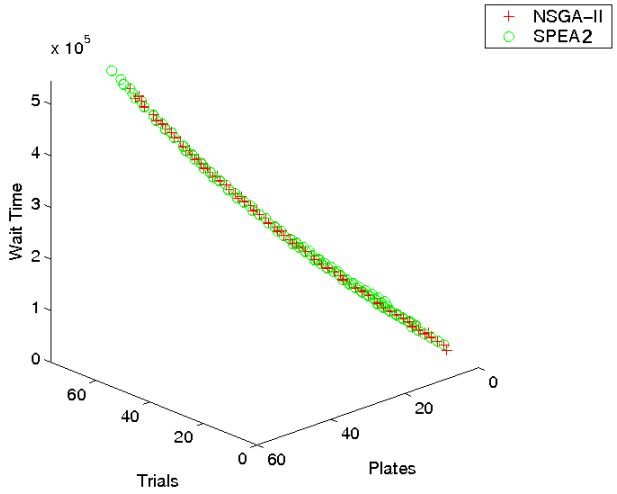

Figure 5: SPEA2 and NSGA-II performance: 20 runs on data for 100 trials. Both metaheuristics result in similar Pareto fronts.

This section presents the results for the metaheuristics evaluated. Section 7 discusses the implications of these results in depth.

Figures 4 and 5 show the behaviour of the three MOGAs: PAES, NSGA-II and SPEA2. SPEA2 and NSGA-II achieve similar results. The fronts that they find are near-identical, with SPEA2 covering more of the Pareto front. PAES covers much less of the Pareto front. However, the fronts that PAES identifies are superior to those uncovered by either NSGA-II or SPEA2.

Figures 6 and 7 show the performance of the the hybrid scatter search metaheuristic, AbYSS, and the particle swarm optimiser, OMOPSO, with SPEA2 shown for the purpose of comparison. AbBYSS finds a similar front to both NSGA-II and SPEA2. However, it truncates the front, and thus does not return values for the lower boundaries of the Pareto front. Nevertheless, it does result in a somewhat better front than either NSGA-II or SPEA.

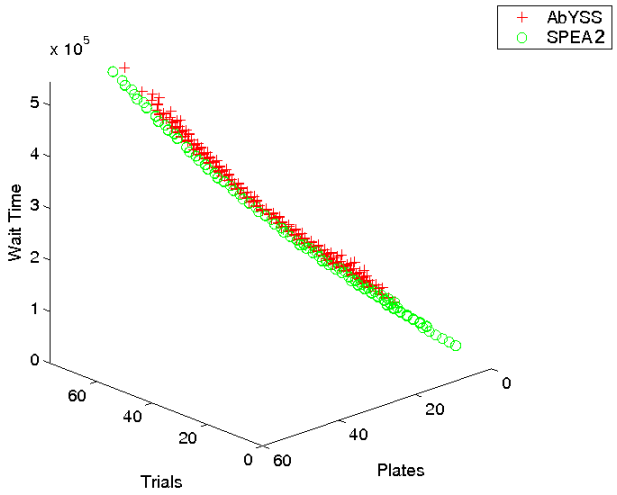

Figure 6: SPEA2 and AbYSS performance: 20 runs on data for 100 trials. The Pareto fronts returned by AbYSS dominate those returned by SPEA2, but AbYSS explores less of the front than SPEA2.

OMOPSO (Figure 7) results in a population that covers less of the front than those populations identified by SPEA2 and NSGA-II, but it identifies superior points to either of these methods, and to AbYSS. The fronts that OMOPSO returns are similar to those returned by PAES.

Mean runtimes were calculated for the five metaheuristics. The PSO OMOPSO was run over 250 iterations rather than the 25,000 generations that were run for the MOGAs and AbYSS. Performance at this number of iterations was similar to the performance of the other metaheuristics. Runtime for each metaheuristic is, on average, less than two minutes, with AbYSS and OMOPSO having mean runtimes under one minute. For a batch process, all of these runtimes are acceptable.

7. CONCLUSION

Each of the evolutionary multi-objective optimisation methods applied to the problem returned Pareto fronts that contained individuals that represent efficient batches of trials.

The particle swarm method, OMOPSO, and the MOGA PAES returned the most efficient batches, but covered only a fraction of the front. It is notable that both of these approaches experienced particular difficulties with the lower part of the Pareto front (few trials, few plates and less exhausted queue time). This is the area of the front that is most rugged: small changes in the number of trials lead to proportionally large jumps in the number of plates required for example.

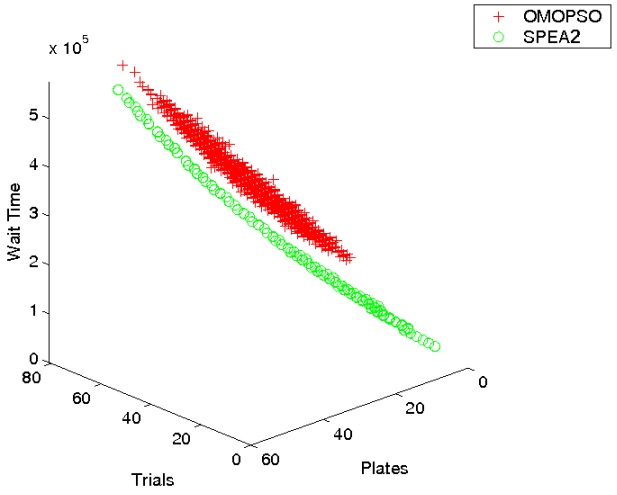

Figure 7: SPEA2 and OMOPSO performance: 20 runs on data for 100 trials. The Pareto fronts discovered by the OMOPSO metaheuristic contain individuals that dominate those discovered by SPEA2, but SPEA2 explores more of the Pareto front than OMOPSO

Both OMOPSO and PAES have aggressive strategies for adding solutions to the archive. This extra selection pressure is likely to be responsible for both the higher performance of these metaheuristics on the experiment flow optimisation problem, and for the lower diversity in the resulting populations. It may be necessary to accommodate a higher number of dominated solutions in the parent population than are retained by these approaches, in order to explore this area of the search space.

Both SPEA2 and NSGA-II also returned similar fronts and both manage to explore the rugged territory in the lower portion of the front. In SPEA2, diversification pressure results from the use of a fitness measure that combines both a dominance measure and a k-nearest neighbour distance measure. The truncation operator also exploits the k-nearest neighbour metric in order to preserve both the diversity of the population, and the solutions that lie at the ends of the front. SPEA2 also admits dominated solutions to the archive whenever the set of non-dominated solutions is smaller than the archive size. The preservation of dominated solutions may be instrumental in the ability of SPEA2 to explore the most rugged areas of the Pareto front. NSGA-II also promotes diversity through the use of a crowding operator.

AbYSS adopts the crowding distance metric from NSGA-II for selecting the individuals that will be copied to the external archive, and the density estimation measure from SPEA2 is used to determine which individuals should be in the reference set. As such it is unsurprising that the diversity of the populations returned by AbYSS should approach that of both NSGA-II and SPEA2.

However, AbYSS is still foiled by the rugged, lower end of the front. This is perhaps the result of maintaining two reference sets: one that privileges diversity and another that privileges dominance. It may be that the segregation of unfit but diverse solutions prevents the development of better solutions in the lower portion of the front, by reducing the opportunity for "diverse" individuals to mate with fitter individuals. Further work is required to determine if this is the case.

AbYSS manages to improve on the Pareto optimality of the front returned by both NSGA-II and SPEA2. The scatter search improvement method, which here, as in PAES, takes the form of a (1+1) evolution strategy, appears to exert a notable pressure towards more optimal solutions by systematically searching the area around each point.

The decision as to which of these metaheuristics best serves our needs for the experiment flow optimisation problem depends on the use to which the population of individuals will be put. A Robot Scientist must select a batch of trials from among the population of batches that allows it to make the best use of the resources available. From the resulting Pareto front, a Robot Scientist must choose which batch of experiments to run together. This may be done by selecting the batch that is most efficient on one of the given objective dimensions from among those Pareto optimal individuals in the front. The dimension chosen depends on the availability of resources or the number of trials queued.

We do not know, *a priori*, what the optimal batches for a given set of trials would be. We do know that an individual batch that contains more trials, eliminates more queuing time and requires fewer plates is preferred over a batch that is weaker in any one or more of those dimensions. To that end, we prefer a front that contains more efficient individuals over one that maintains diversity. OMOPSO and PAES perform better in this respect, and OMOPSO runs slightly faster than PAES.

We have, at this stage, carried out only qualitative evaluation of the performance of these metaheuristics on the experiment flow optimisation problem. Unary quantitative comparisons such as Success Counting and Inverted Generational Distance (see [9] for examples) are not possible as the true Pareto front of this real world problem is not known. Future work will include binary quantitative comparisons, such as Two Set Coverage and Two Set Hypervolume (examples also available in [9]) on the most promising metaheuristics.

Future work will also look at the effect of tailoring the metaheuristics by using different crossover and mutation operators. It is likely that the metaheuristics examined perform differently when using different operators, and that gains and losses in performance may not be uniform across all metaheuristics. We will also attempt to discover the effect of "tuning" the parameters of the metaheuristics and operators. Finally we will examine the suitability of the evolutionary optimisation approach to solving a problem that also includes hard constraints, such as the limits on the numbers of nutrients that can be dispensed simultaneously. We will investigate recent advances in constraint handling for multi-objective evolutionary algorithms, such as the ones described in [5].

This paper demonstrates that evolutionary multi-objective optimisation algorithms are an appropriate method of solving the experiment flow optimisation problem. We have also demonstrated that there are differences in the performance of these algorithms on the experiment flow optimisation problem, and that an aggressive approach is needed, even at the expense of diversity, in order to find highly efficient batches of trials.

8. ACKNOWLEDGEMENTS

I would like to thank Maria Liakata and Jem Rowland of University of Wales, Aberystwyth and Bill Langdon of the University of Essex for their constructive feedback on earlier versions. Thanks are also due to Maria Liakata for Figure 1 and to David Corney of University College London for helpful discussions instrumental in developing this work.

This work was supported by BBSRC grant BB/D00425X/1

9. REFERENCES

[1] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[2] J. J. Durillo, A. J. Nebro, F. Luna, B. Dorronsoro, and E. Alba. jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics. Technical Report ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos, December 2006.

[3] R. D. King, K. E. Whelan, F. M. Jones, P. G. K. Reiser, C. H. Bryant, S. H. Muggleton, D. B. Kell, and S. G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 15, January 2004.

[4] J. Knowles and D. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8:149–172, 2000.

[5] A. Kurpati, S. Azarm, and J. Wu. Constraint handling improvements for multiobjective genetic algorithms. *Structural and Multidisciplinary Optimization*, 23:204–213, April 2002.

[6] A. J. Nebro, F. Luna, E. Alba, A. Beham, and B. Dorronsoro. AbYSS: Adapting Scatter Search for Multiobjective Optimization. Technical Report ITI-2006-2, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos, 2006.

[7] S. O'Hagan, W. B. Dunn, M. Brown, J. Knowles, and D. B. Kell. Closed-loop, multiobjective optimization of analytical instrumentation: gas chromatography/time-of-flight mass spectrometry of the metabolomes of human serum and of yeast fermentations. *Anal. Chem.*, 77:290–303, January 2005.

[8] K. Sastry, D. D. Johnson, A. L. Thompson, D. E. Goldberg, T. J. Martinez, J. Leiding, and J. Owens. Multiobjective genetic algorithms for multiscaling excited state direct dynamics in photochemistry. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1745–1752, New York, NY, USA, 2006. ACM Press.

[9] M. R. Sierra and C. A. C. Coello. Improving pso-based multi-objective optimization using crowding, mutation and epsilon-dominance. In *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005*, pages 505–519. Springer, 2005.

[10] L. N. Soldatova, A. Clare, A. Sparkes, and R. D. King. An ontology for a robot scientist. *Bioinformatics*, 22:e464–e471, 2006.

[11] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Swiss Federal Institute of Technology (ETH), Zurich, ETH Zentrum, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.