Population extremal optimisation for discrete multi-objective optimisation problems

Randall, M.; Lewis, A.

Published in:
Information Sciences

DOI:
10.1016/j.ins.2016.06.013

Population Extremal Optimisation for Discrete Multi-Objective Optimisation Problems

M. Randall*

Faculty of Business, Bond University, Queensland, Australia

A. Lewis*

Institute for Integrated and Intelligent Systems, Griffith University, Queensland, Australia

Abstract

The power to solve intractable optimisation problems is often found through population based evolutionary methods. These include, but are not limited to, genetic algorithms, particle swarm optimisation, differential evolution and ant colony optimisation. While showing much promise as an effective optimiser, extremal optimisation uses only a single solution in its canonical form – and there are no standard population mechanics. In this paper, two population models for extremal optimisation are proposed and applied to a multi-objective version of the generalised assignment problem. These models use novel intervention/interaction strategies as well as collective memory in order to allow individual population members to work together. Additionally, a general non-dominated local search algorithm is developed and tested. Overall, the results show that improved attainment surfaces can be produced using population based interactions over not using them. The new EO approach is also shown to be highly competitive with an implementation

*Corresponding author

Email addresses: mrandall@bond.edu.au (M. Randall), a.lewis@griffith.edu.au (A. Lewis)

of NSGA-II.

Keywords: Extremal Optimisation, Population, Multi-objective Optimisation Problems, Local Search

---

1. Introduction

Over the past few decades, numerous population-based optimisation algorithms have been proposed. From this, a few techniques have emerged as being standard algorithms that researchers and practitioners routinely apply. The more notable of these are genetic algorithms (GAs) [1], particle swarm optimisation (PSO) [2], differential evolution (DE) [3] and ant colony optimisation (ACO) [4]. These have been successfully applied to discrete, continuous, single and multi-objective problems.

One class of algorithms that has not gained as much attention is extremal optimisation (EO) [5, 6]. Unlike the techniques previously mentioned, its standard formulation relies on the manipulation of a single solution only. Only a relatively small number of papers have proposed population models. This paper links the notions of population mechanisms to the need to consistently produce improved solutions to multi-objective problems. It develops appropriate population models and mechanisms to give EO a computational advantage over its canonical form. In particular, the contributions it makes are the development of two alternative EO population mechanisms suitable for multi-objective problems and a generic non-dominated local search technique.

The remainder of the paper is organised as follows. Section 2 gives a brief overview of multi-objective optimisation and the problem that will be used in this study. This will be the first time that this version of the bi-objective Generalised

Assignment Problem (GAP) will be used for benchmark instances. Section 3 provides a thumbnail sketch of extremal optimisation, outlining its canonical mechanics, as well as giving a summary of its applications to multi-objective problems and the competing population models. Section 4 describes a population approach suitable for a range of continuous and discrete multi-objective problems as well as a non-dominated local search technique. Section 5 discusses issues including transition operators, constraint handling and non-dominated local search while the experimental work is carried out in Section 6. The latter includes a comparison to NSGA-II. Finally conclusions and ideas for further investigation are given in Section 7.

2. Multi-objective Optimisation and the Test Problem

For a thorough discussion of multi-objective optimisation and its relation to evolutionary meta-heuristic algorithms, the reader is referred to works such as Coello Coello [7] and Deb [8].

Multi-objective optimisation, by definition, involves the simultaneous optimisation of more than one objective. While each objective requires minimisation or maximisation, this discussion (without loss of generality), will consider objectives that need to be minimised. The minimisation of objectives is commonly expressed according to Equation 1.

$$\text{Minimise: } \overrightarrow{f}(\overrightarrow{x}) = f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_m(\overrightarrow{x}) \tag{1}$$

Where:

$\overrightarrow{f}$ is the set of objective functions,

$\overrightarrow{x}$ is the solution vector and

m is the number of objective functions.

Rather than a single optimal solution being produced, a number of trade-off solutions are generated instead. The key question becomes which solutions should become part of this trade-off (attainment) surface? This is achieved by the process of Pareto dominance ranking. A decision vector $\overrightarrow{x_1}$ dominates $\overrightarrow{x_2}$ (denoted $\overrightarrow{x_1} \prec \overrightarrow{x_2}$) if it is not worse in any objective, and better on one or more objectives. Over a number of iterations, the optimisation algorithm will produce a set of decision vectors which dominate other solutions that have been generated, i.e., solutions will be generated that dominate previously non-dominated solutions. The aim is to produce an attainment surface that is as close as possible to the optimal surface (also known as the *Pareto Front*).

2.1. The Bi-Objective Generalised Assignment Problem - the Test Problem

In comparison to continuous problems, there have been relatively few studies that have considered combinatorial multi-objective problems. Additionally, there are few benchmark problems on which researchers may test their discrete techniques. This paper presents a rarely unused bi-objective version of the generalised assignment problem that is applied to the benchmark instances of Chu and Beasley [9]. In its canonical form, the generalised assignment problem is a problem in which jobs are assigned to agents for these agents to perform subject to capacity constraints. Each job may be performed by one agent only. The aim is to minimise the total cost of assigning the jobs to the set of agents. By its very general nature, it has been applied to such areas as resource scheduling, designing communication networks and vehicle routing.

In a first of its kind, Prakash, Sharma and Singh [10] present a non evolutionary algorithm for solving a bi-objective version of the generalised assignment problem. While maintaining the standard objective function, the second function is non-linear and minimises the maximum resource usage of the agents. This paper, unlike many other Pareto approaches, prioritises the two objectives. Initially, it does this by using a constructive heuristics to optimise objective one. This then becomes the first non-dominated solution. In generating the second non-dominated solution, a constraint is added that restricts the value of the second objective to being smaller than the value of the second objective obtained for the first non-dominated solution. This procedure is repeated for the desired number of non-dominated solutions. Unfortunately, only a numerical example of the optimisation procedure was given, rather than a full suite of results on benchmark problems. It does, however, provide a model of the bi-objective GAP that is used in this paper.

The model of the problem is given in Equations 2-6.

$$\text{Minimise} \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \tag{2}$$

$$\text{Minimise} \max \left\{ \sum_{i=1}^{n} t_{ij} x_{ij} \quad \forall j, 1 \leq j \leq m \right\} \tag{3}$$

s.t.

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \forall i, 1 \leq i \leq n \tag{4}$$

$$\sum_{i=1}^{n} t_{ij} x_{ij} \leq b_j \quad \forall j, 1 \leq j \leq m \tag{5}$$

$$x_{ij} \in \{0,1\} \quad \forall i, 1 \leq i \leq n, \quad \forall j, 1 \leq j \leq m \tag{6}$$

Where:

c_{ij} is the cost of assigning job i to agent j,

t_{ij} is the resource required by agent j to perform job i,

x_{ij} is 1 if job i is assigned to agent j, 0 otherwise,

b_j is the capacity of agent j,

n is the number of jobs and

m is the number of agents.

3. Extremal Optimisation

Extremal optimisation is a nature-inspired strategy for solving combinatorial and continuous optimisation problems [11]. As it is relatively unexplored compared to other techniques such as ant colony optimisation (ACO) [12], genetic algorithms (GAs) [13] and particle swarm optimisation (PSO) [14], there exists wide scope to test and extend its capabilities. Unlike its counterparts, the canonical algorithm manipulates a single solution vector rather than a population of solutions, though population versions have been implemented (e.g., Chen, Lu and Yan [15], Chen, Xie and Chen [16] and Randall et al. [17]). The remainder of this section describes the mechanics of the canonical algorithm, the changes necessary for use on multi-objective problems, as well as the population models that have been used with it.

3.1. Algorithmic Mechanics

EO is loosely based on the principles of the Bak-Sneppen model [18] in which, on occasion, a series of very small changes to the state of a system will lead to a very large change. This is referred to as *punctuated equilibrium*. The model, that has its basis in biology, consists of a number of species arranged in a ring so that each species has two neighbours as well as a fitness value. At each step of the process, the least fit species and its two neighbours are removed and replaced by new species. Over time, the system evolves so that no species' fitness is below a critical value. Boettcher and Percus [5, 11, 19] adapt this notion to the development of EO. The concept of "solution components" replaces "species" as the unit of change, and the ring structure is removed in place of a solution vector. Additionally, neighbouring values do not change when the primary value does.

Solution components are the building blocks of the solution, an example being a job being assigned to a particular agent for the canonical version of the test problem, the generalised assignment problem. In the original version of the EO algorithm, at each iteration, the component whose fitness was worst would be replaced by another solution component value generated at random. However, this choice of always selecting the worst component to modify leads to too greedy a search, and consequently its performance was poor. Like other meta-heuristic algorithms, an element of randomness (in the form of the probabilistic selection of solution components to change) was introduced. This became known as $\tau-$EO [19]. Components are ranked from worst (rank 1) to best (rank n). The parameter τ and the rank controls the selection probability for each solution component [5]. Selection is biased towards the lower ranks (i.e., the worst components). This is achieved using Equation 7.

$$P_i \propto i^{-\tau} \qquad 1 \leq i \leq n \tag{7}$$

Where:

i is the rank of the component,

P_i is the probability ($P_i \in [0,1]$ when normalised) that component i is chosen and

n is the number of components.

Values of τ close to, or equal to, zero produce an undirected random search strategy. Conversely, allowing $\tau = \infty$ gives the original EO algorithm. How does the ranking work practically however? For the generalised assignment problem, the overall total cost of assigning jobs to agents needs to be minimised. Therefore, each component is ordered on its individual assignment cost (of job to agent), with the highest cost assigned a rank of 1 and the lowest a rank of n. This means that the job assignments that are most costly and degrade the solution the most will most likely be chosen.

Algorithm 1 shows the mechanics of a single $\tau-$EO iteration. Note that this canonical algorithm assumes that there is only one objective function that is separable (i.e., each solution component's fitness can be evaluated independently). This is an $O(n \log n)$ algorithm given that Quicksort [20] is used to help calculate the rankings. Typical values for τ are normally between 1 and 2 [11]. Note that the calculation of the P vector values only needs to be done once – before the EO process begins. For the sake of computational ease of implementation, P is normalised.

Algorithm 1 A single $\tau-$EO iteration.

Rank the solution components in x from worst to best according to their adverse effect on the objective function $f(x)$

$i =$ Select a component from x using roulette wheel selection on P

Assign x_i a random (legal) value

$c =$ Evaluate $f(x)$

if c is better than *Best* **then**

  $Best = c$

end if

end

As is evident from the above discussion, one of the structural properties of canonical EO is also one of its potential weaknesses – the requirement that the problem under consideration be a separable one. This means that each solution component can be evaluated separately. There are a wide variety of problems for which, this is not the case. However, some works are beginning to appear that address this issue. Two such are briefly described here. Meneses, Randall and Lewis [21] solve a problem in which the objectives are evaluated through a black box function. Their novel approach incorporating the notion of introducing a pheromone structure from the domain of ACO. It is used to track the relative performance of components via the solutions that they have been incorporated into. Non-dominated solutions (from their components) update the pheromone levels. Therefore components with low levels of pheromone have high values in the P vector. The other approach by Chen, Lu and Yang [22] is used to solve continuous functions in the which the objectives are non-separable. At each iteration of their algorithm, for each solution, n mutations occur on the n solution components

(i.e., one mutation per solution component). The n new solutions are then ranked using dominance ranking, and the solution with rank 0 (i.e., the solution component being the best change, so, by implication the worst solution component to begin with) is always chosen to be changed.

In terms of the bi-objective GAP, and multi-objective problems more generally, ranking the solution components becomes more difficult than single objective problems. One way to do this is on dominance count. That is, those components that have low fitness (i.e., tend not to dominate other components on a quality measure) will have a lower rank than those that don't. This is the approach used to address the non-separable nature of the problem considered in this paper.

Measuring the quality of components is problem-specific. For the GAP formulation in this paper, this is done using the competing component cost (c) and the resource (t) required when a job is assigned to a particular agent. In both instances lower values equate to a better quality component.

3.2. Existing Populations Models and Multi-objective Problems

As previously mentioned, there has been little work done on applying EO to multi-objective problems – particularly discrete/combinatorial ones as well as having population models developed for it. This section summarises the more notable contributions in both regards.

Randall [23] and Randall, Hendtlass and Lewis [17] propose a population extension in which the members interact at either definite intervals [23] or according to a probability function [17]. The latter triggers an interaction event if the objective costs of the members of a particular generation vary widely. Specifically, a probability $p = \left(1 - \frac{cost(best)}{cost(worst)}\right)^{1/l}$ is generated. In it $cost(best)$ is the best cost received in the current population while $cost(worst)$ is the opposite. l is the num-

ber of iterations since an interaction has occurred. This means as time passes, an interaction becomes more likely. As EO does not converge, a large difference between $cost(best)$ and $cost(worst)$ would tend to suggest that the search had lost cohesion. In order to attempt to correct this, the interaction will eliminate the weakest members of the population and replace them by randomly generated members. In the case of Randall [23], this was exactly three members – the worst solution and the two closest in terms of solution similarity, rather than cost similarity. Randall et al. [17] change this model slightly by replacing one of the solutions with the best solution found to date. Given the improvements in this paper, it is unsurprising that it out-performed its predecessor. For both papers, single objective combinatorial problems were used. The generalised assignment problem was studied in both papers; Randall et al. [17] also looked at the bin packing problem and a constrained variant of the hub location problem.

Chen, Lu and Yang [22] propose a population-based EO in which solutions are considered in isolation from one another, rather than as an interactive population like genetic algorithms, ant colony optimisation or particle swarm optimisation. It is also a non-tau version of EO which is applied to the non-separable continuous Zitzler-Deb-Thiele (ZDT) test functions. This is because, at each iteration of the algorithm, for each solution, n mutations occur on the n solution components (i.e., one mutation per solution component). The n new solutions are then ranked using dominance ranking, and the solution with rank 0 (i.e., the solution component being the best change, so by implication the worst solution component to begin with) is always chosen to be changed. If this were a τ version of the algorithm, components with ranks other than 0 would have the opportunity to be changed. If it is the case that more than one component is ranked at 0, a diversity preser-

vation mechanism is invoked. This means that the resultant solution that crowds members in the external archive the least is chosen.

Meneses, Randall and Lewis [21] have recently applied EO to the problem of designing Radio Frequency IDentification (RFID) antennas. This is a discrete, non-separable NP hard problem whose aims are to produce antennas that have high efficiencies and low resonant frequencies (i.e., high read ranges). These values are obtained through a black box evaluation suite. As the problem is to design an antenna as a series of links on a grid, a modified knapsack approach was used (i.e., a component simply incorporated a link, or not, into the design). As EO requires information about each component, a pheromone scheme (adapted from the ACO literature) was used. Furthermore, each time a non-dominated solution is found, the pheromone level on each component value is increased. The algorithm was able to produce very efficient designs with very few function evaluations.

4. Two Population Models

A population of objects (be they solutions to optimisation problems, or people in a country) is more than a collection of non-interacting individuals. It is a concept that embraces the idea of community – the individuals interact with one another, and in turn are affected by one another. This is in the cause of advancing, or optimising, the society in which they live. Modifying a meta-heuristic search algorithm, such as extremal optimisation, to include a population component can be difficult. This is because, as is evident from the previous section, such approaches can be seen as just "bolt-ons" that do not necessarily contribute to the goal of allowing solutions to influence one another in order to produce better solutions.

In this section, two alternative population schemes are defined. As will become evident, they may also be easily combined.

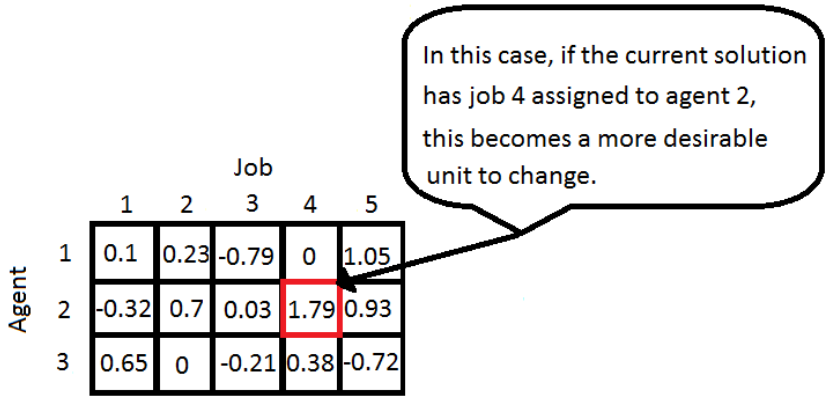
4.1. Collective Memory Scheme

To define an appropriate population strategy, one should take advantage of the key factors of the meta-heuristic under consideration. For EO, without doubt, this is the unique way in which solution components are selected to be changed at each iteration. Presently, each solution component is ranked against the other components in the same solution. This naturally could be extended out across all solutions in a population and achieved by altering the EO generated probability by adding or subtracting a "social" factor. This social factor, which is along the lines of PSO, represents a record of the "badness" of the component and value combination for all solutions from previous generations. The greater the value of the badness, the more this will add to the original probability (as generated through Equation 7). These values are stored in a communal memory matrix that is indexed by component and value. For the GAP, this corresponds to an m by n matrix. Figure 1 gives an example of a collective memory matrix.

To update the memory, a small factor is either subtracted or added to the relevant element of the matrix. This is given by $\frac{s}{N}$ where s is social mediation factor, $0 < s \leq 1$ and N is the size of the population size.

Dividing by N corresponds to exerting relevant influence of members over one another. That is, the members of a small population will have a greater influence over one another than a large population. Working out when to apply the update, and if it should be positive or negative, can be given by a rule base. What needs to be taken into consideration is whether the solution has changed its feasibility status and if there was solution quality improvement or not. The latter can be

Figure 1: An example of the collective memory matrix s for a small problem. Note that a value of 0 signifies that that combination of job and agent has not yet been incorporated into a solution.



determined by working out whether the new solution dominates the old one or vice-versa). The rules, formally stated, are given in Table 1.

Table 1: The rule base for deciding whether the social factor should be applied. Note the last two entries assume that the feasibility status is the same. "Better" and "Worse" refer to better and worse hypervolume values respectively. For all other cases, the factor becomes 0.

Old solution	New solution	Factor
infeasible	feasible	$-\frac{s}{N}$
feasible	infeasible	$\frac{s}{N}$
better	worse	$\frac{s}{N}$
worse	better	$-\frac{s}{N}$

4.2. Interaction/Intervention Population Scheme

This strategy is quite different in approach to the first, but also complimentary to it. In it, a combination of intervention and interaction strategies are used in which community based operations to the members of the current population and the archive are used. To help define the strategy, the following questions are posed:

1. What is the exact nature of the population interactions? Are there alternatives?
2. When would and should such interactions occur? Too often, and it may distort the main optimisation effort as well as consuming excess computational resources. Too few, and it may have no significant effect at all.

In the case of Question 1, the default/control position is to allow individual solutions to evolve completely separately. This was the case for Chen et al [22]. There are a large number of potential actions that could be taken from an interaction. Apart from the control strategy, the following will be implemented.

1. *Canonical Extensions* – The first method is simply an extension of EO's canonical transition mechanism. After an iteration of the population, eliminate the probabilistic worst solution (as determined by a dominance ranking on all the solutions in the current population), and replace it by a randomly created solution. As every iteration may be too frequent, this could be done at specific intervals, such as every thousand iterations. This would then become a parameter of the process. Another extension is to replace all the non-dominant solutions in the population.
2. *Use solution material from the Archive within a genetic algorithm* – The archive can be used for more than just a store of good solutions. These solutions potentially contain valuable material that can be used to form new

solutions to put back in the population. When a population interaction occurs, all of the dominated solutions in the population are replaced by solutions produced by a GA iteration. The idea is motivated by the success of memetic algorithms [24] that often embed attributes of multiple search techniques into the one overall strategy. Crossover and mutation parameters will be given in the next section. Additionally, the number of iterations over which the genetic component is run is also a parameter. Initial experimentation revealed that a value of around 200 proved to be very effective. While solution quality did increase as the number of iterations increased, the runtime started to become very large. The algorithm is given in Algorithm 2

Algorithm 2 The algorithm for the genetic component interaction.

for 1 to the maximum number of genetic iterations **do**

Perform two point crossover using random parents from the Archive (two per new solution) and replace the entire population.

Perform mutation on the new solutions.

Determine which of the new solutions are feasible, and which aren't.

Perform local search on each of the new feasible solutions.

Update the Archive with any improved solutions found.

end for

end

Question 2 is a particularly interesting one, as population-based meta-heuristics often have some form of community interaction as their main means of optimising solutions, at each iteration of the algorithm. As examples, GAs and DE use crossover variants to combine portions of different chromosome/solutions to-

gether. Where optimisation operations are exclusively performed on individuals, like extremal optimisation, adding the dimension of a population means that one has both individual and community operations. Some possible examples of the latter are described above. It becomes evident that community operations needn't be done at each iteration. Two broad alternatives exist. First, perform interactions every fixed number of iterations. This was the approach used by Randall et al. [17]. The second is a little more sophisticated. Rather than performing iterations blindly, only perform them when the need arises. In the case of multi-objective optimisation, and particularly the archive-based Pareto approach adopted in this paper, this need is triggered when there are few, if any, solutions that enter the archive at an iteration. This suggests that the search has stagnated and some form of community based intervention is necessary. A probability function that is inversely proportional to the number of members entering the archive at an iteration (that is, the smaller the number, the larger the probability) is a means of achieving this. The probability function used in this study is given by Equation 8, though other functions are possible.

$$P = \frac{A - N}{A} \times k \tag{8}$$

Where:

P is the probability,

A is the archive size,

N is the number of solutions entering the archive at a particular iteration and

k is a scaling factor, a parameter of the process.

5. Subsidiary Issues

Beyond population mechanics, there are a number of other issues that need to be considered. These are: use of transition operators; constraint handling and local search. Each of these is addressed separately below.

5.1. Transition Operations

In order to select a component for EO to change at each iteration in a multi-objective context, a more complex approach than single objective optimisation is needed. For the latter case, the current value of each component[1] is computed – and then these are ranked from worst to best. The worst components have a greater probability of being changed. With more than one objective, this is more difficult to judge. One way to do this is to use dominance ranking on the components themselves, rather than on full solutions. The dominance ranking can then be mapped directly onto an appropriate EO ranking. In terms of the actual transition, the agent of a chosen job (by EO's canonical mechanics) is changed to a random agent.

5.2. Constraint Handling

As the GAP has constraints, infeasible solutions will inevitably be generated in the course of an EO run. The approach adopted in this paper follows that of Randall [23] and Randall et al. [17] in allowing feasible and infeasible solutions in the system, but subjecting the latter to a process known as *partial feasibility restoration*. It attempts to identify portions of the solution that contribute to the infeasibility the most (in a very greedy way) and change these to values that will

[1] assuming a separable objective function

reduce infeasibility. Obviously, this may still leave an infeasible solution at the end (hence the term "partial"). EO then simply selects its probabilistic worst component in the next iteration on each component's contribution to the infeasibility, rather than its cost. By this method, EO efficiently travels through infeasible space toward feasible space.

5.3. Non-dominated Local Search

To improve solutions that are generated by meta-heuristic search algorithms, it is customary for greedy local search algorithms to be applied. Usually, once a feasible solution has been generated, a series of local transitions are carried out, each transition only being accepted if it improves the solution. Regardless of the transition operators that are used for a particular problem, a general algorithm for multi-objective optimisation is given in Algorithm 3. The main difference between this and other local search algorithms is the acceptance mechanism – only if a new transition creates a solution that dominates the previous solution will it be accepted.

In terms of the local search transition operators for the GAP, two are used, as they have been found to be successful in the past [17, 23] (albeit for the single objective version of the problem). "Move" moves an item from one group to another. The job and agent are chosen such that the (negative) change in the objective function is the greatest. This is a variable length search stopping when an improving move cannot be found. "Swap" works in a similar way except that at each iteration, two items are chosen such that their swap will lead to the most improvement in the objective function.

Algorithm 3 Algorithm for multi-objective local search applied to a single solution.

```
previous_best = Evaluate the objectives of the incoming solution.
while improving solutions are still being received do
  for each possible transition on this solution do
    Try the transition and evaluate its objectives.
    if feasible and it dominates previous_best then
      Accept this as the new solution.
      previous_best = the objective values of the new solution.
    else
      Reverse this transition.
    end if
  end for
end while
```

6. Computational Experiments, Results and Analysis

Section 4 presented numerous novel concepts that can be implemented to produce improved attainment surfaces over and above canonical EO as well as other recognised solvers. This section explains the environment in which the experiments are run, the instances of the test problem used, the design of the experiments, the actual results and the analyses of these.

6.1. Test Environment and Problem Instances

The computing platform used to perform the experiments is a 3GHz Pentium 4 based personal computer. The experimental programs are coded in the C language and compiled with `gcc`.

Each problem instance will be run across ten random seeds to give statistically valid results. A run is terminated if the maximum number of iterations has been reached. A value of 1.4 is used for EO's only parameter, τ, as it has been found to be a good value for similar problems [17, 23].

The large-sized B, C and D problem instances of Chu and Beasley [9] are used. These range in size from five agents to 200 jobs. There are six problem instances for each of the B, C and D categories. Table 2 gives the characteristics of each problem, including the hypervolume reference points and size of the instance[2]. It must be noted that the reference points have been derived through experimentation, as this is the first time this bi-objective formulation has been applied to any GAP instances.

[2]This is given in the name itself, e.g., B5-100 indicates that there are 5 agents and 100 jobs.

6.2. Experimental Design and Results

Given that there are three conceptual developments (i.e., non-dominated local search, the population models, and timing and nature of interactions) and appropriate population size parameters that need to be tested, a large amount of experimental work could be carried out. The experiments were performed over two phases and are detailed below.

6.2.1. Phase 1 – Evaluating the Conceptual Developments on a subset of the Test Problem Instances

This phase tested whether turning local search on gives better results than not using it. In addition, this effect was examined in context with the population/external archive size. The sizes used were $\{20, 50, 100, 200\}$ which gives

Table 2: The description of the problem instances.

Instance	Hypervolume reference	Instance	Hypervolume reference
B5-100	(3500,250)	C5-200	(6000,500)
B10-100	(5000,150)	C10-200	(6500,250)
B20-100	(4000,80)	C20-200	(8000,150)
B5-200	(8000,500)	D5-100	(10000,900)
B10-200	(6500,250)	D10-100	(9000,500)
B20-200	(8000,150)	D20-100	(12000,250)
C5-100	(5000,300)	D5-200	(15500,1750)
C10-100	(3500,150)	D10-200	(16000,900)
C20-100	(3500,65)	D20-200	(16000,500)

eight combinations per instance. These were evaluated across a maximum of 100,000 objective function evaluations per run. As this was an exploratory phase, and as a result of there being potentially a lot of analysis, only a small subset of the problem instances were used: specifically, B20-100, C20-100 and D20-100. This is in accordance with the principles of hierarchical parameter tuning [25].

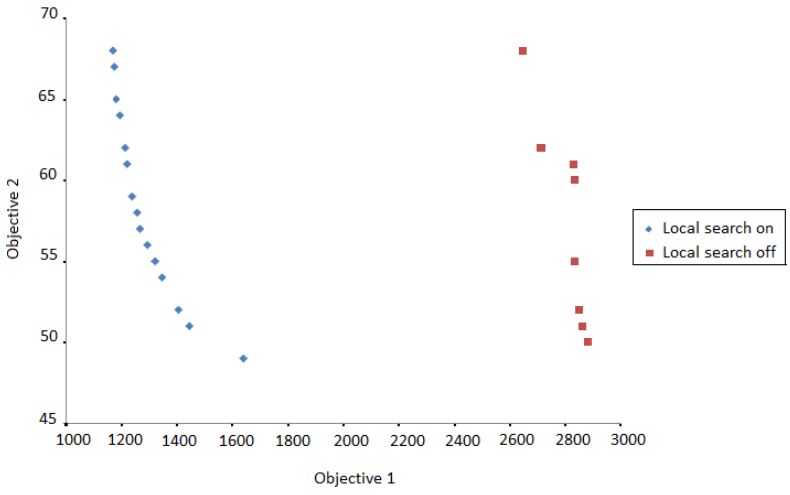
The analysis of Phase 1 fell into three main areas: determining whether multi-objective oriented local search should be used and the best population/archive size; and the most appropriate degree of the social factor. Varying these parameters will affect the quality of solutions. The non-parametric statistical test, Kruskal-Wallis, was used, as the data are highly non-normally distributed. A significance level of $\alpha = 0.05$ is used throughout.

In terms of local search, the results are very emphatic. It outperformed the non-local search version, indicating that a computational advantage was gained given the same amount of resources. The difference between the local search enabled and non-local search versions was statistically significant. Figure 2 shows the two attainment surfaces for local search being switched on and off. Thus, the multi-objective oriented local search will be used by default throughout the rest of the paper.

Now that local search has been established as a necessity, the remaining discussion will emphasise the results that have used local search. For population, a size of 200 members was best. A significant difference was detected and the ordering became (according to the Kruskal-Wallis ranks) $p200 \prec p400 \prec p100 \prec p50 \prec p1$. Note that $p1$ signifies canonical EO.

The next part of this exploratory phase examined the effects of the two interaction models (i.e., the canonical extensions and the one that uses genetic algo-

Figure 2: The attainment surfaces for B20-100 when local search is turned on and when it is not. Both surfaces represent typical runs.



rithms) and collective memory as well as the timing of these (i.e., at fixed intervals or on a calculated needs basis). The parameters required by this model are as follows:

1. *interaction* – This is used to denote whether the replacement, genetic or no interaction, is used.
2. *intervention* – This is used to define whether the interaction will be done at fixed periods or based on need (as defined in Equation 8).
3. *period / k* – The actual value of these parameters.
4. s – The value of the social mediation factor, $0 \leq s \leq 1$.

The local search enabled EO served as the control. Additionally, a non-population version (i.e., a population of one) was also run. If given any of the

parameter settings, improved results could be generated, then these too will be considered as additions to EO. In terms of the parameter values, for the fixed number of evaluations before an interaction, the values of $\{5000, 10000, 20000\}$ were assessed. For the values of k, in the detected approach, scaling values of $\{0.2, 0.5, 0.8\}$ were used. Both sets of values were used with the two interaction models. These values were also used for s.

Tables 3-5 are comparison tables for each of the three test problems (for ease of comparison). Note, in these tables "Min", "Med", "Max" "Stdev" denote minimum, median, maximum and (sample) standard deviation respectively.

The analysis of these tables reveals that the population models certainly produce improved solutions over the non-population (standard EO) version of the algorithm. Distinguishing between the various approaches is a different matter; however, a pattern emerges which becomes more noticeable as the difficulty of the problem instances increases. It is evident that while merely having a population produces very competitive solutions, the enhancements, nearly universally, produce better solutions. This is most apparent for the replacement and genetic strategies for C10-200 and D10-200. Various statistical analyses, using the Kruskal-Wallis technique, confirmed that there was a significant difference between the different EO approaches for each of the test problems. For B10-200, the replacement strategy with a probability of 0.8 worked best, while for the other two problems, the genetic option, with the same probability setting, proved superior. These findings are mirrored when analyses are performed to determine which interaction and intervention strategies are best overall. The differences between B10-200 and the other two may be explained by the nature of the test problems themselves. The C and D versions of these problems progressively have tighter

constraints.

6.2.2. Phase 2 – Final Results

The intent of the final set of experiments was to see what the best versions of the EO population models were capable of, given an extended run length. Additionally, for the sake of comparison, a stand-alone genetic algorithm (based on the one used for the interaction strategy) was run as well. The new number of total function evaluations was 500,000 to allow the meta-heuristics greater scope for exploration. Table 6 shows the EO based results. EO 1 refers to the replacement option with interaction probability of 0.8, while EO 2 is the genetic option with the same probability. By way of comparison, Table 7 show the results of GA and an NSGA-II [26] implementation. Both were tuned in the following ways. Note that the GA was tuned within the EO framework and, in common with the tuning of the EO variants carried out in phase 1 of this investigation, tuning of both NSGA-II and the GA was carried out using the same, reduced subset of test functions used in phase 1.

- For the GA, only two parameters were required to be examined, the mutation rate, and the generational replacement rate. Values of $\{0.1, 0.2, 0.3\}$ and $\{0.2, 0.5, 0.7\}$, respectively, were trialled on the test instances. From this it was determined (via Kruskal-Wallis tests) that a low mutation rate coupled with a low rate of generational replacement worked best. Thus a value set of (0.1,0.2) will be used throughout.

- NSGA-II was allowed the same total function evaluations, and was tested for crossover and mutation rates of $\{90\%, 50\%, 10\%\}$ and $\{10\%, 20\%, 30\%\}$, respectively. By inspection of median hypervolumes attained, it was con-

firmed that a crossover rate of 90% and a mutation rate of 10% gave generally superior performance across the test instances.

Upon visual inspection of Table 6, looking at the bolded figures, it is easy to see that the genetic algorithm based variant of EO (EO 2) is able to find many best solutions (in terms of the descriptive measures) over and above EO 1. It is believed this may be due to the introduction of the diverse material that the genetic intervention can provide. Having said that though, it is clear from Table 7 that the GA component alone is not solely responsible for producing the good solutions. EO 2 produces quite a few more best results on each of the three measures. This shows that EO adds optimisation power to the search process. In terms of the comparison to NSGA-II, it may be noted that it achieved the best results for the test cases with small numbers of agents. The reason for these results is unclear. NSGA-II was included as a point of comparison for the modified EO algorithm, and investigation of its specific, unexpected behaviour was beyond the scope of this paper. It remain, obviously, a question worthy of further investigation. It may also be noted that on a number of test cases NSGA-II was unable to find many feasible solutions, making the statistical analysis of its performance of diminished value, and in one case was unable to find any feasible solutions at all. EO (particularly EO 2), on the other hand, was always able to find feasible solutions while being very competitive with NSGA-II in the number of best solutions it was able to produce. This is very much in evidence on the median measure.

7. Conclusions

Populations are integral to nearly all successful evolutionary optimisation strategies. The collective utility of solutions is harnessed in order to produce improved

solutions that could not be derived from isolated individuals. Techniques such as extremal optimisation, that do not have a population mechanism as standard, often require this in order to be able to find competitive solutions. However, it is not necessarily easy to find an effective strategy, and in some way must be linked to the characteristics of the problems one is studying. The contribution of this paper is the development of two alternative, but complimentary, population mechanisms to target discrete multi-objective optimisation problems, though these can be extended to continuous problems as well. An additional contribution and way to improve solutions was given by a generic non-dominated local search technique. Moreover, a new set of benchmark discrete multi-objective problems have also been introduced.

The evidence from the test problems strongly suggest that the combination of the population models are able to produce very good solutions. Given the same number of function evaluations, it was shown that these results were superior to those that the solver could achieve using a non-interacting population. In comparison to the GA implementation, the modified EO was very competitive, often producing better attainment surfaces. It was also quite competitive with the state-of-the-art NSGA-II algorithm, exceeding its performance on most of the larger problems.

In summary, the contributions of this paper are clear in terms of the advances in the structure of a class of evolutionary algorithms. It was intended to show that, by adding a population component to a single solution technique, Extremal Optimisation, an advantage to that technique can be gained. This was indeed achieved and came with no excessive computational costs. It may be noted that EO did not exceed the performance of comparable algorithms in all cases. However,

as is known in light of the no free lunch theorem [27], this is not the sole criterion of success. Indeed in our case, it was clearly demonstrated the NSGA-II algorithm completely failed in a number of cases. In contrast, the population EO was very robust and was able to produce solutions on each occasion. In addition, it was able to find quite a few improved solutions over NSGA-II.

There are a number of ways in which this work may be extended. The first is to test it on more discrete multi-objective problems, and to extend it to continuous problems as well. This paper only addressed the bi-objective GAP, as the motivation was to test the basic concepts and extensions to EO. With a better understanding of these now, the work will be extended to many objective problems. The collective memory model can also be extended so that this information can be used to eliminate solution components to rank and evaluate at each iteration. This idea belongs to the candidate set literature [28], and may offer improvements in computational efficiency.

One refinement of the proposed approach for population interactions deserves investigation in future work. In Algorithm 2, all dominated solutions were replaced with new solutions derived from the Archive through crossover operations. However, solutions close to the Pareto front may contain useful information, particularly in sparsely populated regions of the front. An approach that uses ranking of dominated solutions, similar to that used by NSGA-II (Non-Dominated Sorting Genetic Algorithm) [26], may be used to preserve "second-rank" solutions. These solutions may contribute to improved coverage of the Pareto optimal set, either via local search mechanisms or by normal evolution of solutions.

Acknowledgements

The authors would like to acknowledge the assistance of Mr Ethan Jackwitz in conducting computational experiments to generate some of the results reported in this paper.

References

[1] S. Sivanandam, S. Deepa, Introduction to genetic algorithms, Springer Science & Business Media, 2007.

[2] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm intelligence 1 (1) (2007) 33–57.

[3] S. Das, P. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31.

[4] M. Dorigo, T. Stützle, Ant colony optimization: overview and recent advances, in: Handbook of Metaheuristics, Springer, 2010, pp. 227–263.

[5] S. Boettcher, A. Percus, Nature's way of optimizing, Artificial Intelligence 119 (2000) 275 – 286.

[6] S. Boettcher, Evolutionary dynamics of extremal optimization, Journal of Computational Interdisciplinary Sciences 2 (2011) 71–78.

[7] C. Coello Coello, A comprehensive survey of evolutionary-based multiobjective optimization techniques, Knowledge and Information Systems 1 (1999) 269–308.

[8] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, Wiley, 2002.

[9] P. Chu, J. Beasley, A genetic algorithm for the generalised assignment problem, Computers and Operations Research 24 (1997) 17–23.

[10] S. Prakash, M. Sharma, A. Singh, Pareto optimal solutions for multi-objective generalized assignment problem, South African Journal of Industrial Engineering 21 (2010) 91–100.

[11] S. Boettcher, A. Percus, Extremal optimization: Methods derived from co-evolution, in: Proceedings of the Genetic and Evolutionary and Computation Conference, Moran Kaufmann, 1999, pp. 825–832.

[12] M. Dorigo, G. Di Caro, The ant colony optimization meta-heuristic, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, 1999, pp. 11–32.

[13] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, Reading MA, 1989.

[14] J. Kennedy, R. Eberhart, The particle swam: Social adaptation in social information-processing systems, in: New Ideas in Optimization, McGraw-Hill, Maidenhead, UK, England, 1999, pp. 379–387.

[15] Y. Chen, Y. Zhu, Y. Lu, Improved extremal optimization for the asymmetric traveling salesman problem, Physica A:Statistical Mechanics and its Applications 390 (2011) 4459–4465.

[16] J. Chen, Y. Xie, H. Chen, A population-based extremal optimization algorithm with knowledge-based mutation, in: Y. Tan, Y. Shi, C. Coello (Eds.), Advances in Swarm Intelligence, Vol. 8794 of Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 95–102.

[17] M. Randall, T. Hendtlass, A. Lewis, Extremal optimisation for assignment type problems, in: A. Lewis, S. Mostaghim, M. Randall (Eds.), Biologically-inspired Optimisation Methods: Parallel Algorithms, Systems and Applications, Vol. 210 of Studies in Computational Intelligence, Springer-Verlag, 2009, pp. 139–164.

[18] P. Bak, K. Sneppen, Punctuated equilibrium and criticality in a simple model of evolution, Physical Review Letters 71 (1993) 4083–4086.

[19] S. Boettcher, A. Percus, Extremal optimization: An evolutionary local search algorithm, in: H. Bhargava, N. Ye (Eds.), Computational Modeling and Problem Solving in the Networked World, Interfaces in Computer Science and Operations Research, Kluwer Academic Publishers, 2003, pp. 61–77.

[20] C. Hoare, Quicksort, The Computer Journal 5 (1) (1962) 10–16.

[21] P. Meneses, M. Randall, A. Lewis, A multi-objective extremal optimisation approach applied to RFID antenna design, in: EVOLVE - A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation II, Vol. 175 of Advances in Intelligent and Soft Computing, Springer, 2012, pp. 431–446.

[22] M. Chen, Y. Lu, G. Yang, Multiobjective optimization using population-based extremal optimization, Neural Computing and Applications 17 (2008) 101–109.

[23] M. Randall, Enhancements to extremal optimisation for generalised assignment, in: M. Randall, H. Abbass, J. Wiles (Eds.), The Third Australian Conference on Artificial Life, Vol. 4828 of Lecture Notes in Artificial Intelligence, Springer, Berlin, 2007, pp. 369–380.

[24] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimization: A literature review, Swarm and Evolutionary Computation 2 (2012) 1–14.

[25] M. Randall, Multiple local neighbourhood search for extremal optimisation, in: Proceedings of the Metaheuristics International Conference, 2013, pp. 278–287.

[26] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multi objective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2002) 182–197.

[27] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.

[28] M. Randall, J. Montgomery, Candidate set strategies for ant colony optimisation, in: Ant Algorithms, Springer, 2002, pp. 243–249.

Table 3: Phase 1 results for B10-200. These values represent hypervolumes from the reference points given in Table 1. 'Single' refers to canonical EO, whereas "Population' is the multi-solution EO without the intervention strategies. For both the 'Replace' and 'Genetic' strategy, the two intervention methods of fixed iterations (first three values) and the detected, k, values (last three values) are given. The following two tables also follow this structure.

Method	Parameter Value	Min	Med	Max	Stdev
Single		42398	53323.5	75473	10926.5
Population		82370	87188.5	93655	4070.3
Replace	5000	82709	87147.5	94370	4246.5
	10000	82705	87142.5	94401	4256.8
	20000	82383	87393.5	93646	4059.6
	0.2	82596	86731.5	93290	3805.9
	0.5	82585	86701	93340	3836.4
	0.8	82722	87455.5	93398	4200.3
Genetic	5000	82359	86645	93221	3843.5
	10000	82490	86474	93186	3784.8
	20000	82477	87292.5	93650	4067.3
	0.2	82625	87056.5	93414	3824.1
	0.5	82513	86803	93598	3921.2
	0.8	82511	86555	93331	3857.9
s	0.2	82396	87214	93527	4066.5
	0.5	82309	86733	93611	3927.8
	0.8	82496	87383	93496	3845

Table 4: Phase 1 results for C10-200. Refer to Table 3 for an explanation of the features of this table.

Method	Parameter Value	Min	Med	Max	Stdev
Single		17810	19937	22041	1657.9
Population		24210	26193	32453	2886.3
Replace	5000	24251	26388.5	32572	2969
	10000	24247	26386.5	32486	2954.5
	20000	24220	26164.5	32514	2908.6
	0.2	24273	26262	32435	2726.7
	0.5	24221	26332.5	32557	2970.5
	0.8	24278	26409	32572	2960.2
Genetic	5000	24310	26380.5	33749	3398.3
	10000	24207	26389	35693	3830.4
	20000	24250	26467.5	32549	2948.1
	0.2	24349	26439	32713	2969.2
	0.5	24363	26439	34002	3313.8
	0.8	24284	26547	34291	3413.1
s	0.2	24211	26170	32482	2898.3
	0.5	24203	26167.5	32510	2908.5
	0.8	24178	26209	32502	2928.9

Table 5: Phase 1 results for D10-200. Refer to Table 3 for an explanation of the features of this table.

Method	Parameter Value	Min	Med	Max	Stdev
Single		298558	399471.5	630066	113379.6
Population		728180	768416	799370	20129
Replace	5000	730596	771273	800623	19774
	10000	728222	769008	798901	20342.6
	20000	728014	768944.5	799035	20198.01
	0.2	727852	771754.5	798233	20404.8
	0.5	728794	771516	800193	20316.4
	0.8	728602	772024	799922	20546.8
Genetic	5000	739916	780529.5	803794	20207.8
	10000	731844	776910.5	801095	20492.3
	20000	739453	771466.5	805987	19540.7
	0.2	739655	781554.5	802961	20676.8
	0.5	741292	784941.5	803256	20935.3
	0.8	739740	783452	802376	20682.5
s	0.2	729651	768563.5	798842	19594.9
	0.5	730250	769179.5	799572	19602.2
	0.8	731745	769325	798898	19154.1

Table 6: The phase 2 results for the two EO variants. Note that boldface values indicate the best values received for Min, Med and Max across this Table and Table 7.

	EO 1				EO 2			
Instance	Min	Median	Max	Stdev	Min	Median	Max	Stdev
B5-100	94137	98755.5	101782	2604.664	99703	104198.5	106389	2183.395
B5-200	558043	574426	585949	9802.196	576989	611116.5	626569	14846.46
B10-100	219691	231192.5	239767	6936.484	**221593**	**236960.5**	242779	8138.798
B10-200	225704	233829	253460	9310.994	**226830**	236724	261579	11099.22
B20-100	82436	87248.5	93704	4056.425	**82986**	87314.5	**93817**	3908.311
B20-200	**543713**	**546863**	**553172**	3558.047	539465	545775.5	548753	2852.738
C5-100	2284701	2305554.5	2335446	16946.86	**2301985**	**2329676.5**	**2360071**	18388.13
C5-200	263206	267094	276418	4128.081	276651	283170.5	298379	7252.545
C10-100	124549	126771.5	129425	1590.653	**125387**	**128754.5**	133659	2469.478
C10-200	**198896**	**228806.5**	253757	18910.07	181564	218945.5	251521	24372.97
C20-100	24253	26207	32534	2888.421	24507	27699.5	34494	3376.83
C20-200	**270011**	**297982.5**	319445	13276.38	269458	295962.5	**322159**	14062.84
D5-100	943170	982318	1024922	23553.77	1057271	1148590	1207370	53501.97
D5-200	149808	203278	242945	31065.26	1345241	1478741	1653667	91433.95
D10-100	432591	446206	456438	6313.08	**582101**	633822	676206	34409.76
D10-200	196192	306434	357587	53252.44	667233	1010097	1131048	142059.7
D20-100	730039	769825.5	799511	19779.67	**762394**	**793975**	**842072**	24881.47
D20-200	153457	224614	278965	39558.01	239536	**390887.5**	536718	89206.98

Table 7: Results of running GA and NSGA-II on the test problems. Note that the 0 entries indicate that NSGA was unable to find feasible solutions. Boldface values indicate the best values received for Min, Med and Max across this Table and Table 6.

	GA				NSGA-II			
Instance	Min	Med	Max	Stdev				
	Min	Median	Max	Stdev	Min	Median	Max	Stdev
B5-100	93954	98531	104200	3514.394	**128610**	**213580**	**250370**	39559.16
B5-200	535706	544886.5	588003	19615.14	**824321**	**927861**	**1126303**	97090.71
B10-100	220996	229361	**247425**	9124.602	121946	172956.5	223440	29192.32
B10-200	222635	231336.5	264675	15115.53	177546	**254081.5**	**299482**	35459.58
B20-100	82774	**88308.5**	93742	3225.715	0	0	35882	16603.78
B20-200	530127	539094.5	543576	4080.025	94468	178642	252310	48815.12
C5-100	2268995	2316721	2357730	25540.32	523376	547381	612122	26280.13
C5-200	247359	271020	296170	14918.21	**1043319**	**1243416.5**	**1320131**	96378.92
C10-100	117530	125031.5	130728	4036.423	80022	101166.5	**153417**	20440.87
C10-200	170773	200876.5	233434	18324.82	160324	226043	**315263**	52373.9
C20-100	**24585**	**28733**	**34992**	3334.501	0	0	24145	8286.963
C20-200	264586	287466	308868	15391.34	0	71550	223956	93707.32
D5-100	1088684	1177462	1311221	79603.69	**1588447**	**2209439.5**	**2739628**	295771.2
D5-200	1100449	1273125	1657301	210915.8	**5994382**	**6394365**	**7182949**	348316.9
D10-100	146538	229148	258128	35870.21	0	**848387**	**1128841**	301992.7
D10-200	787807	911578	1060052	99895.54	**1623617**	**2042716.5**	**2900104**	413851.3
D20-100	747281	781562.5	840238	28562.2	0	0	0	0
D20-200	**310561**	353607.5	488267	57184.21	0	0	**1130795**	514752.3