Entropy-based Adaptive Range Parameter Selection for Evolutionary Algorithms

A. Aleti
Faculty of Information Technology
Monash University, Australia
aldeida.aleti@monash.edu

I. Moser
Faculty of ICT
Swinburne University of Technology, Australia
imoser@swin.edu.au

ABSTRACT
Evolutionary Algorithms are equipped with a range of adjustable parameters, such as crossover and mutation rates which significantly influence the performance of the algorithm. Practitioners usually do not have the knowledge and time to investigate the ideal parameter values before the optimisation process. Furthermore, different parameter values may be optimal for different problems, and even problem instances. In this work, we present a parameter control method which adjusts parameter values during the optimisation process using the algorithm’s performance as feedback. The approach is particularly effective with continuous parameter intervals, which are adapted dynamically. Successful parameter ranges are identified using an entropy-based clusterer, a method which outperforms state-of-the-art parameter control algorithms.

Categories and Subject Descriptors
I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—Heuristic methods

General Terms
Algorithms, Performance

Keywords
Evolutionary algorithms, optimisation, parameter control

1. INTRODUCTION
Evolutionary Algorithms (EAs) have successfully been employed to solve a range of hard optimisation problems [1, 2, 31, 16, 25, 29]. In recent years, it has been acknowledged that the success of these algorithms depends on the numerous parameters that these algorithms have [18, 13, 33, 6], which make the optimisation procedure flexible and efficient for any kind of problem, regardless of the search space difficulty.

Unfortunately, the settings of the parameter values are known to be problem-specific [72], often even specific to the problem instance at hand [5, 40, 39, 19], and greatly affect the performance of the algorithm [33, 6, 26, 14]. In cases where the number of parameters and their plausible value ranges are high, investigating all possible combinations of parameter values can itself be an attempt to solve a combinatorially complex problem [8, 16, 7, 34]. It has also been empirically and theoretically demonstrated that different parameter settings may be required for different optimisation stages of a problem instance [9, 5, 40, 39, 19, 11], and therefore should vary during the search process for best algorithm performance [5, 10, 39, 10].

To address the problem of parameterising Evolutionary Algorithms, adaptive parameter control [42, 15, 10, 11, 21, 22, 24, 27, 38, 44] emerged as a new line of thought. Instead of exploring parameter settings before using an EA, in adaptive parameter control parameters are adjusted during the optimisation process. More specifically, properties of an EA run (such as the quality of the solutions produced) are monitored and the change in the properties is used as a signal to change parameter values.

One of the difficult issues faced in adaptive parameter control methods is the choice for real-valued parameter assignments, such as the mutation rate and the crossover rate. Usually, adaptive parameter control methods [49, 20, 9, 22, 15, 13, 42, 8] discretise the choices for parameter assignments and sample parameter values from the preselected ranges. Insight from existing research [5] shows that initial discretisation can be problematic. As the ranges are fixed, they remain sub-optimal throughout the optimisation process.

Confounding this view, Adaptive Range Parameter Selection (ARPS) [4] was introduced, which uses adaptive value ranges that change dynamically during the optimisation process. ARPS outperforms state-of-the-art parameter control methods, such as Probability Matching (PM) [42], Adaptive Pursuit (AP) [42], Dynamic Multi-Armed Bandit (DMAB) [15] and Predictive Parameter Control (PPC) [8]. However, it was observed that high-performing ranges were sometimes absorbed into very large intervals, making it difficult
for the algorithm to re-establish small, promising areas within the range.

With the goal of handling real-valued parameter assignments effectively, we introduce a new adaptive parameter control approach which uses an entropy-based measure to discretise parameter value ranges. At every iteration, feedback from the search is used to cluster parameter values into successful and unsuccessful groups. The outcome from the clustering process is employed to cut the parameter range into two sub-ranges such that the class information entropy is minimised. Results from the experimental validation of the approach show that the Entropy-based Adaptive Range Parameter Selection (EARPS) outperforms state-of-the-art methods.

2. BACKGROUND

2.1 Evolutionary Algorithms

Generally speaking, Evolutionary Algorithms maintain a population of solutions that evolves by means of the mutation operator, the crossover operator, the selection procedure, and the replacement procedure. The objective is to search the solution space in order to optimise some quality function(s). The optimisation process starts with a set of solutions as initial population, which can be randomly generated, created by applying heuristic rules (e.g. greedy algorithm), or provided by an expert. After the initialisation, EA evolves the population using the crossover, mutation and selection operators. The crossover and mutation operators are applied according to predefined crossover and mutation rates. These operators are applied to specific solutions, which are selected according to the selection procedures. The new individuals (offspring) created by the genetic operators are added to the population. The replacement procedure selects the solutions that will survive in the next generation and removes as many individuals as required to maintain the prescribed population size.

Before using an EA for optimisation, a number of algorithm parameters have to be adjusted, such as the population size, offspring number, selection procedure, mutation rate and crossover rate. A comprehensive overview of the research in various parameters of Evolutionary Algorithms is presented by Eiben and Schütz [15].

2.2 State-of-the-art in Adaptive Parameter Control

Formally, given a set \( \{v_1, \ldots, v_n\} \) of \( n \) algorithm parameters, where each parameter \( v_i \) has \( \{v_{i1}, \ldots, v_{in}\} \) values that can be discrete numbers or intervals of continuous numbers, parameter control has the task of deriving the optimal next value \( v_{ij} \) to optimise the influence of \( v_i \) on the performance of the algorithm. As an example, when the mutation rate \( v_1 \) is dynamically adjusted by considering 4 intervals \( (m = 4) \), \( v_{12} \) stands for a mutation rate sampled from the second interval. In the discrete case of optimising the type of mutation operator \( v_2, v_{22} \) stands for the second operator.

The majority of the adaptive parameter control methods found in the literature [10, 11, 21, 22, 24, 27, 38, 44] belong to the class of probability matching techniques, in which the probability of applying a parameter value is proportional to the quality of that parameter value.

The earliest approaches [22, 21] tended to ‘lose’ value ranges if the feedback from the algorithm was not in their favour in the initial phases of the process. In later work, a minimum selection probability \( p_{min} \) is introduced [22], to ensure that under-performing parameter values did not disappear during the optimisation, in case they were beneficial in the later stages of the search.

These probability matching techniques [10, 11, 21, 22, 24, 27, 38, 44, 17] were based on the notion of selecting parameter values in proportion to their previous performances. One of the more recent and mature examples of probability matching is the work by Igel and Kreutz [22]. Their Equation 1 for calculating the selection probability for each parameter value incorporates the maintenance of a minimum probability of selection.

\[
p_t(v_{ij}) = p_{min} + (1 - mp_{min}) \frac{p_t(v_{ij})}{\sum_{r=1} p_t(v_{ir})} \quad (1)
\]

Probability matching has been criticised for the loose correlation between the reward allocations and the differences in performance with vastly superior values receiving a marginal increase in selection probability.

Adaptive Pursuit (AP) [42] was conceived as an attempt to address this issue, ensuring that clearly superior values are rewarded with a maximum probability of choice.

Even though every parameter value is selected from time to time, in practice the Adaptive Pursuit algorithm spends a number of iterations before responding to a change of best parameter value. To address this problem, the Dynamic Multi-Armed Bandit (DMAB) [15] completely recalculates the probabilities when a change in the effects distribution is detected by using a change detection test, in this case the statistical Page-Hinkley (PH) test. The PH test checks whether the quality of the parameter values has changed. When a change is detected, the algorithm is restarted. As a result, DMAB can quickly identify the new best parameter value without being slowed down by old information.

Adaptive Range Parameter Selection (ARPS) [4] is the first attempt at adapting real-valued parameter ranges. ARPS discretises continuous-valued parameters by partitioning its ranges into two equal intervals. At every iteration, the best-performing interval is subdivided by splitting it in the middle. At the same time, the worst-performing interval is merged with the worse-performing of its neighbours. The selection probabilities of the split intervals are initially the same, the merged interval maintains the neighbour’s probability of selection.

Refining the most successful parameter areas into narrower intervals, the probability of choosing particular values increases. Merging badly performing parameters decreases their probability of selection. An
analysis of the mutation/crossover ranges \cite{4} revealed that ARPS sometimes absorbs high-performing intervals into very large ranges as a result of short-term underperformance. The merged ranges are usually very large and it takes many iterations for an absorbed sub-range to re-establish itself. This behaviour almost certainly affects the performance of the search. Therefore, an improved approach to maintaining parameter value intervals is the subject of this investigation.

3. ENTROPY-BASED ADAPTIVE RANGE PARAMETER CONTROL

At every iteration of an EA run, the chosen parameter values are employed to create different solutions, which are evaluated using the fitness function(s). The output from the evaluation process provides valuable information for the adaptive parameter control, since it can be used to assess the effect of the parameter values on the performance of the optimisation algorithm. The feedback from the search is used to approximate the class information entropy \( E(v_{ij}) \), which is proportional to the average quality of the parameter values in each range. Let parameter value \( v_{ij} \), representing values between \([i, j]\) into two intervals: \( v_{i1} \) representing values in the range \([i, k]\) and \( v_{i2} \) denoting values in the range \([k, j]\). To obtain the minimal entropy, we calculate all entropy values arising from placing \( k \) between each of the adjacent pairs in the sequence \( c_1, c_2, c_1, c_2, \ldots \).

\[
E(v_{i1}) = -\frac{|c_1(v_{i1})|}{|c_2(v_{i1})|} \log_2 \left( \frac{|c_1(v_{i1})|}{|c_2(v_{i1})|} \right) - \frac{|c_2(v_{i1})|}{|c_2(v_{i1})|} \log_2 \left( \frac{|c_2(v_{i1})|}{|c_2(v_{i1})|} \right)
\]

Eq. 2 has to be repeated for each interval \( v_{ij} \). In each case, it calculates the number of ‘matching’ cluster members \( (c_1 \text{ for interval } v_{i1}) \) in one interval as a proportion of all samples located in that interval \( (c_1(v_{i1})) \) and subsequently the proportion of non-matching cluster members \( (c_2 \text{ for interval } v_{i1}) \) of all samples in that cluster.

The overall class information entropy when a range \( v_{ij} \) is partitioned at the cut-point \( k \) into two new ranges \( v_{i1} \) and \( v_{i2} \) is calculated as the weighted average of the class entropies of the sub-ranges. This constitutes the class information entropy and is computed as follows:

\[
\mathcal{E}(k|v_i) = \frac{|c_1(v_{i1})|}{|c_2(v_{i1})|} E(v_{i1}) + \frac{|c_2(v_{i1})|}{|c_2(v_{i1})|} E(v_{i2})
\]

The best cut-point \( k \) is the one that minimises the class information entropy \( \mathcal{E}(k|v_i) \).

The method employs the cut-point \( k \) to partition the parameter range into two sub-ranges. These two intervals are employed to select the parameter values for the next iteration. The selection of parameter values is proportional to the average quality of the parameter values in each range. Let \( p(v_{ij}) \) denote the selection probability of parameter range \( v_{ij} \) and \( q(v_{ij}) \) the combined quality of the members of the interval. The mean quality of the intervals is calculated according to Eq. \( ? \), regardless of cluster membership.

\[
q(v_{ij}) = \frac{1}{|c_2(v_{ij})|} \sum_{c_2 \in v_{ij}} q(c_2)
\]

The probability of sampling from a range \( p(v_{ij}) \) is derived from the combined quality of its members \( q(v_{ij}) \), normalised by the summed qualities of all intervals.

The main steps of EARPS are given in Algorithm 1. Each of the steps is explained in more details in the following sections.
4. EXPERIMENTS

Evolutionary Algorithms are not expected to deliver exact and repeatable results, but to provide good approximate solutions where exact approaches cannot be devised. Hence, results concerning the performance of approximate algorithms such as EAs, are usually reported as mean values over repeated trials. To obtain a fair comparison, the generally accepted approach is to allow the same number of function evaluations for each trial \([30]\). Therefore, for the current comparison, all algorithms trials were repeated 30 times for each optimisation scheme. These values were decided after running the algorithm once for every problem and choosing the value where the quality of the solutions seemed to not improve any more. Nevertheless there are indications that all algorithms still make small but steady improvements after these numbers of evaluations.

4.1 Benchmark Problems

According to Lin et al. [37], differences in performance among approximate algorithms are more likely to be detected statistically if all algorithmic approaches solve the same problem instances. Along this line, three problems were chosen: the generally accepted Royal Road Problem (RRP), the Quadratic Assignment Problem (QAP), and the multiobjective Quadratic Assignment Problem (mqAP) which were especially designed for testing EAs. The problems were chosen due to their dissimilarity, which enables a more informed judgement as to the portability of the approach when applied to an EA. Moreover, all four problems are NP-hard, which makes them difficult to be solved by an exact algorithm and justifies the use of EAs to optimise them.

4.1.1 Quadratic Assignment

The Generalised Quadratic Assignment Problem (GQAP) is one of the most difficult combinatorial optimisation problems. The aim is to assign \(M\) utilities to \(N\) locations with minimal cost. The candidate assignments are evaluated according to equation \([5]\)

\[
\sum_{i,j} B_{ik} \cdot u_{ik} + \sum_{i,j,k,l} C_{i,j,k,l} \cdot u_{ik} \cdot u_{jl}
\]

where

- \(B_{ik}\) is the cost of assigning utility \(i\) to location \(k\)
- \(C_{i,j,k,l}\) is the cost of the flow between neighbouring utilities (given utility \(i\) is assigned to location \(k\) and utility \(j\) is assigned to location \(l\))
- \(u_{ik}\) is 1 if utility \(i\) is assigned to location \(k\), 0 otherwise

GQAP allows for multiple assignments to the same location subject to the availability of space as described by equation \([6]\)

\[
\sum_{i=1}^{M} a_{ik} \cdot u_{ik} \leq S_k (k = 1...N)
\]

where

- \(a_{ik}\) is the space needed for utility \(i\) at location \(k\)
- \(S_k\) is the space available at location \(k\)
The GQAP is a well-known problem and instances of considerable difficulty have been made available as benchmarks. It is a single-objective problem with one objective function that lends itself as quality feedback for the performance assessment of the parameter values.

4.1.2 The Royal Road Problem

Mitchell, Forrest, and Holland \[23\] especially devised the Royal Road (RR) problem to demonstrate that there exist problems which are easier to solve using a Genetic Algorithm than a hill climber. The function of the form \( F : \{0, 1\}^l \rightarrow \mathbb{R} \) is used to define a search task in which one wants to locate strings that produce high fitness values. The string is composed of \( 2^k \) non-overlapping contiguous sections each of length \( b + g \), where \( b \) is known as the block and \( g \) is known as the gap. In the fitness calculation, only the bits in the block part are considered, whereas the gaps make no contribution.

Higher order schemata are formed from sets of the base level blocks, where the base level containing the initial blocks is level 0. The fitness calculation proceeds in two steps, the part calculation and the bonus calculation. The overall fitness assigned to the string is the sum of these two calculations.

The RR function is being used here as a benchmark to match the Genetic Algorithm, whose parameters are being optimised for the experimental results presented. It has also been used by Fialho, Schoenauer and Sebag \[15\], whose results are being used for the comparison.

4.1.3 Component Deployment Optimisation

One of the practical applications of Evolutionary Algorithms is the component deployment problem in embedded systems \[1\], relevant e.g. for the automotive industry. An existing hardware topology is used to run software components which form the basis of the increasingly sophisticated functionality of contemporary cars. The quality of the overall system depends on the choice of hardware unit to host a particular software component (Papadopoulos and Grante \[35\]).

The quality of the system is commonly measured in terms of non-functional attributes such as safety, reliability, performance and maintainability. We model the embedded system as a set of software components and a set of hardware hosts as listed below.

Let \( C = \{c_1, c_2, ..., c_n\} \), where \( n \in \mathbb{N} \), denote the set of software components. The parameters for the software architecture are as follows:

- Component communication frequency \( CF : C \times C \rightarrow \mathbb{R} \), where \( CF(c_i, c_j) = 0 \) if \( c_i = c_j \) or there is no communication between \( c_i \) and \( c_j \).
- Component event size \( ES : C \times C \rightarrow \mathbb{N} \), where \( ES(c_i, c_j) = 0 \) if \( c_i = c_j \) or there is no event occurring \( c_i \) and \( c_j \).

Let \( H = \{h_1, h_2, ..., h_m\} \), where \( m \in \mathbb{M} \), denote the set of hardware resources. The parameters for the hardware architecture are as follows:

- Network bandwidth \( NB : H \times H \rightarrow \mathbb{N} \), where \( NB(h_i, h_j) = 0 \) if \( h_i = h_j \) or there is no network connection between \( h_i \) and \( h_j \).
- Network reliability \( R : H \times H \rightarrow \mathbb{R} \).
- Network delay \( D : H \times H \rightarrow \mathbb{N} \), where \( D(h_i, h_j) = 0 \) if \( h_i = h_j \) or there is no network connection between \( h_i \) and \( h_j \).

The deployment problem is then defined as \( D = \{d \mid d : H \rightarrow C_{sub}\} \), where \( C_{sub} = \{c_2, c_3, ..., c_j\} \subset C \), and \( D \) is the set of all functions assigning hardware resources to components. Note that, since \( C \) and \( H \) are finite, \( D \) is also finite, and represents the set of all deployment candidates.

The deployment problem has many quality-related aspects and is therefore always modelled as a multi-objective problem. Data Transmission Reliability (DTR) follows the definition of Malek \[28\]. Reliability of the data transmission is a crucial quality attribute in a real-time embedded system, where important decisions are taken based on the data transmitted through the communication links. The Data Transmission Reliability (DTR) formulation we use has first been defined by Malek \[28\].

\[
DTR(d) = \sum_{i=1}^{n} \sum_{j=1}^{n} \text{freq}(c_i, c_j) \cdot \text{rel}(d(c_i), d(c_j)) \quad (7)
\]

In embedded systems with their constrained hardware resources, repeated transmissions between software components are discouraged. The Communication Overhead (CO) \[30\] objective attempts to enforce minimal data communication for a given set of components and system parameters. As a network- and deployment-dependent metric, the overall communication overhead of the system is used to quantify this aspect. It is first formalised by Medvidovic and Malek \[50\].

\[
CO(d) = \sum_{i=1}^{n} \sum_{j=1}^{n} \text{freq}(c_i, c_j) \cdot nd(d(c_i), d(c_j)) + \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\text{freq}(c_i, c_j) \cdot ds(c_i, c_j)}{\text{rel}(d(c_i), d(c_j))} \cdot \text{rel}(d(c_i), d(c_j)) \quad (8)
\]

The Scheduling Time (ST) objective is another objective to be minimised. It describes the average time for a hardware unit to complete the round-robin processing of all its assigned components. ST is given by

\[
ST(d) = \frac{1}{m} \sum_{j=1}^{m} \left( \sum_{i \in C_{sub}} \frac{m_i}{ps_j} \right) \quad . (9)
\]

4.2 Evolutionary Algorithm

The Royal Road problem is optimised using an EA with a representation using a binary string. The operators in use are bit-flip mutation and multipoint crossover.
GQAP as an assignment problem with constraints maps $N$ tasks to $N$ resources. Hence the solution representation is a simple array which describes the numbered locations and the values of the array represent the items. Multipoint crossover swaps the assignments between solutions. The mutation operator changes the assigned item of a single location. As we are solving the GQAP, singularity of item occurrence is not mandatory and the solutions are always valid after the operators have been applied.

The industrial problem of software deployment uses an EA with a specialised solution encoding which maps each hardware unit to one or more software components. The crossover operator combines the allocation lists of two solutions' locations (hardware units) and subsequently divides them again into two child solutions avoiding duplication. The mutation operator exchanges the host allocations of two randomly chosen components. The problem definition does not allow for duplication, and a repair operation follows the crossover/mutation move.

Also, the component deployment problem is multiobjective in nature and requires a more specialised approach. One of the state-of-the-art multiobjective EA implementations is NSGA-II, devised by Deb et al. \cite{Deb2002}.

### 4.3 Parameter Optimisation

The crossover and mutation rates are probably the most conspicuous control parameters to optimise in stochastic optimisation \cite{Aleti2011}. Hence, for the benefit of these experiments, only the crossover and mutation rates were varied. Based on preliminary exploration, a range of 0.01 – 0.7 was adopted for the mutation rate and the interval 0.01 – 1.0 was used for the crossover operator.

The parameter control method was invoked every time the optimising EA completed an iteration comprised of 150 function evaluations. The probabilities were calculated and new parameter values were assigned for the next iteration. This process was repeated 20 times. Consequently, each trial is allowed 3000 function evaluations. These settings apply to all benchmark optimisation trials regardless of the problem at hand.

In the case of GQAP and RR, the quality feedback for the parameter values is based on the fitness values returned by the objective function. The multiobjective nature of the deployment problem necessitates the combination of the fitness values from all three objective functions into a single unary measure of solution quality. The most conspicuous choice for this feedback is the hypervolume indicator, which has been described as the most reliable unary measure of solution quality in multiobjective space \cite{Zitzler2002}.

5. RESULTS

6. CONCLUSION

In this paper we have introduced an adaptive parameter control method to adjust variables needed for the parameterisation of Evolutionary Algorithms. According to our knowledge, the best-performing state-of-the-art approaches with the same functionality are AP, PM, DMAB and ARPS. The new approach clearly outperforms the other parameter control methods.

The adaptation of parameter value ranges increases the sampling accuracy, which can explain the superior performance of EARPC compared to other approaches. Furthermore, unlike other adaptive parameter control methods, EARPC does not have any hyperparameter that requires tuning before the optimisation process.

In the future, we will also investigate the use of information about possible correlations of the values of different parameters as well as the potential integration of such information into the meta-optimiser.

7. REFERENCES


Table 1: The means and standard deviations of fitness functions for the 30 runs of each problem instance using different parameter control schemes.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Instance</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PM</td>
<td>AP</td>
</tr>
<tr>
<td>QAP</td>
<td>THO150</td>
<td>9154000</td>
<td>9157000</td>
</tr>
<tr>
<td></td>
<td>WIL100</td>
<td>287100</td>
<td>286800</td>
</tr>
<tr>
<td></td>
<td>TAI256</td>
<td>47450000</td>
<td>47490000</td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RR</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>Instance</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PM</td>
<td>AP</td>
</tr>
<tr>
<td>QAP</td>
<td>THO150</td>
<td>36173.41</td>
<td>33358.02</td>
</tr>
<tr>
<td></td>
<td>WIL100</td>
<td>784.2973</td>
<td>634.0923</td>
</tr>
<tr>
<td></td>
<td>TAI256</td>
<td>261879.1</td>
<td>314044</td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: The Kolmogorov-Smirnov test values of fitness functions for the 30 runs of each problem instance using different parameter control schemes.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Instance</th>
<th>EARPS vs. PM</th>
<th>EARPS vs. AP</th>
<th>EARPS vs. DMAB</th>
<th>EARPS vs. ARPS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>d</td>
<td>p</td>
<td>d</td>
<td>d</td>
<td>p</td>
</tr>
<tr>
<td>QAP</td>
<td>0.8065</td>
<td>2.642e-10</td>
<td>0.8387</td>
<td>2.781e-11</td>
<td>0.9032</td>
</tr>
<tr>
<td></td>
<td>WIL100</td>
<td>3.508e-09</td>
<td>0.8065</td>
<td>3.508e-09</td>
<td>0.9032</td>
</tr>
<tr>
<td></td>
<td>TAI256</td>
<td>1.225e-10</td>
<td>0.9355</td>
<td>3.304e-12</td>
<td>0.9677</td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


