Evolutionary optimization: A big data perspective

Maumita Bhattacharya a,⁎, Rafiqul Islam a, Jemal Abawajy b

a School of Computing and Mathematics, Charles Sturt University, Australia
b School of Information Technology, Deakin University, Australia

1. Introduction

Many real life problems often involve non-linear, high-dimensional, complex search spaces that may be riddled with many local minima or maxima. These issues get incredibly more complicated in problems involving big data. Stochastic methods, such as evolutionary algorithms, perform a more exhaustive search of the model space as compared to deterministic search techniques. However, when dealing with even regular high dimensional problems, it gets difficult or too time consuming for all the model parameters or variables to converge within a given margin of error. In particular, as the dimensions or the number of model parameters increases, so does the required population size due to the exponential scale-up of the search space size. Large population sizes imply large number of cost function evaluations.

The research issue addressed here is that of ensuring constructive population diversity to achieve desirable EA performance for high dimensional optimization problems (Wang et al., 2011). The POPULATION_EA framework proposed in this research is aimed specifically at such problems. While the proposed method is not necessarily scalable to big data optimization problems in its current form, it is a move in the right direction.

1.1. Big data challenges relevant to evolutionary algorithms

Twenty first century has seen the explosion of data collection in many areas. The ease of acquisition of data as a result of technological revolution has made this feasible (Fan, 2013). Massive data and high dimensionality characterizes many contemporary problem domains such as biomedical sciences, engineering, finance and social sciences. This means machine learning problems handling such spatio-temporal problems may have to deal with tens of thousands of features extracted from documents, images and other objects. Typical key features of big data include very large samples as well as very high dimensionality. Sparseness...
of data is another feature associated with big data. Big data are often collected over different locations or platforms contributing to measurement errors and experimental variations.

In evolutionary optimization, high dimensionality contributes to deteriorated performance, increased computational cost, algorithmic instability, spurious correlations, and also incidental noise accumulations. The next section discusses the issues related to high dimensionally and evolutionary algorithms, existing remedial approaches as well as their shortcomings.

1.2. High dimensional problem and EA

Some of the reasons why traditional evolutionary algorithms are not adequate for high dimensional problems are associated with the implicit assumptions about evolutionary algorithm.

First, the average absolute fitness of the population increases continually. The population then seems to gradually lose diversity, until the search converges to the global optimum or premature convergence to some false optimum in the search space.

Many researchers associate the loss of exploratory capability with the loss of diversity and try to increase the diversity of the population, only the individuals with competitively high fitness are tend to be allowed to survive. New “explorer” individuals sampling fairly different regions in the search space normally have low fitness, until sufficient local exploration i.e. exploitation reveals their beneficial characteristics (Crépinsek et al., 2013). Even the individuals with competitive fitness, because of their sparse distribution in distant regions may disappear as a result of insufficient sampling or genetic drift. This problem is order of magnitude higher in case of high dimensional problems due to the sheer volume of the search space.

Second, EA’s response to the characteristics of the search space where problems with enormous (due to high dimensionality) and highly multimodal search spaces are involved.

Many difficult search problems (Bhattacharya, 2007a,b; Bhattacharya, 2008) are made difficult because their search spaces are enormous (due to high dimensionality) and highly multimodal. Many real-world problems fall in this category; for example, even regular engineering design problems may have over 100 design parameters. Here, we have included multimodality along with high dimensionality as in many real world problems, these properties may not occur in isolation.

1.2.1. Enhanced EAs and high dimensional problems

High dimensionality introduces complexities in the search space rendering most common evolutionary algorithms ineffective. There are evidences that even unimodal functions such as the generalized Rosenbrock may show multimodal tendencies in very high dimensional cases. In case of multimodal functions, the number of local optima may increase exponentially with increase in the number of dimensions. The search space and the number of local optima may become too large for EAs to realistically use a population that is large enough to be representative of the actual search space (see Fig. 1).

So far scalability of an evolutionary algorithm is concerned the most crucial task is how deal with the complex search space resulted from high dimensionality. A number of enhanced EAs have been introduced that attempts at manipulating the search space, not necessarily aiming at scalability issues though. Some of the major enhanced or improved variants on traditional EA (as our proposed solution is based on genetic algorithm (GA) we will cite examples of GA variants mainly) are as follows. (1) Fitness sharing and crowding: Crowding (seminal work by De Jong (1975)) and deterministic crowding (Yannibelli and Amandi, 2012; Mahfoud, 1992) avoids redundancy by replacing similar individuals in the population by different individuals. Bycontrolling the number of individuals in localized search spaces or niches, (Yannibelli and Amandi, 2012) attempted to control the average fitness of the population. Genotypic or phenotypic diversity is maintained in fitness sharing by preventing individuals to gather around high-fitness peaks. Such techniques may be influenced by the properties of the search space (Hu et al., 2005). (2) Multi-population and search space division: The multiple subpopulation approach often aims at exploiting each subpopulation by controlling migration. The search space division approach used by Tsutsui et al. (1998) also aimed at working on genetically different populations. In general, both multipopulation and search space division methods focused on population diversity (Yao et al., 2010; Rucinski et al., 2010). Another diversity-focused approach was SCGA (Li et al., 2002) which used similarity-based species protection. (3) Special learning-based approaches: Using a Differential Evolution (DE) implementation, (Wang et al., 2011) addressed scalability by an enhanced learning-based approach. (4) Cooperative coevolution: Cooperative coevolution (Li and Yao, 2009; Yang et al., 2008) is perhaps one of the most popular approaches specifically addressing the issue of high dimensionality in EAs. We shall discuss this approach in more details in the next subsection. (5) Other approaches: Jiang et al. (2011) applied intelligent management. An Artificial Bee Colony (ABC) implementation in (Akay and Karaboga, 2012) applied a constraint handling technique into the selection step of ABC to favour the feasible regions of the search space. Lozano et al. (2011) presented some recent works on scalability of EAs and other related metaheuristics for continuous optimization problems.

It is common knowledge that EAs do not usually fare very well in terms of scalability and unfortunately majority of the techniques discussed above may require unrealistically large population or large number of subpopulations to be effective while dealing with the “curse of dimensionality”. Historically, it may be noted that Koza et al. (1999) used population sizes of 350,000 or larger in order to attain satisfactory results. However, impractically large population size naturally comes with impractically high computational cost and also increasing the population size does not necessarily improve the performance of an EA. Hence, any technique requiring large population size or large number of subpopulations does not provide realistic solution for scalability in case of high dimensional problems.

1.2.2. High dimensionality specific EAs

One of the early attempts at handling high dimensionality is was by Krishnakumar (1989). He proposed a micro-genetic algorithm that uses micro populations to locate promising areas of the
search space. As the small populations are unable to maintain diversity over a number of generations, restarting was introduced whenever population diversity was lost. Restarting also helped in preventing premature convergence. Bretz et al. (2008) introduced a novel Differential Evolution for high dimensional real-parameter optimization. In addition to self-adaptive control parameters, their method uses dynamic population size reduction.

One of the most optimistically researched and conceptually straightforward approaches to handle the “curse of dimensionality” in EA is perhaps the cooperative coevolution technique. Originally, the method was introduced in (Potter et al., 1994) with a genetic algorithm implementation. The idea is to decompose a high dimensional problem into a number of manageable, low dimensional problems for the purpose of solving the main problem. García et al. (2010) used coevolution by decoupling the dimensions of a problem to form subpopulations. Li and Yao (2009) proposed a variant of cooperative coevolution.

In its popular form, the cooperative coevolution technique involves the following stages:

- **Decomposition of the main problem**: Decompose the object vector into a number of manageable, low dimensional subcomponents.
- **Solve the subcomponents**: Optimize the subcomponents by evolving them using an EA.
- **Cooperative combination of the subcomponents**: Combine all the subcomponents.

Cooperative coevolution in its various forms has been quite successful for separable problems and the idea of dealing with lower dimensional subcomponents clearly justifies its application to high dimensional problems. However, due to the very nature of the technique, its application to non-separable problems is not straightforward. Cooperative coevolution does not take into account the inter-dependability of the variables and hence, application of the technique is problem dependent. Several researches have attempted to overcome this constraint of cooperative coevolution with varied degree of success. Yang et al. (2008) proposes a coevolution approach that is targeted at non-separable problems.

The problem decomposition and coevolution in their Differential Evolution implementation use random grouping scheme and adaptive weighting. Despite some success, coevolution-based approaches are yet to overcome many issues to be considered robust and scalable with reasonable computational cost.

Cooperative coevolution by decoupling the dimensions of a problem into a number of manageable, low dimensional problems for the purpose of solving the main problem. Garcia et al. (2010) used coevolution by decoupling the dimensions of a problem to form subpopulations. Li and Yao (2009) proposed a variant of cooperative coevolution.

2. **POPULATION_EA: the proposed framework**

The POPULATION_EA is a hierarchical multipopulation evolutionary algorithm that uses enhanced genetic operations; the algorithm is designed to specifically target large scale/high dimensional problems. The algorithm is close in concept to island model evolutionary algorithm.

The proposed POPULATION_EA model uses a synergistic approach by combining the benefits of the complex population structure and the specialized operator (see Bhattacharya (2006)). Cooperative coevolution approach that is targeted at non-separable problems.

The problem decomposition and coevolution in their Differential Evolution implementation use random grouping scheme and adaptive weighting. Despite some success, coevolution-based approaches are yet to overcome many issues to be considered robust and scalable with reasonable computational cost.

The rest of the paper is organized as follows: The proposed POPULATION_EA framework is detailed in Section 2. Simulation details are presented in Section 3. Section 4 presents the discussion and analysis of simulation results. Section 5 presents the details of application of the proposed algorithm to a real world problem of financial portfolio management. Finally, concluding remarks are presented in Section 6.

2.1. **Algorithm structure**

POPULATION_EA is the main algorithm which in two different levels, employs COUNTER_NICHING_EA in the main population, while maintaining a set of hierarchical, co-evolving populations with less fit individuals.

The POPULATION_EA algorithm is described in Fig. 2. The main functional blocks in the algorithm are as follows:

- **Informed mutation operation**: The algorithm borrows the concept of informed mutation operation from COUNTER_NICHING_EA (see Appendix A). This is achieved by invoking GRID_NICHING (see Appendix B) and INFORMED_OP (see Appendix C) in sequence. The description of this block is given in Section 2.2. As before the purpose is to introduce and maintain constructive diversity in the population. This operation is restricted to the main population only. It may be noted that this single population of restricted size is adequate to efficiently sample the enormous search space associated with high dimensional problems. Hence, a set of coevolving
subpopulations are maintained alongside the main population as mentioned below.

- **Hierarchical sub-populations**: As the EA search progresses in the main population, the lesser fit individuals are moved to a set of hierarchical subpopulations. These subpopulations maintain populations within specified fitness ranges and are evolved as per traditional genetic algorithm. Thus the less fit individuals get a chance to survive to maturity and move back to the main population. The subpopulations are filled to size limit by random sampling. Description of this block is given in Section 2.3.

### 2.2. Exploration with informed operator

The parent population of POPULATION_EA model evolves the population with COUNTER_NICHING_EA technique as described in Bhattacharya (2006). To make this paper self standing we include a brief description of the mechanism here. As described in Bhattacharya (2006), the evolutionary mechanism first extracts information about the population landscape before deciding on introduction of diversity through informed mutation. The aim is to identify locally converging regions or donor regions in the landscape that could spare less fit individuals those could be replaced by more fit individuals. This algorithm uses the pseudo-niching information obtained from grid-niching to guide the genetic operators as above. INFORMED_OP operates on selected regions or communities only. Regular mutation and recombination is performed as usual on the entire population.

Remarks: The COUNTER_NICHING_EA technique is implemented only in the main population. Each one of the sub-populations is evolved using canonical genetic algorithm only. Also, for simplicity the main population is evolved with COUNTER_NICHING_EA a predefined fixed number of times before switching to the migration phase.

### 2.3. Exploitation, avoidance of genetic drift and others with hierarchical population structure

While the COUNTER_NICHING_EA guides EA search based on the information on current population, the hierarchical population scheme allow the apparently under-performing individuals to grow and expands coverage of the search space by sampling in multiple subpopulations. Although the COUNTER_NICHING_EA tries to confront crowding around fitter individuals, selection pressure will eventually force the less-fit individuals in other parts of the decision space to die. In the multi-tier population scheme, a set of sub-populations exist alongside the main population. Individuals are moved into these sub-populations according to their fitness level (see Fig. 3(b)). These co-evolving subpopulations hold individuals within specific fitness ranges and thus prevent unfair competition among individuals belonging to different fitness levels (Hu et al., 2005).

Each sub-population in the multi-tier co-evolving sub-population hierarchy can accommodate individuals within a specified range of fitness. The complete range of possible fitness is computed by the sum of the ranges of the sub-population fitness. An admission fitness threshold is designated to control admission to each sub-population level. Qualified candidates move across the sub-populations synchronously or asynchronously. An escape threshold determines whether or not a candidate should move to a higher fitness level sub-population. Also, there is a lower bound of the fitness threshold for each level. This is same as the escape threshold of the next lower-fitness level sub-population. However, movements of candidates are one-directional (from lower fitness level to higher fitness level) except for the members of the main population (see Fig. 3(a) and (b)). Migration is not restricted to the next higher fitness level and is strictly determined by the fitness of a candidate. Less fit individuals from the main population are either copied or shifted to an appropriate lower fitness level sub-population based on certain spatial or fitness information. Copying instead of complete migration from the main population, allows the candidates to survive in their original environment, and facilitates gain through recombination (Hu et al., 2005). Subpopulations are filled to the population size limit by random sampling.

### 2.4. Relationship among subpopulations: an analysis

In this section we present a simple analysis to determine the relationship among the subpopulations of POPULATION_EA, a multi-population evolutionary algorithm. We base our analysis on Schmitt et al. (1998).
For simplicity of analysis the following assumptions have been made:

- There are only two subpopulations.
- Only an elementary single-point crossover is used.
- Candidates use binary representation.

Let us consider a finite collection $C$ of candidate solution and a function $f : C \rightarrow \mathbb{R}$. The task of the optimization algorithm is to find a candidate solution (or simply a candidate) $c \in C$ so that $f(c)$ is optimal. Let $P$ be the set of all possible populations of size $s$: where, $s \in N$. $P$ is the set of all possible states when the population size $s$ is fixed. $l$ is the length of the binary strings representing the
candidates. So, every population \( p \in P \) can be represented as a bit vector of length \( L = l \cdot s \).

Again, let \( p = (c_1, c_2, ..., c_i) \in P \), \( c_i \in C \), \( 1 \leq i \leq s \) and

\[
\text{mean}(p) = \frac{1}{s} \sum_{i=1}^{n} c_i.
\]

The elementary crossover operation is explained as follows:

Let us consider two candidates \( c_i \) and \( c_j \) where \( 1 \leq i < j \leq n \) and \( c_i = (a_1, ..., a_l) \) and \( c_j = (b_1, ..., b_l) \) for \( k \) \((1 \leq k \leq l)\) to be a crossover point. Then, an elementary crossover operation on \( p \) involves replacing \( c_i \) and \( c_j \) in \( p \) by the offspring \( c_i' \) and \( c_j' \) respectively.

Here, \( c_i' = (b_1, ..., b_k, a_{k+1}, ..., a_l) \) and \( c_j' = (a_1, ..., a_k, b_{k+1}, ..., b_l) \).

Now, let us consider two subpopulations \( p \in P \) and \( q \in P \). If \( \text{mean}(p) = \text{mean}(q) \), then it can be proven that to transform \( p \) into \( q \) (or vis a vis) at most \( l \cdot \lceil s/2 \rceil \) elementary crossover operations will be needed.

**Proof.** Assuming \( l = 1 \) and \( \text{mean}(p) = \text{mean}(q) \), the number of indices \( i \) \((1 \leq i \leq s)\) at which \( p \) and \( q \) differ must be even.

An appropriately chosen elementary crossover operation will be able to correct two of these positions \((1 \leq i, j \leq s)\) simultaneously while generating \( p \) from \( q \). So, in this simple case at most \( s/2 \) such elementary crossover operations will be needed. On the other hand, if \( l > 1 \), then \( l \cdot \lceil s/2 \rceil \) elementary crossover operations will be needed.

### 2.5. Why POPULATION_EA should work for high dimensional problems

As has been explained in Section 1.1, even individuals with competitive fitness, are lost because of their sparse distribution in distant regions may disappear as a result of insufficient sampling or genetic drift. This is a major problem in case of high dimensional problems. This problem is taken care of by allowing less fit individuals to mature in the hierarchical subpopulations.

An EA population with a manageable (to accommodate all operations) population size may be a poor representative of the actual search space as the true search space for high dimensional problems are enormous in size. This problem is addressed in POPULATION-EA by evolving multiple subpopulations (includes resampling to introduce new members), thus attempting to cover larger area of the search space.

### 3. Simulations

As is the standard practice in the evolutionary computation research community, we have tested the performance of the proposed POPULATION_EA algorithm using a set of commonly used benchmark test functions.

It may be noted that POPULATION-EA has been designed for high dimensional, but not for very high dimensional (number of dimensions \( \geq 1000 \)) problems. Also, the algorithm’s performance in a wide range of problem dimensions \((20 \leq n \leq \text{problem dimensions} \leq 100)\) is also of interest. So, first we evaluate POPULATION_EA’s effectiveness for low dimensional problems \((20 \text{ dimensions in our experiments})\). Further, the algorithm is tested on relatively higher dimensional test cases \((50 \text{ and } 100 \text{ dimensions in our experiments})\). Performance of POPULATION_EA has been compared with that of COUNTER_NICHING_EA [Bhattacharya, 2006]. Some of the simulation results have also been compared with the following techniques as reported by Ursem (2002). Further details on these algorithms are as described in Bhattacharya (2006).

- The “standard EA” (SEA).
- The self organized criticality EA (SOCEA).
- The cellular EA (CEA).
- The diversity guided EA (DGEA).

#### 3.1. Test functions

The test suit for our experiments comprises the following benchmark test functions: Ackley’s Path Function \( f_{ack}(x) \), Griewank’s Function \( f_{gri}(x) \), Rastrigin’s Function \( f_{rtg}(x) \), Generalized Rosenbrock’s function \( f_{ros}(x) \), Axis parallel Hyper-Ellipsoidal Function or Weighted Sphere Model \( f_{sp}(x) \), Schwefel Function 1.2 \( f_{sch}(x) \) and a rotated Rastrigin Function \( f_{rot}(x) \). All the benchmark marks used are minimization problems. The descriptions and characteristics of the functions are as given in Table 1.

#### 3.2. Experiment setup

In our simulations we have used the following set up: real-valued encoding for the candidate solutions in the population; parameters \( N \) \((\text{Total population size}) = 250\), \( p_m \) \((\text{mutation probability}) = 0.05\) and \( p_r \) \((\text{recombination probability}) = 0.9\).

As mentioned previously, three variants of the above problems were considered: 20 dimensional, 50 dimensional and 100 dimensional. Reported results were averaged over 30 independent runs. The maximum number of generations in each run are 500, 1000 and 2000 for the 20, 50 and 100 dimensional problem variants respectively, as against 1000, 2500 and 5000 generations in Ursem (2002) for the same set of test cases.

The results reported in Tables 2 and 3 were obtained by running the simulation for fixed number of generations i.e. the maximum number of allowable generations set for the simulation runs.

### 4. Results and discussions

In this section, we present the empirical results obtained by the POPULATION_EA algorithm when applied to the seven test problems described in Table 1 with dimensions 20, 50 and 100.

#### 4.1. General performance of POPULATION_EA

Table 2 presents the error values, \((f(x) - f(x)^*)\) where, \(f(x)^*\) is the optimum. Each column corresponds to a test function. The error values have been presented for the three dimensions of the problems considered, namely 20, 50 and 100. As each test problem was simulated over 30 independent runs, we have recorded results from each run and sorted the results in ascending order. Table 2 presents results from the representative runs: 1st (Best), 7th, 15th (Median), 22nd and 30th (Worst), Mean and Standard Deviation (Std). The main performance measures used are the following:

- **A Performance**: Mean performance or average of the best-fitneess function found at the end of each run. (shown as “Mean” in Table 2).
- **SD Performance**: Standard deviation performance. (Shown as “SD.” in Table 2).
- **B Performance**: Best of the fitness values averaged as mean performance. (Shown as “Best” in Table 2).
The interesting observation here is that POPULATION_GA tends to have rather steady performance across different dimension ranges and also across the various simulation runs. This is an indicator of robustness of performance of the algorithm. POPULATION_EA performs relatively poorly in case of the high epistasis rotated Rastrigin Function $f_{reg}(x)$ (marked in **bold** in Table 2). This can be explained by no free lunch theorem as no single technique can realistically perform well across all test cases.

### Table 1

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
<th>Global Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{stk}(x) = 20 + e^{-20\exp\left(-0.2 \cdot \sum_{i=1}^{n} \cos(2\pi x_i)\right)}$</td>
<td>Multimodal</td>
<td>$f_{stk}(x = 0) = 0$</td>
</tr>
<tr>
<td>$f_{ge}(x) = \frac{1}{2000} \sum_{i=1}^{n} (x_i - 100)^2\cos \left(\frac{x_i - 100}{\sqrt{i}}\right)$</td>
<td>Multimodal, medium epistasis</td>
<td>$f_{ge}(x = 0) = 0$</td>
</tr>
<tr>
<td>$f_{rg}(x) = \frac{n}{10} \left(\sum_{i=1}^{n} x_i - 10 \cos(2\pi x_i) + 10\right)$</td>
<td>Multimodal, no epistasis</td>
<td>$f_{rg}(x = 0) = 0$</td>
</tr>
<tr>
<td>$f_{re}(x) = \sum_{i=1}^{n} (x_i - 10 \cos(2\pi x_i) + 10)$</td>
<td>Unimodal, high epistasis</td>
<td>$f_{re}(x = 1) = 0$</td>
</tr>
<tr>
<td>$f_{re}(x) = \sum_{i=1}^{n} (x_i - 10 \cos(2\pi x_i) + 10)$</td>
<td>Unimodal</td>
<td>$f_{re}(x = 0) = 0$</td>
</tr>
<tr>
<td>$f_{reh}(x) = \sum_{i=1}^{n} (x_i - 10 \cos(2\pi x_i) + 10)$</td>
<td>Unimodal</td>
<td>$f_{reh}(x = 0) = 0$</td>
</tr>
<tr>
<td>$f_{reg}(x) = 10M + \frac{\sum_{i=1}^{n} (x_i^2 - 10 \cos(2\pi x_i))}{2\pi}$</td>
<td>Multimodal</td>
<td>$f_{reg}(x = 0) = 0$</td>
</tr>
</tbody>
</table>

### Table 2

Error values achieved on the test functions with simulation runs for POPULATION_EA. Dimensions of each function considered are 20, 50 and 100. The segment marked in **bold** shows that the performance of POPULATION_EA is rather poor in case of the high epistasis rotated Rastrigin function.

<table>
<thead>
<tr>
<th></th>
<th>$f_{stk}(x)$</th>
<th>$f_{ge}(x)$</th>
<th>$f_{rg}(x)$</th>
<th>$f_{re}(x)$</th>
<th>$f_{re}(x)$</th>
<th>$f_{reh}(x)$</th>
<th>$f_{reg}(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st (Best)</td>
<td>0.00E – 61</td>
<td>4.0E – 62</td>
<td>1.01E – 61</td>
<td>0.5E – 60</td>
<td>1.01E – 60</td>
<td>1.05E – 50</td>
<td>1.01E – 5</td>
</tr>
<tr>
<td>7th</td>
<td>1.00E – 61</td>
<td>4.01E – 62</td>
<td>1.911E – 61</td>
<td>1.05E – 60</td>
<td>1.01E – 61</td>
<td>1.1E – 50</td>
<td>1.59E – 5</td>
</tr>
<tr>
<td>15th (Median)</td>
<td>1.01E – 61</td>
<td>4.9E – 61</td>
<td>1.95E – 60</td>
<td>1.1E – 60</td>
<td>1.1E – 60</td>
<td>1.1E – 50</td>
<td>2.01E – 5</td>
</tr>
<tr>
<td>22nd</td>
<td>1.90E – 61</td>
<td>5.01E – 62</td>
<td>2.00E – 61</td>
<td>1.91E – 60</td>
<td>1.96E – 60</td>
<td>2.01E – 50</td>
<td>2.90E – 5</td>
</tr>
<tr>
<td>20th (Worst)</td>
<td>3.01E – 62</td>
<td>8.11E – 62</td>
<td>3.00E – 61</td>
<td>2.0E – 60</td>
<td>2.0E – 60</td>
<td>2.29E – 50</td>
<td>4.0E – 5</td>
</tr>
<tr>
<td>Mean</td>
<td>1.1E – 61</td>
<td>5.01E – 62</td>
<td>1.99E – 61</td>
<td>1.51E – 60</td>
<td>1.29E – 60</td>
<td>1.29E – 50</td>
<td>3.0E – 5</td>
</tr>
<tr>
<td>50D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st (Best)</td>
<td>0.59E – 40</td>
<td>4.73E – 40</td>
<td>1.1E – 40</td>
<td>1.11E – 40</td>
<td>1.1E – 40</td>
<td>1.1E – 40</td>
<td>1.03E – 30</td>
</tr>
<tr>
<td>7th</td>
<td>0.60E – 40</td>
<td>4.81E – 40</td>
<td>1.1E – 40</td>
<td>1.1E – 40</td>
<td>1.1E – 40</td>
<td>1.1E – 40</td>
<td>1.04E – 30</td>
</tr>
<tr>
<td>15th (Median)</td>
<td>0.61E – 40</td>
<td>4.81E – 40</td>
<td>1.1E – 40</td>
<td>1.1E – 40</td>
<td>1.1E – 40</td>
<td>1.1E – 40</td>
<td>1.04E – 30</td>
</tr>
<tr>
<td>22nd</td>
<td>0.99E – 40</td>
<td>5.1E – 40</td>
<td>1.3E – 39</td>
<td>1.2E – 40</td>
<td>1.2E – 40</td>
<td>1.2E – 40</td>
<td>1.1E – 39</td>
</tr>
<tr>
<td>30th (Worst)</td>
<td>0.99E – 40</td>
<td>5.1E – 40</td>
<td>1.3E – 39</td>
<td>1.2E – 40</td>
<td>1.2E – 40</td>
<td>1.2E – 40</td>
<td>1.1E – 39</td>
</tr>
<tr>
<td>Mean</td>
<td>0.65E – 40</td>
<td>4.82E – 40</td>
<td>1.17E – 40</td>
<td>1.1E – 40</td>
<td>1.1E – 40</td>
<td>1.1E – 40</td>
<td>1.05E – 30</td>
</tr>
<tr>
<td>Std.</td>
<td>1.51E – 41</td>
<td>1.28E – 41</td>
<td>7.89E – 42</td>
<td>3.13E – 42</td>
<td>3.92E – 43</td>
<td>6.01E – 32</td>
<td>0.06E – 1</td>
</tr>
<tr>
<td>100D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st (Best)</td>
<td>1.99E – 21</td>
<td>8.92E – 21</td>
<td>1.2E – 20</td>
<td>1.2E – 20</td>
<td>1.2E – 20</td>
<td>1.15E – 20</td>
<td>1.05E – 15</td>
</tr>
<tr>
<td>15th (Median)</td>
<td>1.99E – 21</td>
<td>8.92E – 21</td>
<td>1.2E – 20</td>
<td>1.2E – 20</td>
<td>1.2E – 20</td>
<td>1.15E – 20</td>
<td>1.05E – 15</td>
</tr>
<tr>
<td>30th (Worst)</td>
<td>2.00E – 21</td>
<td>8.95E – 21</td>
<td>1.2E – 20</td>
<td>1.2E – 20</td>
<td>1.2E – 20</td>
<td>1.15E – 20</td>
<td>1.05E – 15</td>
</tr>
<tr>
<td>Std.</td>
<td>4.07E – 24</td>
<td>1.03E – 23</td>
<td>4.08E – 23</td>
<td>8.26E – 23</td>
<td>4.2E – 20</td>
<td>2.27E – 16</td>
<td>0.05E – 2</td>
</tr>
</tbody>
</table>

problems. Also it may be noted that COUNTER_NICHING_EA showed similar poor performance on this test problem (Bhattacharya, 2006).

4.2. Comparative performance of POPULATION_EA

Table 3 presents the comparative performance of POPULATION_EA as against COUNTER_NICHING_EA and other techniques, namely, (i) Standard EA (SEA), (ii) Self-organized criticality EA (SOCEA), (iii) The Cellular EA (CEA), and (iv) Diversity guided EA (DGEA) as has been explained in Ursem (2002). Results reported for POPULATION_EA were averaged over 50 independent runs in this case to eliminate effects of random results.

4.2.1. Statistical significance of comparative analysis

A t-test (at 0.05 level of significance; 95% confidence) was applied in order to test if differences in the performance of the best average fitness function are statistically significant when compared with the one for the other techniques used for comparison. The P-values of the two-tailed t-test are given in Table 4.

From Table 4 it can be seen that the differences between POPULATION_EA and the other algorithms (except for COUNTER_NICHING_EA) are statistically significant on most of the considered test functions, specifically in the higher dimensional cases.

4.3. Final observations

Overall, the simulation results ascertain POPULATION_EA’s superior performance as regards to solution precision for all seven test cases, across their three dimensional variants, as can be observed from Table 3. This may be attributed to POPULATION_EA’s ability to strike a better balance between exploration and exploitation. Interestingly the performance of POPULATION_EA is very close to that of COUNTER_NICHING_EA for the test cases in their 20 dimensional variants. However, the POPULATION_EA outperforms COUNTER_NICHING_EA when tested on the 50 and the 100 dimensional variants of all the test cases.

5. Application to financial portfolio optimization

The seminal work of Markowitz (1952) set the scene for portfolio optimization research. In this context portfolio optimization is an investment decision-making process to select a set of financial assets that satisfy various investment criteria.

5.1. Problem definition

Following the widely used Markowitz model, a portfolio P can be represented as a set of N real-valued weights (w0, w1, ..., wN) which corresponds to N available assets so that the following constraints are satisfied:

\[ \sum_{i=0}^{N} w_i = 1 \]

\[ 0 \leq w_i \leq 1 \]

The expected return \( R_p \) and the risk \( \sigma_p \) of a portfolio are respectively expressed as follows:

\[ R_p = \sum_{i=0}^{N} w_i R_i \]

\[ \sigma_p = \sqrt{\sum_{i=0}^{N} \sum_{j=0}^{N} \sigma_{ij} w_i w_j} \]

The risk measure of an asset is the variance of the asset’s return over time. The risk of the portfolio \( \sigma_p \) is the covariance between its assets.

The problem of portfolio management can then be viewed as maximizing the expected return \( R_p \) and minimizing the risk \( \sigma_p \) of a portfolio.

The Sharpe ratio is a well known financial indicator that is frequently used to evaluate investments. It can be viewed as an indicator of risk-adjusted return. For simplicity, in our experiments we use the Sharpe ratio as the fitness function. Maximizing the Sharpe ratio is then the objective of the problem. The Sharpe ratio can be defined as follows:

\[ S_i = \frac{R_i - R_b}{\sigma_i} \]
The POPULATION_EA algorithm presented in this paper basically targeted to tackle higher dimensional problem domain. The basic

5.2. Datasets and experiment setup

The simulation data has been extracted from the following two datasets: the NASDAQ 100 index and the NIKKEI 225 index. The NASDAQ 100 is composed of 100 assets related mainly to the technology industry. On the other hand, the NIKKEI 225 index is composed of 225 assets from diverse fields. A subset of each dataset has been extracted for the purpose of our experiments. We have compared the performance of POPULATION_EA against SEA (Standard EA). The parameter settings for POPULATION_EA for this simulation are depicted in Table 5. Different parameter settings have been used for experiments with the two datasets due to difference in data size. Training length indicates how many months prior to the current date will be considered while calculating the expected return. Values of training length for the two sets of experiments (with two datasets) have been computed empirically following the model used in Aranha and Iba (2007, 2009).

5.3. Results and discussions

The comparative performances of POPULATION_EA, SEA (standard EA) and the historical values of the indexes have been presented in Table 6. The index values represent the average trend of the component assets. The goal of our experiments is to see if the proposed POPULATION_EA can “beat the index” or pick up the above average assets.

We have considered the following two performance measures:
1. Cumulative return
2. Sharpe ratio

Cumulative return indicates how much the portfolio’s value increased compared to its initial value (Aranha and Iba, 2007). The Sharpe ratio is a measure of risk adjusted returns; higher the Sharpe ratio, lower the risk.

Results reported in Table 6 presents the average of best results obtained over 50 independent runs. The standard deviation of the average has been indicated within parenthesis.

As is apparent from Table 6, the proposed algorithm (POPULATION_EA) consistently manages to outperform SEA (standard EA) in case of both NASDAQ 100 and NIKKEI 225. It does surpass the index in case of NASDAQ 100. The marginally poorer performance in case of the NIKKEI 225 may be due to volatile or “bubble behavior” of the dataset. No special measures were introduced to handle such volatility as we just wanted to test general suitability of POPULATION_EA for this problem domain. Modifying the fitness function to account for volatility, especially “hararmfull volatility” may prove helpful.

6. Conclusions

The POPULATION_EA algorithm presented in this paper basically targeted to tackle higher dimensional problem domain. The basic

| Function | P_EA**_SEA | P_EA**_SOCEA | P_EA**_CEA | P_EA**_DGEA | P_EA**_C_EA*
|----------|-------------|-------------|-------------|-------------|-------------
| fros(x) 20D | 0.1144 | 0.4263 | 0.625 | 0.9954 | 1 |
| fros(x) 20D | 0.2793 | 0.3349 | 0.4231 | 0.9998 | 1 |
| fros(x) 20D | 0.0009 | 0.0901 | 0.2636 | 0.9999 | 1 |
| fros(x) 20D | 0 | 0 | 0 | 0.0044 | 1 |
| fros(x) 20D | 0 | 0 | 0 | 0.0002 | 1 |
| fros(x) 50D | 0.0903 | 0.217 | 0.4198 | 0.9873 | 1 |
| fros(x) 50D | 0.2037 | 0.2843 | 0.3098 | 0.9725 | 1 |
| fros(x) 50D | 0 | 0 | 0 | 0.0002 | 1 |
| fros(x) 50D | 0 | 0 | 0 | 0.0002 | 1 |
| fros(x) 100D | 0.0891 | 0.1363 | 0.2857 | 0.975 | 1 |
| fros(x) 100D | 0.1337 | 0.2019 | 0.2776 | 0.9546 | 1 |
| fros(x) 100D | 0 | 0 | 0 | 0.0002 | 1 |
| fros(x) 100D | 0 | 0 | 0 | 0.0002 | 1 |
| fros(x) 100D | 0 | 0 | 0 | 0.0002 | 1 |
| fros(x) 100D | 0 | 0 | 0 | 0.0002 | 1 |
| fros(x) 100D | 0 | 0 | 0 | 0.0002 | 1 |
| fros(x) 100D | 0 | 0 | 0 | 0.0002 | 1 |

Table 4
The P-values of the t-test with 99 degrees of freedom. Dimensions of each function considered are 20, 50 and 100. ‘–’ appears where the corresponding data is not available.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>NASDAQ 100</th>
<th>NIKKEI 225</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generations</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>Population size</td>
<td>200</td>
<td>400</td>
</tr>
<tr>
<td>Crossover rate</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Training length</td>
<td>12</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 5
Parameter settings for POPULATION_EA for the financial portfolio optimization experiment.

<table>
<thead>
<tr>
<th>Method</th>
<th>NASDAQ 100</th>
<th>NIKKEI 225</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sharpe ratio</td>
<td>1.73</td>
<td>1.56</td>
</tr>
<tr>
<td>Cumulative return</td>
<td>2.91 (0.02)</td>
<td>1.41 (0.02)</td>
</tr>
<tr>
<td>Sharpe ratio</td>
<td>0.221</td>
<td>0.148</td>
</tr>
<tr>
<td>Cumulative return</td>
<td>0.181 (0.001)</td>
<td>0.191 (0.001)</td>
</tr>
<tr>
<td>Index</td>
<td>0.238</td>
<td>0.138</td>
</tr>
<tr>
<td>SEA (Standard EA)</td>
<td>0.391</td>
<td>0.184</td>
</tr>
<tr>
<td>POPULATION_EA</td>
<td>0.530</td>
<td>0.350</td>
</tr>
</tbody>
</table>

Table 6
Comparative performance of POPULATION_EA, SEA (standard EA) and the historical values of the indexes for the financial portfolio optimization experiment.

framework incorporates three key processes. Firstly, the population's spatial information is obtained with a computationally inexpensive GRID_NICHING algorithm. Secondly, the information is used to identify potential local convergence and community formations and then diversity is intelligently introduced with informed genetic operations; aiming at two objectives: (a) Promising samples from unexplored regions are introduced replacing redundant less fit members of over-populated communities. (b) While local entrapment is discouraged, representative members are still preserved to encourage exploitation. Finally, a multi-tier hierarchical population scheme has been used to prevent premature death of promising but initially under-performing individuals. The strong emphasis on exploration of the COUNTER_NICHING_EA has been balanced in the POPULATION_EA with the help of the multi-population structure. The main conclusions derived from the results of the simulations carried out are as follows:

The empirical results obtained from the simulation runs revealed some interesting trends. POPULATION_EA clearly outperformed other techniques considered for comparison particularly in case the higher dimensional test cases.

In conclusion, we can say that the POPULATION_EA model is specifically meant to target higher dimensional problems.

Appendix B

Algorithm 2: Procedure GRID_NICHING

1: begin
2: Compute grid block cluster densities $D_{Eb}$, where $D_{Eb} = N_{B}/N_{Pop}$, $N_{B}$ and $N_{Pop}$ being the number of individuals in grid block cluster $(B_{i})$ and the population size, respectively.
3: for each grid block cluster $B_{i}$
4: begin
5: if $(D_{Eb} > \text{Threshold}_1)$ then
6: Mark grid block cluster as potential region or community hub $B_{i-HUB}$
7: end if
8: end for
9: (Look for possible cluster around each marked grid block cluster $B_{i-HUB}$)
10: for each marked grid block cluster $B_{i-HUB}$
11: begin
12: while (Number of new members found < Threshold $N_{lim}$) AND (not $\text{overlap with any marked cluster } CLUST_{K}$)
13: begin
14: Expand the region $\Re \rightarrow K_{Block}$ in all dimensions by a predefined step size $\delta$
15: Find members that belong to the expanded region or new non-uniform hypercube $\Re$
16: end
17: Mark the new found community, $CLUST_{K}$ where, $k$ is the cluster number
18: end while
19: end for
20: Return community chart, consisting of list of $CLUST_{K}$
21: end

Appendix C

Algorithm 3: Procedure INFORMED_OP

1: begin
2: Get community chart, consisting of list of $CLUST_{K}$
3: for each cluster $CLUST_{K}$
4: begin
5: if (fitness standard deviation of cluster, $CLUST_{K}$ ≤ average fitness standard deviation of all clusters) AND cluster density, $D_{Eb}$ ≥ average cluster density of all clusters then
6: begin
7: Randomly sample $n$ individuals in virgin zones and replace $n$ worst representatives from the identified cluster, by $n$ best-sampled individuals.
8: end if
9: else if (average cluster fitness $f_{CLUST_{AVG}(K)}$ > Average fitness of all clusters) then
10: begin
11: Mark best individual of the cluster as an elite to be transferred to the new population unchanged, irrespective of the individual’s fitness ranking when the entire population is considered.
12: end else if
13: end for
Algorithm 3: Procedure INFORMED_OP

14: Mutate $P(t)$
15: Recombine $P(t)$
16: Return $P(t)$
17: end

References


