Multi-population Genetic Algorithm for Cardinality Constrained Portfolio Selection Problems

Nasser R. Sabar[1][0000−0002−0276−4704], Ayad Turky[2], Mark Leenders[2], and Andy Song[2]

[1] La Trobe University, Melbourne, VIC 3083, Australia
n.sabar@latrobe.edu.au
[2] RMIT University, Melbourne, VIC 3000, Australia
{ayad.turky, mark.leenders, andy.song}@rmit.edu.au

Abstract. Portfolio Selection (PS) is recognized as one of the most important and challenging problems in financial engineering. The aim of PS is to distribute a given amount of investment fund across a set of assets in such a way that the return is maximised and the risk is minimised. To solve PS more effectively and more efficiently, this paper introduces a Multi-population Genetic Algorithm (MPGA) methodology. The proposed MPGA decomposes a large population into multiple populations to explore and exploit the search space simultaneously. These populations evolve independently during the evolutionary learning process. Yet different populations periodically exchange their individuals so promising genetic materials could be shared between different populations. The proposed MPGA method was evaluated on the standard PS benchmark instances. The experimental results show that MPGA can find better investment strategies in comparison with state-of-the-art portfolio selection methods. In addition, the search process of MPGA is more efficient than these existing methods requiring significantly less amount of computation.

Keywords: Optimisation · Portfolio Selection Problems · Genetic Algorithms · Multi-population GA.

1 Introduction

Investment is one of the most essential activities in the finance industry, and a key mean of stimulating economic growth. A good investment strategy is obviously to achieve maximum return back to the investors while the risk of investment loss should be minimal [1], [2], [3]. In reality high investment profit often associates with high risk. Therefore, professional investors and brokers often maintain a portfolio of investment consisting of a collection of relative small assets instead of a single or a small set of large assets. So the risk can be mitigated if one or two investments went wrong. Setting the optimal portfolio is a key part of daily tasks of portfolio managers. The challenge here is not only finding an optimal

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_10

or near optimal portfolio investment, but also to find a good solution efficiently so the professionals can react to events such as market changes quickly.

The above problem is known as the Portfolio Selection (PS) problem, which is also a challenge for computer scientists and optimisation practitioners. PS is one of the key roles of portfolio mangers whose goal is obviously needed to maximise customer satisfaction. The selection process can be done manually by a manager. However, that is only suitable when this manager is very experienced and the choice is rather limited. In many circumstances formulating the best portfolio strategy by hand is not feasible as the problem is combinatorial in nature, with high computational complexity. For circumstances where the market changes fast or the asset structure is complicated, the complexity of PS can go higher and become difficult to solve even for computing methods. Establishing a fast and effective portfolio selection algorithm remains as a challenge. The PS problem has some notable extensions. The most studied is probably a variation with the addition of cardinality and boundary constraints [3]. Cardinality constraint means the total number of assets to be included in the solution portfolio can not go beyond a certain threshold. Boundary constraint specifies the lower and upper limits of investment that can go to each asset in the formed portfolio. The task is known as cardinality constrained portfolio selection problem.

It is known that finding the optimal portfolio is an NP-hard problem. One of the widely used methods in PS is the Markowitz mean-variance model, which forms a single portfolio. This model captures the expected return and the risk of the portfolio [1] [4]. This model is and still remains as the core of existing PS methodology. However, the practicality of this model in real world scenarios has been criticized because the assumptions of the model are not very realistic. For example, it assumes the returns are in normal distributions. It also assumes that correlations between assets are fixed and never change. Utilizing the Markowitz mean-variance PS model, a PS task can be formulated as a quadratic programming problem [3]. This is an exact method which can find the actual optimal solution. However, that is only feasible when the number of variables in the model is small. When the number of variables is large, exact methods become impractical, if not impossible, to find the optimal solution within an acceptable amount of time.

In contrast to exact methods, heuristic and meta-heuristic approaches search for near-best solutions. It is well known that meta-heuristic algorithms can find good quality solutions within a reasonable period of time [5]. This approach has indeed been introduced to Portfolio Selection problems. That includes the use of Genetic Algorithm [3]], Tabu Search [3], Simulated Annealing [3], Particle Swarm Optimisation [6], Harmony Search [7] and hybrid algorithms [8] [9].

To further improve the PS solving mechanism, we thereby propose a new approach based on Genetic Algorithm (GA). Also we address the extended version of PS which has cardinality and boundary constraints embedded as these constraints are more realistic but add many more difficulties in finding a good solution. GA is a well-known population based meta-heuristic search method which simulates the survival of the fittest principle for problem solving [5] [10]. In the past GA has demonstrated its effectiveness in solving optimisation problems from a wide range of fields that include many difficult real-world applications. However, classical GA does have certain drawbacks. For example the convergence of the GA evolution process can be slow hence affecting the time required for finding a satisfactory solution. In constrained optimisation problems in particular the PS problem in this study, slow convergence may jeopardise GA as the chosen method due to the efficiency issue. One of the causes of this phenomenon is the use of a single population in classical GA, as the exploitation in the search space may not unfold well because the coverage in the search space of only one population may not be sufficient even if the population size is big [11].

To address the aforementioned issues in classical GA, we propose a variation of GA which allows multiple populations co-exist during one GA evolution process. We denote that method as MPGA which is designed for the constrained PS. By MPGA one or more populations will explore the search space of a problem, whilst other populations can perform exploitation in the same space. With the combination of both exploration and exploitation the convergence of a GA search process is expected to be quicker. In addition, MPGA allows good individuals to be passed across different populations. So good genetic materials can be shared periodically to help different populations find better solutions more effectively. The well known PS benchmark dataset [3] is used for performance evaluation in this study. This benchmark has been widely used in the PS literature. On this benchmark MPGA is compared against state of the art portfolio selection algorithms that are widely used by PS researchers, developers and managers.

The rest of the paper is organised as such. Section 2 describes the portfolio selection problem in detail. Section 3 discusses the main components of the proposed method. Section 5 shows the experimental results with the comparison with existing methods. The conclusion of this study is presented at Section 6.

2 Problem Descriptions

This paper focuses on the cardinality constrained portfolio selection problem which has two constraints added, the cardinality constraint and boundary constraint. These constraints are to reduce the transaction cost and avoid investment assets that are too small or too large. Cardinality constraint limits the number of assets to be included in the formed portfolio. Boundary constraint set a lower bound and upper bound for each asset of the formed portfolio. The formulation of the PS model is proposed by [3] [12]:

$$min\ \lambda \left[\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \alpha_{ij}\right] + (1-\lambda)\left[-\sum_{i=1}^{n} w_i \mu_i\right] \tag{1}$$

Subject to

$$\sum_{i=1}^{n} w_i = 1 \tag{2}$$

$$\sum_{i=1}^{n} s_i = K \tag{3}$$

$$\varepsilon_i s_i \leq w_i \leq \delta_i s_i, i = 1, ..., n \tag{4}$$

$$s_i \in \{0, 1\}, i = 1, ..., n \tag{5}$$

where n is the total number of assets, w_i is the proportion of the budget invested in the i-th asset, α_{ij} is the covariance between i-th and j-th assets, λ is the risk aversion, λ in [0, 1], μ_i is the expected return of the i-th asset, K is the number of assets to be invested in assets in a portfolio, s_i is a decision variable represents whether the i-th asset has been selected or not, and ε_i and δ_i respectively are the upper and lower bounds. The cardinality constrained PS model involves two sub-problems: (1) the selection problem that seeks to select a subset of assets and (2) the allocating problem which aims at determining the proportion for each of the selected asset. In the literature, PS formulations are treated as a mixed integer programming [3].

3 Methodology

The classical GA has shown slow convergence and difficulties in handling constrained optimisation problems [11]. While Memetic Algorithms (MAs), which combine GA with local search algorithms, have been proposed to improve the convergence process of GA [11] [13],[14],[15],[16]. A local search algorithm is called at every generation to further improve the generated solutions and exploitation process [17]. Nevertheless, calling the local search algorithm at every generation would be time consuming and may lead to premature convergence. Our proposed multi-population GA (MPGA) is a remedy of this issue of memetic algorithms while still addressing the slow convergence problem [18].

Figure 1 shows the flowchart of the proposed MPGA. It starts from generating a population of random solutions. Then MPGA decomposes the whole population into multiple subpopulations from *Subpopulation* 1 to *Subpopulation* n. The subpopulations are scattered over the search space. This is to encourage exploration. While the search of each sub population acts like exploitation in nearby space similar to that in local search. These subpopulations evolve independently during the search process. From Figure 1 we can see that the process of each subpopulation is identical, all following selection, which is to pick good solutions to produce the next generation; crossover and mutation, which are to produce offspring solutions based on the picked solutions. When better solutions are found, the subpopulation will be updated as shown in the figure.

However at each generation, if the update criterion is met but the stopping criterion is not met, these subpopulations will be combined into a large population for the next generation, which will start from splitting the large population into multiple subpopulations again. So the exploration and exploitation will start

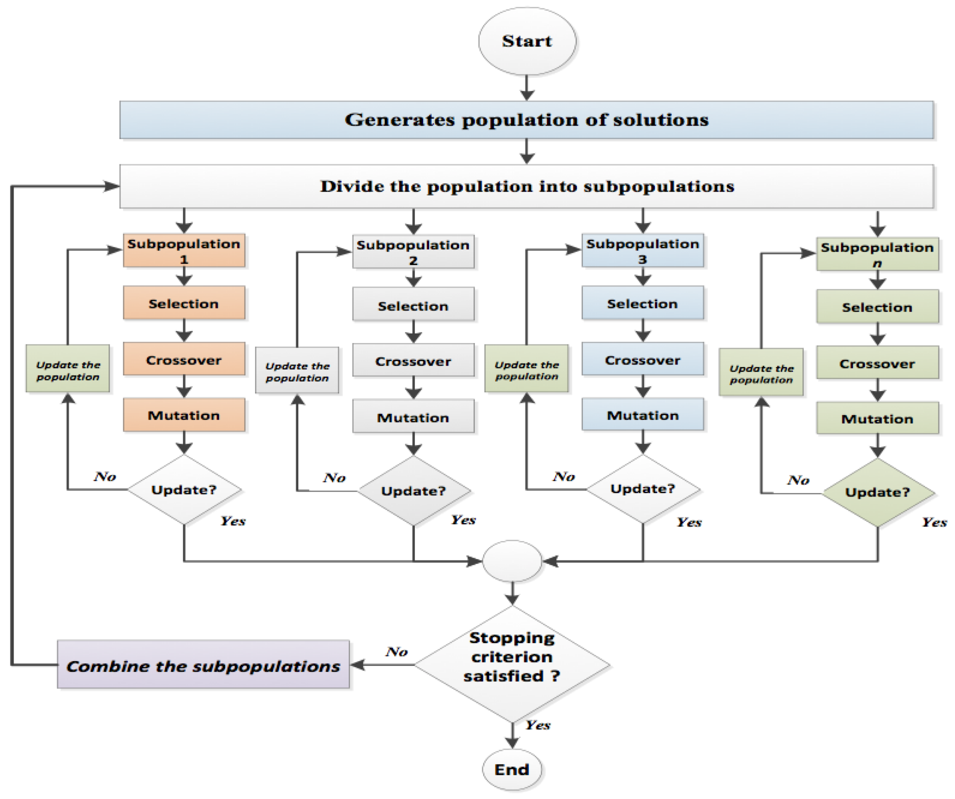
Start
Generates population of solutions
Divide the population into subpopulations
Subpopulation 1
Subpopulation 2
Subpopulation 3
Subpopulation n
Selection
Crossover
Mutation
Update the population
Update?
No
Yes
Stopping criterion satisfied ?
Combine the subpopulations
End

Fig. 1. Visual Representations of the Proposed Multi-population Genetic Algorithm (MPGA)

over again. The update criterion is that the best solution has not been improved for a certain steps. The stopping criterion is that the maximum number of generation has been reached. We can see from the algorithm that the search progress may progress further if it is trapped in a local optima because it can be detected and dealt with by the combining and dividing of populations.

Asset Index	1	2	3	4	5	6	...	n
Selection	1	1	0	0	0	1	...	1
Boundary Value	0.1	0.5	0	0	0	0.2	...	0.1

Fig. 2. An Example of Solution Representation

In terms of problem representation, each individual of GA is a solution for PS. A solution is represented by a two-dimensional array where the array size is equal to the total number of assets n, as shown in Figure 2. The first row represents the selection. The value of each cell is either 0 or 1, while 1 indicates the corresponding asset is selected and 0 is not. The second row represents the boundary value of the chosen asset which takes a real value within the given lower and upper boundary constraint.

Each cell of the first row is randomly assigned either 0 or 1 while makes sure that the total number of $1s$ satisfies the cardinality constraint, the max number of assets in the portfolio. Next, for each column with a 1 at the first row, the corresponding cell in the second row will be assigned with a real value. That value has to be in between the predefined upper and lower bounds to satisfy the constraint. In addition, the sum of these values in the second row has to be 1 indicating 100% allocation of the investment fund.

Once solutions in the initial populations are generated, they will be assigned with fitness values before the standard GA selection process. The calculation of the fitness value is based on Equation (1) described above. The decomposition into n subpopulations is done in a random manner. That is, each subpopulation contains a set of solutions randomly selected from the whole populations. No duplicates will occur in the selections so there is no overlaps between these subpopulations.

All subpopulations use roulette wheel as the selection mechanism. They all use one point crossover operator, one point mutation operator and steady state updating rule. The crossover operator, which exchanges parts between two selected solutions, runs with a low probability and only generates one offspring by picking the best one generated by the parents. This is to increase the efficiency of the search. The mutation operator, which randomly modifies the selected solution, can be viewed as a local search algorithm in MPGA. Mutation has relatively high probability in order to give better chances to perturb cell values to exploit surround areas in the search space. Only those changes that lead to an improvement in the fitness values are accepted in the offspring. If the best solution in all subpopulations cannot be improved for a consecutive number of generations, all subpopulations are merged into one population to combine their genetic materials for the next round of decomposition. By this approach, solutions will be regrouped for the new episode of search. The regrouping will stimulate the search in new areas while still maintain the best solution obtained so far. Hence it has a better chance to overcome local optima. As mentioned early and shown in Figure 1, the entire process repeats until the maximum number of generations is reached.

4 Experiment Settings

This section first discusses the characteristics of instances from the PS benchmark which is used to evaluate the performance of the proposed MPGA. Then the MPGA parameter settings is presented in detail.

4.1 Benchmark instances

The PS benchmark instances from the OR–library are used to evaluate the effectiveness of the proposed MPGA. This benchmark is commonly used in the literature of Portfolio Selection studies [19]. It comprises five different sets each representing the weekly share prices at the stock market of a country [19]. The main characteristics of these five instances are listed in Table 1. In this table, n represents total number of the assets, k represents the maximum number of assets in a formed portfolio (cardinality constraint), ε_i (i=1,...,n) is the lower bound of the asset and δ_i (i=1,...,n) is the upper bound of the asset.

Table 1. Portfolio Selection Benchmark Datasets

#	Data Set	Country	n	k	ε	δ
1	Hang Seng	Hong Kong	31	10	0.01	1
2	DAX 100	Germany	85	10	0.01	1
3	FTSM 100	UK	89	10	0.01	1
4	S&P 100	USA	98	10	0.01	1
5	Nikkei	Japan	255	10	0.01	1

4.2 Parameter settings

The proposed MPGA involves six parameters that need to be set in advance. To calibrate MPGA parameters, we randomly selected two data sets for parameter tuning purpose. The selected sets are: DAX 100 and Nikkei. Next, we conducted a preliminary experiment to set the parameters value of MPGA. For each parameter, we have tested a range of values within the predefined range and the value that leads to good results are used. Also the trade-off between the solution quality and the computational time is considered. Based on this empirical study, parameters are settled with the suggested values which are shown in Table 2. The λ value of Equation (1) was tested using 51 different values and each value is tested for $1000 \times n$ times of evaluations. That is the same as the experiments in [3] and [6].

5 Results and Comparisons

To evaluate the effectiveness of the proposed MPGA, two sets of experiments were conducted. In the first set of experiments, we evaluated the benefit of using multi-population by comparing the results of GA with multi-population (MPGA) and the results of classical single population GA. In addition, we also investigated the impact of the number of subpopulations in MPGA. In the second set of experiments, MPGA was compared with the existing state of the art methods. To make a fair and consistent comparison, our MPGA used the same PS benchmark and applied the same stopping condition as reported in these studies, the maximum number of evaluations.

Table 2. MPGA Parameter Setting

Parameters	Tested Range	Suggested Values	
		Exploration	Exploitation
Crossover rate	0.01-0.99	0.8	0.2
Mutation rate	0.01-0.99	0.3	0.85
Population size	10-500	300	
Number of sub-populations N	2-10	6	
Number of consecutive non-improvement generations	5-100	Every 50 fitness evaluations	
Maximum number of generations $MaxG$	$1000 \times n$		

5.1 Effectiveness evaluation

To ensure a fair comparison, both MPGA and GA have been tested on same population of solutions, stopping condition and computer resources. All the experiments of both MPGA and GA are conducted for 51 independent runs with different random seeds. All five datasets from the benchmark were used in this comparison.

The results from both MPGA and GA are statistically compared using the Wilcoxon test with a confidence level of 0.05. MPGA consistently outperformed classical GA. The actual results are shown in Table 3 which combines a few other comparison results. Here we only show the p-value of MPGA against GA in Table 3. In this table, a p-value <0.05 means that the MPGA is statistically better than GA (shown in bold). A p-value >0.05 means that the difference between these two algorithms are not significant.

Table 3. The p-value of comparing MPGA with GA

	Data Set	p-value
1	Hang Seng	0.059
2	DAX 100	**0.023**
3	FTSM 100	**0.016**
4	S&P 100	**0.037**
5	Nikkei	**0.011**

The p-values tabulated in Table 3 show that MPGA is statistically better than GA on four out of five data sets. From this experiment, we observe that the multi-population approach has a positive impact on the performance GA.

Now we show the studies on the impact of n, the number of subpopulations, to see how this influences the performance of MPGA in term of solution quality as well as the computational cost. MPGA was evaluated on two data sets, DAX 100 and Nikkei. The n value tested from $n = 2$ to $n = 10$ are presented in Table

Table 4. The impact of n: the number of subpopulations, on solution quality and computational cost

	$n = 2$	$n = 4$	$n = 6$	$n = 8$	$n = 10$
λ (DAX 100)	2.8324	2.7461	2.3116	2.3027	2.2875
s (DAX 100)	16.21	17.11	18.21	26.33	28.14
λ(Nikkei)	0.7643	0.7275	0.6328	0.6279	0.6248
s(Nikkei)	56.18	67.26	73.14	105.16	115.71

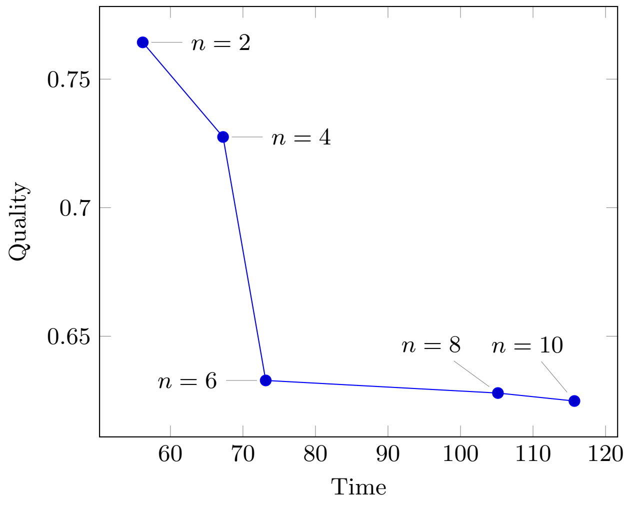

Fig. 3. Runtime and solution quality under different n for Nikkei

4. For each dataset, the solution quality λ and runtime in seconds s are recorded in two rows of the table. It can be seen that with the increased n value, the quality improves as the goal of PS is to minimise the λ value. As expected the computational cost climbs quite quickly with the size n as well.

The relationships between solution quality and run time are also shown in Figures 4 and 3 which are for Nikkei and DAX 100 dataset respectively. From the obtained results, we can see that the best trade-off is when $n = 6$. Although $n = 8$ and $n = 10$ are slightly better, their computational cost are higher that than $n = 6$ and they do not add much extra value to the solution.

5.2 Comparisons with state of the art methods

In this section, the proposed MPGA is compared with the state of the art methods. The methods included in the comparison are the following four:

- Tabu search algorithm (TS) [3]
- Simulated annealing (SA) [3]

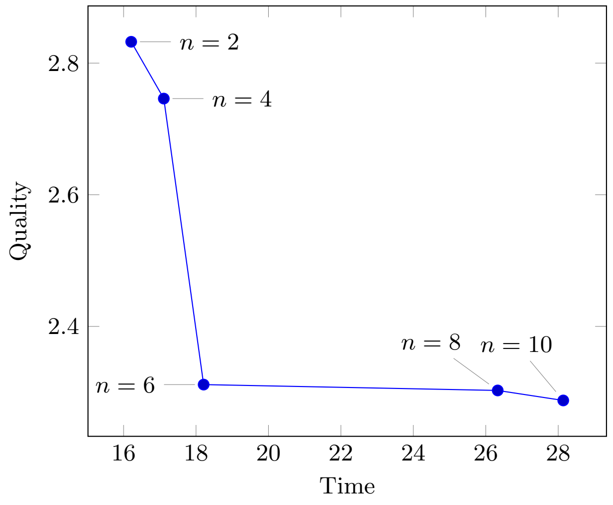

Fig. 4. Runtime and solution quality under different n for DAX 100

- Genetic algorithm (GA) [3]
- Particle swarm optimisation (PSO) [6]

The comparison includes the solutions found by MPGA against those from TS, SA, GA and PSO. The minimum mean percentage error over 51 independent runs was used. Table 5 shows the results from MPGA and the results from the aforementioned four existing PS methods. These results of the compared methods are taken form the corresponding publication. In the table, the best results are shown in bold. Table 5 shows that, on all five tested data sets, the proposed MPGA obtained better results compared to the state of the art methods. When considering the overall average result, which is shown in the last row of Table 5, MPGA is also achieved better solution than other algorithms.

Table 5. MPGA Comparing with Existing State-of-the-art Methods

#	Data Set	MPGA	GA	SA	TS	PSO
1	Hang Seng	**1.0952**	1.0974	1.0957	1.1217	1.0953
2	DAX 100	**2.3116**	2.5424	2.9297	3.3049	2.5417
3	FTSM 100	**1.0543**	1.1076	1.4623	1.6080	1.0628
4	S&P 100	**1.6482**	1.9328	3.0696	3.3092	1.6890
5	Nikkei	**0.6328**	0.7961	0.6732	0.8975	0.6870
Average overall		**1.34842**	1.49526	1.8461	2.04826	1.41516

Table 6. Computation Cost of MPGA and Existing State-of-the-art Methods (in seconds)

#	Data Set	MPGA	GA	SA	TS	PSO
1	Hang Seng	**1.3**	172	79	74	4.8
2	DAX 100	**18.21**	544	210	199	26.8
3	FTSM 100	**28.34**	573	215	246	31.4
4	S&P 100	**30.44**	638	242	225	36.6
5	Nikkei	**73.14**	1964	553	545	75.8

To compare above methods in term of efficiency, we measured the computational runtime of these methods on the five datasets. Table 6 presents the time of MPGA, GA, SA, TS and PSO in seconds. In the table, the best computational time are also highlighted in bold. As can be seen, the computational time of MPGA is considerably lower than that of compared methods on all tested instances. Note that MPGA and all the methods in this comparison use the number of fitness evaluation as the main stopping condition. That is fixed and identical for all runs regardless of the method. Hence the advantage of MPGA is valid.

6 Conclusion

In this work, a multi-population genetic algorithm is presented for solving the cardinality constrained portfolio selection problem. The proposed MPGA can improve the convergence of the search process. The combining and splitting of subpopulations has the effect of hybridising exploration and exploitation in the search space. The sharing of genetic materials during the evolution process encourages the search to step out local optima more effectively. The performance of the proposed MPGA was evaluated on the benchmark datasets of cardinality constrained portfolio selection problem. The experiments showed the effectiveness of MPGA by comparing it with GA as well as the current state-of-the-art methods. In addition, the computational cost of MPGA is much lower than that of these state-of-the-art methods. Thus, we conclude that MPGA, multiple population of GA, is an effective and efficient method for the cardinality constrained portfolio selection problems.

This study of PS is still at its early stage. A few extensions will be investigated in the near future, for example improving the sharing and collaboration between multi-populations and introducing MPGA into other similar domains. In addition we will study the impact of the solution distribution of multiple population.

References

1. Harry Markowitz. Portfolio selection*. *The journal of finance*, 7(1):77–91, 1952.

2. Hal Varian. A portfolio of nobel laureates: Markowitz, miller and sharpe. *The Journal of Economic Perspectives*, pages 159–169, 1993.
3. T-J Chang, Nigel Meade, John E Beasley, and Yazid M Sharaiha. Heuristics for cardinality constrained portfolio optimisation. *Computers & Operations Research*, 27(13):1271–1302, 2000.
4. Harry M Markowitz. *Portfolio selection: efficient diversification of investments*, volume 16. Yale university press, 1968.
5. Michel Gendreau and Jean-Yves Potvin. *Handbook of metaheuristics*, volume 2. Springer, 2010.
6. Guang-Feng Deng, Woo-Tsong Lin, and Chih-Chung Lo. Markowitz-based portfolio selection with cardinality constraints using improved particle swarm optimization. *Expert Systems with Applications*, 39(4):4558–4566, 2012.
7. Nasser R Sabar and Graham Kendall. Using harmony search with multiple pitch adjustment operators for the portfolio selection problem. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 499–503. IEEE, 2014.
8. Graham Kendall and Yan Su. Imperfect evolutionary systems. *Evolutionary Computation, IEEE Transactions on*, 11(3):294–307, 2007.
9. Alberto Fernández and Sergio Gómez. Portfolio selection using neural networks. *Computers & Operations Research*, 34(4):1177–1191, 2007.
10. John H Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.
11. Ferrante Neri and Carlos Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.
12. Rafael Moral-Escudero, Rubén Ruiz-Torrubiano, and Alberto Suárez. Selection of optimal investment portfolios with cardinality constraints. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 2382–2388. IEEE, 2006.
13. Nasser R Sabar and Aldeida Aleti. An adaptive memetic algorithm for the architecture optimisation problem. In *Australasian Conference on Artificial Life and Computational Intelligence*, pages 254–265. Springer, 2017.
14. Nasser R Sabar, Andy Song, and Mengjie Zhang. A variable local search based memetic algorithm for the load balancing problem in cloud computing. In *European Conference on the Applications of Evolutionary Computation*, pages 267–282. Springer, 2016.
15. Nasser R Sabar, Jemal Abawajy, and John Yearwood. Heterogeneous cooperative co-evolution memetic differential evolution algorithm for big data optimization problems. *IEEE Transactions on Evolutionary Computation*, 21(2):315–327, 2017.
16. Nasser R Sabar, Edward Chung, Takahiro Tsubota, Paulo Eduardo Maciel de Almeida, et al. A memetic algorithm for real world multi-intersection traffic signal optimisation problems. *Engineering Applications of Artificial Intelligence*, 63:45–53, 2017.
17. Anmar Abuhamdah, Masri Ayob, Graham Kendall, and Nasser R Sabar. Population based local search for university course timetabling problems. *Applied intelligence*, 40(1):44–53, 2014.
18. Nasser R Sabar and Andy Song. Dual population genetic algorithm for the cardinality constrained portfolio selection problem. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 703–712. Springer, 2014.
19. John E Beasley. Or-library: distributing test problems by electronic mail. *Journal of the operational research society*, pages 1069–1072, 1990.