Using Machine Learning to Improve Evolutionary Multi-Objective Optimization

Rakan Alotaibi[†],

Computer and Information Systems college, Umm Alqura university, Saudi Arabia

Summary
Multi-objective optimization problems (MOPs) arise in many real-world applications. MOPs involve two or more objectives with the aim to be optimized. With these problems improvement of one objective may led to deterioration of another. The primary goal of most multi-objective evolutionary algorithms (MOEA) is to generate a set of solutions for approximating the whole or part of the Pareto optimal front, which could provide decision makers a good insight to the problem. Over the last decades or so, several different and remarkable multi-objective evolutionary algorithms, have been developed with successful applications. However, MOEAs are still in their infancy. The objective of this research is to study how to use and apply machine learning (ML) to improve evolutionary multi-objective optimization (EMO). The EMO method is the multi-objective evolutionary algorithm based on decomposition (MOEA/D). The MOEA/D has become one of the most widely used algorithmic frameworks in the area of multi-objective evolutionary computation and won has won an international algorithm contest.

Keywords:
Multi-objective, MOPs, MOEA, ML, Evolutionary Algorithm.

1. Introduction

The concept of optimization is very important for many of real-world applications and is considered as critical demand for their development and growth. The concept of optimization can be summarized into the process of selecting the best solution among a set of elements based on one or several specific criteria. Using optimization is highly important in many areas such as finance, communications, naval, air and land transport.

In most of optimization problems, the purpose is to optimize one or many objectives. That means to observe that objective and maximize or minimize its value, depending on the issue description and target. In other words, Optimization involves finding the optimal solution for the problem without affecting one of the objectives or making conflict between two objectives. When several objectives are used in the optimization process, the method is called Multi-objective optimization.

The Evolutionary Algorithms (EAs) have been known for their great support in solving all problems of optimization. The concept of these algorithms is based on the Darwin's theory of evolution. Some of the evolutionary mechanisms in Darwin's theory are characterized by population-based algorithms. The feature and resources of this algorithm are able to find many solutions for the optimization problems. In Multi-Objective Optimization Problems (MOOPs), problems may contain more than one objective. The objectives may have relations to each other and will certainly have conflict among them. In some cases, the solution will not only be one solution but will be a set of solutions to be the solutions or keys for the optimization process. These solutions are known as Pareto-optimal solutions. It should be emphasized that the main objective of this type of algorithm is to generate a range of solutions to show the Pareto-optimal front. Several Multi-Objective Evolutionary Algorithms (MOEAs) have been developed with their successful applications within the past two decades. It is essential to refer to the fact that MOOPs are usually developed in all real-world applications. Recently, researchers have attracted their attention to add more developments to this field due to the increasing applications of MOOPs and demand. Researchers have developed new algorithms solving MOOPs, known as multi-objective evolutionary algorithm based on decomposition (MOEA/D), based on the concept and principle of "decomposition".

Evolutionary Computation (EC) uses some algorithms named Evolutionary Algorithms (EA). In this research, Evolutionary Multi-Objective Optimization is achieved based on decomposition (MOEA/D). Decomposition divides the optimization problem with more than one objective into sub-problems and optimize each of them individually and simultaneously.

That means in case of using 0/1 knapsack problem as an example of optimization problem, it turns to multiple knapsacks with each sub-problem is a 0/1 knapsack problem. In knapsack problem, there are a set of objects or items, which have their values and weights. The solution to the problem is to find the best number of objects to be included in the collection in a way where the net weight is less than or equal to a certain top value while the net value

Manuscript received June 5, 2022
Manuscript revised June 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.6.29

is maximized. 0/1 knapsack means that the object will be included or excluded; it cannot be partially added to the knapsack.

Machine Learning (ML) is useful in many applications to let the machine learn from the data by itself. Each of Supervised Learning and Unsupervised Learning can have their own advantages and usages. Applying ML to Optimization process can solve similar problems with similar target by memorizing the information and enforce analogical steps for more accurate results and less processing time.

2. Motivation and Objectives:

The motivation of this research is to study MOPs with different techniques and decide which one gives better results in respect of maximizing the goal. The next step is to enhance to the selected technique trying to achieve better accuracy or lessen the time of optimization process.

This research compares between the results of optimization algorithms with and without the usage of ML. ML shall improve the MOPs results. MOPs with MOEA is used for this research. Decomposition is a step that is added to MOEA to resolve the problem to a number of sub-problems and optimize each one simultaneously and without conflict. The MOEA with decomposition techniques (MOEA/D) is then tested to a Multiple Optimization Problem (MOP) such as 0/1 Knapsack to determine its efficiency.

The workflow of this research will go through the following steps:

- Studying 0/1 multi-objective knapsack problem to understand the nature of the algorithm and its application depending on separate factors.
- Studying the concept of MOEA/D and implement it.
- Applying MOEA/D on 0/1 multi-objective knapsack.
- Using ML on MOEA/D, where the Random Forest method is used. Support Vector Machine (SVM) is used to determine which type of Random Forest to use.
- Comparing the results of the experiment using ML and the experiment results while not using ML

3. MOEA/D: A Multi-Objective Evolutionary Algorithm Based on Decomposition

Recently, a new MOEA has been proposed by Zhang and Li [1]. They proposed a simple but efficient and powerful MOEA that is based on decomposition called the OEA/D. This algorithm works very well on a wide range of multi-objective problems with many objectives and is most successful with a small population. The MOEA/D contains the following features:

- With the MOEA/D, a MOOP is handled as a collection of many single-objective optimization problems.
- It uses scalar optimization methods, such that one scalar optimization problem is linked to one solution. On the contrary, non-decomposition MOEAs do not have the advantage of scalar optimization methods.
- The MOEA/D has a low computational complexity.
- It allows the use of single-objective optimization methods as well as problem specific heuristics or local search.

The framework for MOEA/D can described as following:

Stage 1:
The MOEA/D begins by accepting a MOOP as an input. Then, it must decompose the MOOP that is under consideration into N scalar optimization sub-problems. Any decomposition technique can be used here (e.g., the weighted sum approach).

Stage 2:
This stage starts with the initialization of a set of even spread weight vectors. These weight vectors should satisfy the following two conditions:

$$\lambda^1 + \lambda^2 + \ldots + \lambda^N = 1$$

$$\lambda^1 \in \left\{ \frac{0}{H}, \frac{2}{H}, \ldots, \frac{H}{H} \right\}$$

Where H is a user-defined positive integer. The first condition states that the total sum of all the weight vectors must be 1. The second condition indicates that each individual weight in the set of weight vectors takes a value from weight vectors' count is calculated as follows:

$$N = C_{H-m-1'}^{m-1}$$

Putting m is the number of objectives. For example, by specifying H = 199, one has N = 150 weight vectors for a two objective (m = 2) problem: λ= (0,1) (0.01, 0.99).. (1,0). Take a three objective (m = 3) example. By specifying H = 25, one has N = 351 weight vectors: λ = (0,0,1), (0, 0:04,0:96),….(1,0,0). After the weight vectors have been initialized, one can proceed to the next stage.

Stage 3:
The MOEA/D requires the calculation of the T closest weight vectors in the neighborhood of each weight vector. The value of T is a user-defined positive integer. The MOEA/D uses the standard Euclidean distance to measure the distance between any two weight vectors. After the distances have been calculated, one assigns each λ^i the T closest weight vectors (including λ^i itself) as its neighbours and designates this set as $B(\lambda^i) = \{i_1, I_2,,,,,,i_r\}$ where i is the index to a solution neighbour. After determining the T closest neighbors for each weight vector, one can move to the next stage.

Stage 4:
Now one has N weight vectors as well as T neighbors in $B(\lambda^i)$. So, the MOEA/D can immediately begin the evolutionary process. An initial population is generated randomly or through a problem-specific method. Notice that the population size is the same as N, which is the number of weight vectors. Next, an offspring is generated for each λ^i by applying selection, crossover, and mutation. Two neighbour indices, k and i, are randomly selected from $B(\lambda^i)$ Then, the two neighbour solutions, x^k and x^i, are mated (i.e., undergo crossover and mutation) in order to produce the new solution y. The new solution y is improved by applying a problem-specific repair or an improvement heuristic approach to produce y^i. Then, the new solution y is compared with all of the T neighbours (including λ^i itself) by the scalarizing function (e.g., the weighted sum) with the weight vector of each neighbor. When it is determined that y is fitter than any of the T neighbours y, is replaced with that particular neighbour. This process continues until a stopping condition is met (e.g., a predetermined number of solutions have been examined).

The MOEA/D employs an external population EP to store the non-dominated solutions that are found up to this point in the search process. The EP is updated by the newly generated solution y. All of the vectors that are dominated by $F(y)$ are removed from EP. The preceding information is illustrated below as an algorithm

MOEA/D algorithm

1: Procedure MOEA/D ($MOP_{(x)}, \mathcal{N}, \lambda, T$)
2: Generate initial population $\mathbb{P}$ of size $\mathcal{N}$ randomly
3: Evaluate each member of $\mathbb{P}$
4: Evaluate each member of $\mathbb{EP}$
5: Compute T closest neighbours of each weight vector λ^i set $B(i) = \{i_1, ..., i_T\}$
6: **repeat**
7: for each sub-problem $i = 1, ..., \mathcal{N}$ do
8: Randomly select mating parents from $B(i)$
9: Generate offspring y' by using genetic operators
10: Apply heuristics on to produce y'
11: **for** each j neighbour in $B(i)$
12: **if** x^i is worse than y' regarding to fitness function
13: replace x^i by y'
14: **end if**
15: **end for**
16: Update EP
17: **end for**
18: **until** termination criteria is met
19: **end procedure**

3.1 Multi-Objective Knapsack Problem

The knapsack problem is a combinatorial optimization problem with a group of items. Each item has a weight and a value. The number of items to be set in a collection is determined in order that the net weight is less than or equal to some boundary and also the net value is maximized.

The 0-1 Multi-Objective Knapsack problem is mathematically formulated in equation, with a set of items, n and a set of m knapsacks, with:
P_{ij} >= 0 the profit of item j in knapsack,
w_{ij} >= 0 the weight of item j in knapsack,
ci >= 0 capacity of the knapsack i,

the 0 - 1 multi-objective knapsack problem can be stated as follows: Maximize

$$f_i(x) = \sum_{j=0}^{n} p_{ij} x_j. \quad i = 1....m$$

$$\sum_{j=0}^{n} w_{ij} x_j \leq x_i. \quad i = 1...m$$

$$x = (x_1...x_n) \in \{1.0\}^n$$

x_i = 1 means that item i is selected and inserted into all the knapsacks. The 0 - 1 multi-objective knapsack problem is classified as an NP-hard problem and can be used to model different forms of applications in resource allocation. When m = 1, it is reduced to the 0-1 single knapsack problem

4. Methodology

The target of our research is to solve the Multi-objective Optimization problem by adding a ML component to existent solutions and compare the results. The enhancement added by the study is to show that Machine Learning gives better results solving the optimization problems. Evolutionary multi-objective algorithms optimize many objective functions (two or more) simultaneously which usually have a conflict between each other. The target from this algorithm is to get the set of Pareto optimal solutions as an approximation. The set of solutions will seize the balance among the objective functions.

First, the 0/1 knapsack problem is used as a represent for the optimization problem. It is implemented using decomposition technique with the following parameters are:
- Decomposition technique, the Tchebycheff.
- Number of final solutions,
- Number of total evaluations conducted in current run.
- Mutation rate (set to 0.9).
- Dimension of decision space (set to 250, 500 and 750).
- Dimension of objective space (set to 2).
- Population Size (set to 150, 250 and 300).
- Number of maximum function evaluations in each run (set to 500 * Population Size).
- Neighborhood Size (set to 15 and 30).

4.1 Research Design

The research design shows how the study goes with steps of experiment. The main and first step is to choose the research approach and test its degree of benefit to the study goal. Research design has many tasks and decisions inside it. This study shows the development of research phases, as it used one method for multi-objective optimization and added an enhancement of using ML The first task is the study of previous or related research of the same topic, so we can add enhancement to the latest results in solving optimization problems. It's important to know the valid choices of the number of the objectives for real life optimization issues. There is a dire need for understanding the evolutionary optimization and estimate the complexity for having many objectives to optimize. The other design used is the understanding of how machine learning can enhance the optimization process and what the best technique to choose among different machine learning methods. The practical work must revise or discuss the effects of changing parameters of multi-objective optimization, the parameters of decomposition technique and the parameters of machine learning methodology as well.

4.2 Performance Metrics

This research attempts to recognize key areas where performance measurement can be expected to generate benefit to improve research goals and develop a promising method for solving optimization issues. Then, the decomposition technique is obviously better in terms of accuracy and processing time. Finally, adding machine learning to the multi-objective optimization process proves to show completely accurate results.

Inverted Generational Distance (IGD) is a different method of measuring performance [2]. It is a convergence metric that compares the non-dominated solutions found so far with each element of the Pareto-optimal front. Thus, the IGD metric uses the Pareto-optimal front as a reference. The IGD from the Pareto-optimal front PF* the non-dominated
solutions set P found so far is defined as:

$$IGD\,(PF^*, P\,) = \frac{\sum_{x\ \in P} d(x,\ P)}{|PF^*|}$$

Where *d(x,P)* is the minimum Euclidean distance between x and each element in P. As long as the IGD value is mall. The solutions are more close to the Pareto-optimal front; for example, IGD= 0 leads to the result that the solutions generated are in the Pareto-optimal front. Likewise, the higher the value of IGD obtained, the farther the set of non-dominated solutions is from the Pareto-optimal front.

5. Experiments and Results

The optimization problem adopted by the algorithm is 0/1 multi-knapsack problem. A text file with knapsack problem details. The file has items with their weights and the values of the items in the problem. These items are selected in a true/false manner, that it they are taken or not taken and can not be partially included. The problem has been divided into
sub-problems using Tchebyche_ method as a decomposition technique. Tchebyche_ method tries to eliminate weak or non-dominated points from the optimization problem.

The sub-problems generated are then pass through the next stage, which is genetic algorithm. Crossover and mutation are the variants of the GA. Random Crossover is implemented. Crossover generates a new solutions where the parents are selected from the neighborhood. Machine learning is then applied to the same knapsack problems. Random forest supervised machine learning method is used in the experiment. It first randomly selects some items from

the dataset. Then it builds a decision tree to switch between solutions depending on their rank or votes.

5.1 Datasets

At level one, all the focus was only on whether the implementation was right or wrong because its accuracy gives an indication of the extent of his understanding of the subject. Thus, a group of datasets, with their optimal solutions were used with the application of a simple GA to the single objective knapsack problem. The purpose of using those datasets is to test how accurately the GA is applied to this problem. At level two, the program and the algorithm were different from those in the previous level in that multiple objectives were included. Therefore, a suitable method was needed for testing and evaluating these multiple objectives. The 0-1 multi-objective knapsack problem, which is NP-hard, required special test problems that are suited for it. Ziztler and Theile [3] have proposed a dataset for multi-objective heuristics, and this dataset is commonly used in the EC community. This dataset consists of nine test instances and is widely used for multi-objective knapsack problems.

5.2 Implementation Environment

Python is an open-source computer programming language that is one of the most powerful languages a programmer can use for algorithm development, data visualization, data analysis, and numeric computation. It is widely used to perform engineering and scientific calculations and provides a very extensive library of predefined functions. Most of the applied ML methods used implementations from the Python library scikit-learn. Scikit-learn is an open source machine learning library. It focuses on modelling data and it has predefined algorithms. It provides multiple supervised and unsupervised learning algorithms. Scikit-learn was used with version 3.6 of Python. Thus, the implementation environment will be python. The python program and language will be used to implement and test both the MOEA/D algorithm with the 0-1 multi-objective knapsack problem, and MOEA/D with machine learning.

In order to conduct the experiments, a computer with the following specifications will be used:

- Processor: Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz 2.60 GHz
- Installed memory (RAM): 16.00 GB
- System type: 64-bit Windows 7 Professional

5.3 Results of MOEA/D algorithm

In this experiment, we run the MOEA/D algorithm on the knapsacks problem. It was ran 20-times independently for each 2 objective 250 / 500 /750 items in the MOKP test instance in the process of our experiments. Results are reported in the following tables:

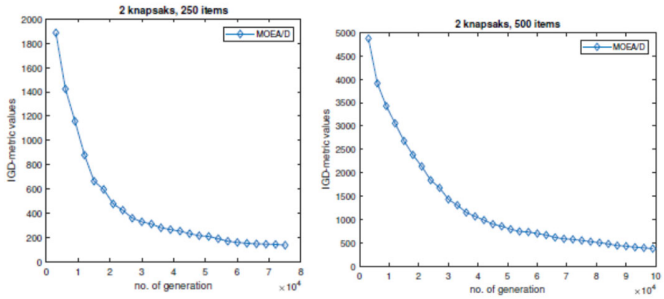
Table 1: IGD metric and run time found by MOEA/D for all the MOKP test instances.

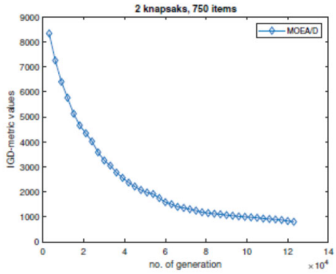
	2-250		2-500		2-750	
	lowest IGD	RT*	lowest IGD	RT*	lowest IGD	RT*
1	143.2006249	201.8944277	444.9408204	318.2083773	944.2398876	609.2434903
2	155.1428247	187.960495	481.8408368	171.9957264	901.6378706	527.4823178
3	157.716762	253.7571235	444.8975882	177.0137358	858.8790042	571.3033804
4	147.9711861	248.9304293	447.6447516	468.516053	849.5828548	705.5217265
5	159.0003191	251.2895554	472.5432384	324.4301339	908.6542245	931.8877485
6	158.7753283	248.8463048	561.120649	533.7609583	889.3826197	944.3672027
7	156.0972566	247.4349942	447.779603	448.2261818	820.1604159	902.8898198
8	181.2698678	246.692415	444.1803696	189.0027585	845.6271408	952.4368346
9	134.9092642	245.1413816	474.6847402	165.2754714	854.8705539	790.5552612
10	168.4431879	247.0970941	575.2301124	355.1320266	917.72472	572.667658
11	169.2604951	247.763091	420.323868	179.2984314	847.9712794	612.856467
12	153.2745276	228.903003	529.432648	194.4231432	941.7161077	602.8994285
13	174.212551	194.5701541	374.4457554	258.6038472	883.5483534	682.1701841
14	157.4169822	190.6685123	540.8662868	225.7145749	817.9117598	342.1558963
15	172.4273628	219.1269798	444.8442634	352.7281141	815.8722525	317.6504582
16	161.1722828	215.6829311	558.2918386	199.0053868	891.8207088	372.2840758
17	197.3757336	195.4597544	443.5074122	181.4257774	784.326483	484.2175806
18	184.5743399	117.2949106	478.2369113	181.0042239	835.1045496	321.9318567
19	171.9516227	200.2353647	523.2851641	193.9720748	848.3065247	338.8102819
20	142.3343938	125.301109	495.0539535	189.1152247	960.1851116	314.1627872

Table 2: The mean and min IGD metric and standard deviation values of the non-dominated solutions found by MOEA/D for all the MOKP test instances

Decomposition Method			MOEA/D		
	M	N	mean	min	Std
Instances	2	250	163.32	132.60	20.38
	2	500	461.22	414.04	29.96
	2	750	872.75	748.18	67.90

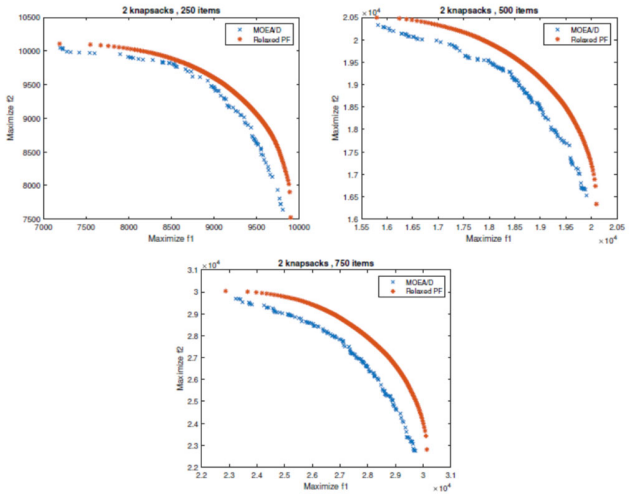
The following figures shows plotting the evolution of the average IGD metric in MOEA/D for 2 objective 250/500/750 items in the MOKP test instances.





Figs.1,2,3 The evolution of the average IGD metric in MOEA/D for 2 objective 250/500/750 items in the MOKP test instances

In the following figures, plotting the non-dominated solutions with the lowest IGD metric values in 20 runs of the MOEA/D in 2 objective MOKP test instances.



Figs 4,5,6: Plot of non-dominated solutions with the lowest IGD metric values in 20 runs of the MOEA/D in 2 objective MOKP test instances with pareto Front PF for each test instances.

5.4 Results of MOEA/D + Machine Learning algorithm

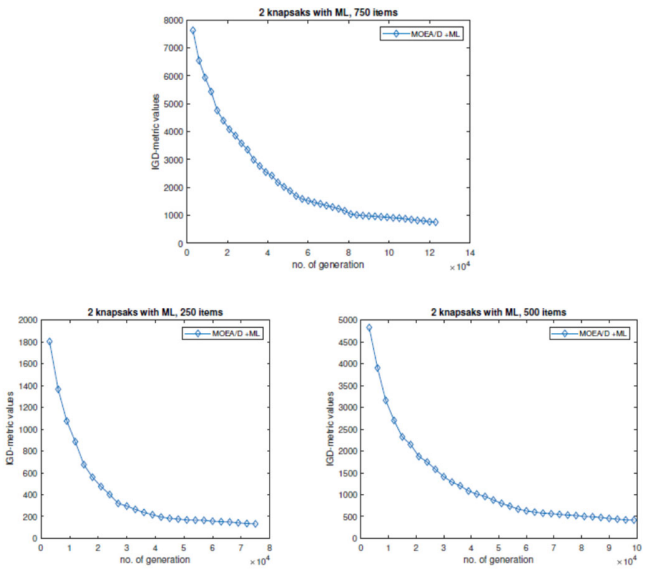
This section presents the results of this paper contribution on multi-objective evolutionary algorithm based on decomposition (MOEA/D). The approach is to add a ML component to previous MOEA/D algorithm. Specifically, a supervised ML algorithm which is the Random Forest algorithm. The learned model constitutes the knowledge which can be then utilized to guide the evolution process within MOEA/D. Simulation results on knapsacks problem are presented. As in the previous experiment where the results of 20 runs were presented. The same is preformed with MOEAD/D+ML solution, this algorithm run 20-times independently for each 2 objective 250 / 500 /750 items in the MOKP test instance in the process of experiments. Table 3 shows the IGD metric and standard deviation values of the nondominated solutions found by MOEA/D+ML for all the MOKP test instances.

Table 3: IGD metric and run time found by MOEA/D + ML for all the MOKP test instances

	2-250 + ML		2-500 + ML		2-750 + ML	
	lowest IGD	RT*	lowest IGD	RT*	lowest IGD	RT*
1	180.6429042	1248.48507	467.7318293	3703.760869	898.0607131	4699.014122
2	161.1662875	2233.147216	439.2102006	4000.437846	850.2706119	4564.377774
3	171.0427098	2319.773402	504.6544971	4319.613534	777.4776097	4589.143422
4	138.6473875	2224.055068	475.2414086	4917.155617	903.183063	6877.553538
5	220.0005072	2241.283895	482.9014256	5032.678095	772.3979987	4054.005288
6	154.7929568	2269.654235	454.1179112	3227.720694	841.6711133	7332.034636
7	148.1262243	1246.020272	454.7149089	3836.878179	819.9176764	8696.820183
8	157.1213696	2505.072016	434.3179699	3183.250341	836.7562743	8629.511545
9	153.9700268	2776.413521	434.3179699	3183.250341	988.2751071	13710.54394
10	132.6049534	2843.268583	487.4828188	2926.776765	914.8562759	4702.511763
11	178.8079799	2578.600995	488.8678765	3218.921171	801.1633573	5687.850672
12	190.9437521	2579.117725	414.4824084	6323.367418	853.1221558	6114.823309
13	154.8621529	2555.215304	489.4826039	5251.212206	900.0366953	6765.169408
14	143.6733459	3490.889297	492.2631043	6510.884768	892.3846692	6692.699329
15	165.1234287	3505.944977	489.167529	6487.367152	834.6932421	10286.77684
16	163.8438069	2065.770016	499.3469834	4523.063763	748.1839264	10376.73166
17	139.9731046	2056.014921	414.0480587	4916.012222	907.3283584	10822.91158
18	178.3777268	2403.198405	415.2555003	4828.487228	924.8551125	10861.91312
19	172.8037285	2509.225351	447.6061161	5069.401755	997.6432935	11039.41574
20	159.9642691	3118.129014	444.3151069	5035.793481	930.13577	11088.0634

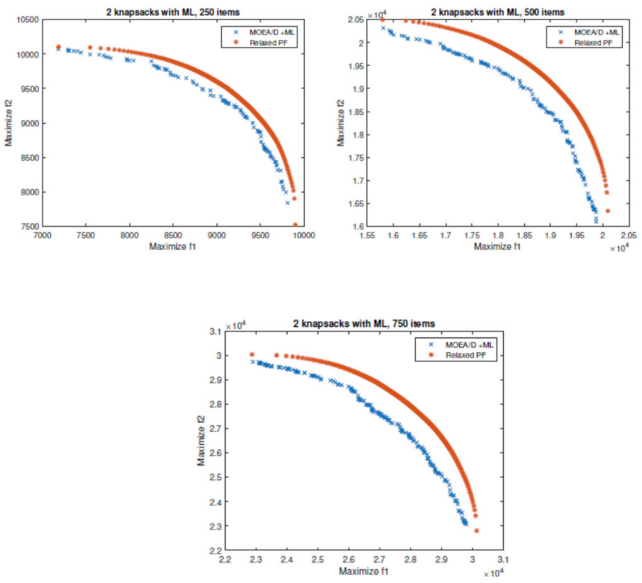
Table 4: The mean and min IGD metric and standard deviation values of the non-dominated solutions found by MOEA/D+ML for all the MOKP test instances

Decomposition Method			MOEA/D + ML		
	M	N	mean	min	Std
Instances	2	250	163.32	132.60	20.38
	2	500	461.22	414.04	29.96
	2	750	872.75	784.18	67.90



Figs 7,8,9: The evolution of the average IGD metric in MOEA/D for 2 objective 250/500/750 items in the MOKP test instances MOEA/D + ML

For this experiment, plotting the non-dominated solutions with the lowest IGD metric values in 20 runs of the MOEA/D + ML in 2 objective MOKP test instances.

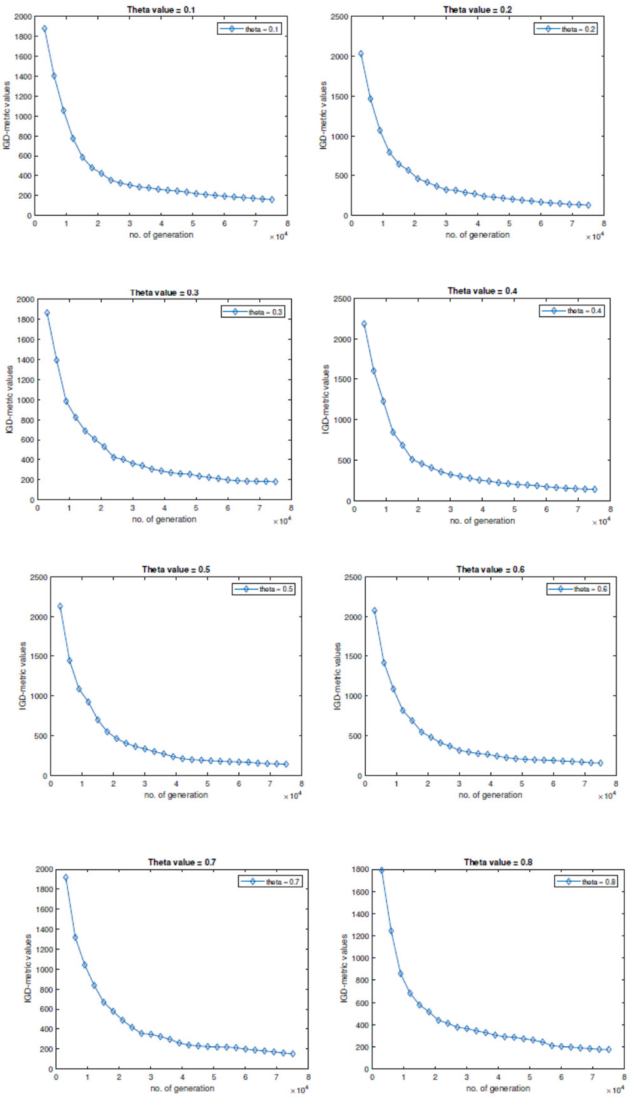


Figs 10,11,12: Plot of non-dominated solutions with the lowest IGD metric values in 20 runs of the MOEA/D + ML in 2 objective MOKP test instances.

Another experiment was conducted with varying the training data size to show its impact. MOEA/D+ML algorithm was trained using nine different sizes of training data set (theta = 0.05, theta = 0.10, ..., theta = 0.80, and theta = 0.9). For each size data, the lowest IGD metric value is reported in table 5 and Figure 13.

Table 5: Theta probability

Theta	IGD	Running time
theta =.05	169.817	3229.082885
theta = 0.1	158.9603	3224.263185
theta = 0.2	**128.493**	2145.154742
theta = 0.3	180.0953	2621.279783
theta = 0.4	139.0953	2169.789729
theta = 0.5	138.9911	3054.430382
theta = 0.6	153.6612	2397.672321
theta = 0.7	152.4231	2449.853865
theta = 0.8	175.7541	2329.893226
theta = 0.9	175.7541	2329.893226



Figs 13-20: (a) Theta probability 0.1 (b) Theta probability 0.2 (c) Theta probability 0.3 (d) Theta probability 0.4 (e) Theta probability 0.5 (f) Theta probability 0.6 (g) Theta probability 0.7 (h) Theta probability 0.8

6. Discussions

The IGD metric was used to assess the performance of the algorithm. In many cases, the Pareto-optimal front is unknown. In such instances, an upper approximation for the Pareto-optimal front is used. For the nine test instances of the 0-1 multi-objective knapsack problem, there exist a very good upper approximation. This upper approximation was produced by Jaszkiewicz. His upper approximation has 202 points for two objective

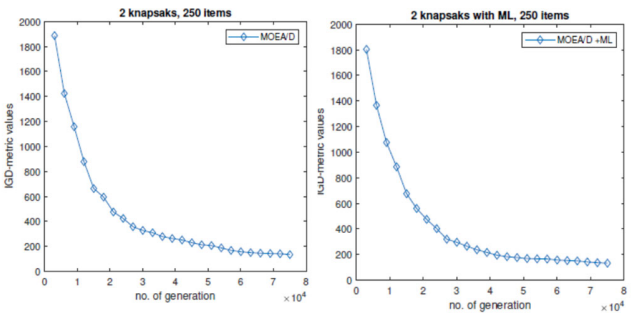
instances. Table 6 shows the mean and the minimum values of the IGD metric in the MOEA/D and MOEA/D with machine learning for all the 0-1multi-objective knapsack problem test instances. It also presents the standard deviation values.

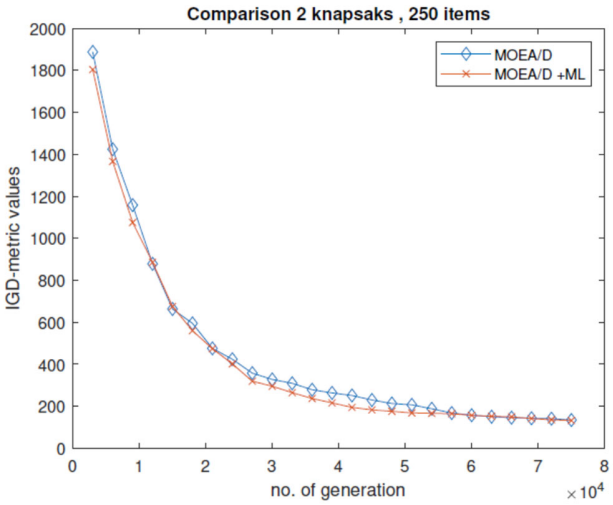
As can be seen in table 6, Among both multi-objective EAs, MOEA/D+ML algorithm seems to provide the best performance for the 2 objective instance with 250 items and 750 items. The minimum IGD values for each test problem with MOEA/D+ML are less than the corresponding IGD values of the MOEA/D algorithm. However, The minimum IGD value for the two objective instance with 500 items obtained with MOEA/D is less than the corresponding IGD values of the MOEA/D+ML algorithm.

Table 6: IGD metric and standard deviation values of the non-dominated solutions found by MOEA/D and MOEA/D+ML for all the MOKP test instances

Decomposition Method			MOEA/D			MOEA/D + ML		
Instances	M	N	mean	min	Std	mean	min	Std
	2	250	162.32	134.9	15.33	163.32	**132.6**	20.38
	2	500	473.61	**374.44**	52.92	461.22	414.04	29.96
	2	750	856.87	784.32	47.89	872.75	**748.18**	67.9

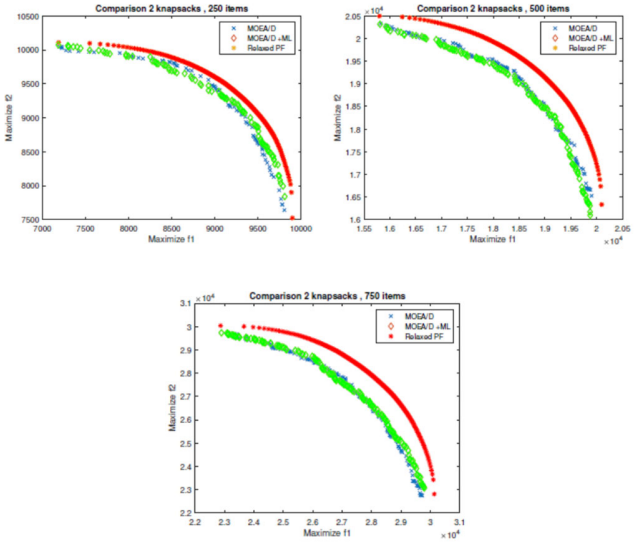
In order to examine the convergence of both algorithms (MOEAD vs MOEAD+ML), the mean IGD metric values for both compared methods (20 independent runs) for the 2 objective instance with 250 items, 500 items, and 750 items are plotted in Figures 21, 22, and 23. These figures show respectively the evolution of the average IGD metric in MOEA/D for 2 objective 250, 500, and 750 items in the MOKP test instances (a) MOEA/D (b) MOEA/D + ML (c) comparison MOEA/Dvs MOEA/D + ML. It can be observed from these figures that the curves of the mean IGD values obtained by MOEA/D+ML reach the lowest positions with the fastest searching speed on 2-250 and 2-750 items. The promising convergence speed of the proposed MOEA/D-ML might be attributed to the adaptive strategy used when applying random forest algorithm.





Figs 21,22,23: The evolution of the average IGD metric in MOEA/D for 2 objective 250 items in the MOKP test instances (a) MOEA/D (b) MOEA/D + ML (c) comparison MOEA/D vs MOEA/D + ML

The results using the PF obtained by both algorithms are shown in figure 24,25 and 26. MOEA/D+ML performs the best in 2-objective optimization problems (250 items and 750 items). The ML enhanced MOEA/D is able to obtain better estimation of Pareto fronts. It behaves better than MOEA/D. But for the 2-objective optimization problem (500 items), MOEA/D slightly outperforms the MOEA/D+ML algorithm.



Figs 24,25,26: Plot of non-dominated solutions with the lowest IGD metric values in 20 runs of the MOEA/D and MOEA/D +ML and showing the comparison for all test instance (a) comparison 2 objective 250 items (b) comparison 2 objective 500 items (c) comparison 2 objective 750 items

To summarize, machine learning techniques can help the EA algorithms search more effectively and efficiently, they also add to computational burden. There is a tradeoff between this benefits and the computational costs. Better understanding and improving cooperation between ML and EC will play a significant role in enhancing EC algorithms with ML techniques efficiently.

7. Conclusion and Future Work

Research in this study worked on solving multi-objective optimization problem. Knapsack problem is a popular example of an optimization. In 0/1 knapsack problem, the items to be optimized can't be partitioned, it's taken or not taken. Using decomposition with Multi-objective optimization makes it easier to solve. Tchebyche_ is the decomposition techniques used in the research.

Genetic Algorithm is an optimization method that imitate the natural evolution process. It's based on population search. There are many applications that Genetic Algorithms are suitable for them. Decision making problems are a popular example of such applications. Optimization, Machine Learning, Robotics and Search problems are other types of applications for which Genetic Algorithms may have proper solutions. Genetic Algorithm has been used to solve each sub-problem generated from the decomposition step. Crossover and mutation are the variants or the operators of the genetic algorithm. In the research, the crossover is random.

Artificial Intelligence, Machine Learning, Evolutionary Multi-Objective Optimization and Genetic Algorithm were the main topic of search in the thesis. It's vital to show the solution of the optimization problem as a search space to solve it. Also, it's needed to design specific operators for the search process so that they generate new solutions as candidates based on the solution presented. Computational complexity of exploring the new solutions reduces as the accuracy of the cost function increases. The techniques of Machine Learning have promising capabilities in the integration with MOEA framework; yet this integration is still in its early stages. Machine Learning supports the solution for Evolutionary Optimization Problems. Machine Learning gives better population diversity and more accuracy. The used ML technique here is Random forest.

Supervised Random Forest has been used for classification problem mainly. It's random as it selects random samples from the given dataset used for the training. The forest comes from the fact that it creates decision trees on the data samples. Then the algorithm form the prediction from each of these trees. Finally it selects the best solution by voting.

This research work has laid down the framework for different operations of Genetic Algorithms and also the hybridization of GA with other search techniques. This paper can be groundwork for further research in this area. Some of the possible directions for future research are highlighted here:

- Use some other techniques like Population Initialization, Population Reproduction and Variation, Algorithm Adaptation and Local Search
- This paper focuses on using ML for Fitness Evaluation and Selection particularly in Reducing Number of Function Evaluations, other direction can be done for Modeling Objective Function

References

[1] Q. Zhang and H. Li, ―MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decomposition,‖ IEEE Trans. Evol. Comput., vol. 11, no. 6, pp. 712–731, Dec. 2007.
[2] M. Sierra and C. Coello (2005). ―Improving PSO-based Multi-objective Optimization using Crowding, Mutation and Epsilon-Dominance.‖ In Third International Conference on Evolutionary Multi-Criterion Optimization, pp. 505-519
[3] Kassu Jilcha Sileyew (August 7th 2019). Research Design and Methodology [Online First], IntechOpen, DOI: 10.5772/intechopen.85731.

Rakan Alotaibi received the B.E. and M.E. degrees, from umm al qura university in 2009and 2020, respectively. In 2009. working as a programing, a Business Analyst (from 2012) in the Dept. of Demand Management , Holy Makkah Municipality, and a Software Project Manager in IT (from 2016), he has been a Digital Transformation Consultant to the Makkah mayor general secretary / Vice CIO and Director of IT Demand Management. Since 2019. His research interest includes Artificial intelligence, Algorithms and machine learning.