Demonstrating Some Modern Multi-Criteria Decision Making Methods Under Uncertainty & Their Python Libraries

Sina Azartash SAZARTA1@JH.EDU
Whiting School of Engineering
Johns Hopkins University

Abstract

This paper presents some modern strategies for using artificial intelligence methods to make optimal decisions. We focus on Multi-Criteria Decision Analysis(MCDA) to aid optimal decision making with many factors, uncertainties, and conflicting components. Specifically, we demonstrate the python implementation of the MCDA algorithms: TOPSIS, VIKOR, PROMETHEE-II, SPOTIS, and COMET using the pyMCDM python library. These algorithms are run on two datasets and then compared against each other in their ability to recommend the optimal choices. The algorithms search through instances in the dataset as alternatives and find the optimal alternative representing the most preferred vehicle. The top 5 vehicles in a dataset of hundreds of vehicles were chosen. The framework for these algorithms are explained and some advanced variations are briefly introduced. Later, we dissent the implications of our research and discuss the merits of our experimental design. Lastly, we make some suggestions for further experimentation investigating the effect of more advanced variants and the involvement of evolutionary methods.

I. Introduction

The task of finding optimal decisions given multiple variables is often referred to as Multi-Criteria Decision-Making (MCDM) and analyzing those decisions is referred to as MCDA [1–3]. Certainly, making optimal decisions has the potential to be incredibly powerful and lucrative for any organization. Professionals can choose the decisions with the best combination of several conflicting variables and unknowns that consistently have better returns than other decisions in the long-term. Unfortunately, even with the power of machine learning, this is no easy task. Modeling decision making in the real world can be highly complex, computationally expensive, and the result is often still an approximation. In this paper, we introduce some cutting edge methods for simple as well as complex optimal decision making problems.

Some foundational topics and nomenclature are introduced surrounding MCDA and illustrated in *Figure 1*. The word ‘criteria’ in MCDA refers to variables involved in making a decision, which are also interchangeably called objectives, factors, or goals. MCDA aims to find a set of decisions which outperform all other decisions or are at least equally good. Decisions are interchangeably referred to as alternatives and solutions [1–3]. The name for this best set of decisions is called the pareto optimal front. A single decision in the pareto optimal front is referred to as pareto optimal. When decision ‘A’ is more optimal than decision ‘B’, decision A is said to dominate decision B. Pareto-Optimality is analogous to the concept of an admissible decision function. Admissible decision functions aim to minimize the risk

associated with making the wrong decision [4]. E-admissible decisions aim to find a decision that maximizes expected utility for at least one probability distribution in a set of multiple probability distributions. However in this study, we focus on the optimal decisions that on average dominate other decisions with respect to all the probability distributions of the criteria involved. In that case where a specific objective in a set of objectives must be maximized, then e-admissibility calculations would be appropriate.

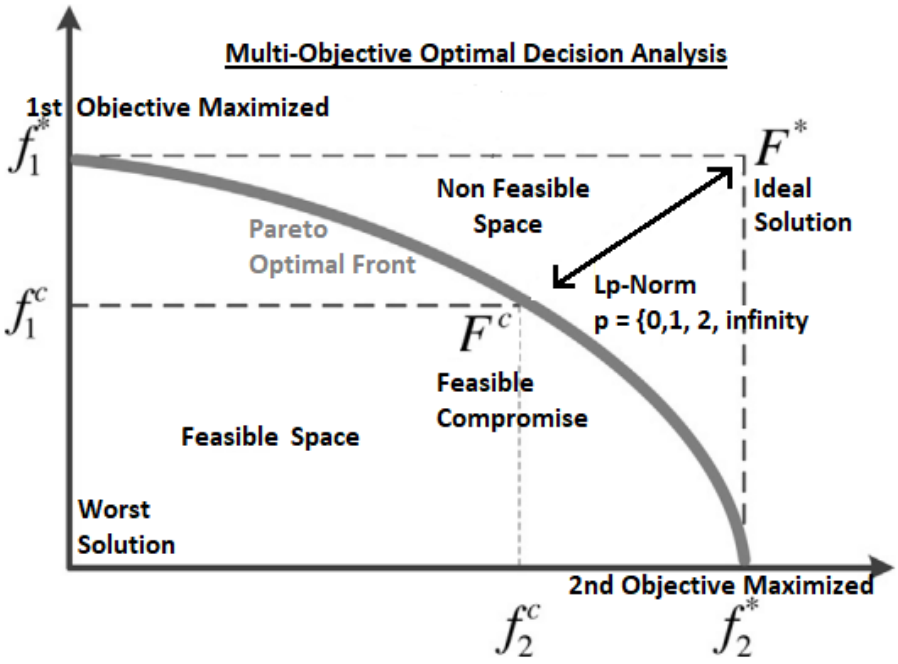


Figure 1 - demonstration of 2 - objective optimization

Finding a set of optimal decisions can be challenging for a variety of reasons. The most straightforward problem occurs when there is not enough data. One type of objective can be underrepresented or there may be too many alternatives for the necessary amount of examples. Also, the constraints of the true problem might be unknown. In other words, the data points may not properly represent enough regions of the decision space and also over or under represent specific areas. Another problem is that the objectives can be conflicting and thus require complex tradeoffs. In addition, each objective can have a different weight of importance on the final optimality. For example, when comparing car models, safety rating would likely be weighted higher than the stylishness of the car. This segues into the next problem, there are many methods of comparing safety rating to stylishness both of which are subjective to quantitative conversions. Both values will have to be normalized and the method of normalization may influence results [1,2,5] . Furthermore, since the model may not capture all factors involved in a decision, the true objective function weights might fluctuate due to confounding variables and unknowns. To add more complexity, objectives can interact with each other making it difficult to attribute the reason why a set of decisions is more effective than others. Either sets of objectives can influence one objective, one objective can influence many objectives, or a set of objectives can influence a set of other objectives. Moreover, as the number of factors increases in a decision, the computational time will increase greatly. For such problems, finding the true optimal set of decisions will be too slow, therefore the set of decisions are approximated. Another noteworthy problem is the rank reversal paradox. The rank reversal paradox occurs when the ranking of the most optimal alternatives is paradoxically influenced by alternatives that are not even close to being optimal. There are four main variations of this. Copying or replacing a non-optimal solution can influence the ranking of the optimal solutions as the first two variations. Ranking the alternatives in subsets and then combining them into the main set may not produce the same ranking as the main set. In the last variation, different sizes and combinations of subsets may produce

directly conflicting rankings of the alternatives [6,7]. All these problems make it difficult to tell if the results of the MCDA method is actually correct.

TOPSIS, VIKOR and COMET are some of the more common MCDA methods. The best method to use depends on the industry domain and type of problem. Each of these methods has multiple variants, but only the main one is discussed here. The Technique for Order of Preferences by Similarity to Ideal Solution (TOPSIS) and VlseKriterijumska Optimizacija I Kompromisno Resenje (VIKOR) are both aggregation methods that calculate an ideal point and then find the closest feasible point as illustrated by *Figure 1*. Briefly, the protocol involves first weighing the criteria and making a decision matrix, the matrix is normalized and each alternative is weighted, the ideal solution is the intersection of each maximum value of each objective, then the Lp-Norm from each alternative to the ideal solution is calculated; the alternatives closest to ideal point are ranked higher [2]. TOPSIS uses euclidean distance, the L2-Norm while VIKOR uses L1 and L-infinity Norm [5]. The Characteristic Objects Method (COMET) . Remarkably, the COMET is resistant to the rank reversal paradox. Briefly, the process involves coding objectives as characteristic objects using fuzzy set theory. The characteristic objects are compared and then each alternative is evaluated as the fuzzy process is reversed back [8]. There are many additional MCDA methods that are not discussed in this paper, but should be made aware of, these are: COmplex PRoportional ASsessment (COPRAS), Stable Preference Ordering Towards Ideal Solution (SPOTIS), Additive Ratio ASsessment (ARAS), COmbined COmpromise SOlution (COCOSO), COmbinative Distance-based ASsessment (CODAS), Evaluation based on Distance from Average Solution (EDAS), Multi-Attributive Border Approximation area Comparison (MABAC), MultiAttributive Ideal-Real Comparative Analysis (MAIRCA), Measurement Alternatives and Ranking according to COmpromise Solution (MARCOS), Operational Competitiveness Ratings (OCRA), Multi-Objective Optimization Method by Ratio Analysis (MOORA), Preference Ranking Organization Method for Enrichment of Evaluations (PROMETHEE), a full Multiplicative form of MOO by Ratio Analysis (MULTIMOORA), and Elimination et choix traduisant la realité (ELECTRE) [5]. Generally, the variations of these algorithms are explained by different weighting methods, different normalization methods, and the use of different statistical correlation coefficients.

Evolutionary algorithms handle some of these challenges through intelligent approximation. Multi-Objective Optimization (MOO) is a family of algorithmic techniques that simultaneously optimize conflicting objectives with reduced time complexity. A parameter represents a value for a single objective function and a set of parameter values represents a solution for the MOO problem. MOO problems on large datasets typically can only handle up to 3 objectives due to increased computation time. When more than 3 objectives are being considered, this method is called Many Objective Optimization (MaOO) which requires a different approach. One of the main approaches is the use of Evolutionary Algorithms which when combined with MOO are called Evolutionary Multi-Objective Optimization (EMO) and Evolutionary Many-Objective Optimization (EMaO). The framework of these algorithms generally involves randomly predicting a set of parameters for the objective functions, comparing the parameters to the best solution and then also making future predictions more intelligent overtime in an evolutionary process. In the evolutionary process, combinations of parameters are generally treated as individuals with genetic material and they can be mutated and exchanged. If a solution provided is less optimal, then future solutions will pick values that are farther away from the current solution. If the solution improves optimality, then solutions will more likely be proposed that are in the vicinity of the current solution.

Some notable EMO algorithms are the Strength Pareto Evolutionary Algorithm (SPEA), SPEA2, Non-dominated Sorting Genetic Algorithm (NSGA), NSGA-II, and Pareto Archived Evolution Strategy (PAES) [9]. Some EMaO algorithms include Approximation-Guided Evolutionary (AGE-II), (HypE) Hypervolume Estimation Algorithm and Adaptive NSGA-III (A-NSGA-III). EMO and EMaO algorithms vary in their ability to: reliably to converge to a set of solutions, produce approximations that are close to the true Pareto-optimal front, and produce diverse solutions that properly represent the entire optimization space.

In this paper, we demonstrate some of the MCDA, and EMO methods through modern python libraries. We showcase the python libraries capabilities and evaluate them. Section II contains a description of the Python Libraries and the designing of the experiments, Sections III shows the results of the executed python code as visualizations including bar graphs, Section IV contains a discussion of the results, and in Section V we conclude with some direction for further research.

II. Method

A. Dataset

Two datasets were used. Two datasets were from the UCI Machine Learning Repository and the other one was a series of random arbitrary mathematical functions. The UCI datasets were used for MCDA algorithms. The datasets can be found by going to the UCI DataSet Repository homepage. The dataset from the machine learning repository was Auto-mpg and Car Evaluation. Auto-mpg contains 398 instances and 8 attributes of car specifications. The attributes are miles per gallon (mpg), number of cylinders, horsepower, weight of the vehicle, acceleration, the year made, the origin, and the car brand name. The year made had values running from 1970 to 1982. Car Evaluation contains 1728 and 6 attributes. The attributes are buying price, maintenance cost, number of doors, capacity of people, luggage capacity size, safety rating and evaluation classification. Since some attributes were not relevant to MCDM, of the 8 attributes for Auto-mpg, only 6 attributes were used for MCDA. The origin and car name brand were not included in the study. Similarly, for the Car Evaluation file, only 4 out of the 6 attributes were used. The number of doors and persons were not included because that was reasoned to be a matter of choice. Evaluation was also not included because that variable was based on the other attributes. Including this term in the analysis would have introduced strong interaction terms which give extra unneeded to specific columns. Finally, the Auto-mpg dataset had numerical data and the Car Evaluation dataset had categorical data. Car Evaluation categorical data had an ordinal relationships including small to big and low to very high. The data was converted into floating points via ordinal encoding.

B. Preprocessing

The complete set of preprocessing actions can be found in the Azartash_Sina_743_Research_Code.ipynb file, a python notebook file. Rows of the datasets represented alternative decisions. Columns of the dataset represented factors. The goal of MCDM algorithms was to find which alternatives were the best. That is, among several attributes of a car, we wanted to find out which car matched our preferences the most optimally .The objective of these datasets was to showcase the MCDA algorithms so we only needed one type of each alternative. For this reason, duplicate rows were deleted. Duplicate alternative decisions were not needed. The remaining size of the dataset was 397 for auto-mpg (1 instance removed), and the remaining size of Car Evaluation was 1628 (100 removed). The Auto-mpg file had 6 missing

values for the attribute horsepower. Instead of discarding these rows, a K-Nearest Neighbor Imputer was used. Null values for both datasets were checked. Both datasets were converted into float values to be used as a numpy matrix. Later these matrices were converted into a Pandas DataFrame. Both datasets were normalized using maximum normalization (each value was divided by the maximum value in its column). Several other normalization methods were experimented on: logarithmic normalization, vector normalization, mean normalization, linear normalization and vector normalization. We decided to go with max_matrix because linear_matrix for the datasets was shown above to be equivalent. The lograthmic_normalization caused an overflow and min_max matrix resulted in loss of information. The values for vector normalization resulted in very small decimals. The values for mean normalization resulted in many positive and negative values which would have been troublesome.

C. Algorithms Implementation

MCDA was based on maximizing some attributes and achieving average on others in unique combinations. The goal was to maximize each attribute as much as possible in a clever combination. For the Auto-mpg dataset, we aimed to minimize buy price and maintenance cost; we aimed to maximize luggage capacity and safety rating. For the Car Evaluation dataset, we aimed to minimize the weight, but maximize mpg, number of cylinders, horsepower, acceleration and year made. In this study we preferred to find cars that are made at a later date. Each dataset also had a set of weights associated with it. We weighted the attributes of the datasets because each attribute has a different level of importance associated with it. For the Auto-mpg dataset, we wanted to find the optimum economy car. The weight of buying cost was set to 75% importance, maintenance costs were set to 85% importance, luggage capacity was set to 50%, and safety rating was set to 100%. For the Car Evaluation dataset, we wanted to find the optimal race car. The weight of miles per gallons was set to 33% importance, the number of cylinders was set to 75%, horsepower was set to 90%, weight was set to 80%, acceleration was 100%, and the year made was 33%. Finally, In order to run the experiment, Python version 3.97 was used. Python implementations of TOPSIS, VIKOR, PROMETHEE-II, SPOTIS, and COMET were implemented. The python library used for MCDA was called pyMCDM by Andrii Shekhovtsov. The library can be obtained by pip install pymcdm. ng For each algorithm, the top choices of the 97.5 quintile were presented as the best alternatives.

III. Results

Below, we present the MCDM algorithms' ability to present the most optimal vehicles to buy. Two tables are shown for each algorithm displaying optimal alternative choices. Each section represents an algorithm and for each section, the top picture is for Car Evaluation, which is identified by 4 original attributes. The bottom picture is based on the Auto-mpg dataset which is identified by 6 original attributes. Each table below has an extra attribute called preference. Preference quantifies how much we prefer this attribute compared to other alternatives. Preference is calculated by comparing it to the ideal point, a theoretical point in which all attributes are maximized. Each table contains the top 5 or less alternatives along with their preference value. An alternative must have a preference in the 97.5 percentile or above to be classified as a top alternative. Preference values are normalized by the maximum preference value. Therefore, a higher preference value indicates that the algorithm is performing better because our alternatives are becoming close to the theoretical ideal alternative.

A. TOPSIS

average preference = 0.8883

	buying	maint	luggage	safety	pref
1196	2.0	1.0	3.0	3.0	0.846855
1520	1.0	2.0	3.0	3.0	0.846855
1625	1.0	1.0	2.0	3.0	0.859592
1628	1.0	1.0	3.0	3.0	1.000000

Table 1: Top Alternatives presented by TOPSIS on Car.Data

average preference = 0.5513

	mpg	cyln	h_power	weight	accel	m_year	pref
13	14.0	8.0	225.0	3086.0	10.0	70.0	0.546121
25	10.0	8.0	215.0	4615.0	14.0	70.0	0.529641
26	10.0	8.0	200.0	4376.0	15.0	70.0	0.540139
27	11.0	8.0	210.0	4382.0	13.5	70.0	0.529198
28	9.0	8.0	193.0	4732.0	18.5	70.0	0.562891

Table 2: Top Alternatives presented by TOPSIS on Auto.Data

Best economy car: 1196, 1520, 1625, 1628
Best race car: 13, 25, 26. 27, 28

B. VIKOR

average preference = 0.8705

	buying	maint	luggage	safety	pref
1196	2.0	1.0	3.0	3.0	0.834677
1520	1.0	2.0	3.0	3.0	0.812634
1625	1.0	1.0	2.0	3.0	0.834677
1628	1.0	1.0	3.0	3.0	1.000000

Table 3: Top Alternatives presented by VIKOR on Car.Data

average preference = 0.8109

	mpg	cyln	h_power	weight	accel	m_year	pref
222	17.0	8.0	110.0	4060.0	19.0	77.0	0.784526
249	19.9	8.0	110.0	3365.0	15.5	78.0	0.785451
277	16.2	6.0	133.0	3410.0	15.8	78.0	0.751805
285	17.0	8.0	130.0	3840.0	15.4	79.0	0.808297
288	18.2	8.0	135.0	3830.0	15.2	79.0	0.806303

Table 4: Top Alternatives presented by VIKOR on Car.Data
Best economy car: 1196, 1520, 1625, 1628
Best race car: 222, 249, 277, 285, 288

C. PROMETHEE-II

average preference = 0.8712

	buying	maint	luggage	safety	pref
1196	2.0	1.0	3.0	3.0	0.818182
1520	1.0	2.0	3.0	3.0	0.818182
1625	1.0	1.0	2.0	3.0	0.848485
1628	1.0	1.0	3.0	3.0	1.000000

Table 5: Top Alternatives presented by PROMETHEE-II on Car.Data

average preference = 0.7137

	mpg	cyln	h_power	weight	accel	m_year	pref
222	17.0	8.0	110.0	4060.0	19.0	77.0	0.660535
252	19.2	6.0	105.0	3535.0	19.2	78.0	0.605134
298	23.0	8.0	125.0	3900.0	17.4	79.0	0.792241
300	23.9	8.0	90.0	3420.0	22.2	79.0	0.875452
347	37.0	4.0	65.0	1975.0	19.4	81.0	0.630368

Table 6: Top Alternatives presented by PROMETHEE-II on Auto.Data

Best economy car: 1196, 1520, 1625, 1628
Best race car: 222, 252, 298, 300, 347

D. SPOTIS

	buying	maint	luggage	safety	pref
1196	2.0	1.0	3.0	3.0	0.913978
1520	1.0	2.0	3.0	3.0	0.913978
1625	1.0	1.0	2.0	3.0	0.919355
1628	1.0	1.0	3.0	3.0	1.000000

Table 7: Top Alternatives presented by SPOTIS on Car.Data

```
average preference =  0.3580
```

	mpg	cyln	h_power	weight	accel	m_year	pref
13	14.0	8.0	225.0	3086.0	10.0	70.0	0.325966
262	19.2	8.0	145.0	3425.0	13.2	78.0	0.321577
288	18.2	8.0	135.0	3830.0	15.2	79.0	0.320486
298	23.0	8.0	125.0	3900.0	17.4	79.0	0.359820
300	23.9	8.0	90.0	3420.0	22.2	79.0	0.443834

Table 8: Top Alternatives presented by SPOTIS on Auto.Data

Best economy car: 1196, 1520, 1625, 1628
Best race car: 13, 262, 288, 298, 300

E. COMET

```
average preference =  0.9824
```

	buying	maint	luggage	safety	pref
1196	2.0	1.0	3.0	3.0	0.974359
1520	1.0	2.0	3.0	3.0	0.974359
1625	1.0	1.0	2.0	3.0	0.980769
1628	1.0	1.0	3.0	3.0	1.000000

Table 9: Top Alternatives presented by COMET on Car.Data

average preference = 0.6501

	mpg	cyln	h_power	weight	accel	m_year	pref
13	14.0	8.0	225.0	3086.0	10.0	70.0	0.654875
25	10.0	8.0	215.0	4615.0	14.0	70.0	0.611720
26	10.0	8.0	200.0	4376.0	15.0	70.0	0.626908
27	11.0	8.0	210.0	4382.0	13.5	70.0	0.607815
28	9.0	8.0	193.0	4732.0	18.5	70.0	0.657783

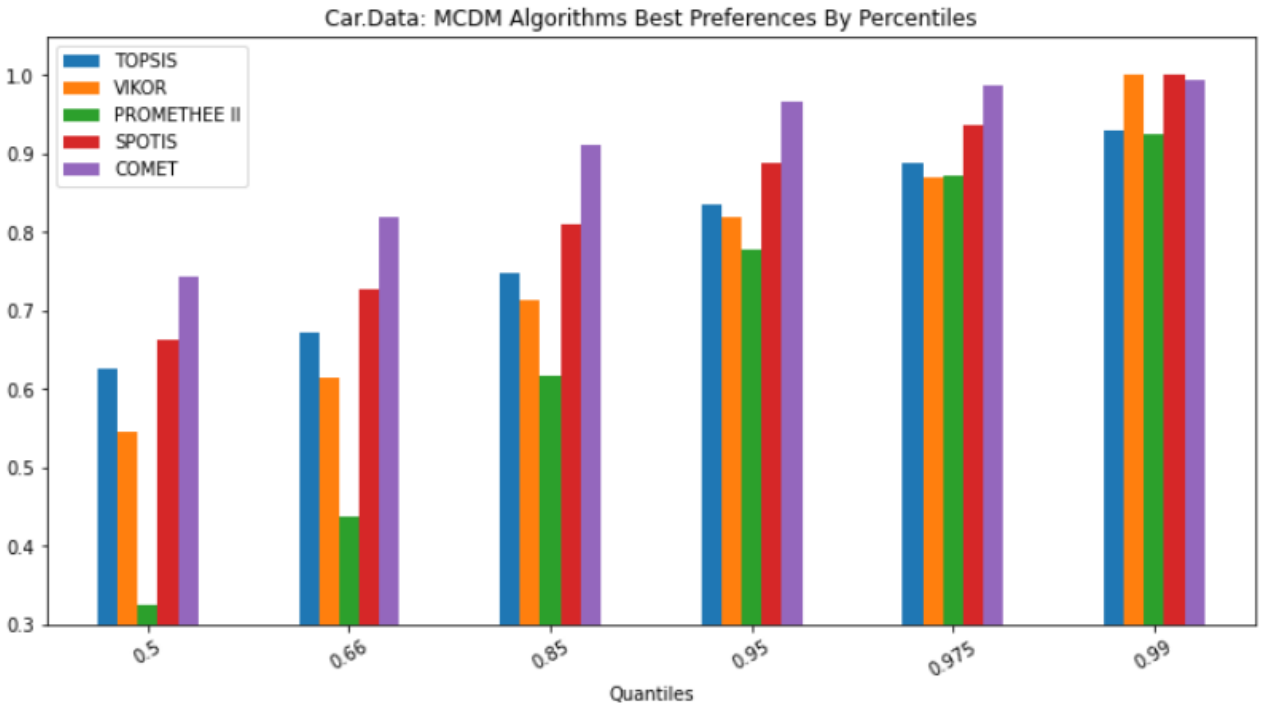
Table 10: Top Alternatives presented by COMET on Auto.Data

Best economy car: 1196, 1520, 1625, 1628
Best race car: 13, 25, 26, 27, 28

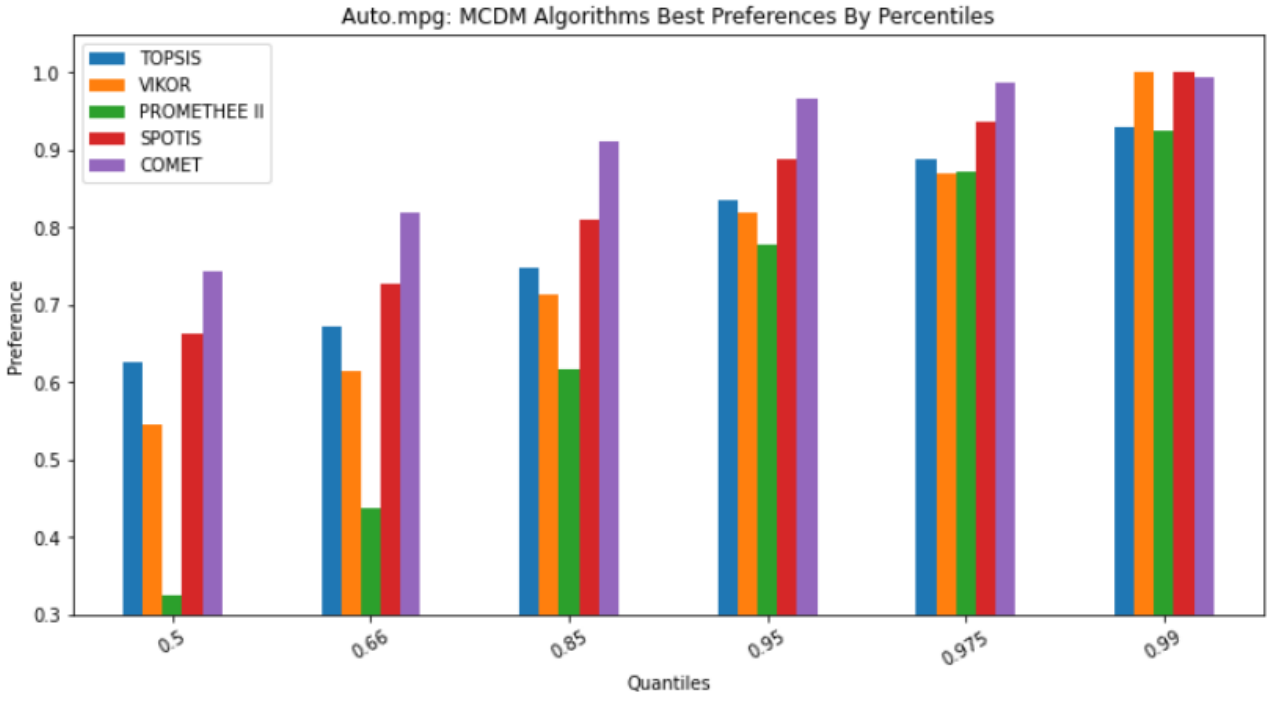
F. Comparison

For the Car Evaluation dataset, the top best economy cars for all algorithms was universally 1520, 1196, 1625 and 1628. For the Auto-mpg dataset, the best race car fluctuated depending on the algorithm. However, there was some overlap between the algorithms. 13,25, 26,27, 28 was reported by both TOPSIS and COMET. Also, PROMTHEE-II, SPOTIS, and VIKOR produced similar alternatives between 220 and 340 with the values 222, 228, 298 and 300 each being repeated twice.

Below, we compare all 5 algorithms side by side for 6 quantiles. The quantile represents the threshold for values to be considered optimal. The preference axis represents the average preference of all alternatives above the threshold. For example, in the Car Evaluation dataset, we see that in 0.66 percentile, the alternatives produced by the SPOTIS algorithm have an average preference a little greater than 70%. That is, we are 30% away from the theoretical ideal point in which all attributes are maximized. Naturally, as we increase the threshold in the quantile range, then the average preference also increases. The purpose of the bar graphs is to show the difference between algorithms for each quantile.



Above, *Fig 1:* Comparing the performance of the MCDA algorithms on the Car Evaluation dataset for six different quantiles. Below, *Fig 2:* Comparing the performance of the MCDA algorithms on the Car Evaluation dataset for six different quantiles.



As expected, TOPSIS, the most popular method performed consistently below other methods. PROMETHEE performed the worst in all situations. PROMETHEEE did not rank the top 50% best alternatives well, but did about as well as TOPSIS for the top alternatives in the 97.5% and 99%

percentile. As expected, COMET, the latest method, did outperform the other methods. VIKOR performed the best overall followed by COMET. VIKOR preformed much better than other methods in the 50th percentile and about equal to COMET in the 99th percentile. COMET did perform slightly better than VIKOR for the 85th percentile and 95th percentile. SPOTIS reliably performed 3rd for all quantiles except the 99th percentile. Overall, at the lower quantiles, there was much more fluctuation in average fluctuation for the algorithms. At the higher quantiles, the average preferences were much more near each other.

IV. Discussion

Which economy car was the best to buy (Auto-mpg) and which was the best race car to buy (Car Evaluation)? For the Car Evaluation dataset, the alternatives 1196, 1520, 1625 and 1628. Knowing if MCDA algorithms produce reliable results can be very difficult [1,2]. We have strong confidence in these alternatives because they were repeated with many algorithms. However, we have medium confidence in the alternatives in the Auto-mpg dataset. Knowing the best race car was more difficult because the best alternatives fluctuate according to the algorithm. Each of the alternatives 13, 25, 26, 27, 28, 222, 228, 298 and 300 were at least repeated twice. The algorithms on the Auto-mpg diverged into two groups with TOPSIS and COMET reporting 13, 25, 26, 27, 28 but PROMETHEE-II, VIKOR, and SPOTIS each reporting 222, 228, 298 and 300. Interestingly, we see that these algorithms are clustered around a set of points. A likely cause for this fluctuation was that Auto-mpg had 6 attributes and Car Evaluation had 4 attributes. MCDA algorithms often perform better on datasets with less attributes[5,8]. Future experimentation can be done to test if removing 2 attributes from the Auto-mpg dataset would report more consistent alternatives by the MCDA algorithms. Another likely source of fluctuation was that only the top 5 alternatives were reported. There could have been the possibility that the top 10 or top 20 choices were very similar. If these choices were similar, then the fluctuation of the best alternatives would be better explained by random fluctuations in the data and confounding variables not related to the MCDA algorithms.

COMET was the clear winner in the best algorithm to use for both datasets and nearly all quantiles. However, at quantiles 0.99, the choice of algorithm mattered much less. This suggests that MCDA algorithms may eventually converge to some optimal choices with high threshold values. In order to test this, future experimentation can focus on testing a larger range of values in high quantiles values. For example, testing 0.95,0.96,0.97,0.98, 0.99, and 0.995. An important point to consider is that the choice of threshold may impact the alternatives reported. We see that in both Figure 1 and Figure 2, TOPSIS, VIKOR and COMET performed better for low alternatives while SPOTIS and PROMETHEE-II performed generally worse. This partly explains why TOPSIS and COMET clustered against VIKOR, SPOTIS and PROMETHEE-II. Further experimentation can be done to see how the optimal alternatives change in response to different threshold values. Likely, an optimal threshold value will need to be tuned through algorithmic hyper parameter tuning techniques.

A very important point about this study is that we are not making any claims about the capabilities of these algorithms whatsoever. In order to do so would require testing these algorithms and their variants on an abundance of standardized datasets specifically for MCDA with much more data analysis. Also, more attention would be needed to the parameters of the algorithms and calculation options. However, what this study does seem to suggest is that the COMET MCDM followed by SPOTIS were generally the best

algorithms suited to the task of finding the best alternatives for these specific datasets. Also, in terms of finding the very best alternatives in the highest percentile threshold values, VIKOR, COMET, and SPOTIS all did about equally well. In regards to future experimentation, we are interested to see if these three algorithms would also perform well on several different datasets and dataset types. There could be some hidden patterns in the datasets or some confounding variables unique to the domain of vehicle selection that is favoring VIKOR, COMET and SPOTIS.

Another very important point is the strong influence of the weighting methods in the MCDA algorithms. Different columns have different influences on the quality of a product and therefore the weighting problem cannot be ignored. For all MCDA algorithms in this study, we subjectively assigned weights to the importance of the columns. For example, miles per gallon and year made was less preferred in the design of a race car and was weighted at 33%. These weights significantly influence the results of the optimal alternatives reported. Reducing a weight for an attribute causes the optimal alternatives to less likely contain that attribute as having a maximized value. How do we know we should not have instead chosen 10%, 20%, or 60%? Choosing weights for the columns is strongly subjective which means that the different users deploying the MCDA algorithms will likely use wildly different weights. One solution for this is using algorithms to determine the best weights. These algorithms work by measuring how much influence an attribute has on making an instance more desirable. An example of such a method is a 2021 study in which Keshavarz-Ghorabaee et. all determined the weights for MCDM based on studying the effects of removing subsets of criteria on the optimal alternatives [10]. The downside to letting algorithms decide the weights is not always a good thing because while humans may value an attribute highly, an algorithm might overstate it or vice versa. Therefore, the engineer will have to decide the appropriate method based on the specific situation and dataset. One solution for this would be to have reinforcement learning algorithms to learn the preferences of the individual. This would require significant architecture additions and might be time consuming on the users part.

There is also the possibility that this study could have been accidentally designed unfairly against TOPSIS and PROMETHEE-II. There are many reasons why this could be the case. First, this study did not do hyper-parameter tuning. The performance of machine learning algorithms often heavily depends on tuning the variables. We did not do hyper-parameter tuning in this experiment because doing proper hyper-parameter tuning can be time intensive and the purpose of this study was to showcase some modern algorithms. Hyper-parameter will be a high priority goal in our future experimentation because doing so might change the relative performance of the algorithms. In other words, PROMETHEE-II for instance might have performed less than VIKOR because PROMETHEE-II was improperly tuned while VIKOR was lucky with the default values. Next, the variants of these algorithms were not deployed in this study. There are several versions of each of the algorithms presented in this study. A modified version of TOPSIS might outperform all the algorithms because it can be more suited to a specific dataset. For this reason, we are interested in trying several more algorithms and their variants in future studies. While choosing the right set of algorithms and the right set of variants to experiment with will likely significantly influence the results, this is a minor point because this is inherent in the field of engineering. There will always be a variety of methods for the engineer to try.

Lastly, this study aimed to use EMaO algorithms for MCDA, but was unable due to a lack of time and several constraints. While evolutionary methods are the frontier of MOO algorithms, finding python

libraries that convert data into objective functions was difficult. Algorithms such as NSGA-III required an objective function in which an attribute already had a curve associated with. Obtaining objective functions would entail machine learning to fit an appropriate curve for each attribute. For future experimentation, we are interested in using symbolic regression for each attribute and then feeding the results into NSGA–III. Another area of our interest is in exploring EMaO methods that handle directly without objective functions. An example of this would be using reinforcement methods to estimate the objective functions of the dataset which are fed into the EMoA.

V. Conclusion

There are some key takeaways for designing the next experiment deploying MCDA algorithms and several future directions to investigate. Reporting the top 5 choices may yield fluctuating results if the best alternatives are already similar. Other strategies need to be explored on how to pick the best alternative. One idea could be to pick a lower quantile and then present the alternatives in those quantiles that were repeated the most by different MCDA algorithms. For example, the top 90 quantile alternatives would be presented and the top 5 most repeated alternatives would be presented. Also, finding the optimal threshold will likely require first finding the optimal tuning threshold to find the alternatives. Of course, future experimentation should focus on a variety of different dataset types with many more algorithms and their different variants. More attention will need to be devoted to determining the weights and achieving solid reasoning for doing so. Using evolutionary methods and reinforcement learning on MCDA is a particular interest of ours.

VI. References

[1] B. Paradowski, W. Sałabun, Are the results of MCDA methods reliable? Selection of materials for Thermal Energy Storage, Procedia Computer Science. 192 (2021) 1313–1322. https://doi.org/10.1016/j.procs.2021.08.135.

[2] V. Kozlov, W. Sałabun, Challenges in reliable solar panel selection using MCDA methods, Procedia Computer Science. 192 (2021) 4913–4923. https://doi.org/10.1016/j.procs.2021.09.269.

[3] M. Yelmikheiev, T. Norek, Comparison of MCDA methods based on distance to reference objects - a simple study case, Procedia Computer Science. 192 (2021) 4972–4979. https://doi.org/10.1016/j.procs.2021.09.275.

[4] A. Wald, An Essentially Complete Class of Admissible Decision Functions, The Annals of Mathematical Statistics. 18 (1947) 549–555. https://doi.org/10.1214/aoms/1177730345.

[5] A state of the art literature review of VIKOR and its fuzzy extensions on applications, Appl. Soft Comput. 46 (2016) 60–89.

[6] W. Sałabun, P. Ziemba, J. Wątróbski, The rank reversals paradox in management decisions: The comparison of the AHP and COMET methods, in: Intelligent Decision Technologies 2016, Springer International Publishing, Cham, 2016: pp. 181–191.

[7] E. Triantaphyllou, Two new cases of rank reversals when the AHP and some of its additive variants are used that do not occur with the multiplicative AHP, Journal of Multi-Criteria Decision Analysis. 10 (2001) 11–25. https://doi.org/10.1002/mcda.284.

[8] A. Bączkiewicz, J. Wątróbski, B. Kizielewicz, W. Sałabun, Towards objectification of multi-criteria assessments: A comparative study on MCDA methods, in: Proceedings of the 16th Conference on Computer Science and Intelligence Systems, IEEE, 2021. https://doi.org/10.15439/2021f61.

[9] S. Chand, M. Wagner, Evolutionary many-objective optimization: A quick-start guide, Surveys in Operations Research and Management Science. 20 (2015) 35–42.

https://doi.org/10.1016/j.sorms.2015.08.001.
[10] M. Keshavarz-Ghorabaee, M. Amiri, E.K. Zavadskas, Z. Turskis, J. Antucheviciene, Determination of Objective Weights Using a New Method Based on the Removal Effects of Criteria (MEREC), Symmetry. 13 (2021) 525. https://doi.org/10.3390/sym13040525.