Generating Alternatives for DEX Models using Bayesian Optimization

Martin Gjoreski
Department of Intelligent Systems
Jožef Stefan Institute
Jožef Stefan Postgraduate School
Ljubljana, Slovenia
martin.gjoreski@ijs.si

Vladimir Kuzmanovski
Department of Computer Science
Aalto University, Finland
vladimir.kuzmanovski@aalto.fi

Department of Knowledge Technologies
Jožef Stefan Institute
Ljubljana, Slovenia

Marko Bohanec
Department of Knowledge Technologies
Jožef Stefan Institute
Ljubljana, Slovenia
marko.bohanec@ijs.si

ABSTRACT

Multi-attribute decision analysis is an approach to decision support in which decision alternatives are assessed by multi-criteria models. In this paper, we address the problem of generating alternatives: given a multi-attribute model and an alternative, the goal is to generate alternatives that require the smallest change to the current alternative to obtain a desirable outcome. We present a novel method for alternative generation based on Bayesian optimization and adapted to qualitative DEX models. The method was extensively evaluated on 42 different DEX decision models with a variable complexity (e.g., variable depth and variable attribute's weight distribution). The method's behavior was analyzed with respect to computing time, time to obtaining the first appropriate alternative, number of generated alternatives, and number of attribute changes required to reach the generated alternatives. The experimental results confirmed the method's suitability for the task, generating at least one appropriate alternative within one minute. The relation between the decision-model's depth and the computing time was linear and not exponential, which implies that the method is scalable.

KEYWORDS

multi-attribute models, method DEX, alternatives, decision support, Bayesian optimization

1 INTRODUCTION

Hierarchical multi-attribute models are a type of decision models [1],[2],[3], which decompose the problem into smaller and less complex subproblems and represent it by a hierarchy of attributes and utility functions. Such decision models are especially useful in complex decision problems [4],[5].

DEX is a hierarchical qualitative multi-attribute method whose models are characterized by using qualitative (symbolic) attributes and decision rules. The method is supported by DEXi [6],[6],[7],[8], an interactive computer program for the development of qualitative multi-attribute decision models and the evaluation of alternatives (options). DEXi has been used to analyze decision problems in different domains in healthcare [9], agriculture [10], [11], [12], economy [13], etc.

A useful extension of DEX would be the possibility to search for new alternatives that require the smallest change to the existing alternative to obtain a desirable outcome. This task is important for practical decision support [14], however the related work on generating alternatives for qualitative multi-attribute decision models is quite scarce. The only related study was presented by Bergez [15], in which the focus is on attribute scoring (and not on the alternatives), and the starting (current) alternative was not taken into a consideration. More specifically, Bergez developed a genetic algorithm for searching a set of the ''worst-best'' i.e., lowest scores for the input attributes that lead to the highest score for the root attribute (the decision model's output), and ''best-worst'' i.e., highest scores for the input attributes that lead to the lowest score for the root attribute.

In this study, we developed a stochastic method for generating alternatives that require the smallest change to the current alternative to obtain a desirable outcome. To avoid combinatorial explosion, the method uses guided search based on Bayesian optimization. The method is evaluated on 42 different qualitative multi-attribute models with a varying complexity. The method's behavior was analyzed with respect to several characteristics including: computing time, time to first appropriate alternative, number of generated (appropriate) alternatives, and number of attribute changes required to reach the generated alternatives.

2 DOMAIN DESCRIPTION

In this study, a set of 42 DEX multi-attribute decision models were used. The models are benchmark mock models, designed by Kuzmanovski et al. [16]. The decision models are designed by taking into account properties such as model depth, distribution of attributes' aggregation weights (weights' distribution), and inter-dependency of attributes (input links). Table 1 presents a summary of the decision models. The weights' distribution is given with descriptive names: skewed, normal, and uniform. All the attributes in the models are defined with same value scale (low, medium, high), including the input and the output attributes. Additional assumption is that all attribute combinations are possible.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Information Society 2020, 5–9 October 2020, Ljubljana, Slovenia
© 2020 Copyright held by the owner/author(s).

Table 1: Properties of the mock DEX decision models.

Leaves	Depth	Weights' distribution			Links	Versions
8	3	skewed,	normal,	uniform	yes	3, 3, 1
9	3	skewed,	normal,	uniform	no	3, 3, 1
19	4	skewed,	normal,	uniform	yes	3, 3, 1
20	4	skewed,	normal,	uniform	no	3, 3, 1
38	5	skewed,	normal,	uniform	yes	3, 3, 1
39	5	skewed,	normal,	uniform	no	3, 3, 1

3 METHOD FOR GENERATING ALTERNATIVES

An efficient search strategy is required to generate alternatives that require the smallest change to the current alternative to obtain a desirable outcome. A naïve approach would be to generate all possible alternatives, or to iteratively generate random alternatives, and to evaluate the outcome for each alternative. However, for reasonably complex decision models, the search space can be enormous, rendering the naïve approaches unsuitable.

A more appropriate approach would be to use informed search based on the history of previously generated and evaluated alternatives. The history can be used to estimate the search space and the behavior of the decision model. Based on that estimation, more promising alternatives can be generated. By focusing on the more promising alternatives the search space is reduced, and consequently, the time needed to find the appropriate alternatives is also reduced. The next subsections describe a stochastic method that uses Bayesian optimization to efficiently generate such alternatives. The method assumes that we do not know the internal rules by which the decision models operate, thus it falls into the category of ''black-box'' optimization techniques. Knowing and utilizing the decision rules might help the search algorithm, but this option was not addressed in this study.

3.1 Implementation

The problem of generating alternatives that require the smallest change to the current alternative to obtain a desirable outcome can be defined as an optimization problem with two objectives: (1) improved outcome (desired output) of the decision model, and (2) maximum similarity between the current alternative $\overline{c}$, and the new proposed alternative $\overline{a}$. For each decision model DM, one alternative can be defined as a tuple of attributes $\overline{a} = (a_1, a_2, \dots, a_n)$, where each attribute can take any value of a limited set of values. Usually, that set includes ordinal values (e.g., low, medium and high) and those values can be encoded with integers (e.g., 0, 1 and 2). Consequently, a distance d between alternatives can be defined over Euclidean space. The specific distance function used by the method is a modified element-wise difference between the candidate alternative $\overline{a}$ and the current alternative $\overline{c}$. This distance considers only the attributes for which the candidate alternative has higher values compared to the current alternative $\overline{c}$.

$$d(\overline{c}, \overline{a}) = \sum \begin{cases} a_j - c_j, & if\ a_j > c_j \\ 0, & if\ a_j \le c_j \end{cases}$$

From the distance function, a similarity function s can be also defined as one minus the normalized distance. The distance is normalized using the maximum plausible distance for the specific problem. For example, if $\overline{a}$ has 20 attributes with possible values between 0 and 2 and each attribute has the highest possible value, and if $\overline{c}$ has only attributes with the lowest possible value (0), then the maximum distance is 20 * 2.

$$s(\overline{c}, \overline{a},) = 1 - \frac{d(\overline{c}, \overline{a})}{\max\_distance}$$

Finally, the optimization function can be defined as:

$$f(\overline{c}, \overline{a}, DM(\overline{c},), DM(\overline{a})) = \begin{cases} s(\overline{c}, \overline{a},), & if\ DM(\overline{a}) > DM(\overline{c}) \\ 0, & if\ DM(\overline{a}) \le DM(\overline{c}) \end{cases}$$

where $DM(*)$ is the output of the decision model for the specific alternative. By optimizing f, the method searches for alternatives that are as similar as possible to $\overline{c}$ and improve the output of the decision model $(DM(\overline{a}) > DM(\overline{c}))$.

In order to apply the Bayesian optimization approach, a surrogate function (a model), an acquisition function, and a generator of alternatives, need to be defined. The surrogate model SM is a model that estimates the objective function for a given alternative as input. Typically, models based on Gaussian Process (GP) [17] are used because by exploiting the mean and the standard deviation of the output distribution, we can balance the trade-off of exploiting (higher mean) and exploring (higher standard deviation). Since GP models are computationally expensive with the complexity of $O(n^3)$, ensemble models such as Random Forest (RF) can be also used [18]. In that case, the mean and the variance are calculated based on the predictions of all base models available in the ensemble. Our method uses RF with 1000 decision trees as base models.

The acquisition function operates on top of the mean and standard deviation of the SM's output. The final version of the method uses the expected improvement (EI) as an acquisition function [19]. This acquisition function checks the improvement that each candidate alternative brings with respect to the maximum known value ($\mu(SM(\overline{a})) - a_b$), and scales those improvements with respect to the uncertainty. If two alternatives have a similar mean value, the one with higher uncertainty ($\sigma(SM(\overline{a}))$ will be preferred by the acquisition function.

Finally, we need to define the generator of alternatives. Our method uses two generators of alternatives: a neighborhood generator and a random generator. Based on the distance function d, neighborhood relation can be defined. Two alternatives $\overline{a_1}$ and $\overline{a_2}$ are considered as neighbors with a degree k, if $d(\overline{a_1}, \overline{a_2})$ = k. . The random generator is a generator of alternatives which: (1) avoids generating known alternatives; and (2) is conditioned by the best-known (with respect to the optimization function) alternative discovered in the previous iterations.

Algorithm 1 presents the implementation of the proposed method. The function *check_promising_values* runs the SM on a set of promising alternatives. This set contains all alternatives that have been previously generated as neighbors to a specific best alternative, but have not been evaluated with the DM because the acquisition function has selected other alternatives. This enables one final check of the most promising solutions which may have been missed because of an earlier bad prediction of the SM.

Algorithm 1:

```
Input: Decision model DM, current alternative CA,
Output: best_alternatives
# parameters and initialization
max_e = 150 # maximum number of epochs
n_candidates = 10 # candidates per iteration
objective_jitter = 0.8 # if an alternative is close to the current
    best (e.g, 75% as good as the current best , the
    alternative's neighbors should be checked)
random_sample_size = 10000
best_alternatives = []
surrogate_model = new Random_Forest()
promising_alternatives_pool = []
#initial values
candidate_alternatives = generate_random_alternatives(10)
real_objective_values = objective_func(DM, CA, alternatives)
surrogate_model.fit(candidate_alternatives, real_objective_values)
known_alternatives.add(candidate_alternatives,
    real_objective_values)
best_alternative,best_score = max(candidate_alternatives
    ,real_objective_values)
neighboring_alternatives= gen_neighborhood(best_alternative)
while counter < max_e do:
    if size(neighboring_alternatives)>0:
    alternatives_pool = neighboring_alternatives
    else:
    alternatives_pool = gen_rand_alternatives(best_alternative,
      random_sample_size)
    # get top ranked (e.g., 10) candidates using the acquisition
      function
    candidate_alternatives, candidate_scores =
      perform_acquisition(alternatives_pool,          n_candidates)
    #evaluation of candidate alternatives
    real_objective_values = objective_func(DM, CA, alternatives)
    known_alternatives.add(candidate_alternatives,
      real_objective_values)
    #update current best and promising alternatives
    i=0
    while i < size(candidate_scores) do:
    if best_score*objective_jitter <= candidate_scores[i] do:
    neighboring_alternatives = gen_
      neighbourhood(candidate_alternatives[i])
    promising_alternatives_pool.add(neighboring_alternatives)
    if  best_score< candidate_scores[i] do:
    best_alternatives = []
    best_alternatives.add(candidate_alternatives[i])
    if  best_score==candidate_scores[i] do:
    best_alternatives.add(candidate_alternatives[i])
    i++
    #update the surrogate model
    surrogate_model.fit(candidate_alternatives, real_objective_values)
    counter++
end
#peform final check of the promising alternatives
best_alternatives =
    check_promising_values(promising_alternatives_pool,best_alt
    ernatives)
return best_alternatives
```

4 EXPERIMENTS

4.1 Experimental Setup

The method was evaluated with the 42 decision models described in Section 2. For each decision model, nine different randomly sampled starting alternatives (current alternatives $\bar{c}$) were sampled. Three of those alternatives were with a final attribute value low, three with a final attribute value medium, and three with a final attribute value high. The desirable outcome was also varied, i.e., from low to medium, from low to high, from high to medium, and from high to low. This experimental setup resulted in 756 different experimental runs. Each experiment was running for a minimum of 100 epochs, a maximum of 150 epochs, and 50 epochs without improvement. The method and the experiments were implemented in Python, and are available online[1].

4.2 Experimental Results

The average experiment duration for the models with depth 3 was less than 5 min. For the models with depth 4, the duration increased for 3 min and for the models with depth 5 the duration increased for additional 3 min. This indicates that the relation between the computational time and the model depth is linear.

The final output of the algorithm is a set of thousands of different alternatives. However, from a user perspective, only one or just a few alternatives should be enough. Figure 1 presents the number of epochs required to generate the first alternative for the most complex models (depth 5). From the figure it can be seen that on average, the first alternatives are generated in the first 10 epochs. For the less complex models, the number of required epochs was less than 5.

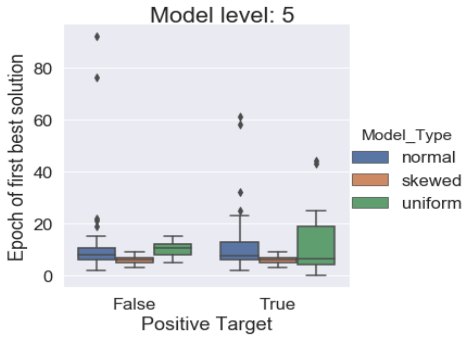

Figure 1: Number of epochs required to generate the first alternative in the final set of alternatives.

In each epoch, the algorithm selects the top 10 alternatives with respect to the optimization score. The higher the score, the better the alternatives are. The selected alternatives depend on the acquisition function, which in turn depends the predictions of the surrogate model. Figure 2 present the average optimization score in each epoch for the most complex models (depth 5). For a comparison, the average optimization score of 10 randomly sampled alternatives at each epoch is also presented (dashed line). From the figure it can be seen that the optimization score of the random samples is significantly lower than the optimization score of the samples selected using the proposed algorithm.

Finally, the presented algorithm is stochastic and the optimality of the solution cannot be guaranteed. One metric that presents the quality of the solutions is the number of attribute changes required to achieve the final solution starting from the current state of the current alternative. Figure 3 presents that metric, which is the same as the distance defined in Section 3.1. From the figure it can be seen that in the majority of the cases, the final solution can be reached with less than 5 attribute changes. Exception of this are the decision models that have a depth 5 and uniform weights' distribution.

[1] Repository link.

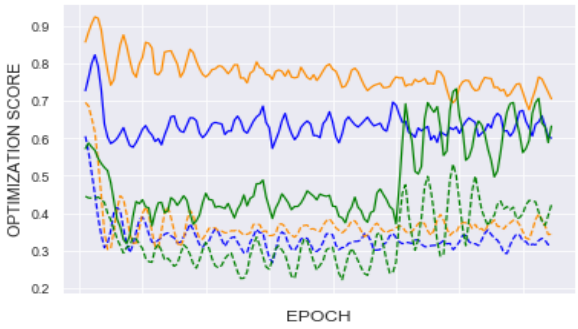

Figure 2: Average optimization score for the decision models with depth 5. Full line - alternatives generated by the surrogate model. Dashed line - random alternatives. The type of attribute weights is color-coded (blue-normal, orange-skewed, green-uniform).

This is because these models have a larger number of input attributes and the uniform distribution requires many attributes to be changed in order for that change to be prolonged to the aggregate attribute. On the other hand, the models with normal and skewed weights' distribution require smaller number of attribute changes for that change to be propagated to the aggregate attributes.

5 DISCUSSION AND CONCLUSION

We presented a novel method for generating alternatives for multi-attribute DEX decision models based on Bayesian optimization. The main goal of the method was to generate alternatives that require the smallest change to the current alternative to obtain a desirable outcome. The method was extensively evaluated on 42 different DEX decision models. The models were with a variable complexity (e.g., variable depth and variable attribute's weight distribution). The method's behavior was analyzed with respect to several characteristics: computing time, time to first appropriate alternative, number of generated (appropriate) alternatives, and number of attribute changes required to reach the generated alternatives.

The experimental results confirmed that the method is suitable for the task i.e., it generates at least one appropriate alternative in less than a minute, even for the most complex decision models. In the majority of the cases, the computing time was lower than that. The discovery of the alternatives was equally distributed throughout the overall runtime. Exception of this is the final check performed by the algorithm (see *check_promising_values* in Algorithm 1), which generates the majority of the alternatives for the more complex models (depth 4 and depth 5). The quality of the alternatives was also appropriate as in the majority of the cases, the generated alternatives could be reached by less than 5 attribute changes. Finally, the relation between the decision-model's depth and the computing time was linear and not exponential, which implies that the method is scalable.

The method implementation considers ordinal attribute values. However, there is possibility for considering other types of distance measures that would work in nominal settings (e.g., Levenshtein distance).

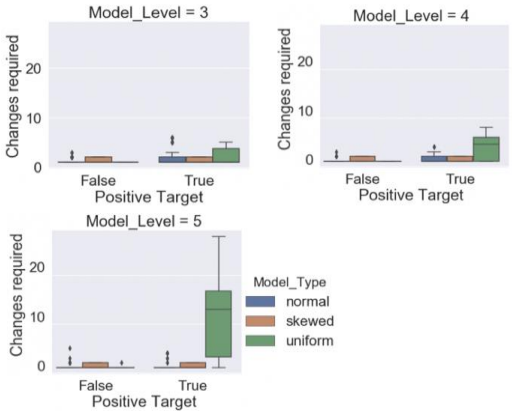

Figure 3: Boxplots for the number of changes required to switch from the starting alternative to the best alternative.

Regarding the future work, the proposed method is stochastic and the optimality of the final solution cannot be guaranteed. In order to do that, the method needs to be validated additionally. Promising options include comparison of the proposed method with deterministic methods and methods that utilize internal rules by which the decision models operate.

REFERENCES

[1] Power, D.J. Decision Support Systems: Concepts and Resources for Managers. Quorum Books, Westport, 2002.
[2] Turban, E., Aronson, J. and Liang, T.-P. Decision Support Systems and Intelligent Systems, Prentice Hall, Upper Saddle River, 7th Edition, 2005.
[3] Mallach, E.G. Decision Support and Data Warehouse Systems. Irwin, Burr Ridge, 2000.
[4] Sadok, W., Angevin, F., Bergez, J.-E., Bockstaller, C., Colomb, B., Guichard, L., Reau, R., Messeau, A. and Doré, T. MASC: a qualitative multi-attribute decision model for ex-ante assessment of the sustainability of cropping systems. Agron. Sustain. Dev. 29, 447–461, 2009.
[5] Munda, G. Multiple criteria decision analysis and sustainable development. In: Multiple Criteria Decision Analysis: State of the Art Surveys, Springer-Verlag, New York, 2005.
[6] Bohanec, M. and Rajkovič, V. DEX: An Expert System Shell for Decision Support. Sistemica 1(1), 145-157, 1990.
[7] Bohanec, M. and Rajkovič, V. Multi-attribute decision modeling: Industrial applications of DEX. Informatica 23, 487-491, 1999.
[8] Bohanec, M. DEXi: Program for Multi-Attribute Decision Making User's Manual." Ljubljana, Slovenia: Institut Jozef Stefan, 2008.
[9] Bohanec, M., Zupan, B. and Rajkovič, V. Applications of qualitative multi-attribute decision models in health care, International Journal of Medical Informatics 58-59, 191-205, 2000.
[10] Bohanec, M., Cortet, J., Griffiths, et al. A qualitative multi-attribute model for assessing the impact of cropping systems on soil quality. Pedobiologia 51, 239–250, 2007.
[11] Bohanec, M., Messéan, A., Scatasta, S. et al. A qualitative multi-attribute model for economic and ecological assessment of genetically modified crops. Ecol. Model. 215, 247–261, 2008.
[12] Coquil, X., Fiorelli, J.L., Mignolet, C., et al. Evaluation multicritère de la durabilité agr environnementale de systèmes de polyculture élevage laitiers biologiques. Innov. Agron. 4, 239–247, 2009.
[13] Bohanec, M., Cestnik, B., Rajkovič, V. Qualitative multi-attribute modeling and its application in housing, Journal of Decision Systems 10, pp. 175-193, 2001.
[14] Debeljak, M., Trajanov, A., Kuzmanovski, V. et al. A field-scale decision support system for assessment and management of soil functions. Frontiers in Environmental Science, 7, p.115, 2019.
[15] Bergez, J.-E. Using a genetic algorithm to define worst-best and best-worst options of a DEXi-type model: Application to the MASC model of cropping-system sustainability. Computers and electronics in agriculture 90: 93-98, 2013.
[16] Kuzmanovski, V., Trajanov, A., Dzeroski, S., et al., M. Cascading constructive heuristic for optimization problems over hierarchically decomposed qualitative decision space. Omega, submitted September, 2020.
[17] Rasmussen C. E. and Williams C. K.I. Gaussian Processes for Machine Learning", MIT Press 2006.
[18] Frank, H., Hoos, H. H., and Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration (extended version). Technical Report TR-2010–10, University of British Columbia, Computer Science, Tech. Rep. 2010.
[19] Lizotte F. Practical Bayesian Optimization. PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 2008.