A Decision Making and Problem Analysis Supporting System

Norbert Sram

Óbuda University, Budapest, Hungary
norbert.schramm@gmail.com; sramm.norbert@phd.uni-obuda.hu

***Abstract**: Decision making can be regarded as an outcome of a cognitive process leading to the selection of a course of action among several alternatives. With the latest advances in decision making systems it is important to study which methods provide better results and in which fields, contexts. By creating a hybrid approach to decision making we can outline the performance and deviations between different methods, but more importantly we establish the foundation for building a hybrid decision making system. With the analysis of the decision making methods, we are able to build a hybrid decision making system, which is a composition and cooperation of different methods and uses the optimal methods for specific contexts. In this paper the author presents an approach to creating and evaluating hybrid decision making systems.*

1 Introduction

There are a lot of decision making algorithms and systems, all of these methods have their advantages and disadvantages. Not one of them is suitable for every type of problem or input. Instead of trying to choose the "correct" method for a specific problem, a better approach could be to combine multiple methods into one decision tree with the "right tool for the job" approach. The presented system is also usable to fine-tune one specific method, by experimenting with different kind of configurations and inputs.

2 Decision Making

Logical decision making is an important part of all science-based professions, where specialists apply their knowledge in a given area to making informed decisions. For example, medical decision making often involves making a diagnosis and selecting an appropriate treatment. Decision making can be based on

different methods like expert systems, fuzzy logic [1, 2, 3, 4, 10] based systems, neural networks [5], probabilistic reasoning [6] and so on. All these methods have their unique approach to come to a conclusion. It's hard to predict, which method is the most suitable for a specific task, which would provide the best overall result. The Pareto principle (80/20 rule) is not suitable for optimizing decision making. This means that it's not enough to finetune or direct attention to the "vital" parts. In case of decision making, the complete solution falls into the "vital" part category. An issue in one part can lead to the exclusion of other parts, thus producing an inaccurate result. The overall best result would be the best possible result from all subsystems, regardless of the method used. To achieve this state, we face the problem of interfacing different approaches. In this paper, the author provides a possible solution to this problem.

3 Hybrid Approach

The presented system is an environment, which is specialized for decision making. Evaluating different methods and generating optimal decision steps. The system is built up form three different abstraction units:

- Control vertices – highlighted points of the decision system
- Transformation algorithms – algorithm to convert from/to domain specific values
- Evaluation algorithms – algorithms to evaluate the effectivness of a step based on the control vertices.

Decision making can be represented as a tree of steps we execute to get a final result. In order to compare different type of decision methods and to merge them into a hybrid decision making system, we need to use a representation which is suitable for comparing and analyzing hierarchical structures. The most adequate representation would be a directed weighted graph of the decision systems. Were each decision system would be a branch of the graph. We need to specify control vertices in the decision graph. These vertices can be viewed as partial results or steps which lead to the final decision. Control vertices are the key points of the decision system and also the comparison point for multiple approaches. Multiple approaches are represented as graph edges which are made up of control vertices. All edges of the graph which belong to control vertices have values. These values represent the effectiveness or the result of the specific method. By executing the decision graph with a specific input we can assign a value to every edge of the graph. From this we can deduce which methods provide a better result for the specified input. This done in multiple steps. First step is to iterate through the edge of the graph with a transformation algorithm, which converts the graph edge values to domain specific or effectiveness values. After the edge value transformation step we can use graph algorithms on our decision graph. Based on the transformation

algorithm we can find the optimal decision tree in the graph for example by running a maximum spanning tree or a minimum spanning tree algorithm on the decision graph [7, 8, 9]. In case of hierarchical systems the order in which the control vertexes are executed is crucial. In those cases we have to use the shortest path algorithm [11].

Transformation algorithm is a function which takes the value of the control vertex edge, which is equal to a result produced by one of the decision methods and maps it to a different domain. So the transformation function has a type of a -> b, where it is possible for a and b to be equal. In most cases the identity transformation is ideal, which means that the edge values of the graph remain the results. The main purpose of the transformation algorithm is to create graphs, which represent the effectiveness of a specific method in a certain context or for a certain unit of measurement. By fusing together all these graphs, it is possible to create a multi criteria based evaluation of the decision methods. The fusion of the graphs is based on the control vertices defined. Every transformation algorithm produces a graph with a domain specific edge values. Between two control vertices there is at least one edge. The fused graph contains all of the edges produced by the transformation algorithms. The importance of the fused graph is that it provides the ability to compare multiple decision methods based on multiple parameters and multiple contexts. This means that we have the ability to evaluate the same method in different contexts, evaluate the effectiveness of a certain method with different parameters. Using the fused graph we create the hybrid decision making system by eliminating the “unnecessary” edges between the control vertices. The elimination is done by the evolution algorithm specified or implemented by the user. We can use the fused graph to create multiple hybrid decision systems by executing multiple, different evolution algorithms on it.

The presented system is based on the basic requirements of a decision making system. Objectives must first be established and must be classified and placed in order of importance. Objectives can be modeled as control vertices. This means that the user needs to decouple the system to sub steps, which correspond to the control vertices. This can be viewed as a disadvantage, because of the additional work, but modularity has its own advantages as well. It's easy to pinpoint the problems in the system. After the specification of the control vertices, the user needs to specify the connection between two control vertices. The connection is the decision method we need to make to reach the specified vertex. The user creates a decision graph for each decision method, the system then fuses these graphs together. The fused graph can be evaluated by specifying an input. This way all steps with all used decision methods can be evaluated. The initial setup can be a lot of work, this can be viewed as a disadvantage, but the information and the test environment gained can have a significant impact over time. Also it's always possible to generate a different hybrid system from the existing fused graph, which may be more suitable for different input values.

The generated hybrid decision graph can be too specific or ideal for only one type of input context. Based on the Pareto principle the optimal branch can be tailored for the "20 part" of the input set. To avoid the disadvantages of input specific training of the system, it can be run in a hosted mode. The hosted mode always executes all the steps represented in the decision graph, it executes the transformation and evolution algorithms. It works with the principle of "just in time", a decision is produced before required, instead of ahead of time training and fixed steps.

4 Case Study

In this chapter, the author will present a simple case study, to demonstrate the system described in the paper. The first step is always to create a control vertex layout of the decision system. A simplified scenario is shown on Figure 1.

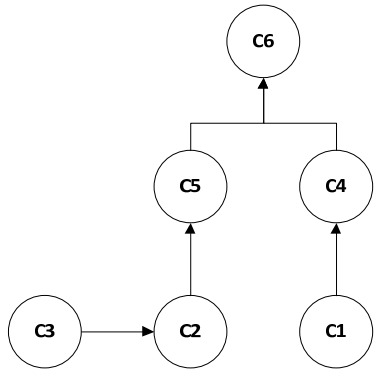

Figure 1
Control vertex layout of a decision system

This tree is a data flow tree of the steps which lead to a decision. It's an abstraction of the system, which does not include any logic; it is only a schematic for different method implementations. Based on this schematic the user can create different implementations, shown on Figures 2 and 3, which represent two separate methods.

The decision system shown on Figure 2 maps directly to the control vertex layout shown on Figure 1, but that's not always the case. The used method can include additional steps; such case is shown on Figure 3. In order to start evaluating a hybrid decision system the two defined decision methods need to be fused into one system.

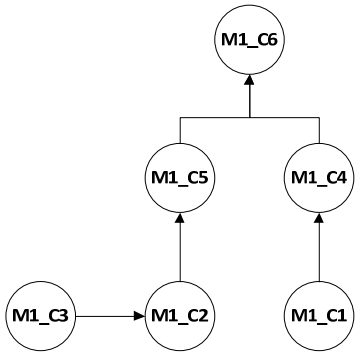

Figure 2
A method specific implementation which does not map directly to a control vertex system

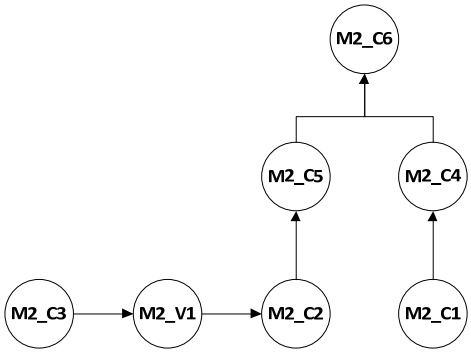

Figure 3
Control vertex layout of a decision system

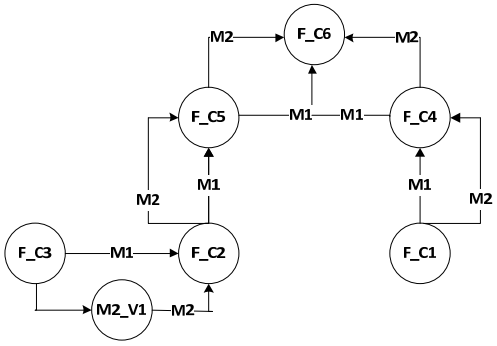

Figure 4
Fused decision making system

As seen on Figure 4 the fused graph includes all the decision steps from both of the methods. These steps are represented as edges (M1, M2) between vertexes. Edges represent operations, not just scalar values. From the fused graph, the system can determine that the input points of the decision system are F_C3 and F_C1, the output or the conclusion is F_C6. From the fused graph the presented system provides an environment which is useful for not only creating hybrid decision systems, but also for evaluating different kind of methods.

The presented example was a simplified case, which is unlike the most problems the users would come across, but it captures the essential approach behind the presented method. Let's take for example Figure 5. It models a complex hierarchical decision system, which has a single input point. The system has a significant number of subsystems. The presented fusing method provides the possibility to freeze certain points of the decision graph. This means that the user has the ability to provide multiple solutions to specific branches in the graph. This is done by contracting the vertexes, which are frozen. For example, in Figure 5 the vertexes from 3 to 11 could be replaced with a single contracted vertex, thus simplifying the control vertex schematic. This is the main strength of the method presented in the system. The user should start out with the decision method most likely to be suitable for the specific problem and step by step freeze branches of the control vertex schematic, providing alternate approaches to parts of the decision tree, which are less than satisfactory.

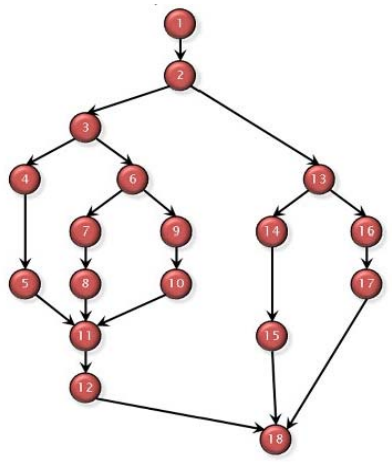

Figure 5
Control vertex layout of a complex hierarchical system

Conclusions

The presented method tries to capture the broad possibilities of decision making into a manageable format and also to provide possibilities for evaluating and mixing different methods. By providing a common format existing algorithms can

be used to deduce information and results. Generic algorithms can be created and used on different decision systems to provide specific information. Based on the collected data and use case scenarios it's possible to deduce input related method specific properties.

Future development plans include the support of broad range of decision methods, with a high level domain specific language to manipulate the decision graphs. Decision step generation based on the common format of the presented method. The generated program would not require the presented system to execute.

References

[1] G Bellmann, R. E., Zadeh., L. A., Local and Fuzzy Logic, Modern Uses of Multiple-Valued Logic, edited by Dunn, J. M., Epstein, G., Reidel Publ., Dordrecht,1977, The Netherlands, pp. 103-165

[2] Dubois, D. Prade, H., What are Fuzzy Rules and How to Use Them, Fuzzy Sets and Systems 84, 1996, pp. 169-185

[3] Dubois, D. Prade, H., Fundamentals of Fuzzy Sets, The Handbook of Fuzzy Sets Series, Kluwer Academic Publishers, Boston, 1999

[4] Fullér, R., Fuzzy Reasoning and Fuzzy Optimization, Turku Centre for Computer science, TUCS General Publication, N0 9, September 1998, ISBN 952-12-0283-1

[5] Simon Haykin, Neural Networks: A Comprehensive Foundation, 1st edition, 1994

[6] Neapolitan, R E, Probabilistic Reasoning in Expert Systems: Theory and Algorithms, 1989

[7] Y. J. Chu and T. H. Liu, On the Shortest Arborescence of a Directed Graph, Science Sinica, Vol. 14, 1965, pp. 1396-1400

[8] J. Edmonds, Optimum Branchings, J. Res. Nat. Bur. Standards, vol. 71B, 1967, pp. 233-240

[9] R. E. Tarjan, "Finding Optimum Branchings", Networks, v.7, 1977, pp.25-35

[10] Márta Takács, Approximate Reasoning in Fuzzy Systems Based on Pseudo analysis and Uninorm Residuum, Acta Polytechnica Hungarica, Volume 1, Issue Number 2, 2004, pp.49-62

[11] B. V. Cherkassky, A. V. Goldberg, T. Radzik, Shortest Paths Algorithms: Theory and Experimental Evaluation, Mathematical Programming, Vol. 73, pp. 129-174, June 1996