ELSEVIER

European Journal of Operational Research 145 (2003) 232–238

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Towards using computational methods for real-time negotiations in electronic commerce

Georgios P. Papamichail *, Dimitrios P. Papamichail

Department of Management Science and Technology, Athens University of Economics and Business, 47A Evelpidon and 33 Lefkados Str., 11 362 Athens, Greece

Abstract

This paper attempts to construct a sound basis to support real-time negotiation decision-making by carefully applying concepts from the area of computational geometry, to meet the increasing decision requirements of electronic commerce. To achieve this, a method is discussed that models decision preferences and traces exact or close matches to these preferences in real-time, therefore allowing the decision maker to form a starting point of negotiations, while being realistic in individual expectations. The paper also discusses the implications and restrictions of such an approach and its future application in the deployment of decision support systems.

Keywords: Decision support systems; Group decisions and negotiations; Computational methods; Electronic commerce

1. Introduction

Electronic commerce is rapidly becoming the principal medium to handle business transactions in the new economy era. Significant investments are taking place in order to automate procurement processes, standardize the form in which organisations state their requirements and define product characteristics (Min and Galle, 1999). In its first steps, this automation was realized through the design and implementation of electronic data interchange (EDI) messages which would accommodate most of the transaction and negotiation cases between cooperating parties (MacDowell, 1993). During recent years, the Internet is used as a facilitating means of these processes, leveraging existing EDI infrastructure and allowing purchasers to do business with all of their trading partners electronically, whether the trading partner is EDI-enabled or not (Copeland and Hwang, 1997).

In all situations it is necessary to support the negotiating parties in such a way that they can assess their similarities or differences in real-time, while identifying the key-issues that construct the basis of their negotiation (Franklin and Reiter, 1996). If such situations are examined carefully, it can be seen that their complexity lies mainly in the difficulty of identifying exact or close matches between counter-party needs and objectives, in the discrimination of preferences, and, finally, in making choices among alternatives (Foroughi et al., 1995). Furthermore, the overall underlying

* Corresponding author. Tel.: +30-10-8203-671; fax: +30-10-8828-078.

E-mail address: pmichael@aueb.gr (G.P. Papamichail).

0377-2217/03/$ - see front matter
PII: S0377-2217(02)00531-3

business goals require the simultaneous consideration of a variety of issues, thus making negotiations more complex or even impossible (Jarke et al., 1987). Also, it is has been observed that during the first stages of the decision-making process it is important to be able to identify identical or similar opinions, options or arguments between negotiating parties in order to produce a solid consensus building basis. The computational decision support and negotiation methods should be implemented in such a way as to enforce their adoption and deployment in a variety of organizational environments and avoid forcing the end-user to comply with proprietary standards. All the above render electronic commerce a challenging area for the design and implementation of negotiation methods and techniques within distributed environments of high uncertainty and increased information asymmetry. Existing approaches and techniques such as data mining and on-line analytical processing often prove to be time-consuming and data intensive (Chaudhuri et al., 2001), and, require customized design and careful adaptation to the varying decision needs (De Sousa et al., 1999).

In this paper an analytical approach is developed whereby the arguments, opinions and views of decision makers are quantified as a range of values within a user-defined interval. Value intervals are arranged as multi-dimensional "argument vectors" with the dimensions defined by a number n of decision parameters for each negotiating party. In another step, each n-dimensional argument vector is associated to an iso-oriented rectangle, where each one of the n-dimensions corresponds to each side of the rectangle. In this manner, it is possible to map and identify exact or close matches between opinions and preferences as an intersection problem among corresponding iso-oriented rectangles. The intensity of similarity between decision views is proportional to the area of intersection among corresponding rectangles, respectively. Rectangle intersection problems have been successfully addressed in the area of computational geometry. A set of algorithms exist for their solution in real-time situations with affordable memory allocation costs.

In the following sections, a method is presented that models decision arguments and traces exact or close matches to these in real-time, therefore allowing the decision maker to form a starting point of negotiation, while being realistic in expectations. The paper also discusses the research and practical implications of such an approach and its future application in the deployment of decision support systems.

2. Research approach

During the last few years, a number of computational models were created to support effective decision-making during negotiations (Bellucci and Zeleznikow, 1998). This growing interest in the use of information technology in supporting negotiations is mainly due to two main reasons: the first has to do with the actual time, around 20% of total working hours, spent by managers in negotiations resolving conflict (Shea, 1983) while the second refers to the increasing complexity of the issues and parameters under consideration. The existing computational models try to produce modern decision support and negotiation systems that are more transparent to the end-user via efficient methods and algorithms, and, relatively simple and friendly user interfaces. A variety of approaches have been developed for attacking complex problems involving, among others, cooperative situations (Wong, 1994), distributed forming of decision alternatives (Vetchera, 1994), and group problem description in complex or ill-structured situations (Agarwal and Prasad, 1994). This paper introduces concepts from the area of computational geometry to construct a unified support infrastructure for efficient decision-making in situations that require cooperation or negotiation among a large number of parties. The idea is based on the observation that the problem complexity faced by decision makers, is mainly heightened by their effort to identify the most common, less controversial key-issues, while trying to maximize consensus among the various negotiating or cooperating parties.

2.1. The distributed decision-making and negotiation problem

Fig. 1 depicts a decision-making situation giving the correspondence between the negotiation

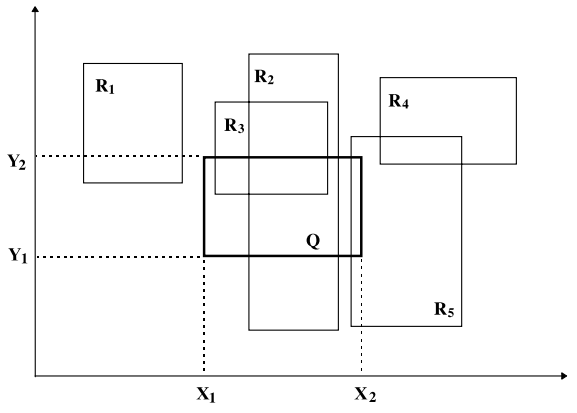

Fig. 1. The decision scene.

and computational geometry problems. Assume that a decision maker expresses his view by stating, in order to preserve simplicity, only two arguments. In the first argument he defines a preference interval, which lies between the values X_1 and X_2, and in the second another preference interval, which lies between the values Y_1 and Y_2. In this way a rectangle is formed, named Q, which represents the specific view of the decision maker. Assume also that the other decision-making or negotiating parties have expressed their views and that these are represented in the same manner. The rectangles formed in this representation form the so-called "decision scene".

In the example given in Fig. 1 it can be seen that there exists a variety of intersection areas between the rectangle Q and rectangles of the decision scene: a large area with R_2 and R_3, a smaller one with R_5 and no intersection with R_1 and R_4. More specifically, in the case of R_3, two of its vertices are enclosed by the Q rectangle, while in the case of R_5, its projection in the x-axis intersects with that of Q. In these two cases the decision problem of the decision view similarity measurement is reduced to the calculation of the corresponding intersecting area. In the intersection between rectangles Q and R_2 a particularly interesting situation arises: R_2 is enclosed in one dimension by Q, which means that the corresponding decision views have a part in common, while it intersects with Q in the other dimension. In the context of negotiations this can be considered equivalent to a case where agreement exists in all parameters but one, in which negotiating efforts should concentrate in order to facilitate the other negotiating party in making a counter-offer.

In this manner, tracing common views or identifying important issues for consensus achievement can be treated within the subject of rectangle intersection. It is important to note that the computational algorithms that have been developed to solve such problems can be utilized also towards building efficient information systems in terms of factors such as anonymity, increased focus on the task, interactive offering and counter-offering and incremental negotiations. Moreover, these algorithms can be used to enhance the ability to negotiate along more dimensions than just price in complex tasks such as electronic commerce. The negotiating parties do not have to give a priori their final preferences but can do so after the computational methods have calculated and suggested a set of possible exact or close matches to what they have described as their preliminary views. The following section outlines the research in the area of rectangle intersection in order to justify its use in the negotiation decision-making area.

2.2. The rectangle intersection problem

By the term *rectangle intersection problem* in the plane, we mean the problem of determining the subset of a set of n iso-oriented rectangles that intersect a given query rectangle Q. Although it has been widely researched in the computational geometry literature it still remains an interesting topic due to its many practical applications.

Edelsbrunner and Maurer (1981), presented a solution to the general rectangle intersection problem with $O(\log^{d-1} n + K)$ time and $O(n \log^d n)$ space complexities, where K is the size of the answer, and d is the number of space dimensions, using layered structures. In a dynamic environment the above time is increased to $O(\log^d n + K)$. Edelsbrunner and Maurer adopted a unified approach for orthogonal intersection searching which is particularly suited to rectangle intersection searching.

Lee and Wong (1981) have solved the intersection problem in $O(\log^{2d-1} n + K)$ time and $\theta(n \log^{2d-1} n)$ space complexities when all inputs are not known before the pre-processing stage. They also presented a unified approach which solves the same problem when all inputs are known at the pre-processing stage, in $O(n \log n + n \log^{2d-3} n + K)$ time. Both results hold for the containment and edge intersection problems as stated in Lee and Wong (1981).

Later research concentrated on individual problems of rectangle containment and enclosure. The solutions of such individual cases are usually more complex and have worse time bounds than the general case, since the special characteristics of each distinct case, which are not crucial in the general case, should now be taken into consideration and handled separately. The three separate sub-cases are:

- The problem of determining the rectangles with one or more of their vertices contained in the query rectangle (rectangle containment). This case has been solved in d-dimensions using $O(\log^d n + K)$ query time and $O(n \log^{d-1} n)$ space executing a two-dimensional range search (Willard and Luecker, 1985).
- The problem of determining the rectangles that enclose the query rectangle (rectangle enclosure). This particular case has been solved (Bistiolas et al., 1993) using $O(\log^{2d-1} n + K)$ query time and $O(n \log^{2d-2} n)$ space (dynamic version).
- The problem of determining the rectangles that intersect the query rectangle, but do not belong to the solutions of the cases previously described. More formally, the rectangles which belong to the set of answers are those whose horizontal (vertical) sides enclose the corresponding sides of the query and at least one of their vertical (horizontal) sides is enclosed by the corresponding sides of the query (see Fig. 1, the intersection between Q and R_2), the so-called 'cross' rectangle intersection case. This has been solved in d dimensions using $O(\log^{2d-3} n \log \log n + K)$ time and $O(n \log^{2d-3} \times n)$ space complexities for the unrestricted d-dimensional space, where n is the number of the rectangles, K is the size of the answer and $O(\log^{d-1} M + K)$ time and $O(n\sqrt{\log M}^{2d-3})$ space complexities for the restricted universe (grid), where n is the number of the rectangles, K is the size of the answer and M is the upper limit of the grid coordinates (Kapelios et al., 1995).

3. The rectangle intersection approach in negotiation decision-making

In negotiation decision-making, each one of the negotiating parties wants to know, across most of the negotiating stages, which of his/her views, arguments or ideas are the most acceptable among the negotiating players and to which degree. This knowledge allows to focus on separate issues or combinations of items under negotiation, which incorporate maximum preference similarities, leading thus to higher levels of consensus while minimizing concession.

For the sake of simplicity the two-dimensional case—or better two-argument case—will be discussed, in which each negotiating party defines two intervals, one for each argument. These intervals represent the preferred range of values for each party. Assume that all negotiators have declared their individual preferences or value intervals, which form the "decision scene". When a new preference query is declared it is represented by its coordinates in the two axis forming a four-element tuple $[x_{Q\mathrm{l}}, x_{Q\mathrm{r}}, y_{Q\mathrm{b}}, y_{Q\mathrm{t}}]$. A random argument of those forming the scene is represented by the tuples $[x_{i\mathrm{l}}, x_{i\mathrm{r}}, y_{i\mathrm{b}}, y_{i\mathrm{t}}]$ $(1 \leqslant i \leqslant n)$. The pairs $[x_{i\mathrm{l}}, y_{i\mathrm{b}}]$, $[x_{i\mathrm{r}}, y_{i\mathrm{b}}]$ represent a two-dimensional point or, more precisely, vertices of a rectangle. The 2-fold structure is formed as a two-dimensional range tree for the $2n$ points:

$$[X_{1l}, Y_{1\mathrm{b}}], [X_{1\mathrm{r}}, Y_{1\mathrm{b}}], \ldots, [X_{n\mathrm{l}}, Y_{n\mathrm{b}}], [X_{n\mathrm{r}}, Y_{n\mathrm{b}}]$$

This is modified in the second dimension in order to be combined in a priority search tree (see Fig. 2). In other words, every node v of the range tree in the first layer (see Fig. 2—the T tree) points to a priority search tree for the $[Y_{i\mathrm{b}}, Y_{i\mathrm{t}}]$ pseudopoints, which correspond to leaves of the sub-tree rooted at v (see Fig. 2—the T_v tree). In this way all the coordinates of the rectangles are stored in the

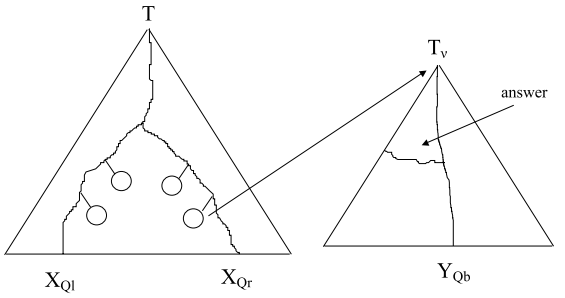

Fig. 2. The 2-fold structure.

structure. Finally, the priority trees are organized having the largest value y_{it} stored in the root.

3.1. The algorithm

Let T be the first layer of the structure and T_v the second layer which is associated with each internal node v of T. The algorithm that answers the query is described below:

Step 1: Search for X_{Ql} και X_{Qr} in tree T. In this way two paths are defined, as illustrated in Fig. 2. Let $u_1, u_2, \ldots, u_i$, be the sons of the nodes forming the paths to X_{Ql} και X_{Qr}, without belonging to the paths themselves and they also lie between the two paths. Observe that the trees T_{u1}, T_{u2}, ..., T_{ui}, of the second layer store all the rectangles, having at least one of their vertical sides included between the lines $X_1 = X_{Ql}$ και $X_2 = X_{Qr}$.

Step 2: Search for Y_{Qb} in every second layer tree T_{ui} (Fig. 2 shows only one of them). The answer is contained between the search paths to Y_{Qb} and the leftmost leaf of the tree. We traverse top-down all T_{ui} trees and report all pairs $[Y_{ib}, Y_{it}]$ (corresponding rectangles) until the condition $Y_{Qt} < Y_{it}$ holds (the largest Y_{it} values are pushed to the root).

Additional inspection is needed for the path leading to Y_{Qb}, the rightmost of the examined paths. This is so because in this path pseudopoints can be stored (corresponding rectangles) with $Y_{ib} > Y_{Qb}$ while all the paths on the left have surely stored points with $Y_{ib} < Y_{Qb}$. So, for each possible answer from this path, it must be examined if the above condition holds also. This additional traversal of the path does not change the overall time complexity.

3.2. Time and space analysis

Theorem. *The space occupied by the structure is* $O(n \log n)$, *while the answer is calculated in time* $O(\log n \log \log n + K)$, *where K is the cardinality of the answer and n the number of rectangles.*

Proof. The total space occupied by our two-layered structure is $O(n \log n)$, because every point $[x_{il}, Y_{ib}]$, or $[x_{ir}, Y_{it}]$ is stored in $O(\log 2n)$ nodes of the range tree (first layer) and every priority tree occupies $O(n_i)$ space ($n_i \leqslant n$) for every one of the n_i pseudopoints $[Y_{ib}, Y_{it}]$, that are stored in it. As the time bound in the static case is concerned, we consume $O(\log^2 n + K)$ since in Step 1 we traverse two paths of length $\log n$ and in Step 2 we answer the enclosure query using a priority tree in $O(\log n_i + K_i)$ with $n_i \leqslant n$ (K_i is the answer taken by every second layer tree T_{ui}) for every node $u_1, u_2, \ldots, u_i$, found in Step 1. The time complexity can be reduced to $O(\log n \log \log n + K)$ by using extended priority trees (Fries et al., 1987) as the second layer of our structure instead of a simple priority search tree. □

3.3. The d-dimensional case

Considering the problem in d dimensions, it can be stated that the d-dimensional rectangles that belong to the set of answers are those that enclose the query rectangle in $d-1$ dimensions and at least one of their sides is partly included, without the vertices, in the query rectangle, one according to each dimension. More precisely, let $[x^1_{il}, x^1_{ir}], [x^2_{il}, x^2_{ir}], \ldots, [x^d_{il}, x^d_{ir}]$ be the pairs of coordinates in each of d axis of the rectangle and $[x^1_{Ql}, x^1_{Qr}], \ldots, [x^d_{Ql}, x^d_{Qr}]$, the corresponding pairs of the query rectangle. It is necessary now to find all the rectangles whose $d-1$ pairs of coordinates enclose as intervals the corresponding pairs (intervals) of the query rectangle and at least one of the two coordinates of the last pair is contained in the corresponding pair of the query rectangle.

To expand the structure to more than two layers, in order to answer the query in d dimensions, it must be noticed that the layers added must solve the problem of segment enclosure. Having these observations in mind, one must simply add two

layers of range trees for every dimension above two, where the problem is set. Each one of the two layers of range tree stores rectangles according to one of the two coordinates $[x_{il}^j, x_{ir}^j]$ in the j dimension, and one has simply to search the first tree for rectangles having $[x_{il}^j < x_{Q\mathrm{l}}^j]$ and the second tree for the ones having $[x_{ir}^j > x_{Q\mathrm{r}}^j]$. As a result, the structure in d dimensions will be a $(2d-2)$-dimensional range tree modified in the last layer, as described previously, such that every node of layer $2d-3$ points to a priority search tree.

The next step is to determine the rectangles that have at least one side partly contained in the query rectangle in the first dimension and enclose the query rectangle in the other $d-1$ dimensions. The other $d-1$ symmetric instances of the problem are solved in an analogous way. The first layer of the structure stores the $2n(2d-2)$-dimensional points of the $[x_{il}^1, x_{il}^2, x_{il}^3, \ldots, x_{il}^d], [x_{ir}^1, x_{ir}^2, x_{ir}^3, \ldots, x_{ir}^d]$ in order to find the rectangles that have at least one coordinate according to dimension 1 in the range $[x_{Q\mathrm{l}}^1, x_{Q\mathrm{r}}^1]$. The subsequent $2d-4$ layers are range trees that were used to solve the enclosure problem according to $d-2$ dimensions, each pair of layers solves the problem in one dimension. At layer $2d-3$ the problem is already solved according to $d-1$ dimensions and it remains to select from those rectangles found the ones that also enclose the query according to the d dimension. Using the priority tree in the same way as with two dimensions, the answers can be located in only one layer.

From the discussion above, it follows that the overall query time in d dimensions is $\mathrm{O}(\log^{2d-2} n + K)$. A time reduction in $\mathrm{O}(\log^{2d-3} n \log\log n + K)$ can be achieved if the ordinary priority tree of the last layer is replaced with an extended priority search tree. The space bound of the query is $O(n\log^{2d-3} n + K)$, since each one of the n rectangles is stored in $2d-3$ layers of range trees and the priority tree used for the last layer occupies linear space. A more detailed analysis of the algorithm not only in the dynamic but also in the static case can be found in Kapelios et al. (1995).

4. Implications

This method provides to groups of decision makers or negotiating parties the ability to compute in real-time all relevant arguments in terms of corresponding rectangles. These are enclosed by a given rectangle in all dimensions but one, in which they intersect. This rectangle intersection computational method, due to the data structures it uses, can be easily integrated not only in relational database systems but also in stand-alone decision methods. In the former case it can be used to compare preference value intervals while in the latter to create an algorithmic computational layer, thus leading the decision-making method to more specialized choices.

The computational geometry methods presented in this paper are suitable for the implementation of e-commerce negotiation systems since:

- They can be integrated into the back ends of electronic catalog systems, based on relational databases and existing EDI systems. Furthermore, they can handle heterogeneous databases, allowing negotiation support systems to work in diverse organizational environments. In this way, the quantity and quality of information available to the negotiating parties can be expanded and the negotiation systems have the potential to leverage their capacity to manage large quantities of data.
- They can be implemented in a distributed fashion, allowing the efficient scaling of corresponding decision support systems, supporting both real-time and asynchronous negotiations. Also, they can help the decision maker bargain with several other parties simultaneously.
- They can take direction and input from either a human user or another software program and negotiate a solution with other negotiating tools.
- They can assist the decision maker to negotiate along multiple preference dimensions apart from price for the products or solutions he is searching for, such as delivery time, financing terms, etc.
- They support the ability to propose and counter-propose solutions or withdraw from the bargaining process if an agreement is not reached.
- Finally, they can provide the decision maker with a means of visualizing pair-wise comparisons in two dimensions through the

representation of the different views of the negotiating parties as iso-oriented rectangles.

5. Conclusions

This paper presented the mapping of negotiation and group decision-making situations as known geometrical problems which can be solved in real-time. It also discussed the capability of such an approach in integrating negotiation support systems with legacy systems. In particular, a specific exemplary solution was examined, the so-called cross-rectangle intersection problem (Kapelios et al., 1995), which allows the comparison of different multi-preference views.

The paper also considered the implications and restrictions of applying such methods in decision support systems while taking into account known difficulties such as the integration of such methods with other approaches (e.g. multi-criteria decision-making). Future research will aim at designing a decision support system, which will integrate such computational methods and experiment with real-time implementation in decision-making situations. Such a case exists when, for example, intelligent agents evaluate bids or offers, while taking into account existing constraints, beliefs and preference intervals (Papazoglou, 2001), one of the most common real situations in electronic commerce and real-time negotiations between trading parties.

References

Agarwal, R., Prasad, K., 1994. A blackboard framework for the design of group decision support systems. Behaviour and Information Technology 13 (4), 277–284.

Bellucci, E., Zeleznikow, J., 1998. A comparative study of negotiation decision support systems. In: Proceedings of the Thirty-First Hawaii International Conference on Systems Sciences, vol. 1. IEEE Computing Society, Los Alamitos, CA, USA, pp. 254–262.

Bistiolas, V., Sofotassios, D., Tsakalidis, A., 1993. Computing rectangle enclosures. Computational Geometry Journal: Theory and Applications 2 (6), 303–308.

Chaudhuri, S., Dayal, U., Ganti, V., 2001. Database technology for decision support systems. Computer 34 (12), 48–55.

Copeland, K.W., Hwang, C.J., 1997. Streamlining procurement through electronic commerce: An internet approach. In: Khosrowpour, M. (Ed.), Proceedings of the 1997 Information Resources Management Association International Conference. Vancouver, BC, Canada, pp. 357–363.

De Sousa, M.S.R., Mattoso, M., Ebecken, N.F.F., 1999. Mining a large database with a parallel database server. Intelligent Data Analysis 3 (6), 437–451.

Edelsbrunner, H., Maurer, H.A., 1981. On the intersection of orthogonal objects. Information Processing Letters 13 (4–5), 177–181.

Foroughi, A., Perkins, W., Jelassi, M.T., 1995. An empirical study of an interactive, session-oriented computerized negotiation support system (NSS). Group Decision and Negotiation 4, 485–512.

Franklin, M.K., Reiter, M.K., 1996. The design and implementation of a secure auction service. IEEE Transactions on Software Engineering 22 (5), 302–312.

Fries, O., Mehlhorn, K., Naeher, S., Tsakalidis, A., 1987. A $\log\log n$ structure for three-sided range queries. Information Processing Letters 25, 269–273.

Jarke, M., Jelassi, M.T., Shakun, M., 1987. Mediator: Towards a negotiation support system. European Journal of Operational Research 31 (3), 314–334.

Kapelios, V., Panagopoulou, G., Papamichail, G., Sirmakessis, S., Tsakalidis, A., 1995. The 'Cross' rectangle intersection problem. The Computer Journal 38 (3), 227–235.

Lee, D.T., Wong, C.K., 1981. Finding intersection of rectangles by range search. Journal of Algorithms 2, 337–347.

MacDowell, N., 1993. Electronic commerce—A new partner in support of public policy. CMA 67 (7), 38.

Min, H., Galle, W.P., 1999. Electronic commerce in business-to-business purchasing. International Journal of Operations and Production Management 19 (9), 909–921.

Papazoglou, M.P., 2001. Agent-oriented technology in support of e-business. Communications of the ACM 44 (4), 71–77.

Shea, G.F., 1983. Creative Negotiating. CNI Publishing, Boston.

Vetchera, R., 1994. Composite alternatives in group decision support. Annals of Operations Research 51, 197–215.

Willard, D.E., Luecker, G.S., 1985. Adding range restriction capability to dynamic data structures. Journal of the Association for Computing Machinery 32, 597–617.

Wong, S.T.C., 1994. Preference-based decision-making for cooperative knowledge-based systems. ACM Transaction on Information Systems 12 (4), 407–435.