4. Gerstner, W.: Spiking Neurons. In: Maass, W., Bishop, C.M. (Eds.) Pulsed neural networks. MIT Press (1999)
5. Hebb, D.: The organization of behavior. John Wiley and Sons, New York (1949).
6. Izhikevich, E.M.: Simple Model of Spiking Neuron. In: IEEE Transactions on Neural Networks, Volume 14, Number 6, November 2003. IEEE (2003)
7. Kempter, R., Gerstner, W., van Hemmen, L.: Hebbian learning and spiking neurons. Physical Review E, 59:4498-4514. (1999)
8. Maass, W.: Computing with Spiking Neurons. In: Maass, W., Bishop, C.M. (Eds.) Pulsed neural networks. MIT Press (1999)
9. Maass, W.: Networks of spiking neurons: the third generation of neural network models. In: Transactions of the Society for Computer Simulation International, vol. 14, issue 4 (December 1997), pp. 1659-1671. (1997)
10. Salerno, M., Casali, D., Constantini, G., Carota, M.: Fully Asynchronous Neural Paradigm. In: Proceedings of the 15th IEEE Mediterranean Electrotechnical Conference (MELECON 2010), pp. 1039-1043. IEEE (2010)
11. Seung, H.S.: Learning in Spiking Neural Networks by Reinforcement of Stochastic Synaptic Transmission. In: Neuron, Volume 40, Number 6, 18 December 2003, pp. 1063-1073. Elsevier (2003)
12. Zuters, J.: Spiking Neural Networks to Detect Temporal Patterns. In: Haav, H.M., Kalja A. (eds.) Frontiers in Artificial Intelligence and Applications, vol. 187, Databases and Information Systems V - Selected Papers from the Eighth International Baltic Conference, DB&IS 2008, pp. 131-142. (2009)

Information Systems and AI Technologies

Interactive Inductive Learning System: The Proposal

Ilze Birzniece

Riga Technical University, Department of System Theory and Design,
Kalku 1, LV – 1658, Riga, Latvia
ilze.birzniece@rtu.lv

Abstract. Inductive learning system learns classification from training examples and uses induced rules for classifying new instances. If a decision cannot be inferred from system rule base, a default rule is usually applied. No approaches with human interaction exist that would provide model of interactivity appropriate for dealing with non-classifiable instances. In the paper a new interactive approach is proposed where in uncertain conditions interactive inductive learning system can ask for human decision and improve its knowledge base with the rule derived from this decision. Problems and solutions of incorporation of human-made decision into rule base and aspects of choosing between static and incremental learning algorithms are analyzed in the context of proposed approach.

Keywords: Inductive learning, interactive inductive learning, machine learning, human-computer interaction, integrity constraints

1 Introduction

The ever-growing importance of machine learning in multiple problem domains has been highlighted in many articles, e.g. [1,2,3,4,5, and 6]. In general machine learning algorithms are domain independent. However, there are domains where not only a high predictive accuracy, but also interpretability of generated descriptions or patterns of underlying structures is essential. Different applications may demand different description forms, i.e., a reasoning system should be able to transform its results from one form of representation to another. Inductive learning algorithms are more preferable than other machine learning methods in systems, where understanding of decision making steps and further processing of classification results is needed. Expert systems are examples of such systems where the rules induced by learning algorithms can be used [4].

Inductive learning algorithms are widely used in machine learning tasks and they hold a strong position as reliable classification methods that can explain their decision making process [3]. Still the inductive learning has capabilities and potential to improve its performance and to extend its area of application. One of the problems to be solved in inductive learning is inability of classifier to always derive the class membership for new instance using only its existing knowledge base.

There are three main goals of this paper: (1) to show the need for the additional approach for dealing with instances that can't be classified using induced rules set; (2) to prove that non-classifiable instances can be handled with the help of human; (3) to explain the procedures of human-classified instance incorporation into existing rule base. This paper extends previously started research [7] on outlining interactive inductive learning system.

The paper sections are organized as follows. Section 2 discusses inductive learning, particularly static and incremental learning approaches, subsequently bringing forward problems with new instance classification. Section 3 is dedicated to looking into the problems of non-classifiable instances and their existing solutions. Section 4 gives an overview of different existing approaches reported in related work on interactive inductive learning. Considering advantages and drawbacks of other described approaches, the new system of interactive inductive learning is proposed in Section 5. As the proposed system involves human in instance classification, the potential problems of adoption of human decision and their solutions are discussed in Section 6. Conclusions are given in Section 7.

2 Inductive learning

Induction is a process of conversion of particular facts into general regularities. In computer science inductive learning is learning by example, where a system tries to induce a general rule from a set of observed examples [8]. This involves classification – assigning the name of a class to every particular input. Classification is important to many problem solving tasks. Inductive learning methods are considered attractive for many real-life applications (e.g., medical diagnostic [9], industrial visual inspection [10]), most due to their interpretability.

In general the classification task with inductive learning involves classifier forming and applying. The process of classifier forming consists of two subtasks, classifier training and testing. Example set can be accumulated form observations, generated by expert or from both. The class for every single record in this set is known.

In this paper two terms – "example" and "instance" - are used. Although they both refer to separate data unit described by particular attributes and values, their meanings are specified in the context of paper. Using term "example" unit for classifier training is meant (item in training set), whereas "instance" denotes new unit to classify with no classification known. In other words, observed unit in classifier training stage is called "example" while unit in classifier applying stage is called "instance". Instance can turn into example if it is classified and joint to training set.

2.1 Static and incremental learning methods

There are several dimensions along which learning algorithms can be classified, e.g., form of classifier (decision tree or rules), position of hyperplanes (axis-parallel or oblique). Depending on the way of learning, inductive learning methods can be divided in incremental and nonincremental (or static) ones. In context with problem addressed in this paper it is worth to go into detail with these two types of learning

algorithms. According to [11], static algorithms are appropriate for learning tasks in which a single fixed set of training examples is provided while incremental algorithms are appropriate for learning tasks in which there is a stream of training examples. In the incremental case, the algorithm revises the current concept definition in response to each new training example, in order to avoid rebuilding the whole classifier each time a new example is observed. Incremental learning algorithms select training instances more carefully thus resulting in smaller (more general) tree, if classifier forms a decision tree [11].

In the real-world domains two problems known as hidden context and concept drift are faced [12]. Hidden context means that target concept may depend on variables, which are not given as attributes. When changes in hidden context induce changes in the target context the problem of concept drift appears. These problems are discussed in more details in [12, 13]. Incremental learning has a facility to adapt to changes in the target concept when creating classifier for real-world data streams [12]. Such ability is hindered or even impossible for non-incremental learning methods. Incremental classifiers in form of rules take advantage over decision trees by not being hierarchically structured thus allowing updating or removing concept description without heavily affecting the learning efficiency [12]. Incremental learning algorithms can be divided in tree groups depending on example memory they have.

1. *Full example memory* stores all training examples, which allows for efficient classifier restructuring and good accuracy but needs a huge storage. Such algorithms are ID5, ID5R, ITI [11,13].
2. *No example memory* stores only statistical information and thus loses the accuracy but saves a storage space. Examples of this group are ID4, STAGGER, AQ11 algorithms [13].
3. *Partial example memory* only stores selected examples, which is compromise between storage space and accuracy. Most popular methods with partial example memory are HILLARY, FLORA, AQ-PM [12,13].

One of the reasons that motivate to study and implement learners with partial example memory is awareness of the fact that "humans not only store concept descriptions, but also remember specific events" [13]. Inductive learning system has to choose these specific events. For the partial example memory there are three main techniques used for choosing examples (events) to store.

1. Select representative examples. Some criteria that recognize the importance of example is needed.
2. Remember consecutive sequence of examples over window of time. This window can be fixed or changing. The size of the window has to be established.
3. Keep extreme examples that lie on or near the boundaries of current concept description. These border examples need to be updated over the time.

All these techniques have problems to be solved to maintain classifier accuracy. No such problems appear with static learning algorithms, so it may not be worth to implement an incremental algorithm for constant or slowly changing learning set. Every static learning method can be used in incremental way by simply storing all past training examples, adding them to new ones, and re-applying the algorithm [13]. Disadvantages of using static algorithm in incremental manner include the need to

store all training examples. On the other hand, such a system is less sensitive to ordering effects than incremental learners because it uses all available data to build classifier. Ordering effects become apparent when different ordered sequences of examples lead to different learning results [14]. Most popular static learning algorithms are ID3, C4.5, CART, AQ, and CN2 [2,4].

2.2 Problems with new instance classification

There are still problems to solve in inductive learning. One group of problems is conflicts that can emerge in the classification stage of new instance. Such conflicts occur in the following situations [15]:

1. At least two rules predicting different classes cover an instance;
2. None of the rules fit an instance.

First problem has been solved with various methods, e.g., ones described in [16, 17]. In situation when the instance satisfies more than one rule inductive learning algorithm AQR (inductive learning system that uses the basic AQ algorithm) predicts the most common class of training examples that were covered by those rules [17]. Other common, easy and efficient approach is the best rule strategy that takes in count consistency and completeness of every rule to measure its quality [15,16].

The existing approaches for solving the problem of second situation most often include the use of the default rule. This approach does not work appropriately in all domains. The problem of dealing with instances that can't be classified is discussed in next section.

3 Background in dealing with non-classifiable instances

Even if classifier accuracy after test results was good, it often happens that none of the rules fits the instance or tree cannot classify the incoming instance. It is possible that some unique or exceptional instance has arrived, or rules do not cover all areas in problem domain.

There are several methods to deal with this problem. AQ and CN2 algorithms apply the default rule that predicts the most common class in particular data set if none of generated rules fits the instance [17]. This approach is comprehensible and acceptable but it does not work well in all domains. If data set contains many classes and, moreover, if all of them occur equally frequently, assigning one certain class to all unclassified instances will not lead to high accuracy of the classifier. It is obvious, that using default rule in domain with 10 classes, which occur in similar frequency, could lead to average classification accuracy 10% for non-classifiable instances with every 10^{th} instance classified correctly. That is inexcusably low result.

Within the domains where it is more critical to detect one class than the others, e.g., medical diagnosis, this most relevant class is usually assigned, when classification can't be clearly made, with the purpose not to make a serious mistake. Yet such a method is not appropriate for all situations. Unbalanced data sets, where one class occurs much more frequently than other, often appear in many practical

applications and it is important to deal with classification also in these domains as accurate as possible [18].

In [12] incremental rule learning based on example nearness from numerical data streams is presented. This learning system uses two types of rules. Consistent rules classify new instances by covering them and inconsistent rules classify instances by distance as the nearest neighbour algorithm. Instance not covered by any rule is classified based on the label associated with the reliable rule that has smallest distance. This approach is specific and cannot work with nominal attributes where nearness can't be calculated.

Summarising the existing approaches, one can assume that if in traditional process of classification no rule that can classify a particular instance is found, a guess is performed. Most frequently this "guess" means the default rule.

As the classification tasks are getting more complicated, computer program often meets the situations when classification can't be made with existing rule base. Leaving a decision making to some predefined algorithm is not always the best solution, and it is also not the only possibility. Some machine learning systems attempt to eliminate the need for human interaction, while others adopt a collaborative approach between human and machine. In situation when new instance can't be unequivocally classified, a collaborative approach between machine and system user (human expert) would be useful. The paper proposes to construct new inductive learning system that could ask human to make a classification in previously described situation and improve its knowledge base with a rule derived from this „experience".

4 Related work in interactive learning

Several research papers of last twenty years refer to concept „interactive inductive learning" or explore the idea of human interaction in concept learning process. Systems and approaches proposed in these papers are from distinct fields and suggest the different types of human interaction. The following types of human interaction are described in [19,20,21,22,23,24, and 25]:

1. Systems where human feedback is asked to evaluate only the given result (decision or prediction).
2. Systems that learn concept classification based on classification by human.
3. At first human is giving his/her knowledge to system and affirming the rules that are induced by the system afterwards.
4. Human evaluates and selects rules induced by system in classifier forming stage.
5. Learning systems where human is the learner and computer should be able to interact in user-friendly way.

Full survey of above-mentioned related works can be found in [7]. Interaction with a human in these systems takes place in different phases of learning. Depending on phase in inductive learning process where human interaction is expected, the diagram for abstract comprehension of different existing approaches to interactive inductive learning has been created (Fig. 1).

In the stage of classifier forming and testing, data are passed to the learning system (phase A) and rules are given to output (phase B). In the stage of classifier applying, new instance (instances) with no classification is provided to the classifier (phase C) and a decision of its class is expected to be received (phase D). The circles with letters in Figure 1 denote the particular phase in inductive learning where the interaction is expected. The following human actions are possible in learning phases:

A Forming of classifier learning data, selection of examples for input.
B Extraction, processing, and selection of rules.
C New instance handling in classifier applying.
D The decision handling after classification of new instance.

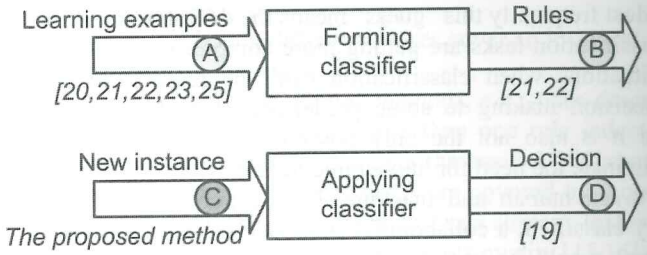

Fig. 1. Phases when human can interact with classifier

Human interaction in classifier training (phase A in Fig. 1) has been practised in two ways: as learning from the examples and categories shown only by the human [21,25] or learning from human answers to questions generated by the system [17,23]. Systems described in [21] and [22] ask for the human advice both in forming of input data and evaluating training results (phases A and B in Fig. 1). According to [19], the human feedback is asked after decision to improve the search results (phase D in Fig. 1).

None of methods described in related work provide appropriate model of interactivity for solving the inductive learning problem with classifying instances that do not fit any of rules in knowledge base. As stated in Section 3, current approaches to this problem do not work well in all situations. Therefore new computer–human interactivity model for dealing with non-classifiable instances approach is proposed.

5 The proposed interactive inductive learning system

Summarising advantages and drawbacks of previously overviewed approaches to human involvement into inductive learning the new system dealing with non-classifiable instances is proposed. This system would interact with human in phase C (Fig. 1) and only if it is needed.

The proposed system involves human when it meets new instance not consistent with the rule base, thereby replacing default rule. Whole cycle of interactive inductive learning system involving human is depicted in Fig. 2.

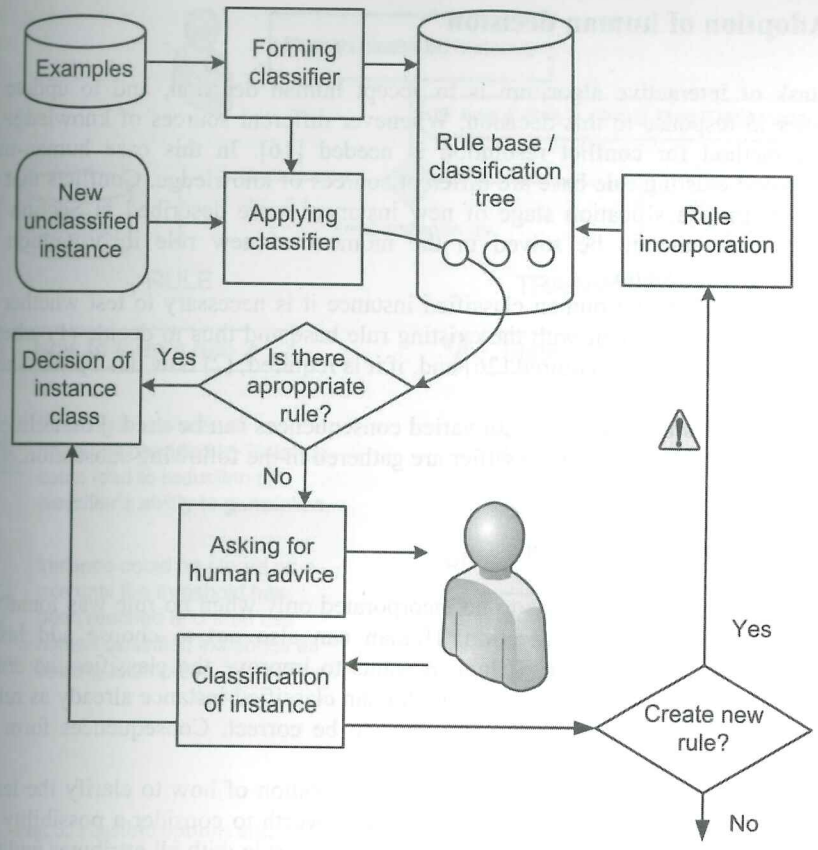

Fig. 2. Classification with human interaction

The structure of the system tends to provide mutual independence for both classifier and human. This position includes the accomplishment of the following properties.

- System is not dependent on human; it in principle operates by itself.
- Human isn't bended to the system to answer its questions systematically.

More benefits can be gained from expert knowledge, not only correct classification of every single instance passed to the expert. Human decision can also be used to improve the rule base as the human-given advice can be saved and formed as a new rule to be incorporated into existing rules. This is done only if the human wants his decision to be saved for further use. However, the possible consistency threats of incorporating human-based rules are to be taken into consideration. It is important to feature the system with integrity constraints and a control mechanism for rule consistency check between existing knowledge base and new input information. There are several options how exactly to add new knowledge to the classifier. These questions will be discussed in next section.

6 Adoption of human decision

The task of interactive algorithm is to accept human decision, and to update the classifier in response to this decision. Whenever different sources of knowledge are used a method for conflict resolution is needed [16]. In this case human-made decision and existing rule base are different sources of knowledge. Conflicts that can emerge in the classification stage of new instances were described in Section 2.2. Conflicts that have to be solved in the moment of new rule incorporation are discussed in this section.

Before incorporating human classified instance it is necessary to test whether the rule obtained is consistent with the existing rule base and thus to decide (1) whether an update of rule base is required [26] and, if it is required, (2) how this update should be performed.

Different learning strategies with varied consequences can be used. Possibilities of human decision processing in classifier are gathered in the following subsection.

6.1 Knowledge incorporation

First of all, new knowledge has to be incorporated only when no rule was found and human was asked for classification. Human can also not to choose add his/her decision to existing rule base. If human wants to improve the classifier the crucial question to discuss is whether to treat the human classified instance already as rule or just as new training example that is known to be correct. Consequences form this decision are depicted in Fig. 3.

If a human decision is treated as a rule, the question of how to clarify the length and essential attributes of this rule emerges. It is worth to consider a possibility that there are many attributes describing instance. Storing rule with all attributes and their values represented in instance is ineffective in both from memory space and generalization viewpoint. In longer term this approach could reduce classifier's ability to generalize thus decreasing predictive accuracy.

If new classified instance is treated as new example to learn from, further actions depend on learning strategy chosen for classifier in general. In case static inductive learning method is used to train classifier, new example is enclosed in training set and classifier should be created from scratch. Such an approach is time and computational resources consuming and ineffective comparing to incremental learning methods if instances are presented serially [11]. This may be not the critical judgement as long as proposed interactive learning system is not offered as real-time learning system. Obviously it depends on particular domain and learning task. A step towards computationally more effective classifier rebuilding would be setting a threshold and waiting for more examples to come and only then incorporating examples into classifier when limit of instances count is reached. Until this limit is reached instances could be stored as rules, thus merging both opportunities – treating human decision temporary as rule and incorporating it within the training examples in order to generalize it afterwards.

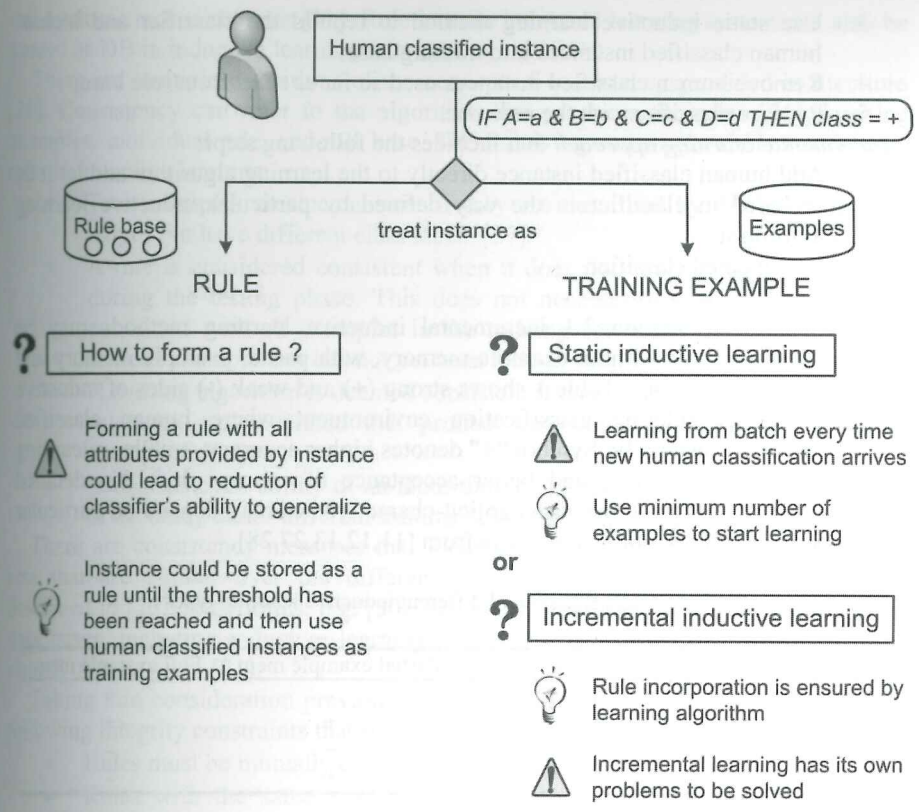

Fig. 3. Possible options and problems with human classified instance incorporation

Finding the examples number for threshold is a subject of further studies and most likely will depend on (1) time needed to build classifier from initial training set and (2) frequency of human classified instance appearance. The higher are the costs for learning from batch, the higher the minimum of examples limit will be set.

Use of incremental learning method has its advantages and drawbacks. The use of incremental method is desirable because incorporation of human-made decision and rule base integrity will be under competence of the learning algorithm. On the other hand, using incremental learning algorithms the open question of examples' memory size has to be solved [13].

Considering options from Fig. 3, the following preferable approaches for human knowledge incorporation into existing rules set arise.

Threshold based static learning approach that includes the following steps:

1. Set the threshold - positive number, representing number of instances classified by human.
2. Store human classified instance as a rule with all attributes and their values unless the threshold has been reached;

3. Use static inductive learning method to rebuild the classifier and include human classified instances into training base;
4. Remove human classified instances used so far as rules from rule base;
5. Replace classifier with the new one.

Incremental learning approach that includes the following steps:

1. Add human classified instance directly to the learning algorithm and let it be included in classifier in the way defined by particular inductive learning algorithm;
2. Use updated classifier.

As described in Section 2.1 incremental inductive learning methods may be divided into algorithms without example memory, with partial example memory and with full example memory. Table 1 shows strong (+) and weak (-) sides of inductive learning types considering classification environment where human classified instances arrive occasionally. Symbol "+" denotes higher accuracy, smaller relearning costs, smaller storage costs, and better acceptance of concept drift. No detailed comparison has been given because explicit characteristics may depend on particular algorithm used. Table 1 summarizes data from [11,12,13,27,28].

Table 1. Characteristics of different inductive learning types

	Static	No example mem.	Partial example mem.	Full example mem.
Accuracy	+	-	-	+
Relearning costs	-	+	+	+
Storing costs	-	+	+	-
Concept drift	-	+	+	-

According to [13], static learning approach could lead to higher accuracy of classifier, compared to incremental approach using partial or no examples' memory.

If the expected frequency of additional instance arrival is high, then incremental learning algorithm should be considered. Depending on storage amount, desired accuracy, and other considerations, the method with full, partial or no examples' memory is to be chosen.

The choice of learning strategy becomes an optimization task when predictive accuracy, relearning costs, storing costs, and other relevant characteristics should be weighted. There are no context and domain independent reasons to define one learning method to be better than others [29]. Which method to choose depends on type of the problem, requirements, constraints, background knowledge, and many other issues.

6.2 Rule base consistency

The integrity maintenance methods have to be defined to prevent updates from violating integrity constraints [30]. General database (DB) integrity constraints are described in literature, e.g. [30,31]. In design of a DB it is important activity to identify the integrity constraints that must hold on the DB, and that are used to detect

and evaluate inconsistencies [31]. Rule base in inductive learning system can also be treated as DB in inductive learning context.

There are multiple definitions of the concept of consistency provided in literature [26]. Consistency can refer to the algorithm that produces the rules [26], learning examples, individual rule, and rule base. The following statements about consistency refer to inductive learning.

- Two examples are inconsistent if they are described by the same variable values but have different class labels [27].
- A rule is considered consistent when it does not cover a negative example during the testing phase. This does not necessarily mean that the rule is consistent with all examples in the training set because it may contradict with an example that has not yet been tested.
- "Learning algorithm is deemed consistent if under different training sessions, rule sets are generated that produce the same classification of unseen examples" [26].
- "Consistency is ability of an algorithm to extract rules with the same degree of accuracy under different training sessions" [26].

There are consistency measures that focus on rule learner ability to provide rule sets that are similar over the different runs of learning method, e.g., measure described in [26]. Algorithms, e.g. [30], that can be used for integrity maintenance in data bases, including inductive learning rule bases, have been developed and used already for the years [31].

Taking into consideration previous statements, in inductive learning rule base the following integrity constraints that must hold [26]:

- Rules must be mutually exclusive, non-overlapping.
- Rules with the same attributes and values mustn't assign different class labels.

Let's discuss how proposed approaches (threshold based static learning and incremental learning) for human decision incorporation into existing rule base hold the integrity constraints. The first statement is important for keeping the rule base clean and avoiding unnecessary rules. If the initial rule base in classifier meets demands of non-overlapping and non-contradictory rules as it should be ensured by inductive learning algorithm which was used for rule generation from batch examples, then both proposed incorporation methods maintain continuous consistency within rule base. This is achieved because new rule is to be added only if no rule can classify the instance thus ensuring consistency of new rule to existing rules set. Of course, the situation can emerge when rules that were consistent are no longer consistent when new instance to be classified arrives. For example, rule base contains rules "*IF A=a AND B=b THEN class* = +" and "*IF C=c AND D=d THEN class* = -". Instance "*A=a AND B=b AND C=c AND D=d*" arrives. Two rules ask to classify this instance in distinct classes. This situation should be handled with approaches described in Section 2.2. This problem can't be managed in rule incorporation moment. It also has to be understood that not always a new unclassified instance will lead to additional rule; rather some existing rules could be changed after rebuilding classifier as the new instance is included in the training set.

Next subsection provides an example of new rule incorporation into existing rule base with threshold based static learning approach (Section 6.1). Incremental learning

approach isn't demonstrated because there are many different incremental methods which operate with new instances accordingly to their algorithms, whereas threshold based static learning approach has not been described before.

6.3 An example

With this elementary example the operation of threshold based static learning approach will be demonstrated. Table 2 shows initial training examples for classifier forming in domain of natural phenomena for season prediction. There are three nominal attributes, namely, temperature, leaf status of trees, and weather. Class label determines the season.

Table 2. Training examples for season prediction

	Temperature	Weather	Trees	Season
1	moderate	rainy	yellow	autumn
2	moderate	sunny	bare	spring
3	high	sunny	green	summer
4	low	sunny	bare	winter

Static inductive learning algorithm (RULES-3 algorithm by Aksoy [3]) has been applied and extracted rule set is given in Table 3.

Table 3. Initial rule base

Rule base
IF weather = rainy THEN season = autumn
IF temperature = moderate AND whether = sunny THEN season = spring
IF temperature = high THEN season = summer
IF temperature = low THEN season = winter

Two thresholds for human-classified instance incorporation in existing rule base are chosen, *one* and *three* instances, respectively. Threshold = 1 means that classifier is rebuilt after every human classified instance. Let's observe how classifiers react to next three instances to classify.

New instance "*trees = green AND temperature = moderate AND weather = rainy*" arrives. None of the existing rules fit the case. Human classifies it as "spring". In example with threshold = 1 classifier is rebuilt and new rule "*IF trees = green AND temperature = moderate THEN season = spring*" added. Another classifier includes full instance with its classification at the end of rules list.

Table 4. Rule base after one human classified instance

Threshold = 1	Threshold = 3
IF weather = rainy THEN season = autumn IF temperature = moderate AND weather = sunny THEN season = spring IF temperature = high THEN season=summer IF temperature = low THEN season = winter **IF trees = green AND temperature = moderate THEN season = spring**	IF weather = rainy THEN season = autumn IF temperature = moderate AND weather = sunny THEN season = spring IF temperature = high THEN season= summer IF temperature = low THEN season = winter **IF trees = green AND temperature = moderate AND weather = rainy THEN season = spring**

Next instance "*trees = green AND temperature = moderate AND weather = cloudy*" appears. First classifier covers it with just created rule. Second classifier transfers instance to human who classifies it as "spring". New temporary rule is added.

Table 5. Rule base after two human classified instances

Threshold = 1	Threshold = 3
IF weather = rainy THEN season = autumn IF temperature = moderate AND weather = sunny THEN season = spring IF temperature = high THEN season= summer IF temperature = low THEN season = winter **IF trees = green AND temperature = moderate THEN season = spring**	IF weather = rainy THEN season = autumn IF temperature = moderate AND weather = sunny THEN season = spring IF temperature = high THEN season= summer IF temperature = low THEN season = winter **IF trees = green AND temperature = moderate AND weather = rainy THEN season = spring** **IF trees = green AND temperature = moderate AND weather = cloudy THEN season = spring**

Third instance is "*trees = bare AND temperature = moderate AND weather = cloudy*". None of classifiers can define a class label for this instance and human decides it is winter. Classifier rebuild should be performed in both classifiers, because threshold in second classifier is reached as third human classified instance arrived.

Table 6. Rule base after three human classified instances

Threshold = 1	Threshold = 3
IF weather = rainy THEN season = autumn IF temperature = moderate AND weather = sunny THEN season = spring IF temperature = high THEN season= summer IF temperature = low THEN season = winter **IF trees = green AND temperature = moderate THEN season = spring** **IF trees = bare AND weather = cloudy THEN season = winter**	IF weather = rainy THEN season = autumn IF temperature = moderate AND weather = sunny THEN season = spring IF temperature = high THEN season= summer IF temperature = low THEN season = winter **IF trees = green AND temperature = moderate THEN season = spring** **IF trees = bare AND weather = cloudy THEN season = winter**

Both classifiers now look the same as the temporary rules have been removed and generalization performed. This small case showed that classifier with lower threshold disturbed human more rarely (threshold = 1 asked for human advice two times while classifier with threshold = 3 did it three times). Classifier with lower threshold managed to generalize previous user decision sooner and next similar instance could be classified by rule, not human. Although immediate instance incorporation gives better results of classifier generalization and is likely to disturb human more rarely than threshold higher than one, it requires more computations. Classifier with threshold higher than one has smaller ability to generalize between the classifier updates. Yet temporary rule helps in situations where exactly the same instance as previous has to be classified and system is expected to disturb human more rarely than threshold based approach without interim rules at all.

7 Conclusion

The paper extends previous research of the author [7] on outlining interactive inductive learning system. The proposed interactive inductive learning system deals with non-classifiable instances by asking human for decision and improving its knowledge base with the rule derived from this human-made decision.

Main improvements since previous research are the following. More related works on computer – human interaction in learning systems have been summarized. Possible solutions of incorporation of human-made decision into rule base have been discussed. Process of creating and adding a rule to the rule base after human classification has been studied.

This paper proposes two models of human-made decision incorporation into rule base. One way is to use threshold based static inductive learning method and relearn classifier from batch after one or more human classified instance arrives. If more than one instance is expected, previous instances are kept as rules until new classifier is built. Other way is to use incremental learning method and incorporate human decision right into classifier as the example comes. Which method to choose depends on particular problem domain, requirements, constraints, frequency of human classification, time needed for classifier rebuilding, learning algorithm, and other conditions. Aspects of choosing between static or incremental learning algorithms have been analyzed.

Potential problems and solutions of incorporation of human-made decision into rule base have been discussed. It is shown that the proposed system can provide rules consistent with existing rule base.

The proposed system has a limitation: in complex domains with many features it could be hard for human to give the classification at once basing on all provided features. In this case a technique should be developed for grouping or structuring the information provided for system user to classify the instance.

The paper describes work in progress. Further, particular algorithms should be chosen and implemented to experiment with both models. Threshold for instance number in static learning approach is a subject of further studies, too, and most likely will depend on time needed to build classifier from initial training set and frequency

of user classified instance appearance. Different types of incremental learning also have to be studied and most appropriate ones chosen.

Acknowledgements. This work has been supported by the European Social Fund within the project „Support for the implementation of doctoral studies at Riga Technical University".

References

1. Street, N.W.: Oblique Multicategory Decision Trees Using Nonlinear Programming. Informs Journal on Computing. No.1, 25--31 (2005)
2. Pham, D.T., Afify A.A.: Applications of machine learning in manufacturing. In: Intelligent Production Machines and Systems. 1st I*PROMS Virtual International Conference, pp. 225--230. Elsevier, Great Britain (2005)
3. Aksoy, M.S.: Dynamic System Modelling Using Rules 3 Induction Algorithm. Mathematical and Computational Applications. No.1, vol.10, 121--132 (2005)
4. Cios, K.J, Kurgan L.A.: Hybrid Inductive Machine Learning: An Overview of CLIP Algorithms. In: New Learning Paradigms in Soft Computing. vol.84, pp. 276--321. Physica-Verlag GmbH, Heidelberg (2002)
5. Kaufman, K., Michalski, R.S.: The AQ18 Machine Learning and Data Mining System: an Implementation and User's Guide. Machine Learning and Inference Laboratory, George Mason University, USA (2000)
6. Ramakrishnan, N.: The Pervasivness of Data Mining and Machine Learning. Data Mining for Software Engineering. Computer, pp. 28--29 (Aug. 2009)
7. Birzniece, I. From Inductive Learning towards Interactive Inductive Learning, paper accepted for the Scientific Proceedings of Riga Technical University, Applied Computer Systems, Riga (2010), in press.
8. Marshall, D: Inductive Learning, http://www.cs.cardiff.ac.uk/Dave/AI2/node144.html
9. Chao, S., Wong, F., Li, Y.: An Incremental and Interactive Decision Tree Learning Algorithm for a Practical Diagnostic Supporting Workbench. In: Fourh Intrnational Conference on Networked Computing and Advanced Information Management. vol.2, pp. 202--207 (2008)
10. Aksoy, M.S.: Applications of RULES-3 Induction System. In: Proceedings of the Innovative Production Machines and Systems (2008)
11. Utgoff, P.E.: Incremental induction of decision trees. Machine Learning. No.4, 161--186. (1989)
12. Ferrer-Troyano, F., Aguilar-Ruiz, J.S., Riquelme, J.C.: Incremental Rule Learning based on Example Nearness from Numerical Data Streams In: Proceedings of the 2005 ACM Symposium on Applied Computing, pp. 568--572. ACM, New York (2005)
13. Maloof, M.A., Michalski, R.S.: Incremental Learning with Partial Instance Memory. Artificial Intelligence, vol.154, No.1-2, 95 - 126 (2004)
14. Cornuéjols, A.: Getting Order Independence in Incremental Learning. In: Brazdil, P. (ed.) Proceedings of the European Conference on Machine Learning. LNCS, vol. 667, pp. 196--212. Springer-Verlag, London (1993)
15. Kwedlo, W., Kretowski, M.: An Evolutionary Algorithn for Cost-Sensitive Decision Rule Learning. In: Raedt, L.D. and Flach, P.A. (eds.) Proceedings of the 12th European

Conference on Machine Learning, LNCS, vol.2167, pp. 288--299. Springer-Verlag, London (2001)
16. Torgo, L.: Rule Combination in Inductive Learning. In: Brazdil, P. (ed.) Proceedings of the European Conference on Machine Learning, LNCS, vol.667, pp. 384--389. Springer-Verlag, London (1993)
17. Clark, P., Niblett, T.: The CN2 Induction Algorithm. Machine Learning Journal. No.3, 261--283 (1989)
18. Zhang, J., Nloedorn, E., Rosen, L., Venese, D. Learning Rules from Highly Unbalanced Data Sets. In: Fourth IEEE International Conference on Data Mining, pp. 571-- 574 (2004)
19. Okabe, M., Yamada, S.: Interactive Web Page Retrieval. In: Active Mining: New Directions of Data Mining. pp. 31--40, IOS Press, Amsterdam (2002)
20. Tanumara, R.C., Xie, M., Au, C.K.: Learning Human-Like Color Categorization through Interaction. International Journal of Computational Intelligence. No.4, 338--345 (2007)
21. Buntine, W., Stirling, D.: Interactive Induction. In: Hayes, J.E., Michie, D., and Tyugu, E. (eds.) Machine intelligence 12: Towards An Automated Logic of Human Thought. pp. 121--137. Clarendon Press, New York (1991)
22. Hadjimichael, M., Wasilevska, A. Interactive Inductive Learning. International Journal of Man-Machine Studies. No.2, 147 -- 167 (1993)
23. Wong, M.L., Laung, K.S.: Data Mining Using Grammar-Based Genetic Programming and Applications. 228 p. Kluwer Academic Publishers, USA (2000)
24. Li, X. et.al.: Learning in an Ambient Intelligent World: Enabling Technologies and Practices. IEEE Transactions on Knowledge and Data Engineering. No.6, pp. 910--924 (2009)
25. Minka, T.P., Picard, R. W. Interactive Learning with a „Society of Models". In: Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition. IEEE, pp. 447 -- 452 (1996)
26. Huysmans, J., Baesebs, B., Vanthienen, J.: A New Approach for Measuring Rule Set Consistency. Data & Knowledge Engineering. vol.63, No.1, 167--182 (Oct. 2007)
27. Utgoff, P. E.: An Improved Algorithm for Incremental Induction of Decision Trees. Technical Report. University of Massachusetts (1994)
28. Giraud-Carrier, C., Martinez, T.: An Incremental Learning Model for Commonsense Reasoning. In: Proceedings of the Seventh International Symposium on Artificial Intelligence, pp. 134--141 (1994)
29. Duda, R.O., Hart, P. E., Stork, D. G.: Pattern Classification 2nd Ed, 654 p. Wiley-Interscience (2000)
30. Bry, F., Decker, H., Manthey, R.: A uniform approach to constraint satisfaction and constraint satisfiability in deductive databases. In: Proceedings of 1st Extending Data Base Technology, pp. 488 --505. Springer-Verlag, Venice (1988)
31. Fathi, M., Alimohammadi, A.: Inconsistency Detection between Spatial Rules in Land Use Planning Application
http://www.gisdevelopment.net/application/urban/products/ma0793.htm
32. Yasdi, R.: Learning Classification Rules from Database in the Context of Knowledge Acquisition and Representation. IEEE Transactions on Knowledge and Data Engineering. vol.3, No.3, pp. 293--306 (Sept. 1991)

Another View on Territory Fair Division Problem

Andre Veski[1] and Leo Vohandu[2]

[1] Ehitajate tee 5, Tallinn, Estonia
Technical University of Tallinn, Institute of Informatics
andre.veski@gmail.com
[2] Ehitajate tee 5, Tallinn, Estonia
Technical University of Tallinn, Institute of Informatics
leov@staff.ttu.ee

Abstract. This paper analyses the territory fair division problem posed by Hugo Steinhaus studying the solutions given by different algorithms on a large generated set of inputs. Defined are measures to characterize the initial task and a proposed solution. Main algorithm used is Adjusted Winner, developed by S. Brams and A. Taylor. We compare it to combinatorial enumeration result and some algorithms proposed for experimentation by authors. Moreover another degree of freedom is added by allowing uncertainty to the initial value representation of items to be divided based on the example of territorial division. For uncertainty handling we use belief system from Dempster-Shafer Theory [3].

Keywords: Fair division, Dempster-Shafer Theory

1 Introduction

Division problem is a more general task of the well known partitioning problem. Several different techniques have been developed to provide a fair division of goods or a single good (e.g. a cake). In general there are two different classes of techniques: turn based information sharing and decision making; or providing full information to a mediator who then proposes a solution. In this paper we concentrate on the later type of algorithms, where we have all the information available.

At first it may seem that fair division is unreachable, because every partaker has their own subjective opinion about the goods being distributed. Ultimately just because of those subjective estimations, fair division is possible. Unfortunately most of the world relies on experts' 'objective' opinions and not on their own attitudes.

Initially this problem was described by Steinhaus [1], [2] as the problem of fair division. In his example two heirs have a territory to divide between them and both expect to get half of it. They draw a vertical dividing line that would split the territory to two subjectively equal parts [1] (Fig. 1). At first both players receive a piece of the territory they value more highly per area unit. In our

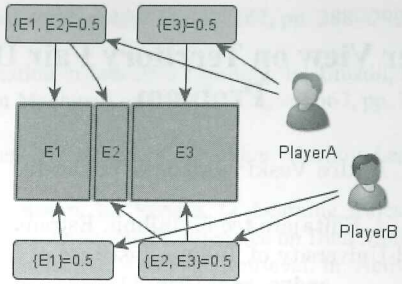

Fig. 1. Division

example E1 would be attributed to B and E3 to A. The remaining piece, E2, can be divided similarly or split randomly, since each of the players have already gained half.

To characterize a fair distribution, we use three measures proposed by Brams and Taylor [4]: *envy, equality* and *efficiency*. Envy shows how much a participant in a distribution desires some one else's gains. Efficiency illustrates the final result of a partaker: how much he finally got compared to his original expectation. A division is efficient when every participant got at least what he bargained for. Equality shows similarity in each participant's efficiency. All three are connected with each other in a certain way as we will see later. **In order for the division to be fair it has to be envy-free, efficient and equal as possible**. There are cases where these measures contradict each-other and one can not drive all of them to the maximum. Compared to a simpler partitioning problem where we have only one measure – equality. We also look how to fuse these three measures and use a single measure to evaluate the result, namely total product value of result for each player.

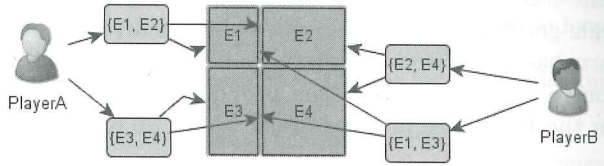

Fig. 2. Conflicting division

In our paper we also add an another degree of freedom to the problematique of fair division methodologies on an example of territory division. Instead of having non-crossing division lines as in Fig. 1 we have crossing lines as in Fig. 2. Hence we get total of four items to be divided, but valuations only for item sets of two. This means we have some level of uncertainty. While we know how participants value part of their division, we don't know how it translates to each

Table 1. Valuations

	E1	E2	E3
A	0.44	0.6	0.50
B	0.50	0.5	0.45

of the four individual section created as depicted on Fig. 2. Investigating this situation is what we concentrate on in the following sections.

2 More formal description of the problem

As already mentioned the problem of fair division is similar to the known NPC partitioning problem, where items with different values need to be partitioned into two or more distinct sets all with equal total value. Often total equality is not achievable then we have to use a measure of fitness – difference between total values. Having n items to be partitioned into two sets A and B, the goal is to minimize the difference

$$D(A,B) = \left| \sum_{i=1,a_i \in A}^{n} a_i - \sum_{j=1,a_j \in B}^{n} a_j \right| \tag{1}$$

For example having items with values {1, 2, 3} then these can be easily divided into two equal sets {1, 2} and {3}. The simplest algorithm to solve this task is to start from the largest element, assign it to a set, then second largest to the set with a lower value, until there are no more items left. This solution method is presented in more detail in algorithm MFP. Partitioning problem can be generalized by adding some degrees of freedom. For example in dividing fairly we have to take into account differences of opinion, i.e. same items can have different values depending to which set they are assigned to – A or B.

Algorithm 2.1: ALGORITHM MFP.(E)

[Initialize]$A \leftarrow newEmptySet(), B \leftarrow newEmptySet()$
while not $E.empty()$
do
- $m \leftarrow E.MaxFreeElement()$
- [Select lower value set]**if** $A.total() < B.total()$
  - **then** $A.add(m)$
  - **else** $B.add(m)$

return (A, B)

For our example on Fig. 1 let us assume that A and B have the following valuations for items in E1, E2, E3 as in Table 1. The valuations are normalized so that for both A and B the total value of the heritage would be 100. Comparing this example to the previous partitioning we see that finding a solution isn't that

simple anymore as with partitioning single value items. Although the solution base size of the task is the same – n^m, where n is the number of items and m number of players. For comparison we also compare a brute force selection to other algorithms. A good algorithm to solve this has been developed by Brams and Taylor [4], called Adjusted Winner. That method is really a MFP counterpart for fair division problem. In the first round the main goal is to return maximal efficiency by assigning each item to the highest bidder. In the second adjustment step equalize by giving most similarly valued items to the worst-off player.

Algorithm 2.2: ALGORITHM AWT.(E)

[Initialize] $D \leftarrow newEmptyDivision()$,
$n \leftarrow E.NumberOfItems(), v \leftarrow 0$
for $i \leftarrow 1$ **to** n
do { [Select largest and add]
if $PlayerAItem[i] > PlayerBItem[i]$
then $D.Add(PlayerAwithItem[i])$
else $D.Add(PlayerBwithItem[i])$
[Adjust] **if** $D.Total(PlayerA) > D.Total(PlayerB)$
then repeat
{ **for** $i \leftarrow 1$ **to** n
do { $c \leftarrow 2/((PlayerAwithItem[i])/(PlayerBwithItem[i]))$
[Select best] **if** $v < c$
then $v \leftarrow c, s \leftarrow i$
$D.Give(Item[s]toPlayerB)$
until $D.Total(PlayerA) <= D.Total(PlayerB)$
else if $D.Total(PlayerA) < D.Total(PlayerB)$
then switch players and goto step [Adjust]
return (D)

As mentioned the problem can be generalized further, when participant valuations are uncertain. For example when for player A the total value of E1, E2 is 73, but it is not known how to break it down further to valuations for individual items, E1 and E2. It might be that it is 6 and 67, but that is uncertain. Or the example on Fig. 2, where values for item-set of two is 0.5 for respective players. Further we might have some knowledge how individual pieces are valued, but are uncertain on some amount.

On Fig. 3 we have values for Player A, where the value of E1, E2=50 and the individual items E1=42 and E2=6. The value left over from 50-42-6=2 is the value that comes from owning both of these items together. In other words it is the value of the connection between those two items. This handling of uncertain valuations is based on Dempster-Shafer Theory (DST) [3]. Even with adding uncertain valuations the size of the task is still the same - n^m, since the number of individual items is still the same and only a combination of single items can form a solution. The matter that changes with added flexibility is the

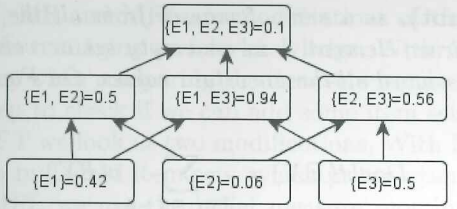

Fig. 3. Valuations

difficulty of finding a good solution. When deciding to which set the item should be assigned with each degree of freedom more information has to be processed in order to make that decision. Moving from the partition problem to fair division the added complexity is that valuations for both players have to be compared and with adding uncertainty as valuations on a lattice (Fig. 1) we also have to take into account the subsets that the partitioned sets contain. Going forward we describe briefly the necessary part of DST [3]. On a valuation lattice (Fig. 3) there are at least three ways of looking at each item set:

1. Total value of items which our current item contains - belief
2. Total value of items in which our current item is part of - plausibility
3. Total value of items in which our current item is not part of and does not contain itself - doubt

The base values for each item or set of items, is called mass or basic probability assignment in DST [3], and is the value only for that particular element. The mass values for all item sets in the power set add up to 1 and value of an empty set is 0. Meaning that for single items the mass is the value of that item and in larger item sets the value of the connection. In our example from Fig. 3 the mass for E1, E2 is 2 and the belief value is 50. The definitions for item set value functions are given below.

Definition 1 (Belief). *is a sum of masses from all the sets where observed element A is a part of or in other words a lower bound for an item set A that contains the certain knowledge about the item set. On Fig. 4 we have belief for E6 = {E2, E3} presented in red.*

$$Belief(EX) = \sum_{EY \subset EX} v(EY) \tag{2}$$

Definition 2 (Plausibility). *is a sum of masses from all the item sets where the union with the observed element A is not an empty set or in other words an upper bound for a set A that A would have if it got assigned all the uncertain values. On Fig. 5 there is plausibility for E1 in red.*

$$Plausibility(EX) = \sum_{EY \cap EX \neq \emptyset} v(EY) \tag{3}$$

Definition 3 (Doubt). *is a sum of masses from all the item sets where the union with the observer element A is an empty set or value that A can never have even if it got assigned all the uncertain values. On Fig. 4 there is doubt for E1 presented in red.*

$$Doubt(EX) = \sum_{EY \cap EX = \emptyset} v(EY) \tag{4}$$

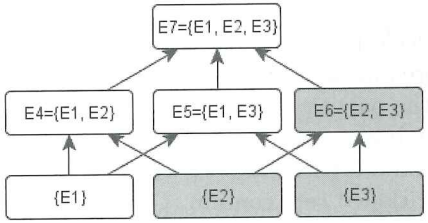

Fig. 4. Belief E6, Doubt E1

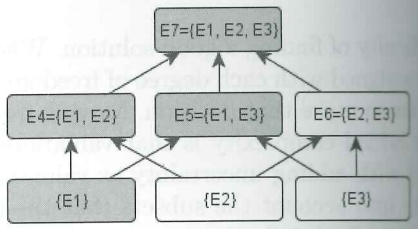

Fig. 5. Plausibility E1

3 Algorithms with uncertainty

So far we have examined two algorithms for fair division: Maximal First and Adjusted Winner. As a next step we need to figure out how to handle uncertain valuations. Handling uncertainty in algorithms can be done in different ways. In the following we'll explore two of them.

1. Using different value functions from DST to determine a value for an item to compare on, e.g. belief or plausibility
2. Start the division process with different volumes sets of item, e.g. on the second level on the valuation lattice where in all item sets there are two items. Without uncertainty all item sets have a volume of 1. With uncertainty as in our example, on Fig. 3, we choose item sets with a volume 1 and 2. Here we test starting from the middle level of the lattice of volume 2.

More precisely we'll compare results from four algorithms.

1. Enumerating all possible sets and simply selecting the best (Algorithm CT and Algorithm C)
2. 2-step Adjusted Winner (Algorithm AWT and Algorithm AW)
3. Maximal valued First (Algorithm MFT, Algorithm MF and Algorithm MFL)

In other words all algorithms will be used in two settings

1. Model with uncertainty (Algorithms C, AW, MF and MFL)
2. Model without uncertainty (Algorithms CT, AWT and MFT)

Algorithms AW, MF and MFL are modifications of their counterparts CT, AWT and MFT from managing certain knowledge. With AW we still loop only over single items, but when comparing the item sets we use plausibility function. We also added a step to check if we can add some item sets already covered by single items. On MFT we look at two modifications. With MF we start the item sets enumeration on mid-sized item sets, which in our example is item sets with size 2. And with MFL we use the belief measure, not just mass, but still go through single items.

Algorithm 3.1: ALGORITHM C/CT.(E)

[Initialize] $p \leftarrow 0, D \leftarrow newEmptyDivision()$
repeat
  $s \leftarrow NextSolutionCombination()$
  [Compare new solution to current solution]
  if $s.product > p$
    then $D \leftarrow s, p \leftarrow s.product$
until $OutOfSolutions()$
return (D)

Algorithm 3.2: ALGORITHM MFT.(E)

[Initialize] $D \leftarrow newEmptyDivision()$,
$n \leftarrow E.NumberOfSingleItems()$
for $i \leftarrow 1$ **to** n
do
  [Select largest and add]
  if $PlayerAItem[i] > PlayerBItem[i]$
    then $D.Add(PlayerAwithItem[i])$
    else $D.Add(PlayerBwithItem[i])$
return (D)

Algorithm 3.3: ALGORITHM MF.(E)

[Initialize] $D \leftarrow newEmptyDivision()$,
$n \leftarrow E.NumberOfItemsVolume(2)$
for $i \leftarrow 1$ **to** n
do
  [Select largest and add]
  if $PlayerAItem[i].Belief > PlayerBItem[i].Belief$
    then $D.Add(PlayerAwithItem[i])$
    else $D.Add(PlayerBwithItem[i])$
return (D)

Algorithm 3.4: ALGORITHM MFL.(E)

[Initialize] $D \leftarrow newEmptyDivision()$,
$n \leftarrow E.NumberOfSingleItems()$
for $i \leftarrow 1$ **to** n
do {
  [Select largest and add]
  if $PlayerAItem[i].Belief > PlayerBItem[i].Belief$
    then $D.Add(PlayerAwithItem[i])$
    else $D.Add(PlayerBwithItem[i])$
}
$np \leftarrow E.NumberOfNonSingleItems()$
for $i \leftarrow 1$ **to** np
do {
  if $D.Contains(PlayerA, Item[i].Elements)$
    then $D.Add(PlayerAwithItem[i])$
    else if $D.Contains(PlayerB, Item[i].Elements)$
    then $D.Add(PlayerBwithItem[i])$
}
return (D)

Algorithm 3.5: ALGORITHM AW.(E)

[Initialize] $D \leftarrow newEmptyDivision()$,
$n \leftarrow E.NumberOfSingleItems()$
for $i \leftarrow 1$ **to** n
do {
  [Select largest and add]
  if $PlayerAItem[i].Plausib > PlayerBItem[i].Plausib$
    then $D.Add(PlayerAwithItem[i])$
    else $D.Add(PlayerBwithItem[i])$
}
[Adjust] $v \leftarrow 0$
if $D.Total(PlayerA) > D.Total(PlayerB)$
  then repeat
  for $i \leftarrow 1$ **to** n
  do {
    $a \leftarrow PlayerAwithItem[i].Belief$
    $b \leftarrow PlayerBwithItem[i].Belief$
    $c \leftarrow 2/(a/b)$
    [Keep best matching item] **if** $v < c$
      then $v \leftarrow c, s \leftarrow i$
  }
  $D.Give(Item[s]toPlayerB)$
until $D.Total(PlayerA) <= D.Total(PlayerB)$
  else if $D.Total(PlayerA) < D.Total(PlayerB)$
  then switch players and goto step [Adjust]
$np \leftarrow E.NumberOfNonSingleItems()$
for $i \leftarrow 1$ **to** np
do {
  if $D.Contains(PlayerA, Item[i].Elements)$
    then $D.Add(PlayerAwithItem[i])$
    else if $D.Contains(PlayerB, Item[i].Elements)$
    then $D.Add(PlayerBwithItem[i])$
}
return (D)

4 Measuring solutions and tasks

With each degree of freedom there are more possibilities to evaluate the fitness of a solution. On the simple partitioning task there is a measure of difference of values in both sets. As the number of partitions grows new measures come up such as total difference of values in pairwise comparison etc.

A fair division is usually described by three measures according to Brams and Taylor [4]: efficiency, envy and equality and we also use total product. The latter is also recommended by Nguyen and Kreinovich in [5] and is known in game theory as Nash's Bargaining Solution [6].

Definition 4 (Efficiency (Pareto efficiency)). *is the total value of the solution. Which is actually a condition where no player can be made better off by making someone else worse off. But if an item can change owners and create more value from that, the initial owner can be compensated by some uniform measure - i.e. money.*

$$Efficiency(D) = \sum_{i=1, a_i \in D}^{n} a_i \tag{5}$$

Definition 5 (Envy (Envy freeness)). *is the amount by which in one players valuations other players result was larger than his. The total envy is total sum on directed pairwise comparisons.*

$$Envy(A, B) = max\left(\left(\sum_{i=1, a_i \in A}^{n} a_i - \sum_{j=1, b_j \in B}^{n} a_j\right), 0\right) \tag{6}$$

$$Envy(B, A) = max\left(\left(\sum_{i=1, b_i \in B}^{n} b_i - \sum_{j=1, a_j \in A}^{n} b_j\right), 0\right) \tag{7}$$

$$TotalEnvy(A, B) = Envy(A, B) + Envy(B, A) \tag{8}$$

Definition 6 (Equality). *is the amount by which end results differ for each player and is calculated as a sum of pairwise differences*

$$Equality(A, B) = \left| \sum_{i=1, a_i \in A}^{n} a_i - \sum_{i=1, b_i \in B}^{n} b_i \right| \tag{9}$$

Definition 7 (Product). *is the total product of all players end results*

$$Product(A, B) = \sum_{i=1, a_i \in A}^{n} a_i \cdot \sum_{i=1, b_i \in B}^{n} b_i \tag{10}$$

Table 2. Example calculations

	Player A	Player B
Result	0.50, 0.6	0.50
Efficency	$0.6 + 0.50 + 0.50 = 1.06$	
Envy	$0.67 - (0.27 + 0.6) = 0.34$	$0.61 + 0.5 - 0.34 = 0.33$
Equality	$0.67 - 0.61 - 0.05 = 0.01$	
Product	$0.56 \cdot 0.50 = 0.28$	
Conflict	$1 - 0.44 \cdot 0.05 - 0.44 \cdot 0.45 - 0.06 \cdot 0.5 - 0.06 \cdot 0.45 - 0.5 \cdot 0.5 - 0.5 \cdot 0.05 = 0.448$	
Difference	$\frac{1}{2} \cdot (\lvert 0.44 - 0.50 \rvert + \lvert 0.6 - 0.5 \rvert + \lvert 0.50 - 0.45 \rvert) = 0.06$	
Uncertainty	0.0	

Envy and equality are two sides of the same coin. In partitioning task we had equality as our primary measure of fitness – as difference in an absolute sense. Since each set has now multiple different values, based on every players individual values, we also get the envy measure – as difference in a relative sense. Both are similarly calculated as a pairwise sum of differences.

Efficiency is a new kind of measure. With simple partitioning this measure does not really carry any information since with every solution it would have the same value. With solving the fair division problem, since the total value of both partitioned sets can be different with each solution, it is important to make sure that we would get the maximum possible total value.

Similarly measuring the fitness of a solution, there are measures to characterize the initial players valuations. To get a better understanding what kind of solutions can be expected from certain type of inputs. So these measures try to characterize the original task.

Definition 8 (Conflict). *defined as a conflict measure in the DST [3]*

$$Conflict(A, B) = 1 - \sum_{i=1}^{n} \sum_{j=i, a_i \cap b_j = \emptyset}^{n} a_i \cdot b_j \tag{11}$$

Definition 9 (Difference). *is a pairwise difference in participants' valuations. This is recommended by authors and as we see later has a good correlation with the solution.*

$$Difference(A, B) = \frac{\sum_{i=1}^{n} |a_i - b_i|}{2} \tag{12}$$

Difference describes the total difference in comparing players pairwise. The more different the players' valuations are the more efficient should the solution be. In extreme cases where valuations are completely opposite the result would be double of that the players initially subjectively expected.

Table 3. Valuation example

	E1	E2	E3	E4	E1, E2	E3, E4	E1, E3	E2, E4
Player A	0	0	0	0	0.5	0.5	0	0
Player B	0	0	0	0	0	0	0.5	0.5

Table 4. Valuation example

	E1	E2	E3	E4	E1, E2	E3, E4	E1, E3	E2, E4
Player A	0.5	0	0.5	0	0	0	0	0
Player B	0	0.5	0	0.5	0	0	0	0

Definition 10 (Uncertainty). *amount of uncertainty in the valuations as total sum of mass on item sets with greater volume than 2.*

$$Uncertainty() = \sum_{i=1, |a_i|>1}^{n} a_i + b_i \tag{13}$$

5 Algorithm comparison set-up

We take the same situation as described before, where two participants had to divide a territory. They split the territory as on Fig. 2 and initially we have only evaluations as given in Table 3.

To compare the results for all the algorithms we look at a large generated set of different inputs, basically taking assumptions for values for single items E1 to E4. Our goal is to look at many different settings. Some of them have uncertainty, some of them are closely valued, some are very different. To make sure we have all the possible settings we generate all different combinations, all together about 97 thousand different settings. We start with assessments as in Table 3, go through all the possible values with step of 0.1 for items E1 through E4 and the last setting being the task in Table 6. Between all those we also have tasks as in Tables 4 and 5.

1. Table 3 example has high uncertainty and high conflict.
2. Table 4 example has no uncertainty, high conflict and high difference.
3. Table 5 has some conflict, some uncertainty and by our definition no difference.

We also have some constraints on tuples we look at, mainly to reduce the number of settings we eliminate some symmetric tasks. If we already looked at a task in Table 3 we would skip the task described in Table 6. Moreover the belief for {E1, E2} and {E3, E4} for Player A will always have to equal 0.5 and respectively {E1, E3} and {E2, E4} for Player B. So if we have value 0.1 on E1 from Player A we know that the value for E2 can be at most 0.4. Meaning that

Table 5. Valuation example

	E1	E2	E3	E4	E1, E2	E3, E4	E1, E3	E2, E4
Player A	0.2	0.2	0.2	0.2	0.1	0.1	0	0
Player B	0.2	0.2	0.2	0.2	0	0	0.1	0.1

Table 6. Valuation example

	E1	E2	E3	E4	E1, E2	E3, E4	E1, E3	E2, E4
Player A	0	0.5	0	0.5	0	0	0	0
Player B	0.5	0	0.5	0	0	0	0	0

we do not need to generate all values from E1 to E4, but only from E1 to E4 since their sum is fixed.

Secondly to compare algorithms that can't handle uncertainty, we need to transform the input. For this we use the plausibility measure. For all single items we calculate their plausibility and then normalize the result to add up to 1. This solution basically distributes the uncertainty evenly between all items.

6 Assessing metrics

How to assess solutions and by what metric to make a decision for a better solution? In [4] there are 3 metrics also described here: Envy, Efficiency, Equality and additionaly we use Product. Usually in examples provided by Brams and Taylor [4] efficiency is preferred until the total reaches 1 and after that more effort is put on equality.

Table 7. Correlations

	Efficiency	Envy	Inequality	Product
Efficiency	1			
Envy	-0.21	1		
Inequality	0.18	0.80	1	
Product	0.65	-0.76	-0.59	1

On Fig. 6 we can see the relation between all the four metrics. These values have been generated from all possible solutions (16) with Algorithm C for each task described in the previous section. This results in total about 1.6M cases. In a rough summary the properties for the metrics are (based on Fig. 6):

1. Product has a good correlation with Efficiency (Table 7)

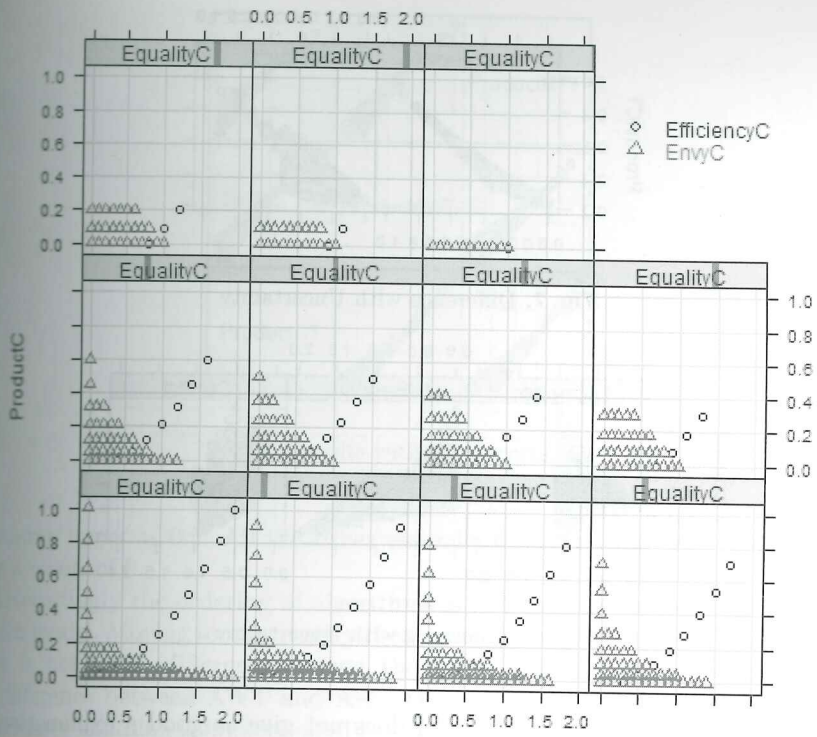

Fig. 6. Solution metrics

2. The highest Product is almost always with low Envy. There are always tasks where Envy is unavoidable, with the extreme case being the example in Table 3 in the previous section
3. With a higher inequality the product approaches 0, as can be seen from the chart (Fig. 6) starting from bottom left and moving to upper right

Since all the correlations with the product are in the right direction for us (Table 7), i.e. maximizing efficiency and minimizing envy and inequality, we will use it to evaluate the fitness of our solutions. Moreover there is more justification for using the Nash's Bargaining solution in [6] and [7].

7 Initial Results

Solving tasks in the model without uncertainty, the resulting solution is usually well predicted with the initial difference of opinions between the players. The greater the difference the better the result will be. This is best illustrated on Fig. 7.

As expected, adding uncertainty creates additional level of complexity. The differences of opinion don't have so direct impact on the result as before. On

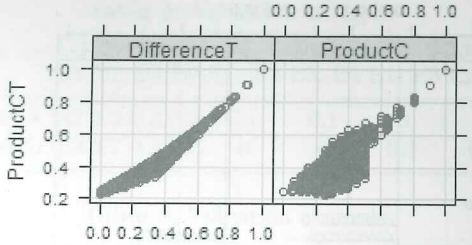

Fig. 7. Difference with Uncertainty

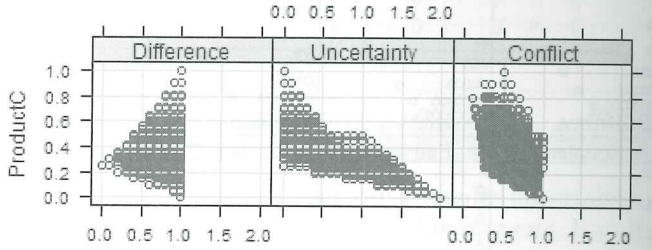

Fig. 8. Difference with Uncertainty

Fig. 8 we see that the difference itself does not give as good a explanation as before. It still has similar direction as with the certain model, the best result is achieved only with the biggest difference, but not the other way around. Moreover from Fig. 8 we see that there is no definite impact by uncertainty and conflict either. Although there is some direction with uncertainty, i.e. the greater uncertainty implies lower results, but with decreasing uncertainty the variance on the result increases.

Next looking at how different results algorithms from our algorithm, with (Fig. 10) and without (Fig. 9) uncertainty. On the certain side (Fig. 9) we see that all the algorithms produce quite similar results. Obviously the CT algorithm has the best result, then AWT and lastly MFT. Most of the results from both comparison (AWT and MFT) algorithms produce quite near results to the ideal CT, both have some deviation, but not very much. In general there is room for improvement for both algorithms. And lastly AWT manages always to produce better result than MFT, at least with certain valuations.

The outcome is different in the situation of having uncertainty (Fig. 10). Again algorithm C is the best, but the second best is not so clear anymore. Algorithm MF produces the strangest result of them all. Most of the solution product values are in the interval [0.25, 0.5], which has to be the result of the selection of the starting point, i.e. operating with item sets of two. This would explain the upper bound 0.5, which can be achieved when one party gets his compound set with the value on 0.5 and the other party values both of the remaining items as 0.5. Lower bound 0.25 is achieved when first party again gets his two items with total value of 0.5 and the other party values the sum of the

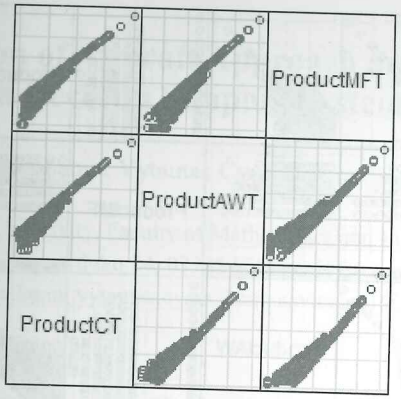

Fig. 9. Results without Uncertainty

remaining items as 0.5. All the other cases lie between those two situation or have a result of 0.

Interestingly the ordering of algorithms is not so clear anymore as with the certain model. Moving to uncertainty then between the MFT and MFL algorithm there is not much difference, except the latter can handle uncertainty, as is the difference between AWT and AW. Also in both cases, AW and MFL, the uncertainty handling is implemented in a similar way. And as is visible on Fig. 10 MFL is in some cases able to achieve better results than AW. This is definitely one of the places to start creating an even better algorithm.

8 Conclusion

Looking at the various charts it is apparent that the quality of the solution is determined by limits on item values. Although in the case of uncertainty the determination is not that clear anymore. Meaning that we should expand the metrics even more, to clarify the connection between the task and the solution.

Moreover the setup of our test environment is quite limited. Currently we only tested the case of two participants, who had 4 items to share. Making the task more complex can reveal new insights on the algorithms. Next steps on the test settings should be:

1. Expanding the task for 3 and more participants
2. Expanding the item space and therefore the valuation lattice will be more complex
3. Resetting the initial task to allow more freedom, i.e. item set values in [0;1]

The most interesting point on algorithm were the differences between MFL and AW. Both have their advantages in different situation. So one one of areas for further research is to understand how to merge them to a single algorithm. The MF algorithm did not perform as well as others, but it might still have

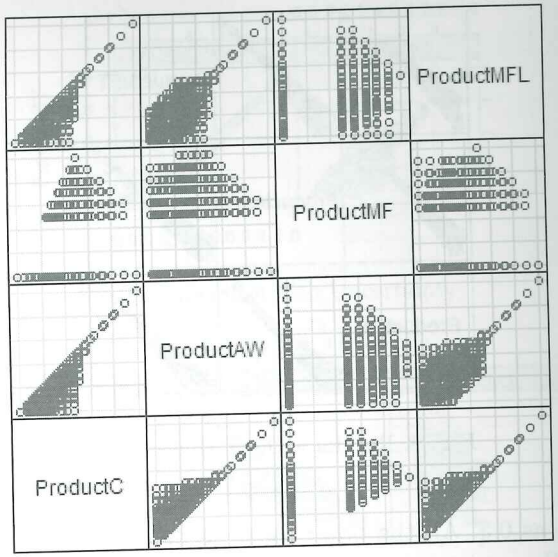

Fig. 10. Results with Uncertainty

its benefits. This and other algorithms should still be tested in more complex situations to appreciate the shortcomings or as it may turn out the benefits.

References

1. Hugo Steinhaus, The problem of fair division, Econometrica, 16, 101-104 (1948)
2. Hugo Steinhaus. Mathematical Snapshots, New York (1969)
3. William L. Oberkampf, Jon C. Helton. Evidence Theory for Engineering Applications Appeared in Engineering Design Reliability Handbook, Danvers (2004)
4. Steven J. Brams, Alan D. Taylor. Fair Division: From cake-cutting to dispute resolution, Melbourne (1996)
5. Hung T. Nguyen, Vladik Kreinovich, How to divide a Territory: A New Simple Formalism for Optimization of Set Functions, International Journal of Intelligent Systems, 223-251 (1999)
6. Duncan Luce, Howard Raiffa. Games and Decisions: Introduction and critical survey, Dover (1989)
7. Ken Binmore. Natural Justice, Oxford (2005)

Visualization of Aircraft Approach and Departure Procedures in a Decision Support System for Controllers

Kristina Lapin, Vytautas Čyras, Laura Savičienė

Vilnius University, Faculty of Mathematics and Informatics,
Naugarduko 24, 03225 Vilnius, Lithuania
{kristina.lapin, vytautas.cyras, laura.saviciene}@mif.vu.lt

Abstract. This paper reports on 3D visualization of airport requirements for approach and departure trajectories. The work is undertaken within the FP6 SKY-Scanner project that develops a new lidar (laser radar, LIght Detection And Ranging) equipment. The fusion of radar and lidar data improves aircraft surveillance within aerodrome traffic zone (ATZ). We propose a new surveillance paradigm which enables the controller to estimate visually a current situation without mental calculations of altitude and distance. Trajectory requirements are defined threefold: normative regulation of aircraft trajectories, vertical and horizontal safety distances, and flight plans. The paper proposes a visualization paradigm based on 2D and 3D view integration. Trajectory constraints are shown in 2D projections. The paper explores how innovative visualizations create new opportunities for controlling air traffic in the ATZ.

Keywords: 3D visualization, controllers' needs, situational awareness, air traffic control.

1 Introduction

An airport comprises a large amount of specialized systems such as passenger and baggage registration, gate for flight assignment, approach and departure management, etc. The paper is devoted to decision making for air traffic control (ATC) systems dispatchers. The research is on the development of a decision support system (DSS) for a new generation air traffic management (ATM) paradigm (Fig. 1). The work is conducted within the FP6 SKY-Scanner[1] project. The topics include aircraft collision risk evaluation and a DSS.

Air traffic management rules combine technological, economics and regulatory aspects that synchronize plans and actions of different stakeholders in both airborne and ground systems. ATC is a service provided by the ground-based controllers for the purpose of preventing collisions and maintaining an orderly flow of traffic [11].

[1] EU FP6 TP1.4 Aeronautics and space, TREN-4-Aero. Title: "Development of an Innovative LIDAR Technology for New Generation ATM Paradigms" (SKY-Scanner), 2007-2010, http://www.sky-scanner.it/

Each airport determines its approach and departure procedures. Each flight is performed according to an assigned flight plan.

New ATM systems have long implementation cycles. Research and development projects generate ideas that are elaborated in subsequent projects. A proposed solution is verified from different perspectives.

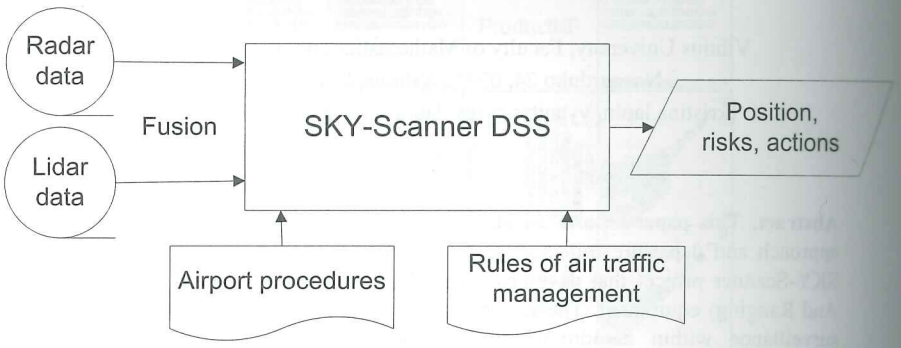

Fig. 1. The scope of SKY-Scanner

The presented work is performed as a part of the Single European Sky Air traffic management Research (SESAR) programme [14]. SESAR aims at increasing the capacity and efficiency of the European Air Traffic Management (ATM) system [13]. SKY-Scanner explores, among others, avoiding the risks identified by SESAR.

Constantly rising air traffic requires more information to display on a dispatcher's screen. With an increased amount of information the risk of a human error grows. The reason is that errors appear when important information is mixed with less important one and some aspects are accessed by scrolling. Therefore visualizations aim to enable the user to focus on the large volume of data without losing context awareness.

SKY-Scanner solution employs the visualization ideas proposed in 3D-in-2D Displays for ATC project [15]. The resulting system extends controller's capabilities to control the adherence of aircraft landing and take-off trajectories with airport procedures. Current systems leave the final landing and initial take-off phases under pilot's responsibility. The controller gives the pilot a clearance to land. The pilot performs the maneuver and reports. The controller can also observe the landing visually from the tower. The SKY-Scanner system visualizes an actual situation on the screen. The system enables a human operator to estimate visually whether the aircraft meets trajectory constraints.

1.1 Background to the Project

The SKY-Scanner system aims at detecting and tracking aircraft up to at least 6 nautical miles from the ATZ barycentre [12]. In the terminal area an aircraft climbs to the desired altitude. While achieved the destination airport it descents from the cruise altitude and lands. The manoeuvres are performed at low altitude.

Current ATC systems are based on radar devices that have significant disadvantages. The radar receives signals reflected from rain, trees and the ground. It is difficult to distinguish aircraft targets and background clutter at low altitude. In most cases in ATZ, the radar cannot determine the height to the accuracy needed as altitudes are below 300 m. Radar equipment has a margin of error of 200 m. Hence, approach and departure procedures cannot be tracked with a proper accuracy when the altitude is less than the equipment's margin of error.

The lidar accuracy has a margin of error measured in centimetres and is exact when directed to the target. An approximate position received from the radar facilitates the deflection of the lidar to the target. When the target is found, the lidar switches to the tracking mode and provides the exact target position for the SKY-Scanner system.

Surveillance equipment (primary radar, secondary radar, etc.) is used to establish the controllers' situational awareness of their assigned airspace. A lidar is installed on ground. Unlike other surveillance systems (secondary surveillance radar or automatic dependent surveillance broadcast), the lidar requires no additional equipment to be installed on the aircraft. The lidar is more precise than the primary radar but it may function worse in certain weather conditions such as rain and fog.

The combination of both devices (Fig. 1) covers their shortcomings. This also enables to track the flight from the beginning till the end. Allowed distances between aircraft are extracted from the rules of air traffic management. Airport procedures define the Ought to Be trajectories. DSS output provides the fused aircraft position and estimated risks of violations. Lidar and radar data are fused in order to better estimate the risks and to propose corrective actions for the controller.

1.2 Motivation for the Paradigm

User needs analysis was performed reviewing the literature. Mainly EUROCONTROL project results and the reports of the National Aerospace Laboratory (NLR) in the Netherlands were concerned.

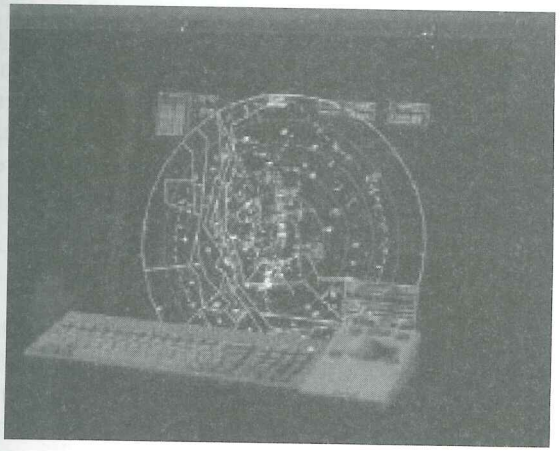

Fig. 2. Typical radar display

Approaches are classified as either precision or nonprecision, depending on the accuracy and capabilities of the navigational aids used. There are different procedures for different navigational aid types. The number of controlled parameters is also different. Precision approaches utilize both lateral (localizer) and vertical (glide path) information. Nonprecision approaches provide lateral course information only. Procedures depict prescribed altitudes and headings to be flown, as well as obstacles, terrain, and potentially conflicting airspace. In addition, they also list missed approach procedures and commonly used radio frequencies (Fig. 4).

The approach procedures determine a certain approach structure defined by approach stages and fly-over points (Fig. 5). A visualization model proposed further is based on this structure.

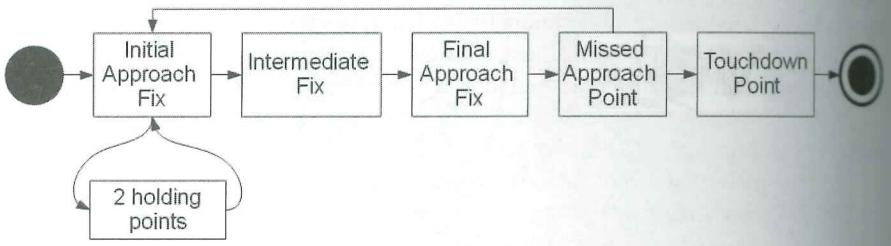

Fig. 5. Approach procedure in terms of fly-over points

An instrument approach is divided into five segments: initial, intermediate, final, and missed approach [8]:

1. Arrival: where the pilot navigates to the Initial Approach Fix (IAF), and where holding (keeping an aircraft within a specified airspace while awaiting further clearance) can take place.
2. Initial Approach: the phase of flight after the IAF, where the pilot commences the navigation of the aircraft to the Final Approach Fix (FAF), a position aligned with the runway, from where a safe controlled descent back towards the airport can be initiated.
3. Intermediate Approach: an additional phase in more complex approaches that may be required to navigate to the FAF. Intermediate Approach begins at the Intermediate Fix (IF).
4. Final Approach: between 4 and 12 nautical miles (NM) of straight flight descending at a set rate (usually an angle of between 2.5 and 6 degrees).
5. Missed Approach: an optional phase; should the required visual reference for landing not have been obtained at the end of the final approach, this allows the pilot to climb the aircraft to a safe altitude and navigate to a position to hold for weather improvement or from where another approach can be commenced.

3 Related Work on DSS Visualization

The way computer systems present information to a human controller is of crucial importance for the effectiveness of future ATC systems. Considering an increasing

amount of information made available, a traditional 2D radar representation becomes overloaded [6]. Therefore 3D interfaces are being created. A 3D view requires significantly less cognitive effort to interpret altitude information. However, pure 3D visualizations make distance estimation inconvenient due to perspective distortion effects. The camera restricts the view to a certain sector. Therefore, the controller cannot oversee global traffic.

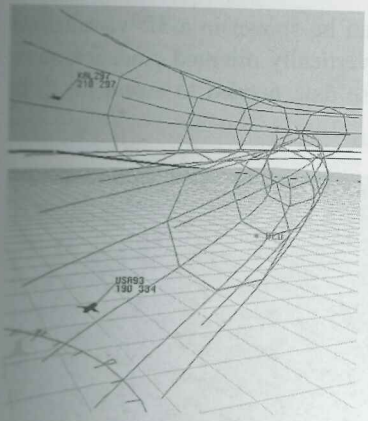

Fig. 6. Ghost plane airspace mode [2]

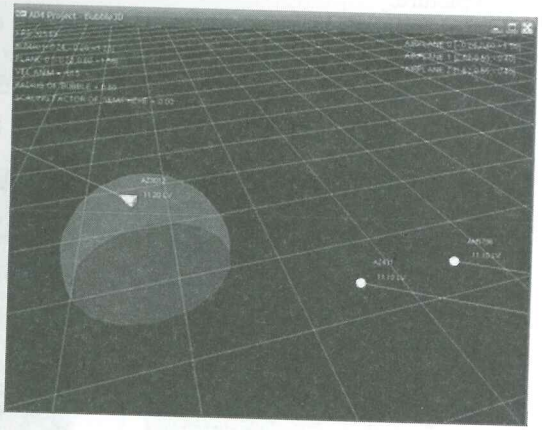

Fig. 7. Lens display [10]

3D visualizations suit better for integrated attention tasks, such as instructing an aircraft to descend and turn to intercept the localizer. 2D suits better for focused attention tasks, such as estimating the exact aircraft altitude in a moment.

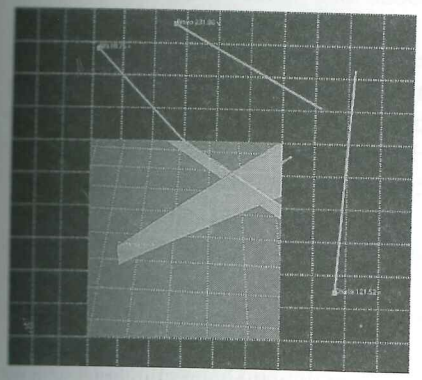

Fig. 8. Picture within a picture [10]

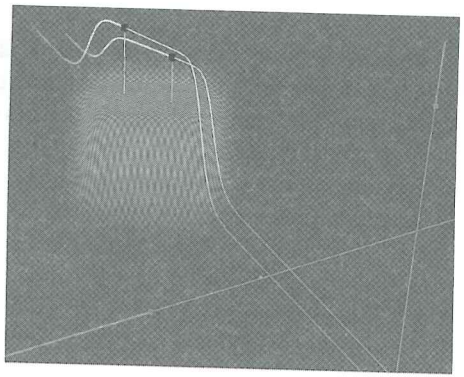

Fig. 9. Distortion display [10]

A method of 3D visualizations for ATC consists of strict 3D and 2D/3D combination displays [15]. For example, a strict 3D was used to detect conflicts in the terminal area of Boston Logan Airport [2]; see Fig. 6. This solution requires filtering to reduce controller's overload. Each zone requires a different mode of visualization. This example shows that pure 3D has certain disadvantages. Just to mention a few,

there is no possibility to oversee the global traffic out of the camera view, then a difficulty to locate traffic at the far end of the scene [3].

A combination of 3D visualization and a 2D view enables to see both the contextual and the altitude information at the same time [1]. The research identified a strategy to select a portion of the main 2D view and represent it in 3D (Fig. 7, 8, and 9). Thus continuity is preserved. It is easier to identify which aircraft is which within a 3D picture.

2D walls with the projections of an aircraft can be shown in a 3D visualization. The Wall View with Altitude Rulers presents a vertically oriented gradation on the wall (Fig. 10). This provides the user with precise data needed to assess the traffic situation or guide aircraft accurately [15].

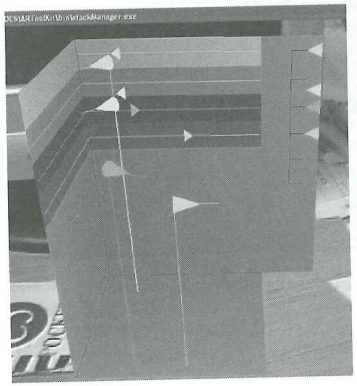

Fig. 10. Wall View with Altitude Rulers allows easily read altitudes [15]

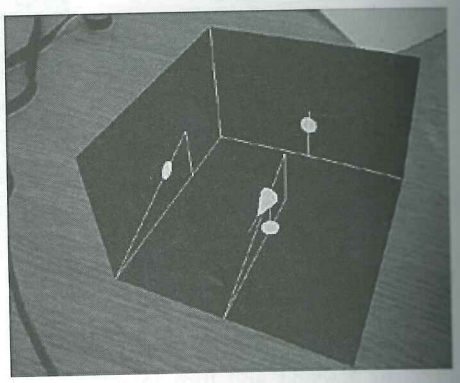

Fig. 11. Wall View of Approach Control enables to control adherence to assigned procedure [15]

The Wall View of Approach Control (see Fig. 11) enables the controller to check whether an aircraft adheres the assigned climb/descent procedures. The view shows unambiguously whether the procedure is strictly followed. This method requires a precise aircraft position that cannot be achieved with the sole radar equipment. Current surveillance technologies cannot implement those visualizations because of inaccuracy of radar devices.

4 A Visualization Model in SKY-Scanner

The DSS visualization subsystem enables controllers to perceive and interact with the following information:

1. aircraft positions
2. predicted trajectories
3. detected risks.

Minimizing clutter and distractions is vital to controllers [3]. Hence, on the one hand, it is important to visualize all required information. On the other hand, this should be done with minimal means in order not to clutter the screen.

4.1 The SKY-Scanner Method

Approach procedures present trajectory constraints in a profile view; see an example in Fig. 12. This view is convenient for aeronautics professionals. The constraints are presented in alphanumeric texts. The constraints can be projected in the Wall View with Approach Control (Fig. 11). Thus display cluttering is reduced.

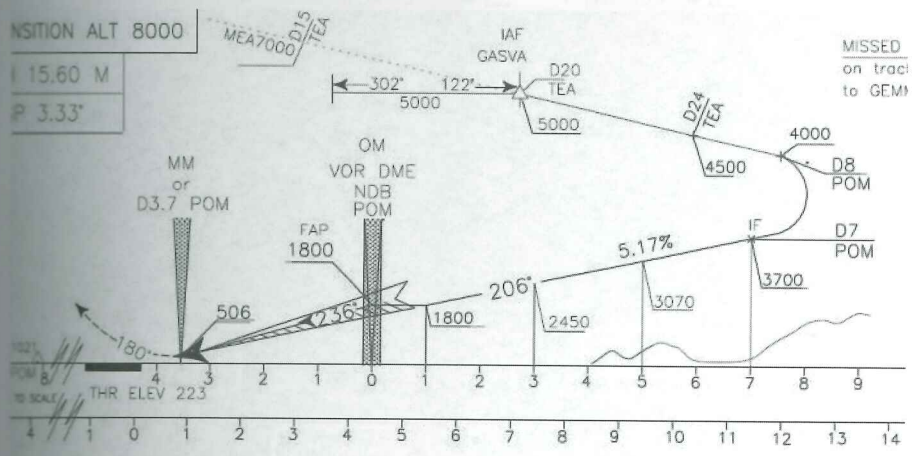

Fig. 12. Example profile view of the approach procedure

In SKY-Scanner, the Wall View with Approach Control is enhanced with semitransparent curtains. The walls cover a significant area of the screen. They also hide a context view behind and underneath them. Semitransparent curtains is a solution to avoid the hiding.

An airport zone is divided into two vertical spaces. The space below a determined altitude (transitional altitude) is allowed for an aircraft which obtained a landing clearance. With regard to this space, the approach and departure procedures are visualized.

The space above a determined altitude is devoted to aircraft which approach the airport from outside. In this space, the task of the controller is to ensure appropriate horizontal and vertical separations. Therefore, the altitude rulers are integrated with vertical curtains. The number of rulers depends on waiting loops determined in a concrete airport.

The rulers enable the controller to monitor a holding stack of landing aircraft. SKY-Scanner integrates two interface models:

4. Wall View with Altitude Rulers.
5. Wall View with Approach Control.

4.2 Terrain

A benefit of 3D visualizations is in representing 3D ground surface together with altitude. A drawback is that a realistic map may provide the controller with little

useful information. Instead, a simple 2D color map based on height can highlight major orientation features [9].

An example of a generalized terrain is shown in Fig. 13. Terrain peculiarities are important for an airport near mountains.

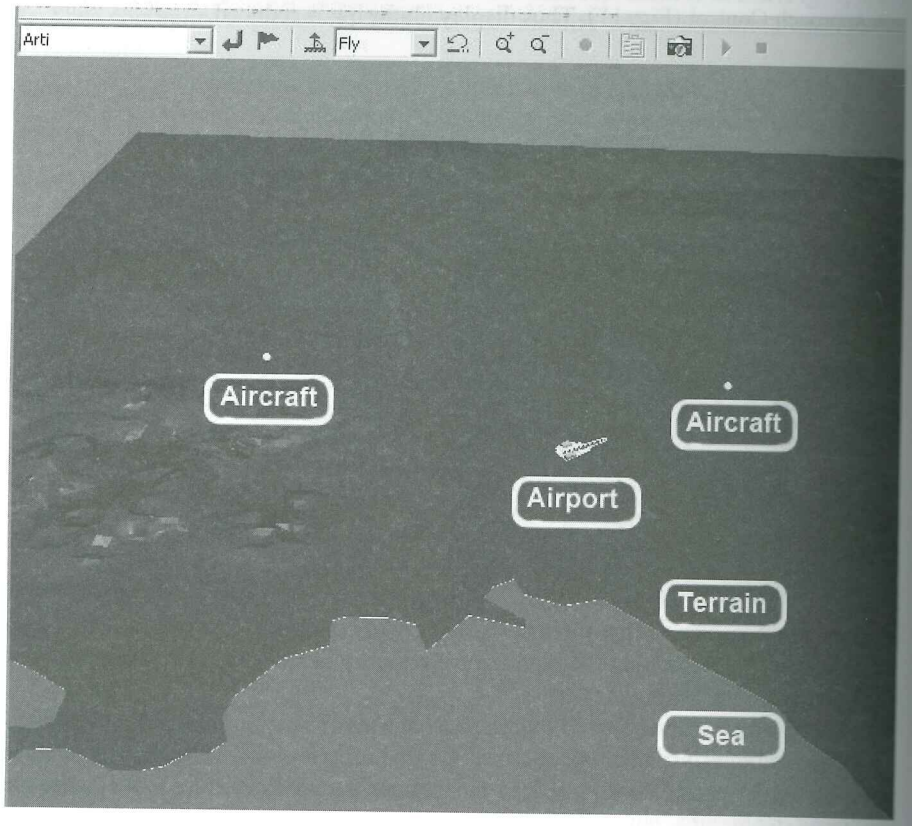

Fig. 13. Generalized terrain model. The airport is a white icon in the center. Small white indicators depict two aircraft

4.3 Modeling the Curtains

Opaque walls can be replaced with transparent curtains. This enables a transparent view. The surroundings are seen like through a curtain. Transition height is represented with a different color – like in the Wall View with Altitude Rulers (Fig. 10). A trajectory is represented with its projection lines on the curtains. White indicators show an exact position of the aircraft in Fig. 14. Following are other features of this model (Fig. 14):

- FAF and IAF are visualized for the procedure; notice dashed lines.

- The approach trajectory is rotated about 90 degrees. Thus, the "back" curtain is clearly seen. The profile view of the procedure which is parallel to the runway is represented on the "back" curtain.
- The approach trajectory is not shown, only the projections of the aircraft position. The reason is that due to the selected viewing angle a representation would be imprecise and bring little information.

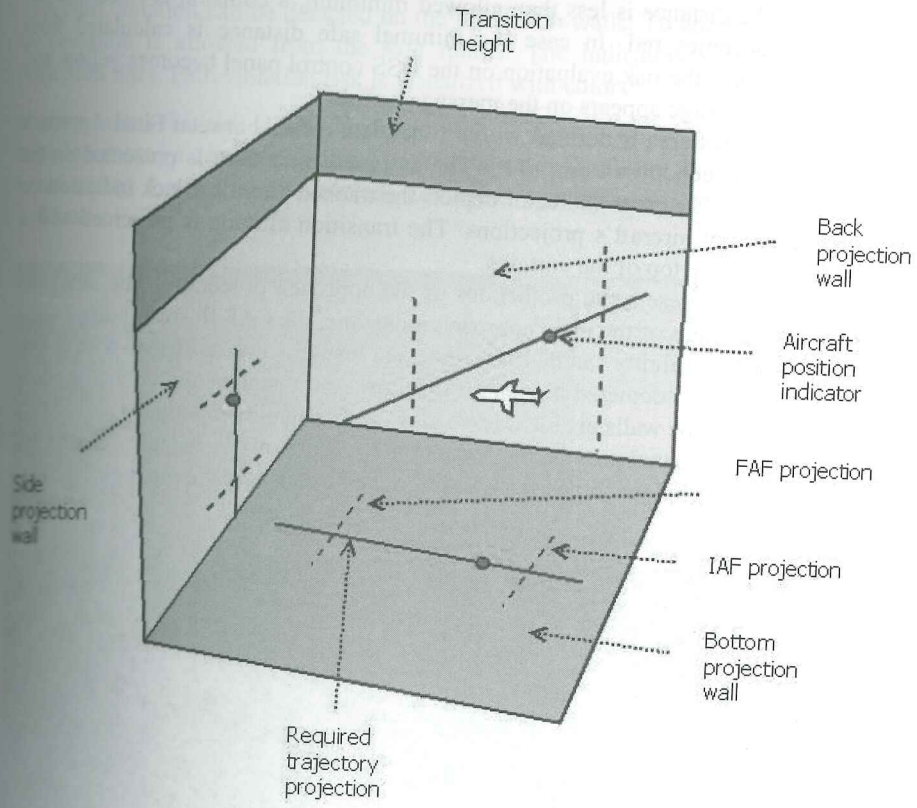

Fig. 14. SKY-Scanner visualization model

5 A Decision Support Scenario for SKY-Scanner

A situation is presented in a 3D view with a generalized landscape and tracks detected by the SKY-Scanner system. This screen imitates the situation that is viewed from the tower but without distracting details. Violations are visualized in 3D view and explained on the message board.

The whole zone under observation is divided into two areas:

1. A soft control area where the collision risk between the detected airplanes is tracked and the altitude is controlled;

2. A strict control area where a certain a landing procedure is assigned to the curtain track and the constraints (altitude, speed, track) are followed.

In the soft control area the aircraft altitude is observed. The longitudinal and vertical distances between aircraft are calculated, too. The main 3D window presents the context view; see Fig. 13. Here aircraft are represented with white indicators. Two aircraft are depicted in an example situation: the first is landing and the second is taking off. If the distance is less than allowed minimum, a collision is fixed and the aircraft icon becomes red. In case the minimal safe distance is calculated from predicted positions, the risk evaluation on the DSS control panel becomes yellow and an appropriate message appears on the message board.

The strict control area is defined within 6 nautical miles, between Final Approach Fix (FAF) and Touchdown Point (TP). The assigned procedure is presented on the curtains (Fig. 15). The green indicator depicts the tracked aircraft. Black indicators on the curtains present aircraft's projections. The transition altitude is presented with a different color on the top of the curtains.

The white lines present the projections of the approach procedure. The blue lines present the projections of the main approach milestone, the FAF fly-over point.

Aircraft position validity can be easily detected visually and confirmed with colors. The tracked aircraft is depicted in green if it follows the assigned approach procedure. The projections on the walls enable tracking until the touchdown point.

The DSS control panel presents the current information about the actual tracks and possible risks. 2D window comprises user interface buttons and the message board.

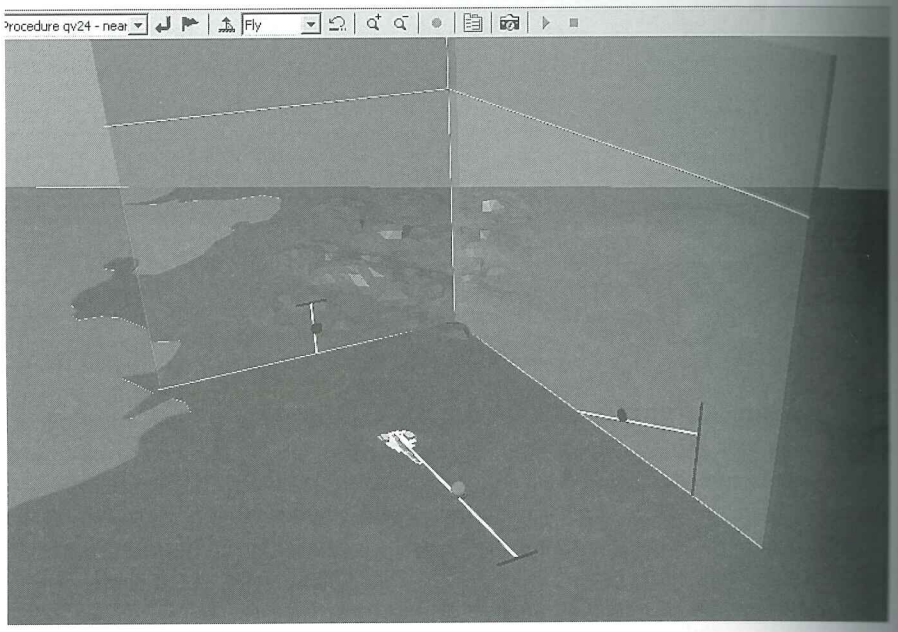

Fig. 15. Approach procedure violations are visible in the real time. The green indicator depicts a tracked aircraft. Two black indicators show the projections of the actual position.

When aircraft receives a clearance for landing, the DSS support scenario is as follows (Fig. 16):

1. Turn on projection curtains.
2. Select an approach procedure assigned. The projections of the procedure appear on the curtains.
3. Observe the situation: aircraft position should be on the projection lines.

A path violation can be detected on the projection walls. To adhere the procedure, no deviation is allowed from the safe funnel. The indicators have to follow the projection lines. Path violation risk is visualized with colors:

- green means that risk is evaluated zero
- yellow means a certain risk of the path violation in the future
- red means that a violation already occurred. Aircraft position is out of the safe funnel.

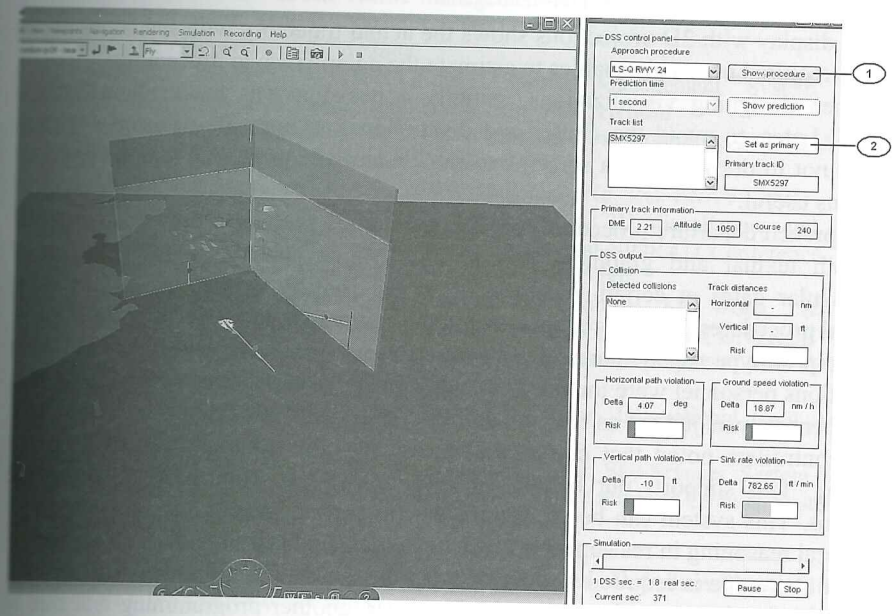

Fig. 16. 3D view with control panel.
Button 1 turns on curtains. Button 2 selects the tracked aircraft.

The upper part of the control panel shows DSS output. The 3D window attracts the controller's attention when the aircraft deviates from the projection lines. The control panel shows a violated parameter. Risk violation for each tracked parameter is depicted in green, yellow or red and also with a real number from 0 to 1.

5 Conclusions

The paper reports on the ongoing study of a new ATM paradigm. It is based on (1) lidar and radar data fusion, and (2) 3D visualization of aircraft trajectories within aerodrome traffic zone. A system will enable a human controller to view on a screen both the Is trajectory and the Ought to Be trajectory. Aircraft tracking with lidar enables the controller to observe flight phases that are near the ground. This is not possible using the sole primary radar. Hence, a decision support system will allow for a limited control over the aircraft in approach/departure phases.

In the DSS prototype, two ideas are integrated when visualizing trajectory restrictions. Human operator needs are satisfied in the following way:

- 3D display improves situational awareness; the airport environment is depicted with essential terrain obstacles;
- 3D walls with airport procedures reduce the cognitive workload.

3D display with 2D projections shows the actual trajectory with spatial restrictions on the required one. This enables to estimate the compliance to procedures.

When designing visualizations, initially a decision was made about "what to draw". Later, to avoid the clutter and distractions, an opposite decision was made, "what not to draw". Thus a screen became cleaner. Controllers found the projection curtains useful.

Approach/departure procedures provide a complex normative regulation composed of both textual and graphical descriptions. Major constraints form a series of rectangular gates at certain distances from the runway. Moreover, an airport has several procedures. Therefore representing the constraints in a DSS is a challenge.

The prototype was demonstrated to expert users. Air traffic controllers and airline operations personnel were included. The data included real-life tracks and procedures. The scenario was judged as realistic and effective while tracking aircraft in the ATZ. The controllers noted that currently they are not able to track aircraft during the landing. This supports our belief that such a DSS can contribute to future ATM systems. This system can be viewed as an information system that supports spatial-temporal reasoning in real-time air traffic management tasks.

The prototype is required to develop with MATLAB. This symbolic mathematical tool suits for demonstration purposes. However, another programming tool shall be chosen for industrial implementation. MATLAB is too slow for real time applications. Consider scalability issues.

The decision support scenario can be elaborated to visualize predicted positions and conflicts. The current prototype informs this on the message board.

References

1. Amaldi, P. , Fields, B., Rozzi, S., Woodward, P., Wong, W.: Operational Concept Report, Vol. 1 Approach Control, Vol. 2 Tower Control (No. OCR2-AD4-WP2-MU). Interaction Design Centre, Middlesex University, London, UK (2005)
2. Azuma, R., Daily, M., Krozel, J.: Advanced Human-Computer Interfaces For Air Traffic Management and Simulation. American Institute of Aeronautics and Astronautics. AIAA Flight Simulation Technologies Conference. (1996) http://www.cs.unc.edu/~azuma/AIAA.pdf
3. Azuma, R., Neely, H., Daily, M., Geiss, R.: Visualization Tools for Free Flight Air-Traffic Management. In: IEEE Computer Graphics and Applications, vol. 20, no. 5, pp. 32–36. IEEE Press (2000) http://www.cs.unc.edu/~azuma/cga2000.pdf
4. Bourgois, M., Cooper, M., Duong, V., Hjalmarsson, J., Lange, M., Ynnerman, A.: Interactive and Immersive 3D Visualization for ATC. 6th USA-Europe ATM R&D Seminar. Baltimore, USA (2005) http://www.atmseminar.org/past-seminars/6th-seminar-baltimore-md-usa-june-2005/papers/paper_040
5. van Es, G.W.H.: Review of Air Traffic Management-related accidents worldwide: 1980 - 2001. Technical report, NLR-TP-2003-376 (2003) http://www.nlr.nl/smartsite.dws?id=2888
6. EU Transport Research - 4D Virtual Airspace Management System (2005) http://ec.europa.eu/research/transport/projects/article_3722_en.html
7. Instrument Procedures Handbook. U.S. Department of Transportation, Federal Aviation Administration, Flight Standards Service (2007)
8. ICAO – Instrument Approach Chart, Napoli/Capodichino, No. 352. ENAV (2003)
9. Lange, M., Hjalmarsson, J., Cooper, M., Ynnerman, A.: 3D Visualization and 3D and Voice Interaction in Air Traffic Management. The annual SIGRAD Conference. Umea, Sweden (2003). http://www.ep.liu.se/ecp/010/005/ecp01005.pdf
10. Rozzi, S., Boccalatte, A., Amaldi, P., Fields, B., Loomes, M., Wong, W.: D1.1: Innovation and Consolidation Report. Technical Report, EUROCONTROL (2007) http://www.eurocontrol.int/eec/gallery/content/public/documents/projects/CARE/CARE_INO_III/3D-2D_Innovation_and_consolidation.pdf
11. Rules of the Air and Air Traffic Services, 13th Edition. International Civil Aviation Organization (1996)
12. Salerno, M., Costantini, G., Carota, M., Casali, D.: The Sky-Scanner System for Air Traffic. Management: a Simulation Software. International Journal of Circuits, Systems and Signal Processing. vol. 4, issue 1, pp. 1–8 (2010) http://www.naun.org/journals/circuitssystemssignal/19-209.pdf
13. SESAR Air Transport Framework: The Current Situation. Definition Phase, Milestone Deliverable D1. SESAR Consortium (2006) http://www.eurocontrol.int/sesar/gallery/content/public/docs/DLM-0602-001-03-00.pdf
14. SESAR Air transport Framework: The performance target. Definition Phase, Milestone Deliverable D2. SESAR Consortium (2006) http://www.eurocontrol.int/sesar/gallery/content/public/docs/DLM-0607-001-02-00a.pdf
15. Wong, B.L.W., Rozzi, S., Boccalatte, A., Gaukrodger, S., Amaldi, p., Fields, B., Loomes, M., Martin, P.: 3D-in-2D Displays for ATC. In: 6th EUROCONTROL Innovative Research Workshop, pp. 42–62 (2007) http://inoworkshop.eurocontrol.fr/index.php?option=com_content&view=article&id=32&Itemid=16