Can Soft Computing Techniques Enhance the Error Diagnosis Accuracy for Intelligent Tutors?

Nguyen-Thinh Le and Niels Pinkwart

Clausthal University of Technology Germany
{nguyen-thinh.le,niels.pinkwart}@tu-clausthal.de

Abstract. Problems for which multiple solution strategies are possible can be challenging for intelligent tutors. These kinds of problems are often the norm in exploratory learning environments which allow students to develop solutions in a creative manner without many restrictions imposed by the problem solving interface. How can intelligent tutors determine a student's intention in order to give appropriate feedback for problems with multiple, quite different solutions? This paper focuses on improving the diagnosis capabilities of constraint-based intelligent tutors with respect to supporting problems with multiple possible solution strategies. An evaluation study showed that by applying a soft-computing technique (a probabilistic approach for constraint satisfaction problems), the diagnostic accuracy of constraint-based intelligent tutors can be improved.

Keywords: Soft computing, constraint satisfaction problems, error diagnosis, intelligent tutoring systems.

1 Introduction

Intelligent tutors which are able to deal with problems that have multiple solution variants usually have to face the challenge of diagnosing the student's intention, i.e., determining which solution strategy the student is pursuing as she is trying to solve a given problem. Diagnosing the solution strategy intended by a student is important, because only if this is done correctly, an accurate error diagnosis can be conducted and, in consequence, appropriate feedback on the student's solution can be given.

There are two main and established approaches for building intelligent tutors: model-tracing [1] and constraint-based modeling [2]. While a model-tracing system is able to diagnose the student's intention by monitoring and relating the student's problem solving steps to the correct solution paths captured in the cognitive model [3], constraint-based tutors do usually not contain sufficient information to decide on the most plausible hypothesis about the student's intention underlying a student's solution. Constraint-based error diagnosis can be conceived as a constraint satisfaction problem. If a student solution is correct, then all constraints will be satisfied. If an erroneous student solution is evaluated, an inconsistency between the erroneous student solution and the constraint system occurs, i.e., one or more constraints will be violated. In this case, the problem of error diagnosis is considered over-constrained.

S.A. Cerri et al. (Eds.): ITS 2012, LNCS 7315, pp. 320–329, 2012.

The goal of constraint-based error diagnosis is not to search for a correct solution, but rather to identify the constraint violations which lead to the inconsistency between an erroneous solution and the constraint system. The result of this constraint-based error diagnosis – the constraints that are violated and those that are not – is a good starting point for giving feedback to students if there is only one main solution for a problem (possibly with slight variations that the constraint system can accommodate). Yet, it is off less use if different solution strategies (i.e., different constraint sets) for a problem are possible.

To build constraint-based intelligent tutors which *are* able to handle problems with multiple solution strategies, the approach presented in this paper adopts a soft computing technique for solving constraint satisfaction problems: a probabilistic framework. For that purpose, each constraint is associated with a constraint weight which represents heuristic information indicating the importance of the constraint. As we will argue in this paper, these weights can be used to hypothesize the student's intention underlying his solution.

In the next section, we review some typical soft computing techniques for solving constraint satisfaction problems and argue why we choose the probabilistic approach. Then, we briefly describe a weighted constraint-based model which can be used to build intelligent constraint-based tutors for problems with multiple solution strategies. Next, we show an evaluation study which confirms that by applying soft computing techniques, the diagnostic accuracy for constraint-based intelligent tutors can be enhanced as compared to traditional constraint-based modeling approaches. Finally, we summarize the benefits of the weighted constraint-based model and propose some future work.

2 Soft Computing for Constraint Satisfaction Problems

To deal with the issue of over-constrained satisfaction problems, some researchers have attempted to distinguish the level of importance between constraints. Here, hard constraints represent conditions which must always hold, and soft constraints represent preferences which should be satisfied when possible. Several techniques have been devised to express soft constraints and to allow their violation. The most popular approaches include fuzzy constraint satisfaction problems (CSPs) [4], cost-minimizing CSPs [5], partial CSPs [6], and probabilistic CSPs [7].

A partial CSP framework attempts to soften a constraint satisfaction problem by changing the domain of variables/constraints or a constraint system in several ways: by 1) enlarging the domain of a variable, 2) enlarging the domain of a constraint, 3) removing variables of a constraint, or 4) removing a constraint from the constraint system. This approach is not appropriate for enhancing the error diagnosis capability of a constraint-based intelligent tutor due to the following reason. To choose the most plausible solution strategy, we need to consider all possible evidence (based on solution components), whereas a partial CSP framework attempts to eliminate constraints which can be violated by a student solution and thus, evidence supporting the process

of hypothesizing the student's intention during error diagnosis is also eliminated. As a consequence, the diagnosis capability of a constraint-based intelligent tutor would be degraded.

While a partial CSP framework requires satisfying a partial set of constraints, the fuzzy CSP and the cost-minimizing CSP approaches allow all constraints to be satisfied by defining a preference ranking of the possible instantiations according to some criteria depending on the constraints. The solution of a fuzzy/cost-minimizing constraint satisfaction problem is the instantiation which meets the highest satisfaction degree. The fuzzy CSP framework associates a level of preference with each instantiation of variables in each constraint and searches a solution by maximizing the satisfaction degree of the least preferred constraint. On the contrary, in a cost-minimizing CSP framework instantiations are assigned with a cost and the goal is to find a solution which minimizes the total sum of costs of the chosen instantiation for each constraint. These approaches are very appropriate for problem situations where preference levels for certain instantiations of the constraint variables are available. They are, however, not well suited for improving the capability of constraint-based error diagnosis. The problem of error diagnosis in a constraint-based tutor is a situation where it is almost impossible to specify instantiations of constraint variables in advance because the amount of constraints required to model domain knowledge is relatively high and the space of possible instantiations is large.

A probabilistic CSP framework, finally, contains a set of constraints. Each of these constraints is intended to represent a piece of knowledge. It is associated with a probability of relevance. That is, some constraints can be specified as relevant to the problem with complete certainty, and for some others it can be specified that they may or may not be relevant to this problem. It is usually assumed that the probabilities of two different constraints are independent from each other. A solution of the probabilistic constraint satisfaction problem is an instantiation of all variables that has a maximal probability. A probabilistic CSP framework can be used to model situations where each constraint can be specified with a certain probability. Since such a situation is applicable to constraint-based intelligent tutors, Le and Pinkwart proposed to adopt the probabilistic approach for enhancing the diagnosis capability of traditional constraint-based tutors [8].

In the approach pursued here, a probability associated with each constraint indicates a measure of the importance of a constraint and is being referred to as a constraint *weight*. Applying the probabilistic CSP approach, the automated evaluation of student solutions resembles the assessment of written examinations by a human tutor: not only the quantity of the "correct" statements made by the student is important for the final mark, but also the importance of the contained statements.

In our approach, one goal for using constraint weights (in addition to coming to a more realistic estimation about solution quality by considering different importance degrees for different constraints) is to choose the most plausible hypothesis about the solution strategy pursued by a student.

3 Weighted Constraint-Based Models

As presented in more detail in [8], a weighted constraint-based model (WCBM) consists of a semantic table, a set of weighted constraints, and transformation rules. The WCBM model assumes that a problem can be solved by applying different alternative solution strategies, and each of them can be implemented in different variations. The semantic table is used to model alternative solution strategies and to represent generalized components for each solution strategy. Constraints are used to check the semantic correctness of the student solution with respect to the requirements specified in the semantic table and to examine general well-formedness conditions for a solution. Transformation rules serve to extend the coverage of a solution space (for instance, they allow for including general rules such as math equations, e.g., X(Y+Z) = XY+XZ, into the diagnosis process). The process of diagnosing errors in a student solution performed by a WCBM tutor consists of two interwoven tasks (hypotheses generation and hypotheses evaluation) which take place on two levels (strategy and solution variant level). First, on the strategy level, the system generates hypotheses about the student's intention by iteratively matching the student solution against the solution strategies that are specified in the semantic table. Then, once a solution strategy has been matched, the process initiates hypotheses about the student's solution variant by matching components of the student solution against corresponding components of the selected solution strategy. Next, hypotheses generated on the solution variant level are evaluated, and the most plausible variant of the student solution (within a strategy) is chosen. In this process, hypotheses are evaluated with respect to their plausibility by multiplying the weights of constraints which are violated by that hypothesis according the following formula:

$$\text{Plausibility}_{\text{Prod}}(\text{H}) = \prod_{i=1}^{N} Wi, \text{ where } \text{W}_\text{i} \text{ is the weight of a violated constraint} \quad (1)$$

On the strategy level, the hypothesis with the highest plausibility score (note that important constraints have weight values close to 0, while less important ones have weights close to 1) corresponds to the solution strategy which the student has most likely intended to pursue in his solution. This hypothesis is selected, and diagnostic information is derived from constraint violations resulting from the plausibility computation of the selected hypothesis.

4 Evaluation

In [8] it has been shown that an intelligent tutor built based on the weighted constraint-based model is better than a corresponding intelligent tutor that is built based on a classical constraint-based modeling approach with respect to evaluating *intention analysis*. The intention analysis of a tutor is the capability to hypothesize the solution strategy underlying the student solutions correctly. In this paper, we intend to go the next step and compare the *diagnostic validity* of a weighted constraint-based tutor with a corresponding traditional constraint-based tutor. Evaluating the diagnostic

validity means determining whether the diagnostic result is acceptable with respect to a gold standard. The diagnostic validity partially depends on the capability of intention analysis, because if the intention of the student is hypothesized wrongly, this makes it more difficult to detect errors with a high validity.

4.1 Design

To compare the diagnostic validity of the weighted constraint-based model with the classical constraint-based modeling approach, we used two versions of INCOM, a tutoring system for logic programming. The first one applies the weighted constraint-based model (INCOM-WCBM). A modified version of INCOM (INCOM-CBM) corresponds to a classical constraint-based tutor and uses constraints without weight values. Classical constraint-based tutors have no "standard" way of dealing with multiple solution strategies that each come with different constraint sets. To realistically compare INCOM-WCMB and INCOM-CBM, such a feature for plausibility of hypotheses about the solution strategy intended by the student had to be added to INCOM-CBM. We did this by summing up the number of constraint violations caused by each hypothesis:

$$\text{Plausibility}_{\text{Add}}(H) = |C| \tag{2}$$

C is the set of all constraint violations caused by each hypothesis H. This approach seems quite natural and straightforward in a situation where constraints do not have weights but can just either be violated or nor – it models the idea that if a student solution violates X constraints for the constraint system corresponding to solution strategy A and Y(>X) constraints for the constraint system corresponding to solution strategy B, then the student has most likely pursued strategy A. That is, the plausibility of a hypothesis is associated to the number of corresponding violated constraints.
We collected exercises and solutions from past written examinations for computer science (specifically, a course in logic programming and AI) and input them into the two systems under comparison (INCOM-WCBM and INCOM-CBM). In total, we collected 221 student solutions, where the solutions have been collected based on the following criteria: 1) any piece of code which satisfies minimal requirements of interpreting it as a Prolog program is considered a solution, 2) syntax errors in the solutions are ignored (because during the written examination session students did not have access to a computer), 3) both correct and incorrect solutions are taken into account. Following are short versions of the seven tasks we selected:

1. Access to specific elements within an embedded list;
2. Querying a data base and applying a linear transformation to the result;
3. Modification of all elements of a list subject to a case distinction;
4. Creation of an n-best list from a data base;
5. Computing the sum of all integer elements of a list;
6. Counting the number of elements in an embedded list;
7. Finding the element of an embedded list which has the maximum value for a certain component.

To define a gold standard, we invited a human expert in logic programming to inspect all errors (i.e., diagnosis results) provided by the INCOM-WCBM system after analyzing 221 student solutions, either confirming or rejecting it. In addition, the human tutor had the possibility to add general comments which are not specific to the presented errors, for example, if he thought that crucial errors have been missed (due to high resource requirements for this human expert tasks, we did not involve multiple graders).

Once the gold standard was specified, we are able to determine the set of *gold standard errors* (which should have been identified by the system) and *gold standard not-errors* (which should not have been identified by the system). The sets *retrieved errors* and *not-retrieved errors* are the results of the systems diagnoses.

4.2 Results

To measure the diagnostic validity of an intelligent tutor, we use the metrics Recall and Precision. With respect to Table 1, Precision and Recall are defined as follows [9]:

$$Recall = \frac{A}{A + C};\ Precision = \frac{A}{A + B}$$

Table 1. Categories for Precision and Recall

	Gold standard Errors	**Gold standard Not-errors**
Retrieved errors	A	B
Not retrieved errors	C	

Under these definitions, a high precision means that the model is based on fairly reliable constraints which have a low risk of producing false alarms, i.e., the developer was careful to avoid particularly risky constraints. A high recall, on the other hand, means that the diagnosis has a good coverage, i.e., it considers a sufficiently rich set of relevant constraints.

Table 2 summarizes the results of system diagnoses of INCOM-WCBM and INCOM-CBM with respect to diagnostic validity. From this table, we can notice three aspects. First, the precision of INCOM-WCBM is high (0.953), as is the recall (0.97). The latter indicates that the set of weighted constraints covers possible errors in the domain of logic programming sufficiently. As such, one can state that the diagnostic validity of WCBM-INCOM is high: one can expect this system to give appropriate feedback to students also in the situation of tasks that have multiple possible solution strategies (as is the case for most of the tasks we considered). Would these good results also have been possible without the constraint weights? Table 2 gives an answer to this: The second claim we can make is that the precision of INCOM-WCBM is remarkably higher than the one of INCOM-CBM (0.459). This can be attributed to the weight values associated to each constraint in the INCOM-WCBM, because constraint weights are used to determine the student's intention and to control the error diagnosis process.

Table 2. Evaluation of the diagnostic validity

	INCOM-WCBM		**INCOM-CBM**	
Task	Precision	Recall	Precision	Recall
1	0.9	0.93	0.466	0.724
2	0.941	1	0.666	0.875
3	1	1	1	1
4	0.907	0.929	0.488	0.909
5	0.991	0.983	0.208	0.297
6	0.961	0.961	0.198	0.359
7	0.974	0.987	0.19	0.185
Avg.	**0.953 (sd.=0.04)**	**0.97 (sd.=0.03)**	**0.459 (sd.=0.3)**	**0.621 (sd.=0.33)**

Third, we notice that while the precision of INCOM-WCBM seems to be stable across the seven tasks, the precision of INCOM-CBM tends to decrease from task 4 on. This can be explained by the fact that the complexity of tasks 4-7 is higher than the one of tasks 1-4. In addition, we can see that both INCOM-WCBM and INCOM-CBM reach their maximum precision value at Task 3. Yet, this has to be interpreted in the light of the fact that only four student solutions were available for this task, and all of them contained very few errors.

We next want to illustrate the difference of diagnostic validity between INCOM-WCBM and INCOM-CBM using an erroneous example student solution for Task 6:

```
countz(N,L):- L=[], N is 0.
countz(N,L):- L=[Head|Rest], countz(N1, Rest), N is N1+1.
```

Task 6 can, among others, be solved by applying either a naive recursive strategy or a tail recursive strategy. Applying the weighted constraint-based model, INCOM-WCBM produced two hypotheses on the strategy level. The first hypothesis (H1) is that the student has implemented the naive recursive strategy and the student solution has violated three constraints, i.e., the solution has three errors (Table 3).

Table 3. Hypothesis 1 of INCOM-WCBM: The naive recursive strategy

ID	**Weight**	**Feedback**
s7c	0.8	At the position **N,** a number is expected. countz(**N**,L):- L=[], N is 0.
p5c	0.8	The variable **Head** in the clause body is not used. Is it superfluous, or did you forget a subgoal to use it, or should it be an anonymous variable? countz(N,L):- L=[**Head**|Rest], countz(N1, Rest), N is N1+1.
s5b	0.1	The arithmetic subgoal is superfluous. countz(N,L):- L=[], **N is 0**.

Since the system does not allow the subgoal "N is 0" as a value assignment, the first constraint violation indicates that the variable **N** needs to be instantiated with a number and the third constraint violation shows that the arithmetic subgoal is not required. The second constraint violation shows that a variable **Head** is present but not used. Applying formula (1), the plausibility of this hypothesis is 0.8*0.8*0.1=0.064.

The second hypothesis (H2) is that the student has implemented the tail recursive strategy. Following this hypothesis, the student solution violated constraints as listed in Table 4. These constraint violations occurred because the system tried to match the student solution with components of the tail recursive strategy which requires the following clauses: 1) the main clause which calls the accumulative predicate, 2) the base case of the accumulative predicate, and 3) a recursive clause which accumulates a value using an accumulative variable. Since the student solution could not be matched well to the tail recursive strategy (it violated five constraints, each with weight 0.01), the plausibility of this hypothesis is 0.01^5. As such, H2 is less plausible than hypothesis H1. As a result, INCOM-WCBM decided that the student has most likely pursued the naive recursion strategy.

Table 4. Hypothesis 2 of INCOM-WCBM: The tail recursive strategy

ID	Weight	Feedback
s7g1	0.01	If you want to implement a non-recursive clause, at least one clause must have been specified as non-recursive.
s7g	0.01	A base case is missing.
s7h	0.01	A recursive case is required. Or did you forget a subgoal in a clause body?
s7i	0.01	countz/2: this predicate definition has more base cases than required. **countz(N,L):- L=[], N is 0.**
s7j	0.01	countz/2: this predicate definition has more recursive cases than required. **countz(N,L):- L=[Head|Rest], countz(N1, Rest), N is N1+1.**

INCOM-CBM also generated two hypotheses. The first hypothesis (H1A) is that the student has implemented the naive recursive strategy. This hypothesis caused, in addition to three constraint violations in Table 3, two others, which address the arithmetic argument in the second clause. (`countz(N,L):- L=[Head|Rest], countz(N1, Rest), N is` **`N1+1`**.):

1. At the position **N1**, a constant number is required.
2. At the position **1**, a variable is required.

These additional constraint violations resulted from the fact that INCOM-CBM was not able to choose the most plausible hypothesis generated on the solution variant level. By matching the arithmetic term N1+1 of the student solution against the semantic table, two hypotheses have been generated: H1_1={map(N1, N1'); map(1,1)} and H1_2={map(N1, 1); map(1,N1')}, where N1'+1 is a corresponding arithmetic

term specified in the semantic table. INCOM-CBM chose the second hypothesis on the solution variant level and forwarded this to the strategy level. Applying formula (2), the plausibility for hypothesis H1A is the number of violated constraints, i.e., the plausibility is 5.

The second hypothesis (H2A) INCOM-CBM has generated is that the student has implemented the tail recursive strategy. Diagnosing errors following this hypothesis, INCOM-CBM produced the same five constraint violations as in Table 4. That is, the plausibility of this hypothesis is also five and equal to the plausibility of hypothesis H1A. Thus, INCOM-CBM was not able to decide on the most plausible hypothesis about the student's solution variant, because two hypotheses of INCOM-CBM (naive recursive and tail recursive) have the same plausibility score (each hypothesis produces five constraint violations). Therefore, the system could not decide which strategy was most likely pursued by the student. Many of the cases where INCOM-WCBM had a higher diagnostic validity than INCOM-CBM can be explained in a similar fashion: the weights outperformed the simple counting of constraint violations.

4.3 Possible Limitations

Overall, our results indicate that adding constraint weights can improve the error diagnosis of constraint-based intelligent tutors. Yet, our results concerning the diagnostic validity of INCOM-WCBM might be a bit optimistic, because our method of determining the gold standard was based on actual system's diagnosis results. This might have created a bias toward these error interpretations. Other comparable tutor systems for programming, e.g. PROUST [10], APROPOS2 [11], and Hong's Prolog tutor [12], which also provide problems with multiple solution variants, defined the gold standard by hand analysis. That is, a human expert analyzed each student solution and detected errors independent from the system's diagnostic result. However, this way of defining a gold standard by hand analysis is not well-suited for constraint-based tutors due to two reasons. First, the human expert has to know the large set of constraints (the current implementation of INCOM includes 147 constraints [13]) which represent error types, and relate every error detected in a program to a corresponding constraint. This is a very laborious undertaking for a human expert. Second, a constraint can be relevant to different components of the same solution many times. If a human expert has to assign a detected error to one of the existing constraints, she would have to iterate through the list of constraints as many times as the system does. This is a bothersome and error prone task. Hence, we specified the gold standard in a way that provides a balance between human and system orientation.

Another possible limitation is that, in our study, we compared INCOM-WCMB to a classical constraint based tutor which made use of a reasonable but quite straightforward method for determining student strategies. Adding more advanced features (i.e., more sophisticated methods for guessing solution strategies) could probably have increased the diagnostic validity of our "control condition" INCOM-CBM. Yet, it remains to be shown if, with additional features but without weights, the diagnostic validity could have reached the level that can be reached with weighted constraints as available in INCOM-WCMB.

5 Conclusion

In this paper we have presented a weighted constraint-based model for intelligent tutors. We also demonstrated an evaluation study which compared the diagnostic validity of a tutor applying the weighted constraint-based model and a classical constraint-based tutor for the same domain in logic programming. The evaluation study showed that the precision of error diagnosis provided by the weighted constraint-based tutor (0.953) is remarkably higher than the one of the classical constraint-based tutor (0.459). From this result and the evaluation study in [8], we can conclude that the error diagnosis capability of constraint-based tutors can be improved if constraints are enriched with weight values which represent the importance of a constraint. In the future, we plan to test the applicability of the weighted constraint-based model in other domains.

References

1. Anderson, J.R., Corbett, A.T., Koedinger, K.R., Pelletier, R.: Cognitive tutors: Lessons learned. Journal of the Learning Sciences 4, 167–207 (1995)
2. Ohlsson, S.: Constraint-based Student Modeling. In: Greer, J.E., McCalla, G.I. (eds.) Student Modelling: The Key to Individualized Knowledge-based Instruction, pp. 167–189. Springer, Berlin (1994)
3. Anderson, J.R., Betts, S., Ferris, J.L., Fincham, J.M.: Neural imaging to track mental states while using an intelligent tutoring system. Proceedings of the National Academy of Science 107, 7018–7023 (2010)
4. Dubois, D., Fargier, H., Prade, H.: Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. Applied Intelligence 6, 287–309 (1996)
5. Schiex, T., Cedex, C.T., Fargier, H., Verfaillie, G.: Valued constraint satisfaction problems: Hard and easy problems. In: Joint Conference in AI (1995)
6. Freuder, E., Wallace, R.: Partial constraint satisfaction. Artificial Intelligence 58, 21–70 (1992)
7. Fargier, H., Lang, J.: Uncertainty in Constraint Satisfaction Problems: A Probabilistic Approach. In: Moral, S., Kruse, R., Clarke, E. (eds.) ECSQARU 1993. LNCS, vol. 747, pp. 97–104. Springer, Heidelberg (1993)
8. Le, N.-T., Pinkwart, N.: Adding Weights to Constraints in Intelligent Tutoring Systems: Does It Improve the Error Diagnosis? In: Kloos, C.D., Gillet, D., Crespo García, R.M., Wild, F., Wolpers, M. (eds.) EC-TEL 2011. LNCS, vol. 6964, pp. 233–247. Springer, Heidelberg (2011)
9. Rijsbergen, C.J.V.: Information retrieval, 2nd edn. Butterworths, London (1979)
10. Johnson, W.L.: Understanding and debugging novice programs. Artificial Intelligence 42(1), 51–97 (1990)
11. Looi, C.-K.: Automatic debugging of Prolog programs in a Prolog intelligent tutoring system. Instructional Science 20, 215–263 (1991)
12. Hong, J.: Guided programming and automated error analysis in an intelligent Prolog tutor. International Journal of Human-Computer Studies 61(4), 505–534 (2004)
13. Le, N.-T.: Using weighted constraints to build a tutoring system for logic programming. PhD Thesis. University of Hamburg (2011)