On dealing with dynamic utility of learned knowledge, a case study in geometry problem-solving task

Masaki Suwa and Hiroshi Motoda
Advanced Research Laboratory, Hitachi Ltd.,
2520, Hatoyama, Saitama, 350-03, Japan

Abstract

Cost-effective utility of learned knowledge consists of two terms; the cumulative matching costs of testing to apply the knowledge in problems and cumulative speed-up effects its application has on problem-solving time. To evaluate these two factors correctly, measuring dynamic utility of learned knowledge when it is *actually applied* to problems is required. In that scheme, however, since utility value of knowledge is in general negative before its application brings about speed-up effects in some problems, we encounter a new issue. What kind of strategy should be taken to retain and/or discard knowledge with currently negative utility value during a training session? This is important because it determines the available set of knowledge to be provided for the subsequent problem in each stage within a training session. The strategy taken greatly affects not only the problem-solving costs needed for a training session but also the kinds of the knowledge that gains a positive estimation at the end of the training session and thus will be provided for a test session. This issue is crucial especially in situations where there are strong *interactions* of co-existing knowledge during a training session. Five strategies are presented in the domain of geometry problem-solving and evaluated from the following criterion; a desirable strategy may be the one that allows really useful knowledge to gain a positive estimation at the end of a training session, with minimizing the problem-solving costs during the session.

1 Introduction

Measuring cost-effective utility of learned knowledge, i.e. matching costs needed for testing to apply it[1] and speed-up effects its application has on problem-solving time, is essential to avoid learning paradox, i.e. learning may degrade problem-solving performance (Gratch *et al.* 1991a; Minton 1985; Submanian *et al.* 1990).

Various techniques of measuring utility value have been proposed so far. In the PRODIGY/EBL system (Minton 1990), speed-up effect is approximately estimated by *initial observation* when the knowledge was learned

[1] This includes the cost taken in testing to apply the knowledge but in vain.

from a problem, since it's difficult to measure it when the knowledge has been actually applied. This simplification was criticized because approximation only by a single instance from which the knowledge was learned does not reflect fluctuating factors of speed-up effects by various problem contexts, i.e. *interactions* from other co-existing knowledge and the kinds of currently existing problem statements (Gratch *et al.* 1991b). Gratch proposed a method in which the incremental utility is measured every time a piece of knowledge is learned, and if it is statistically assured as positive after several measurements then it will be adopted as control strategy (Gratch *et al.* 1992). However, this technique neglects the cost that will be spent in testing to apply knowledge in vain after adopted as a member of available control knowledge. Actually we have encountered in the domain of geometry such knowledge that exhibits big speed-up effects in its application but does not pay as a whole because the cost of testing to apply it in vain to problems is extremely large. These disadvantages are attributed to *staticly* measuring utility when knowledge is learned. From these concerns, our view in this paper is that *dynamic*[2] analysis of cost-effective utility of learned knowledge is required when it is actually applied, *not* when it has been learned[3].

In the scheme of dynamic measurement of utility values, however, we encounter a new technical issue about how to deal with gathered utility data. Utility value of knowledge is in general negative at first due to the cumulative cost of testing to apply the knowledge, before its application brings about speed-up effects in some problems. This means that we cannot discard a piece of knowledge because it has a negative utility value at present, i.e. it may be useful knowledge which will have positive speed-up effects in the near future[4]. Therefore, in this paper we examine what kind of strategies should be taken to retain and/or discard knowledge of currently negative utility value during a training session. This is important because it determines the available set of knowledge to be provided for the subsequent problem in each stage within a training session. The strategy taken greatly affects not only the problem-solving costs during the training session but also the kinds of the knowledge that gains a high utility value at the end of the training session. Under an undesirable strategy, some pieces of really useful knowledge which should have scored high utility value may

[2]Due to dynamic measurement of cost-effective utility, the utility value of knowledge is inevitably influenced by interactions of currently available knowledge. We will mention it in the related work section.

[3]As many researchers pointed out, dynamic measurement of speed-up effect is difficult and time-consuming. We show later a simplified technique of measuring it when knowledge is applied, but discussing the technique itself is not the main issue of this paper, as we say in this introduction.

[4]Things would be simpler in static analysis because initial estimation of speed-up effects may give a positive value.

not be found in the list of available knowledge when a training session ends, and it will degrade performance in the subsequent test session.

2 Strategies for dealing with utility values

Some variations of the following two strategies will be evaluated in this paper.

1. To arrange the pieces of learned knowledge in a descending order of utility value every time a problem is solved and utility values are measured, and then to retain the best N pieces of knowledge in terms of utility value in the list of available knowledge for the next problem.
2. To discard such knowledge as soon as possible during a training session whose utility has been assured as "bad". For judging the "badness" of knowledge, we use the technique of *statistical range estimation* of the true population mean of utility values.

Dynamic utility values are dealt with under each variation of the above strategies during the training session and after that a test problem is solved by use of only the knowledge which has scored a positive utility as a total. And we evaluate the feasibility of those strategies according to the following standard;

1. whether or not the strategy can reduce the problem-solving costs during the training session as much as possible,
2. whether or not the strategy can estimate knowledge of really high utility as positive and available for the test session.

Before going into the detail of the experiments done in geometry problem-solving domain, we will characterize the domain itself, the learning system and the kind of learned knowledge in the next section. Further we will describe briefly the technique of dynamic analysis of utility value.

3 Characterization of the learning system

3.1 Domain

Geometry problem-solving is to prove a goal statement deductively by use of domain knowledge when some problem statements are given. The characteristics of this domain are

- that a huge variety of instatiations are produced by preconditions and/or consequent-parts of domain knowledge in applying it and this may potentially impose heavy burden of matching costs on problem-solving performance,
- and that not only goal-oriented backward reasoning but also bottom-up forward reasoning is essential to constructing a successful proof-tree efficiently especially in complicated problems (Sweller 1988),

- and therefore that search control knowledge suggesting which domain knowledge should be used in a forward manner to an already asserted problem statement plays a significant role in avoiding irrelevant paths of inference.

The first point connotates that in geometry domain there could be potentially heavy interactions from other co-existing knowledge and/or other factors that will greatly affect dynamic cost-effective utility data. This requires us to provide a feasible strategy of dealing with dynamic utility values.

3.2 Search control knowledge

We do experiments in this paper on the learning system PCLEARN (Suwa *et al.* 1994a; 1994b). It chunks essential features from the problem diagrams that will work as search control knowledge. The learned knowledge helps a problem solver select appropriate domain knowledge to be applied to explored problem statements in a forward manner.

The kind of search control knowledge the PCLEARN system learns is called "perceptual-chunks (or schemas)" (Greeno 1983) and its significant role in controlling problem solver's search has been extensively studied in the domain of geometry problem-solving (Koedinger *et al.* 1990; McDougal *et al.* 1992; McDougal *et al.* 1993). The characteristic of a perceptual-chunk is that it is acquired from a problem as *a chunk of diagram elements which are meaningful and grouped together with each other* (Suwa *et al.* 1994a; 1994b). In this sense, a perceptual chunk can be regarded as a local chunk frequently found in the diagrams of many problems. Therefore, it does not contribute to directly accomplishing the goal statement, unlike search control knowledge learned by goal-oriented learning technique, such as explanation-based learning method (Minton *et al.* 1989). Rather, a perceptual-chunk only helps problem solvers do *local search* at a local control decision node during problem-solving traces.

The "freeness" of perceptual-chunks from the goal structure of the problem from which they are learned is advantageous in assuring higher applicability to many future problems (Suwa *et al.* 1994a). But its "localness" is a major weak point at the same time because it may not always control search in a correct direction globally. Therefore we need to justify whether the learned perceptual-chunks are worth being retained as available knowledge. Only empirical evaluation of their utility values gives us this justification by actually applying them in future problems with different problem situations (Suwa *et al.* 1993).

3.3 Measuring cost-effective data of applied knowledge

We briefly describe the method of measuring cost-effective utility of perceptual chunks. The utility value of a perceptual-chunk is calculated through-

out its use over many problems according to the following formula,

$$Utility = TotalEffects - TotalMatchCosts, \quad (0.1)$$

where $TotalEffects$ is the cumulative speed-up effect that results from applying the perceptual-chunk frequently, and $TotalMatchCosts$ is the cumulative time cost spent in testing to apply the perceptual-chunk in vain over frequent testings during many problems. Every time a problem is solved, the matching costs are measured for each perceptual-chunk, and if a perceptual-chunk has been applied in solving the problem, speed-up effect by its use is also calculated. In recording both values, we normalize them by the complexity of the current problem in question[5]. This is intended to prevent costs and/or effects measured in larger problems from preferrably governing the utility of knowledge.

If we intend to correctly make dynamic analysis of speed-up effect a piece of knowledge brings about, running the system with and without the knowledge on each problem would be required, and this would have to be done for each piece of knowledge in question. Doing so requires solving the same problem multiple times and costs too much, as many researchers pointed out (Minton 1990; Gratch *et al.* 1991b). In this paper we simplify the analyzing process by comparing the current solution trace obtained by applying learned knowledge with the solution trace constructed when the same problem was solved without using any learned knowledge[6]. The speed-up effect is calculated according to the following formula;

$$Effect = NoMacroModeCosts - MacroModeCosts, \quad (0.2)$$

$MacroModeCosts$ is the summation of the matching costs taken at the following nodes of the current solution trace; (a) the node (N_{apl}) to which the perceptual-chunk has been applied, (b) all the other nodes that had to be verified for applying the chunk, and (c) all the nodes newly produced by its application. $NoMacroModeCosts$ is the summation of the matching costs taken at the following nodes of the solution trace constructed without using any perceptual-chunks; (a) the ones that correspond to the above three kinds of nodes and (b) the ones along irrelevant paths that are invoked from the node corresponding to N_{apl}, if any. Note that applying a perceptual chunk to a node N_{apl} may save the costs in the irrelevant paths from N_{apl} and it is a major positive source of speed-up effect. On the other hand, $MacroModeCost$ will be potentially added up at every relevant node as the number of other co-existing perceptual-chunks increases. This is interactions from other knowledge and may be a negative source of speed-up effect.

[5] We approximate problem complexity by the problem-solving time spent in solving the problem without using any learned knowledge.

[6] Doing so requires solving the same problem in advance without any search control knowledge. The issue about the cost spent in learning has been a controversy in uitlity analysis. See related work for discussions.

After calculating speed-up effect for each of all the perceptual-chunks applied to a problem, we sum up the effects of all the perceptual chunks. Ideally the total should be equal to the problem-solving time savings obtained by applying those perceptaul-chunks, i.e. the difference of the problem-solving cpu-time currently measured and the cpu-time spent in solving the same problem without using any perceptual-chunks. But in many cases both will not be equal. Therefore, we assume that the descrepancy of both values is attributed to interactions among the applied perceptual-chunks and add up/subtract the value of discrepancy divided by the number of applied chunks to/from the evaluated speed-up effect of each perceptual-chunk.

4 How to deal with utility data

4.1 Experimental setting

We provided twenty geometry problems selected from junior high school reference books. The categories of problems selected are resticted to "features of triangles" and "parallel lines and angles". Since problems of the same category would share some common perceptual chunks, they can be good examples as an experimental platform to address utility problem. The givens and goals of the problems are shown in Fig. ??.

Out of the 20 problems, we assigned 19 problems to a training session and after repeating the training session twice[7] the remaining one problem is to be solved as a test session. In a training session, 19 problems are arranged in the descending order of problem complexity, and each of them is solved by use of the currently available set of perceptual-chunks. In a test session, the remaining one problem is solved using only those perceptual-chunks which have scored positive utility values at the end of the training session. Since the number of the ways of leaving one out for a test session is twenty, we did the above experiments for all the 20 cases. So we got 19 $\times$ 2 $\times$ 20 problem-solving data for training sessions and 20 data for test sessions.

We provided the following five strategies of dealing with cost-effective utility data, out of which the first three are variations of the best-N strategy mentioned in Section 1 and the last two are variations of judging the "badness" of knowledge. We will do the above experiments for each of these five. Five strategies are

1. To adopt the best 20 pieces of knowledge as available (named as "best-20"),

[7] We repeat this twice because the training session of 19 problems is too short to allow some pieces of really useful knowledge to gain positive estimation. Positive estimation can be obtained only after the knowledge has been applied to problems several times.

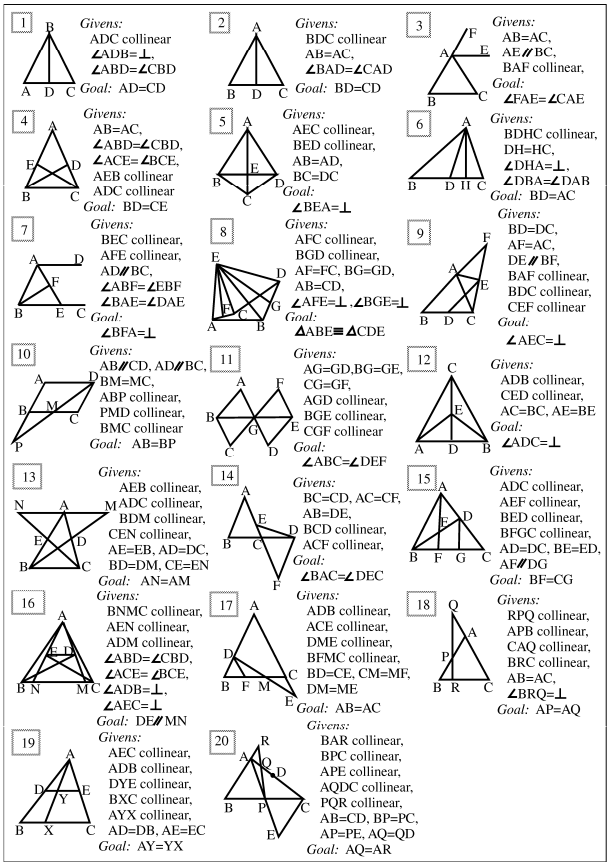

FIG. 0.1. the geometry problems used for experiments

2. To adopt the best 40 pieces of knowledge as available (named as "best-40"),
3. To adopt the best 60 pieces of knowledge as available (named as "best-60"),
4. To discard the knowledge whose population mean of utility values has been assured to be negative by statistical range estimation,
5. To discard not only the knowledge of 4 but also the knowledge which has never been applied but has taken too much cumulative matching costs[8].

[8] In this experiment, the knowledge whose cumulative costs exceed 0.2 in terms of the value normalized by problem complexity is judge to be "bad".

The first three are intended for investigating how large buffer is needed to still retain candidates of high-utility knowledge before gaining a positive estimation. The last two are intended for positively discarding bad knowledge as soon as possible, instead of waiting for their utility values to underscore the lowest in the "best-N" knowledge. The variations in the two are due to the difference about what would be regarded as "bad".

4.2 Statistical range estimation

In general there will be a discrepancy between the average of samples($\overline{x}$) and the true population mean(μ). We employ a technique of statistically estimating the possible range of μ at a certain confidence from the gathered data on $\overline{x}$ and the number of samples (n). The average of samples $\overline{x}$ is known to have a distribution around μ according to the variance of the current samples (s^2) and the degree of freedom ($n-1$). So, if we set

$$t = \frac{\overline{x} - \mu}{s/\sqrt{n-1}}, \tag{0.3}$$

we already have a table of the distribution of t-value. According to this table, we can say at a confidence δ that t-value exists between $-f(\delta)$ and $f(\delta)$, where $f(\delta)$ is a function of δ. Thus, we can estimate the range of μ at a confidence δ as follows;

$$\overline{x} - f(\delta) \cdot \frac{s}{\sqrt{n-1}} \leq \mu \leq \overline{x} + f(\delta) \cdot \frac{s}{\sqrt{n-1}} \tag{0.4}$$

The true utility is calculated from the true population mean of both matching costs and speed-up effects by the following formula;

$$utility = n_{benefit} \cdot \mu_{benefit} - n_{cost} \cdot \mu_{cost} \tag{0.5}$$

For each of the matching costs and speed-up effects, we can statistically estimate their range of existence. In order to assure the utility of a piece of knowledge as negative, we have only to prove the maximum of *utility* is less than zero by calculating it from the maximum of speed-up effects and the minimum of matching costs. We can assure this at a confidence of $\delta_b \times \delta_c$, where δ_b is the confidence in estimating the range of speed-up effects and δ_c is the confidence in estimating the range of matching costs. In the experiment we made estimation at the confidence of 60 %.

4.3 Experimental Results

For each of the five strategies of dealing with utility data, Table **??** shows the following; (1) the average of cpu-time costs over 20 test problems, (2) the average of cpu-time costs over 760 training problems, (3) the number of problems out of 20 test problems, in which positive speed-up effects by use of learned knowledge, negative effects and no applications of knowledge are observed respectively. Cpu-time costs are normalized by problem

complexity. Below we wrap up the observations about the characteristics of the employed strategies.

Table 0.1 problem-solving performance in training and test sessions for each strategy of dealing with utility values

	The test 20 problems			Average costs	
	positive effects	no application	negative effects	test sessions	training sessions
best-20	4	14	2	1.12	1.22
best-40	8	11	1	0.95	1.45
best-60	8	10	2	0.92	2.94
neg. assure*	7	11	2	0.90	2.22
neg.assure + no-application*	12	6	2	0.77	1.28

neg.assure* -- the strategy of discarding the knowledge which has been statistically assured as negative,
no-application* -- the strategy of discarding thse knowledge which has never been applied and has taken too much costs.

1. In using the "best-20" strategy, the number of problems in which no knowledge was applied is the largest and the number of problems in which positive speed-up effects were observed is the smallest. Consequently, the average cost in the test sessions becomes larger than 1.0. The reason is as follows; since the volume of the list of available knowledge during training sessions, i.e. in this case 20, is too small, some pieces of knowledge which should have scored a positive value at the end of a training session have dropped out of the list before gaining positive utility value in the training session. This result suggests that **in order to estimate good knowledge as positive at the end of a training session, we need a certain volume of buffer of knowledge that allows currently negative knowledge to be retained available during the session.**
2. Retaining too much knowledge during training sessions, however, brings about rather undesirable outcome. **As the number of available knowledge retained increases, the average training costs tend to become extremely large undesirably.** Especially, the use of "best-60" strategy costs about three times larger problem-solving cpu-time than using problems without any learned knowledge, while the average cost of test sessions scores less than 1.0 desirably. This is mainly because the solver uses such knowledge that is actually bad but is retained still in the list of currently available knowledge, and also because the situation becomes aggravated by a *snowballing*

effect where using bad knowledge will degrade the problem-solving proof-trees in the training sessions and thus more and more bad knowledge will be learned from the proof-tree.

3. The strategies of **removing bad knowledge as soon as possible** when it is assured to be bad are effective in **allowing really good knowledge to be estimated as positive** at the end of training sessions, with restricting **the training costs consumed to not so expensive degree.** Especially, in using the fifth strategy, the number of problems in which positive speed-up effects are observed is the largest and the number of problems in which no knowledge was applied is the least and consequently the average cost in test sessions is 0.77 while the average cost in training sessions is 1.28.

Table ?? shows the following for each case of using the five strategies; (1) the average of the number of knowledge that is still retained in the list of available knowledge when a training session ends[9], over the 20 cases, and (2) the average of the number of knowledge that has obtained positive utility estimation out of the pieces of knowledge of (1), over the 20 cases.

Table 0.2 the characteristics of the knowledge retained at the end of training session

	all the knowledge*	positively-estimated knowledge **
best-20	20.0	3.5
best-40	40.0	6.2
best-60	60.0	3.8
neg. assure	62.8	5.9
neg.assure + no-application	45.1	9.3

* the average number of all the retained knowledge at the end of a training session , over the 20 cases

** the average number of positively-estimated knowledge at the end of a training session, over the 20 cases

1. The number of the positively estimated knowledge in case of "best-20" strategy is the smallest. This observation also tells that the volume of the retained knowledge is too small.
2. It is surprising that the number of positively estimated knowledge in "best-60" strategy is fewer than that in "best-40" strategy. This is

[9]This includes not only knowledge of pisitive utility but also knowledge of negative utility.

mainly caused by "**interactions with other bad knowledge**; remember that, as we mention the method of measuring cost-effective utility value, $MacroModeCost$ includes the matching costs at the nodes that have been produced by applying the knowledge itself. This means that, when a piece of knowledge has been applied, if bad knowledge requiring too much costs co-exists together, it will potentially degrade the utility value of that applied knowledge. Consequently this will make it difficult for the knowledge to obtain positive estimation finally. In other words, **the property of bad knowledge will ruin the property of the others.**

3. The last two strategies contribute much to preventing knowledge from being affected by interactions with bad knowledge; according to the average number of retained knowledge when a training session ends over the 20 cases, the fourth strategy is compatible with the "best-60" and the fifth one is compatible with the "best-40". In each of both comparisons, the average number of the positively estimated knowledge is larger in the statistical strategy, and also the average costs of both test and training sessions shown in Table ?? are smaller in the statistical strategy.
4. According to the above discussions, the fifth strategy of removing both (1) the knowledge that has been statistically assured as negative and (2) the knowledge that has never been applied but has taken too much matching costs is the most desirable to avoid interactions with bad knowledge and to obtain good performance in test sessions with keeping training costs not so expensive.

4.4 Current limitations and discussions

In the experiments shown in this paper, the best result of the test costs was 0.77 in the fifth strategy. This is worse than we expected first. We mention its reason although this paper does *not* target at implementing a system that exhibits as much learning effects as possible. The main reason is that such problems that do not inherently share perceptual-chunks as partial problem structures with other problems are included in the twenty problems. This is obviously found, as shown in Table ??, from the observation that under all the five strategies some common problems did not allow for any applications of learned knowledge when they are solved in test sessions. Although we noticed it by observing problem-solving performances during all the training sessions, we didn't change the experimental conditions, thinking that we should not artificially design an ideal environment of training sessions. Those problems are in a way unrelated factors that may potentially degrade learning effects. We rather have to note, however, that under these tough conditions some learning effects as well as the differences of the employed strategies are clearly observed.

The technique of range estimation employed for judging the badness of knowledge could be employed for judging the "goodness" of knowledge as well. We can think of an alternative strategy to use only the knowledge that has been *statistically assured* to be positive utility[10] for test sessions. Although the feasibility of this strategy is an open question at present, we currently have a negative view against it. The set of knowledge that has scored a positive value in the current version consists of three types; (1) such knowledge which is applied frequently and exhibits some speed-up effects whenever it is applied, (2) such knowledge which exhibits frequently small negative effects (or is rarely applied) but sometimes big positive speed-up effects, and (3) such knowlede which exhibits sometimes big negative effects but frequently some positive effects (resulting in positive estimation as a total). If we assure the goodness of knowledge as mentioned above, only the knowledge of the first class would be selected as available for test sessions. That may be problematic because it may potentially cause the situation again that no knowledge is applied in many problems of test sessions. We have to examine it in near future.

We repeated each training session twice to make the most of the statistical strategies under a restricted source of geometry problems. Ideally we should have collected more numbers of different problems. Repeating a training session twice, however, might suffice because when the same problem is solved for the second time the utility of the knowledge learned from the problem for the first time can be testified in the different contexts of interactions with co-existing knowledge from the first situation.

The order of problems in a training session may potentially influence the set of learned knowledge. In this paper, we fixed it in the descending order of problem complexity. This may have been one of the factors governing the costs in training sessions and test sessions. We have to examine this factor in future.

5 Related work

Most studies on learning systems have so far evaluated cost-effective utility in a *static* way, i.e. it is meaured simultaneously when knowledge has been learned, because it is the simplest way. They don't have to solve the same problem multiple times. On the other hand, dynamic analysis in this paper requires solving the same problem without using any learned knowledge in advancce. This is obviously more time-consuming than static analysis. But our view is that for the purpose of obtaining dynamic utility value of a piece of knowledge, comparing the current solution trace obtained by applying the knowledge with the one obtained without any knowledge is indispensable. Our simplified technique is still less time-comsuming than the

[10]Notice that this is different from the knowledge that has scored a positive utility value at the end of a training session. In this paper the latter strategy has been taken.

ideal method of solving the same problem with and without the knowledge for each piece of knowledge in question. Due to this simplification, however, our technique would yield rather rough estimation of utility values. But we assume that we can compensate this roughness by doing statistical estimation.

We do not address *interaction problem* in the same sense as Gratch deals with in his paper (Gratch *et al.* 1992). Gratch addresses it in the following way; in general the measured utility values inherently include some interactions from the current available set of control knowledge. He proposes a framework where only the knowledge which gains good interaction from the current available set will be adopted as new control knowledge. This was possible only by gathering static utility value of knowledge simultaneously in learning. In this respect, he makes the most of interactions among knowledge toward the goal of improving the planner's performance in a hill-climbing way. On the other hand, we view interaction problem differently; we understand that interactions between knowledge are one of the fluctuating factors like other factors, e.g. high dependence of utility values on the already existing problem statements, and also that deterioration of training costs due to applying learned knowledge before knowing its utility is ineviable as a cost of learning. In other words, this matches well human way of learning; when actually using knowledge in real problems causes fatal failure, we human learn to avoid using it.

This attitude toward interaction problem is originated from the following notion. We think that we do not necessarily have to improve the system's performance in a hill-climbing way. Rather, if there should be a set of perceptual-chunks that have high-utility as a whole in any contexts including interactions from other knowledge and/or other fluctuating factors, we want to obtain such an organized set of perceptual-chunks in the domain in question. It may be a success if we can finally come up with satisfactory performance by doing so, even if we pay some costs to reach the final state.

6 Conclusion

We address the issue of how to deal with dynamic utility values of knowledge during a training session, the main factor determining the available set of knowledge to be provided for the subsequent problem in each stage within the session. The strategy taken has a great impact not only on problem-solving costs spent in the training session but also whether or not really useful knowledge can gain positive estimation at the end of the training session and therefore on the problem-solving performance in the subsequent test session.

Five strategies were examined in this paper. They are (1) using the best 20 knowledge in terms of cost-effective utility values, out of all the

learned knowledge, (2) using the best 40 knowledge, (3) using the best 60 knowledge, (4) discarding the knowledge as soon as possible whose utility has been statistically assured as negative, and (5) discarding not only the knowledge of (4) but also the knowledge which has never been applied but has taken too much costs.

We have come up with the following insights.

- Some buffer to retain knowledge of currently negative utility values in the list of available knowledge is needed before some of them gets positive estimation in the subsequent training session.
- Too much volume of buffer, however, will undesirably cause *interactions from bad knowledge* and *snowballing effects* during training sessions. If the two effects coincide with each other, the problem-solving costs in training sessions will extremely pile up and, what is worse, the utility value of even useful knowledge will be improperly underestimated due to interactions and discarded from the list of available knowledge. The latter phenomenon badly affects problem-solving performance in test sessions.
- The strategies of discarding "bad" knowledge as soon as possible are effective in allowing really useful knowledge to gain positive estimation at the end of training sessions, with restricting the training costs to a certain degree.

Although this is a trial of fundamental examination, further investigations along this direction will be an important standard for dealing with utility problem in general domains where matchings in applying knowledge impose a heavy burden on problem-solving performance.

Bibliography

1. Gratch, J. and DeJong, G. (1991a). A Hybrid Approach to Guaranteed Effective Control Strategies, *Proceedings of ML-91.*
2. Gratch, J. and DeJong, G. (1991b). On comparing operationality and utility, *Tech. Report, UIUCDCS-R-91-1713, University of Illinois at Urbana-Champaign.*
3. Gratch, J. and DeJong, G. (1992). COMPOSER: A Probabilistic Solution to the Utility Problem in Speed-up Learning, *Proceedings of AAAI-92*, 235–240.
4. Greeno, J. G. (1983). Forms of understanding in mathematical problem-solving, *Paris, S.G et al. eds., Learning and Motivation in the Classroom.* Lawrencce Erlbaum Associates.
5. Koedinger, K. R. and Anderson, J. R. (1990). Abstract planning and perceptual chunks: elements of expertise in geometry, *Cognitive Science 14*, 511–550.

6. McDougal, T. and Hammond, K. (1992). A recognition model of geometry theorem-proving, *Proc. of the 14th Annual Conference of the Cognitive Science Society*, 106–111.

7. McDougal, T. and Hammond, K. (1993). Representing and using procedural knowledge to build geometry proofs, *Proc. of AAAI-93*, 60–65.

8. Minton, S. (1985). Selectively generalizing plans for problem solving, *Proceedings IJCAI-85*, 596–602.

9. Minton, S., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O. and Gil, Y., (1989). Explanation-based learning: a problem solving perspective, *Artificial Intelligence 40*, 63–118.

10. Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning, *Artificial Intelligence 42*, 363–391.

11. Subramanian, D. and Feldman, R. (1990). The utility of EBL in recursive domain theories, *Proceedings of AAAI-90*, 942–949.

12. Suwa, M. and Motoda, H. (1993). A perceptual criterion for visually controlling learning, *Proceedings of 4th international workshop on algorithmic learning theory, Lecture Notes in AI 744*, 356–369, Springer-Verlag.

13. Suwa, M. and Motoda, H. (1994a). Learning perceptually-chunked macro-operators, *Machine Intelligence 13*, Oxford University Press (in press).

14. Suwa, M. and Motoda, H. (1994b). PCLEARN: A model for learning perceptual-chunks, to appear in *Proceedings of the 16th Annual Meetings of the Cognitive Science Society*, Atlanta, Georgia.

15. Sweller, J. (1988). Cognitive load during problem-solving: effects on learning, *Cognitive Science 12*, 257–285.