Problem Order Implications for Learning Transfer

Nan Li, William W. Cohen, and Kenneth R. Koedinger

School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh PA 15213 USA
{nli1,wcohen,koedinger}@cs.cmu.edu

Abstract. The order of problems presented to students is an important variable that affects learning effectiveness. Previous studies have shown that solving problems in a blocked order, in which all problems of one type are completed before the student is switched to the next problem type, results in less effective performance than does solving the problems in an interleaved order. While results are starting to accumulate, we have little by way of precise understanding of the cause of such effect. Using a machine-learning agent that learns cognitive skills from examples and problem solving experience, SimStudent, we conducted a controlled simulation study in three math and science domains (i.e., fraction addition, equation solving and stoichiometry) to compare two problem orders: the blocked problem order, and the interleaved problem order. The results show that the interleaved problem order yields as or more effective learning in all three domains, as the interleaved problem order provides more or better opportunities for error detection and correction to the learning agent. The study shows that learning when to apply a skill benefits more from interleaved problem orders, and suggests that learning how to apply a skill benefits more from blocked problem orders.

Keywords: learning transfer, learner modeling, interleaved problem order, blocked problem order

1 Introduction

One of the most important variables that affects learning effectiveness is the order of problems presented to students. While most existing textbooks organize problems in a blocked order, in which all problems of one type (e.g. learning to solve equations of the form $S_1/V{=}S_2$) are completed before the student is switched to the next problem type, it is surprising that problems in an interleaved order often yields more effective learning. Numerous studies have experimentally demonstrated this effect (e.g., [18, 6, 2, 9, 23, 4, 17, 7]). However, the cause of the the effect is still unclear. A computational model that demonstrates such behavior would be a great help in better understanding this widely-observed phenomena, and might reveal insights that can improve current education technologies.

- Skill divide (e.g. -3x = 6)
- Perceptual information:
  - Left side (-3x)
  - Right side (6)
- Precondition:
  - Left side (-3x) does not have constant term
- Operator sequence:
  - Get coefficient (-3) of left side (-3x)
  - Divide both sides with the coefficient (-3)

Fig. 1. A production rule for divide.

In this paper, we conducted a controlled-simulation study using a machine-learning agent, SimStudent. SimStudent was trained on real-student problems that were of blocked orders or interleaved orders. We then tested whether the advantages of interleaved problem orders over blocked problem orders are exhibited in all three domains. After that, we carefully inspected what causes such effect by inspecting SimStudent's learning processes and learning outcomes, which are not easily obtainable from human subjects.

2 A Brief Review of SimStudent

SimStudent is a machine-learning agent that inductively learns skills to solve problems from demonstrated solutions and from problem solving experience. It is an extension of programming by demonstration [8] using inductive logic programming [13] as an underlying learning technique. In the rest of this section, we will briefly review the learning mechanism of SimStudent. For full details, please refer to [10].

SimStudent learns production rules as skills to solve problems. During the learning process, given the current state of the problem (e.g., $-3x = 6$), SimStudent first tries to find an appropriate production rule that proposes a plan for the next step (e.g., *(coefficient -3x ?coef) (divide ?coef)*). If it finds a plan and receives positive feedback, it continues to the next step. If the proposed next step is incorrect, negative feedback and a correct next step demonstration are provided to SimStudent. The learning agent will attempt to learn or modify its production rules accordingly. If it has not learned enough skill knowledge and fails to find a plan, a correct next step is directly demonstrated to SimStudent for later learning.

Figure 1 shows an example of a production rule learned by SimStudent in a readable format[1]. A production rule indicates "where" to look for information in the interface, "how" to change the problem state, and "when" to apply a rule. For example, the rule to "divide both sides of $-3x=6$ by -3" shown in Figure 1 would

[1] The actual production rule uses a LISP format.

be read as "given a left-hand side (*-3x*) and a right-hand side (6) of the equation, when the left-hand side does not have a constant term, then get the coefficient of the term on the left-hand side and divide both sides by the coefficient."

As there are three main parts in a production rule, SimStudent's learning mechanism also consists of three parts: a "where" learner, a "when" learner, and a "how" learner. The "where" learner acquires knowledge about where to find useful information in the GUI. For example, for the step *divide -3*, *-3x* and *6* are the useful information, the GUI elements associated with them are *Cell 21* and *Cell 22*. The learning task is to find paths that identify such elements. All of the elements in the interface are organized in a tree structure. For instance, if the GUI has a table in it, the table node has columns as children, and each column has multiple cells as children. For each cell, SimStudent uses a *deep feature* learning mechanism that acquires knowledge on how to further parse the content in each cell into a cell parse tree. When given a set of positive examples (i.e., GUI elements associated with useful information in the steps), the learner carries out a specific-to-general learning process (e.g., from *Cell 21* to *Cell ?1* to *Cell ??*). It finds the most specific paths that cover all of the positive examples.

The "when" learner acquires the precondition of the production rule that describes the desired situation to apply the rule (e.g. *(not (has-constant ?var1))*) given a set of *feature predicates*. Each predicate is a boolean function of the arguments that describes relations among objects in the domain. For example, *(has-coefficient -3x)* means *-3x* has a coefficient. The "when" learner utilizes FOIL [15] to acquire the precondition as a set of feature tests. FOIL is an inductive logic programming system that learns Horn clauses from both positive and negative examples expressed as relations. If a step is either demonstrated to SimStudent or receives positive feedback, that step is a positive example for FOIL; otherwise, a negative example.

The last component is the "how" learner which acquires knowledge about how to change the problem state. Given all of the positive examples and a set of *basic operator functions* (e.g., *(divide ?var)*), the "how" learner attempts to find a shortest operator function sequence that explains all of the training examples using iterative-deepening depth-first search.

3 Problem Order Study

To get a better understanding of how and why problem orders affect learning efficiency, we carried out a controlled simulation study on SimStudent given different problem orders.

3.1 Methods

To ensure the generality of the results, we selected three math and science domains: fraction addition, equation solving, and stoichiometry. Both the training and testing problems were selected from problems solved by human students in classroom studies. SimStudent was tutored by interacting with automatic tutors that simulate the automatic tutors used by human students.

Fraction Addition: In the fraction addition domain, SimStudent was given a series of fraction addition problems of the form

$$\frac{numerator_1}{denominator_1} + \frac{numerator_2}{denominator_2}$$

All numerators and denominators are positive integers. The problems are of three types in the order of increasing difficulty: 1) *easy problems*, where the two addends share the same denominators (i.e., $denominator_1 = denominator_2$, e.g., $1/4 + 3/4$), 2) *normal problems*, where one denominator is a multiple of the other denominator (i.e., $GCD(denominator_1,\ denominator_2) = denominator_1$ *or* $denominator_2$, e.g., $1/2 + 3/4$), 3) *hard problems*, where no denominator is a multiple of the other denominator (e.g., $1/3 + 3/4$). In this case, students need to find the common denominator (e.g. 12 for $1/3 + 3/4$) by themselves. Both the training and testing problems were selected from a classroom study of 80 human students using an automatic fraction addition tutor. The number of training problems is 20, and the number of testing problems is 6.

Equation Solving: The second domain in which we tested SimStudent is equation solving. Equation solving is a more challenging domain since it requires more complicated prior knowledge to solve the problem. For example, it is hard for human students to learn what is a coefficient, and what is a constant. Also, adding two terms together is more complicated than adding two numbers.

In this experiment, we evaluated SimStudent based on a dataset of 71 human students in a classroom study using an automatic tutor, CTAT [1]. The problems are also in three types: 1) problems of the form $S_1 + S_2V = S_3$, 2), $V/S_1 = S_2$, 3) $S_1/V = S_2$, where S_1 and S_2 are signed numbers, and V is a variable. Note that the terms in the above problem forms can appear in any order, and surrounded with parenthesis. There were 12 training problems, and 11 testing problems in the experiment.

Stoichiometry: Lastly, we evaluated SimStudent in a chemistry domain, stoichiometry. Stoichiometry is a branch of chemistry that deals with the relative quantities of reactants and products in chemical reactions. We selected stoichiometry because it is different from equation solving and fraction addition in nature. In the stoichiometry domain, SimStudent was asked to solve problems such as "How many moles of atomic oxygen (O) are in 250 grams of P_4O_{10}? (Hint: the molecular weight of P_4O_{10} is 283.88 g P_4O_{10} / mol P_4O_{10}.)". 8 training problems and 3 testing problems were selected from a classroom study of 81 human students using an automatic stoichiometry tutor [11].

To solve the problems, SimStudent needs to acquire three types of skills: 1) unit conversion (e.g. 0.6 kg H_2O = 600 g H_2O), 2) molecular weight (e.g. There are 2 moles of P_4O_{10} in 283.88 $\times$ 2 g P_4O_{10}) , 3) composition stoichiometry (e.g. There are 10 moles of O in each mole of P_4O_{10}). The problems are of three types ordered in increasing difficulty, where each later type adds one more skill comparing with its former type.

Measurement: To measure learning gain, the production rules learned by SimStudent were tested on the testing problems each time tutoring was done on a single training step. For each step in the testing problems, we measure a *step score* for it. In math and science problems, there is often more than one way to solve one problem. Hence, at each step, there is usually more than one production rule that is applicable. In this case, among all possible correct next steps, we count the number of correct steps that are actually proposed by some applicable production rule, and report the step score as the number of correct next steps covered by learned rules divided by the total number of correct next steps plus the number of incorrect next steps proposed by SimStudent, i.e.,

$$\frac{\#OfCorrectNextStepsProposed}{Total\#OfCorrectNextSteps + \#OfIncorrectNextStepsProposed}$$

For example, if there are four possible correct next steps, and SimStudent proposes three, of which two are correct, and one is incorrect, then only two correct next steps are covered, and thus the step score is $2/(4 + 1) = 0.4$. We report the average step score over all testing problem steps for each curriculum.

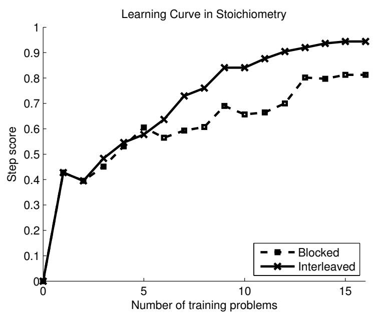
3.2 Blocked vs. Interleaved Problem Orders

To manipulate the order of problems given to SimStudent, for each domain, we first grouped the problems of the same type together. Since there were three types of problems, we had three groups in each domain: *group -1*, *group - 2*, and *group - 3*. Then, there were six different orders of these three groups. For each order (e.g. *[group - 1, group - 2, group - 3]*), we generated one blocked-ordering curriculum by repeating the same problems in each group right after that group's training was done (e.g., *[group - 1, group - 1', group - 2, group - 2', group - 3, group - 3']*). To generate the interleaved-ordering curriculum, the same problems will be repeated once the whole set of problems were done (e.g, *[group - 1, group - 2, group - 3, group - 1', group - 2', group - 3']*).

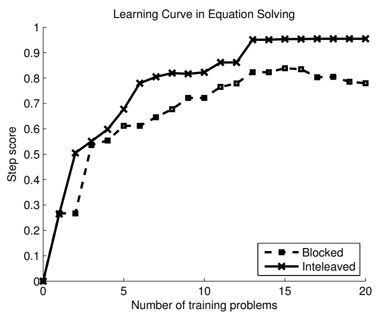
After this manipulation, we ended up having 12 curricula of different orders for each domain as shown in Table 1. Six of them were blocked-ordering curricula, whereas the other six were interleaved-ordering curricula. SimStudent was trained and tested on all these curricula, the results are the average step scores over curricula of the same type (blocked or interleaved).

Blocked-Ordering Curricula	Interleaved-Ordering Curricula
1, 1', 2, 2', 3, 3'	1, 2, 3, 1', 2', 3'
1, 1', 3, 3', 2, 2'	1, 3, 2, 1', 3, 2'
2, 2', 1, 1', 3, 3'	2, 1, 3, 2', 1', 3'
2, 2', 3, 3', 1, 1'	2, 3, 1, 2', 3', 1'
3, 3', 1, 1', 2, 2'	3, 1, 2, 3', 1', 2'
3, 3', 2, 2', 1, 1'	3, 2, 1, 3', 2', 1'

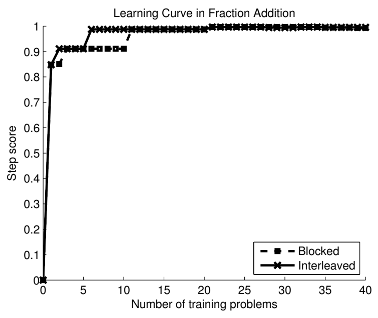
Table 1. 12 curricula of different orders for each domain.



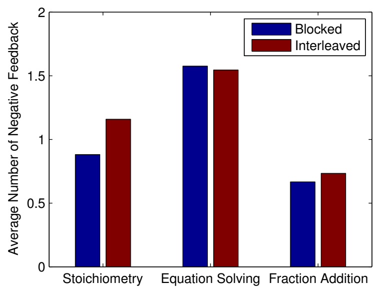
(a)



(b)



(c)



(d)

Fig. 2. Learning curves of blocked-ordering curricula vs. interleaved-ordering curricula in three domains, a) stoichiometry, b) equation solving, c) fraction addition, and the average number of times SimStudent receives negative feedback for each skill across three domains.

3.3 Results

Figure 2 shows the learning curves of SimStudent trained on blocked-ordering or interleaved-ordering curricula. As we can see in the graph, in all three domains, the interleaved-ordering curricula yielded as or more effective learning than the blocked-ordering curricula.

In the domain of stoichiometry, the step score of the interleaved-ordering curricula was 0.944, whereas the step score of the blocked-ordering curricula was 0.813. A sign test between pairs of step scores achieved by the associated interleaved-ordering and blocked-ordering curricula (e.g., *[group - 1, group - 2, group - 3, group - 1', group - 2', group - 3']* vs. *[group - 1, group - 1', group - 2, group - 2', group - 3, group - 3']*) showed that, after trained on 40 problems, the interleaved-ordering curricula is significantly ($p < 0.05$) more effective than the blocked-ordering curricula.

Similar results were also observed in the equation solving domain. The interleaved-ordering curricula again showed a benefit (0.955 vs. 0.858) over blocked-ordering curricula. The sign test also demonstrated significant ($p < 0.05$) advantages of interleaved-ordering curricula over the blocked-ordering curricula.

In fraction addition, SimStudent got an average step score of 0.995 when trained with interleaved-ordering curricula, which is slightly higher than the step score SimStudent received (0.993) when trained with blocked-ordering curricula. There was no significant difference between the two conditions.

3.4 Implications for Instructional Design

We can inspect the data more closely to get a better qualitative understanding of why the SimStudent model is better and what implications there might be for improved instruction. In two of three domains, interleaved-ordering curricula are more advantageous than blocked-ordering curricula. These results provide theoretical support for the hypothesis that when teaching human students in math and science domains, an interleaved problem order yields better learning than a blocked problem order.

To better understand the cause of the advantages of interleaved-ordering curricula, we further measured the amount of negative feedback received by SimStudent, as it is one of the important factors in achieving effective learning. The amount of negative feedback is assessed by the average number of times SimStudent received negative feedback for each skill. As presented in Figure 2(d), the SimStudent given interleaved-ordering problems receives significantly ($p < 0.05$, 31.5%) more negative feedback than the SimStudent trained on blocked-ordering problems in stoichiometry, and 10.0% more negative feedback in fraction addition.

One possible explanation for this is when problems are of an interleaved order, SimStudent may incorrectly apply the production rules learned from previous problem types to the current problem, even if the current problem is of another type. In this case, SimStudent receives explicit negative feedback from the tutor. In contrast, when trained on blocked-ordering curricula, SimStudent has fewer opportunities for incorrect rule applications, and thus receives less negative feedback. Since the negative feedback serves as negative training examples of the "when" learning, more negative feedback in the interleaved problem order case enables SimStudent to yield more effective "when" learning compared to blocked problem orders. Although SimStudent received approximately the same amount of negative feedback ($p = 1$, -1.9%) in the blocked problem order case and interleaved problem order case, a careful inspection shows that negative examples from other problem types are sometimes more informative than those from the same problem type. For example, in algebra, during the acquisition of the skill "subtract", the SimStudent given blocked-ordering problems learned that when there is a constant term in either side of the equation (e.g., term S_2 is a number in $S_1V{+}S_2{=}S_3$), subtract both sides with that number (e.g., *(subtract S_2)*). But it failed to learn that there must be a plus sign before S_2. In the interleaved condition, SimStudent received negative feedback when it tried to subtract both sides with S_2 when given problems of type $S_1/V{=}S_2$. Then, the SimStudent given interleaved-ordering problems modified its when-part. The updated production rule became, "when there is a constant term that follows a plus sign in either side of the equation, subtract both sides with that number."

We conjecture that the frequent use of blocked examples in textbooks might relate to perceived memory limitations of students. SimStudent currently does not have any severe memory (or retrieval) limitations (e.g., it remembers all past examples no matter how long ago). SimStudent would need to have some memory limitations if it were to have a bigger knowledge base or to better model humans. If it did, the benefits for blocking may go up, and in particular for "how" learning. Let's consider a fixed memory size for SimStudent, which means SimStudent is only able to remember a fixed number of most recent training examples. SimStudent receives training examples of "how" learning only when the current step is demonstrated or SimStudent applies a production rule correctly. Hence, in the blocked problem order case, SimStudent maintains all the training examples of the current problem type unless the number of training examples exceeds the memory limit. In contrast, when trained on interleaved-ordering curricula, SimStudent needs to remember training examples for multiple problem types. For any specific production rule, the number of training examples will be smaller than that given a blocked-ordering curricula, which could result in less effective learning than the blocked-ordering case.

This also relates to VanLehn's work on "learning one subprocedure per lesson" [20]. If a subprocedure is achieved in the same way, that is, with the same how-part in the production rule, then as Vanlehn suggested, problems of blocked orders are more beneficial. However, for production rules/procedures to differentiate across subgoals, the when-part needs to be acquired and in that case, interleaving problems of different types is important.

In summary, the study shows that learning when to apply a skill benefits more from interleaved problem orders, and suggests that learning how to apply a skill benefits more from blocked problem orders. Therefore, when tutoring students in domains that are more challenging in "how" learning, we suggest that the problems presented to students should be of blocked orders. If the learning task requires more rigorous "when" learning, interleaved-ordering problems should be preferred.

4 Related Work

The main objective of this work is to better understand how and why problem orders affect learning outcome using a learning agent. A considerable amount of research has demonstrated the effectiveness of interleaved problem orders. Shea and Morgan [18] were the first that showed problems of a random order yields better performance in retention and transfer tests than students trained on problems of a blocked order, and named this effect as the *contextual interference (CI) effect*. The CI effect compares random problem orders and blocked problem orders, not interleaved problem orders and blocked orders, but the results should be similar since the main point is whether consecutive problems should be of the same or different types. That is, random problem orders have lots of interleaving. After that, a growing number of studies (e.g., [6, 2, 9, 23, 4, 17, 7]) have repeatedly observed the CI effect in different tasks. Other studies on relatively complex tasks (e.g., [19]) or novices (e.g., [5]) have yielded mixed results. To explain the CI phenomenon, researchers have proposed several hypothesis including the elaboration hypothesis [18], the forgetting or reconstruction hypothesis [9], etc. More details on these hypotheses are available in [22], however, all are described in fairly ambiguous language and none have the precision of a computational theory. In contrast, SimStudent provides a precise, unambiguous implementation of how and why interleaving may be effective.

Research on task switching [12] shares a resemblance with our work. It shows that subjects' responses are substantially slower and more error-prone immediately after a task switch. Our work differs from this research in that we focus on learning tasks. During the learning process, switching among problems of different types also increases the cognitive load, but causes more effective learning.

Other research on creating simulated students [21, 3, 14] and simulating expert memory [16] also share some resemblance to our work. VanLehn [21] created a learning system and evaluated whether it was able to learn procedural "bugs" like real students. To the best of our knowledge, none of the above approaches made use of the models to simulate the advantage of interleaved or random problem orders over blocked problem orders.

5 Concluding Remarks

In spite of the promising results, there remain several fruitful future steps. First, the current study used only one set of problems in each domain. To evaluate the generality of the claim, we should carry out the same set of experiments using other problem sets or in other domains. Second, we would like to carry out more studies in which SimStudent has limited memory, and validate whether "how" learning gains more from blocked problem orders in this case. Last, future research could apply the theoretical implications in a study on human students, and evaluate the validity of the recommended tutoring strategy.

In this paper, we carried out a controlled simulation study to gain a better understanding of why interleaved problem orders generate more effective learning than blocked problem orders. We measured the learning effectiveness of a machine-learning agent, SimStudent, in three domains given different problem orders. The results show that since the interleaved problem order yields more opportunities for error detection and correction, the SimStudent trained by interleaved-ordering curricula achieved better performance than the SimStudent trained by blocked-ordering curricula.

References

1. Aleven, V., Mclaren, B.M., Sewall, J., Koedinger, K.R.: A new paradigm for intelligent tutoring systems: Example-tracing tutors. International Journal of Artificial Intelligence in Education 19, 105–154 (April 2009)
2. Carnahan, H., Van Eerd, D.L., Allard, F.: A note on the relationship between task requirements and the contextual interference effect. Journal of Motor Behavior 22(1), 159–169 (1990)

3. Chan, T.W., Chou, C.Y.: Exploring the design of computer supports for reciprocal tutoring. International Journal of Artificial Intelligence in Education 8, 1–29 (1997)
4. Del Rey, P.: Effects of contextual interference on the memory of older females differing in levels of physical activity. Perceptual and motor skills 55(1), 171–180 (1982)
5. French, K.E., Rink, J.E., Werner, P.F.: Effects of contextual interference on retention of three volleyball skills. Peceptual adn Motor Skills 71, 179–186 (1990)
6. Gabriele, T.E., Hall, C.R., Buckolz, E.E.: Practice schedule effects on the acquisition and retention of a motor skill. Human Movement Science 6, 1–16 (1987)
7. Jelsma, O., Pieters, J.M.: Practice schedule and cognitive style interaction in learning a maze task. Applied Cognitive Psychology 3(1), 73–83 (1989)
8. Lau, T., Weld, D.S.: Programming by demonstration: An inductive learning formulation. In: Proceedings of the 1999 international conference on intelligence user interfaces. pp. 145–152 (1998)
9. Lee, T.D., Magill, R.A.: The locus of contextual interference in motor-skill acquisition. Journal Of Experimental Psychology. Learning Memory And Cognition 9(4), 730–746 (1983)
10. Li, N., Cohen, W.W., Koedinger, K.R.: Integrating representation learning and skill learning in a human-like intelligent agent. Tech. Rep. CMU-MLD-12-1001, Carnegie Mellon University (January 2012)
11. Mclaren, B.M., Lim, S.j., Koedinger, K.R.: When and how often should worked examples be given to students? new results and a summary of the current state of research why isn't the science done? Cognitive Science pp. 2176–2181 (2008)
12. Monsell, S.: Task switching. Trends in Cognitive Sciences 7(3), 134–140 (2003)
13. Muggleton, S., de Raedt, L.: Inductive logic programming: Theory and methods. Journal of Logic Programming 19, 629–679 (1994)
14. Pentti Hietala, T.N.: The competence of learning companion agents. International Journal of Artificial Intelligence in Education 9, 178–192 (1998)
15. Quinlan, J.R.: Learning logical definitions from relations. Mach. Learn. 5(3), 239–266 (1990)
16. Richman, H.B., Staszewski, J.J., Simon, H.A.: Simulation of expert memory using epam iv. Psychological Review 102(2), 305–330 (1995)
17. Sekiya, H., Magill, R.A., Anderson, D.I.: The contextual interference effect in parameter modifications of the same generalized motor program. Research quarterly for exercise and sport 67(1), 59–68 (1996)
18. Shea, J.B., Morgan, R.L.: Contextual interference effects on the acquisition, retention, and transfer of a motor skill. Journal of Experimental Psychology Human Learning Memory 5(2), 179–187 (1979)
19. Tsutsui, S., Lee, T.D., Hodges, N.J.: Contextual interference in learning new patterns of bimanual coordination. Journal of Motor Behavior 30(2), 151–157 (1998)
20. Vanlehn, K.: Learning one subprocedure per lesson. Artificial Intelligence 31, 1–40 (January 1987)
21. VanLehn, K.: Mind Bugs: The Origins of Procedural Misconceptions. MIT Press, Cambridge, MA, USA (1990)
22. Wulf, G., Shea, C.H.: Principles derived from the study of simple skills do not generalize to complex skill learning. Psychonomic bulletin review 9(2), 185–211 (2002)
23. Young, D.E., Cohen, M.J., Husak, W.S.: Contextual interference and motor skill acquisition: On the processes that influence retention. Human Movement Science 12(5), 577–600 (1993)