When does Changing Representation Improve Problem-Solving Performance ?

Robert Holte
Computer Science Dept.
University of Ottawa
Ottawa, Ontario
Canada K1N 6N5
holte@csi.uottawa.ca

Robert Zimmer and Alan MacDonald
Dept. of Electrical Engineering
Brunel University
Uxbridge, Middx.
England UB8 3PH

Abstract

The aim of changing representation is the improvement of problem-solving efficiency. For the most widely studied family of methods of change of representation it is shown that the value of a single parameter, called the expansion factor, is critical in determining (1) whether the change of representation will improve or degrade problem-solving efficiency and (2) whether the solutions produced using the change of representation will or will not be exponentially longer than the shortest solution. A method of computing the expansion factor for a given change of representation is sketched in general and described in detail for homomorphic changes of representation. The results are illustrated with homomorphic decompositions of the Towers of Hanoi problem.

Definitions

The following definitions of the basic elements of problem solving are used throughout the paper. Only the definition of "solution" is non-standard.

A **state** is an atomic object.
An **action**, or **operator**, is a function mapping states to states.
A **plan**, or **path**, is a sequence of actions.
A **goal** is a set of states.
A **problem** is a pair consisting of an initial state and a goal.
A **problem space** is a pair consisting of a set of states and a set of actions.[1]

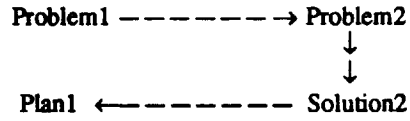
A **solution** (of Problem <S0,G>) is a sequence
S0-A1-S1-A2-...-S(n-1)-An-Sn
where Si is a state, Ai is an action mapping S(i-1) to Si, and Sn is in the goal G.

A **solution plan** (of Problem <S0,G>) is a sequence of actions mapping S0 to a state in G.

[1] This definition assumes all sets of states are allowable as goals, that all State-Goal pairs are allowable as Problems, that all sequences of composable actions are allowable as Plans, etc.. This assumption is not essential to any of the results that follow.

Change of Representation

Problem space Pspace2 is a change of representation of problem space Pspace1 if there exists a relation, R, between the two problem spaces that "preserves" solutions in the following sense. If R maps a problem, Problem1, in Pspace1 to Problem2 in Pspace2, and Solution2 is a solution of Problem2, and R maps Solution2 to Plan1 (a plan in Pspace1), then Plan1 is a solution plan of Problem1. This definition is depicted in the following diagram.

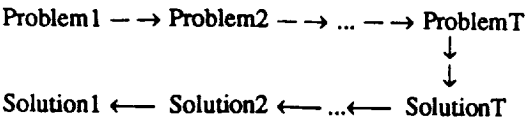


This is an extremely general definition, presupposing nothing about the nature of the indivudal mappings, nor anything about how problem spaces or the mappings are implemented.

The net effect of change of representation is to "decompose" problem-solving in Pspace1 into a three step computation:
(1) translating a given problem into a problem in Pspace2
(2) problem-solving in Pspace2, and
(3) translating the solution back into Pspace1.

Change of representation may be applied recursively to any of these three steps. Most commonly (for example in "hierarchical" problem-solving (Knoblock,1991)), it is applied repeatedly to step (2) until this step becomes

trivial. Diagrammatically, this may depicted as:



The rightmost problem space in this diagram can always be assumed to be the trivial space consisting of one state and one operator (the identity). In this way the explicit problem-solving step is entirely eliminated: a problem is solved by being translated into the trivial problem space and then translating the solution back into the original problem space. The total cost of problem-solving, then, is the sum of the costs of the two translations.

Solution Refinement

Within the preceding general strategy for problem-solving by change of representation there are many possible variations. One of these variations, called "solution refinement" is the subject of this paper. Solution refinement is defined by the two following properties. First, the only complex computation is the translation of a solution in Pspace(K) to a solution in Pspace(K-1). This computation is called refinement. Second, refinement preserves the structure of the solutions, in the following sense. Suppose the solution in Pspace(K) is

S0-A1-S1-A2-...-An-Sn

A refinement of this solution must have the form

X0-I0-RS0-RA1-X1-I1-RS1-RA2-...-Xn-In-RSn

where

- RSi is a state in Pspace(K-1) corresponding to state Si,
- RAi is an action in Pspace(K-1) corresponding to action Ai and defined on RSi,
- Xi is the result of applying action RAi to state RS(i-1) (X0 is the start state in the problem to be solved in Pspace(K-1)),
- and Xi-Ii-RSi is a solution to the problem of getting from Xi to RSi (if it happens that Xi=RAi, then Ii is empty).

Every action in a solution has a counterpart in the refinement of the solution, and usually there will be new actions added (the non-empty Ii). Therefore, a refinement will usually be longer, and can never be shorter, than the solution it is based on. In other words, as the initial "trivial" solution is translated back to become a solution in Pspace1 it grows longer and longer — it expands each time it is refined. The "expansion factor" (pp.10-11, (Stefik & Conway,1982)) is defined as the average ratio of the length of a refinement to the length of the solution from which it was derived. An equivalent definition, which will be useful later, is that the expansion factor is the average number of states in the segments Xi-Ii-RSi.

Solution refinement, in various forms, is the oldest and most widely studied method of change of representation. It is most often associated with the use of "abstractions", as in ABSTRIPS (Sacerdoti,1974), NOAH (Sacerdoti,1977), ALPINE (Knoblock,1991), and ABTWEAK (Yang & Tenenberg,1990). But solution refinement, as a strategy for problem-solving, is equally applicable to many ways of decomposing a problem space, not only to abstraction. For example, in our research (Zimmer et al.,1991), Pspace2 may be any refinable homomorphic image of Pspace1: that is, the mapping between Pspace1 and Pspace2 may be any many-to-one mapping of states to states and operators to operators[2] such that: (1) the behaviour of operators is preserved, and (2) there exists a refinement of every solution in Pspace2. Examples of solution refinement and homomorphic decompositions are given in the next section.

The Towers of Hanoi Problem

Although there are several different ways to define the Towers of Hanoi problem space, in this paper we will follow the standard definition. A state is defined by naming the peg on which each of the disks sits. There are 3 pegs, and any disk may be on any peg, so if there are D disks there are 3^D states. An operator is defined by naming a disk and a direction (clockwise or anticlockwise): thus there are 2D operators, given D disks. The effect of an operator is to move the specified disk from its current peg to the next peg in the specified direction. An operator is defined on a state only if all the disks smaller than the disk to be moved are on the peg that is

[2] We are currently exploring the use of many-to-many mappings of operators, called "distributed representations" in (Holte,1988).

not affected by the operator.

In the 2-disk Towers of Hanoi problem there are 9 states, { <S,L> | S, L ∈ {1,2,3}}, where S indicates the peg of the smaller disk and L the peg of the larger disk. The 4 operators are SC, SA, LC, LA, where S (L) indicates the smaller (larger) disk, and C (A) indicates clockwise (anticlockwise).

In the 1-disk Towers of Hanoi problem, there are 3 states {<1>,<2>,<3>}, indicating the position of the lone disk, and two operators, C and its inverse A.

The Standard Decomposition

There are exactly four homomorphisms of the 2-disk space, as defined above, onto the 1-disk space. The standard one can be summarized in English as: "ignore the position of the smallest disk". Formally, in this decomposition state <S,L> is mapped to state <L>, operators LC and LA are mapped to C and A, respectively, and operators SC and SA are mapped to the identity.

To illustrate the use of this decomposition in problem-solving, consider the problem in which the start state is <1,1> and the goal is <3,3>. This 2-disk problem is mapped into the 1-disk space. The translated problem has <1> as a start state and <3> as a goal. The 1-disk solution is <1>-A-<3>. This solution implicitly has identity operators acting on states <1> and <3>. Refinement must now map this solution into a solution to the original problem. Operator A maps back uniquely to the operator LA, but the states do not map back uniquely, nor do the implicit identity operators. For example, the identity operators on state <1> map back to any sequence of operators which, when applied to a state in which the larger disk is on peg 1, lead to a state in which the larger disk is on peg 1.

Our refinement algorithm works by translating a solution in Pspace(K), SolutionK, into a sequence of subproblems to be solved in Pspace(K-1). Each State-Operator fragment in SolutionK is translated into a goal to be solved starting at the final state of the previously solved subproblem (or, in the case of the first goal of this form, starting at the given start state of Pspace(K-1)). In the present example, SolutionK is <1>-A-<3>: this is translated into the goal "<1>-A", whose meaning is "reach a state in which operator LA is applicable and the larger disk is on peg 1". Problem-solving commences from the start state in Pspace(K-1), <1,1>, and proceeds, as usual, until this goal is satisfied. In this case the solution is <1,1>-SC-<2,1>. This solution is the refinement of the <1>- segment of SolutionK. Note that the one state in SolutionK has been expanded into 2 states in this refinement: this expansion factor is the key in determining whether this decomposition will improve or degrade the efficiency of problem-solving.

Continuing with the example, operator LA is added to the solution, along with the state (<2,3>) to which it leads from <2,1>. The state <2,3> will be the start state for the next subproblem in the refinement process. Because we have finished with all the operators in SolutionK, only the final refinement subproblem remains: the goal is to reach <3,3>, the goal state in Pspace(K-1). Problem-solving commences from the state <2,3> and finds the solution <2,3>-SC-<3,3>. This is the expansion of the -<3> segment of SolutionK: as before, there is an expansion factor of 2. The final solution to the original problem is created by linking together all the solutions to the refinement subproblems, giving <1,1>-SC-<2,1>-LA-<2,3>-SC-<3,3>.

Non-Standard Decompositions

In the first non-standard decomposition, state <S,L> is mapped to state <P> if S and L are both equal to P or if both are different than P. Thus, states <1,1>, <2,3>, and <3,2> map to state <1>, states <2,2>, <1,3>, <3,1> map to state <2>, and states <3,3>, <1,2>, <2,1> map to state <3>. Operators SA and LC both map to operator A, and SC and LA both map to C.

In the second non-standard decomposition, state <S,L> is mapped to state <S>: that is, the position of the larger disk is ignored. Thus, states <1,1>, <1,2>, and <1,3> map to state <1>, states <2,1>, <2,2>, <2,3> map to state <2>, and states <3,1>, <3,2>, <3,3> map to state <3>. Operators LA and LC map to A and C, respectively, and SA and SC both map to the identity.

In the final decomposition, state <S,L> is mapped to state <S-L+1>, where the subtraction is done modulo 3. In other words, the mapping is based on the relative positions of the two disks. States in which the two disks are on the same peg — <1,1>, <2,2>, and <3,3> — are mapped to state <1>. States in which the smaller

disk is one peg "ahead" of the larger disk — <2,1>, <3,2>, and <1,3> — are mapped to state <2>. And states in which the smaller disk is one peg "behind" the larger disk — <1,2>, <2,3>, and <3,1> — are mapped to state <3>. Operators LC and SC both map to operator C, and operators LA and SA both map to operator A.

When any of these decompositions is applied to the N-disk Towers of Hanoi problem space the resulting space is isomorphic to the (N-1)-disk space. Hence the same decomposition can be applied repeatedly to produce a sequence of successively smaller problem spaces ending with the trivial problem space.

Problem-solving Efficiency

The aim of all kinds of change of representation, including solution refinement, is to improve the efficiency of problem-solving. Consequently, it would be useful to be able to predict the change in problem-solving efficiency that would result by making a particular change of representation. This ability would enable a system to select the most efficient among a set of possible changes of representation — for example, to select the best of the four decompositions of the 2-disk Towers of Hanoi problem. And, accompanied by an estimate of the problem-solving efficiency of the original problem representation, this ability would enable a system to determine whether any of the changes of representation is actually an improvement.

It is not difficult to analyze the efficiency of solution refinement methods under the assumption that the expansion factor at every level is the same. Let A be the number of nontrivial problem spaces, and X be the expansion factor. Then the length of the final solution is X^A. If W[X] denotes the amount of "work" required to refine a single state-operator solution fragment, then the total amount of work required to create the final solution is $W[X]*(X^A-1)/(X-1)$.

In his thesis (Knoblock,1991), Craig Knoblock observes that if X is a constant and A is proportional to the logarithm of the optimal solution length, then the work required by solution refinement is exponentially less than the work required by a brute force problem-solver in the original (undecomposed) problem space. These circumstances hold when the standard decomposition is used to solve Towers of Hanoi problems in which all disks are initially on the same peg.

This formula for "work" provides a direct way to evaluate the efficiency of different decompositions of a problem space, providing that one can compute W[X] and measure the values of X and A for a given decomposition. In fact, the only real difficulty is the calculation of X. The number of non-trivial problem spaces is normally self-evident, and the term W[X] is almost always negligible compared to X^A. Note that with the values of X and A we can calculate the expected length of a solution as well as the expected amount of work required to create it.

To see how to calculate X, recall that X, the expansion factor, is (by definition) equal to the average number of states in the segments "Xi-Ii-RSi" that are inserted during refinement. If the method used to change representation imposes constraints on the possible values of Xi and RSi, then these constraints may provide enough information to compute an expected value, or at least an upper bound, on X. For example, in homomorphic decompositions it must be the case that Xi and RSi are "equivalent", i.e. that they are mapped to the same state by the homomorphism. Given this fact, the expected value of X is simply the "average" length of the shortest path (operator sequence) between each possible pair of equivalent states. "Average" is in quotes because the actual probability of encountering each of the <Xi,RSi> pairs in practice is normally unknown.

To illustrate this computation, consider the standard decomposition of the 2-disk Towers of Hanoi problem space. 9 different <Xi,RSi> pairs can be constructed from the 3 states that map to <1>. Of these 9 pairs, 3 are of the form <S,S>, 3 are of the form <S,SC(S)>, and 3 are of the form <S,SA(S)>. The shortest path connecting S to S has a length (number of states) of 1, and the shortest path connecting S to SC(S) or SA(S) has a length of 2. The same analysis holds for the the states that are mapped to <2>, and for those that are mapped to <3>. Therefore the expected value of X, assuming all pairs of equivalent states are equiprobable, is (3*1 + 6*2)/9, or 5/3. This turns out to be impossibly low — in the N-disk Towers of Hanoi problem X must be larger than twice the Nth root of 2/3 — an indication that all pairs are not actually equiprobable. Nevertheless, this value may still be useful to

compare with the value of X computed in the same manner for the other decompositions.

The three other decompositions all have the same expected value of X, namely 2.56. This is considerably larger than the value for the standard decomposition. Thus we expect that the standard decomposition will produce shorter solutions with less work than the other decompositions. To test this prediction, the standard decomposition and the second non-standard decomposition were used to solve all N-disk problems in which all disks are initially on peg 1. Work is measured as the number of arcs traversed by a breadth-first problem-solver before finding a solution[3]. The results of this experiment are:

# of Disks	WORK	
(N)	Standard	Non-Standard #2
2	8.7	12.8
3	25.4	47.1
4	63.0	133.0
5	142.0	328.0

# of Disks	SOLUTION LENGTH	
(N)	Standard	Non-Standard #2
2	3.0	3.2
3	5.7	6.7
4	11.0	14.3
5	21.7	29.9

The ratio of successive solution lengths gives a true indication of the actual expansion factors of the two decompositions: X is optimal (slightly less than 2.0) for the standard decomposition and 2.1 for the non-standard decomposition. The difference in expansion factors is much smaller than predicted, but still results in a significant difference in solution lengths and the work required.

If a formula is available to compute the expected amount of work required for problem-solving in the original (undecomposed) problem space, then this can be compared to the work formula for solution refinements to determine whether a given decomposition will improve or degrade efficiency. In the N-disk Towers of Hanoi problem space the expected amount of work is half the number of arcs in the entire space (assuming that the problem-solver never traverses the same arc twice), which is given by the formula $3*(3^N-1)/2$. Because this formula has the same form as the work formula for solution refinement, it follows immediately that a decomposition will degrade performance on the N-disk Towers of Hanoi if and only if its expansion factor is 3 or greater.

In the same way that the work required with and without a change of representation can be compared, so too can solution length be compared. A breadth first problem-solver always finds a minimal length solution. In the N-disk Towers of Hanoi problem space, the minimum solution length, for the average problem in which all disks are initially on peg 1, is $(2^{(N+1)}+1)/3$. Comparing this to the expected solution length for solution refinement, it follows that a decomposition will produce exponentially longer solutions whenever its expansion factor is greater than 2.

The fact that the critical value of the expansion factor is different for solution length and work-required leads to the apparent paradox that some decompositions will construct exponentially longer solutions and yet do exponentially less work. In fact, the second non-standard decomposition exhibits this phenomenon, as the following data shows (the experimental conditions are the same before).

# of Disks	WORK	
(N)	Original Space	Non-Standard #2
2	10.8	12.8
3	37.8	47.1
4	118.8	133.0
5	361.9	328.0

# of Disks	SOLUTION LENGTH	
(N)	Original Space	Non-Standard #2
2	3.0	3.2
3	5.7	6.7
4	11.0	14.3
5	21.7	29.9

[3] Unlike the problem-solver in Knoblock's analysis, this problem-solver never traverses the same arc twice. This simple bookkeeping usually results in an exponential reduction in the work required.

Conclusion

The aim of changing representation is the improvement of problem-solving efficiency. For the most widely studied family of methods of change of representation it has been shown that the value of a single parameter, called the expansion factor, is critical in determining (1) whether the change of representation will improve or degrade problem-solving efficiency, and (2) whether the solutions produced using the change of representation will or will not be exponentially longer than the shortest solution. A method of computing the expansion factor for a given change of representation has been sketched in general and described in detail for homomorphic changes of representation. The results have been illustrated with homomorphic decompositions of the Towers of Hanoi problem.

References

Holte, R. C. (1988), "An Analytical Framework for Learning Systems", Ph.D. dissertation, Brunel University. also TR-AI88-72, Computer Sciences Dept., Univ. of Texas at Austin.

Knoblock, C.A. (1991), "Automatically Generating Abstractions for Problem Solving", technical report CMU-CS-91-120, Computer Science Department, Carnegie-Mellon University.

Knoblock, C.A., J.D. Tenenberg, and Q. Yang (1991), "Characterizing Abstraction Hierarchies for Planning", Proc. IJCAI, pp.692-697.

Sacerdoti, E. (1974), "Planning in a hierarchy of abstraction spaces", Artificial Intelligence 5(2):115-135.

Sacerdoti, E. (1977), "A Structure for Plans and Behaviour", American Elsevier Publishers.

Stefik, M. and Lynn Conway (1982), "Towards the Principled Engineering of Knowledge", The AI Magazine, vol.3, no.3, pp. 4-16.

Yang, Q. and J.D. Tenenberg (1990), "Abtweak: Abstracting a nonlinear, least commitment planner", Proc. AAAI, pp.204-209.

Zimmer, R., Alan MacDonald, and Robert Holte (1991), "Reasoning about Representations: Towards the Automation of Representation Change", Proceedings of the Florida Artificial Intelligence Research Symposium, pp. 201-205.