AN ANALYSIS OF ABSTRACTION IN PROBLEM SOLVING

Richard E. Korf

CUCS-108-84

An Analysis of Abstraction in Problem Solving

Richard E. Korf

Department of Computer Science
Columbia University
New York, N.Y. 10027
(212)280-8193
Korf@Columbia-20.ARPA

March, 1985

Abstract

A quantitative model of abstraction in problem solving is presented which explains how and to what extent it reduces the amount of search necessary to solve a problem. It is shown that a single level of abstraction can reduce search time by a factor of the square root of the size of the original space. Multiple hierarchical levels of abstraction can reduce the search complexity of a problem from linear in the size of the original problem space to logarithmic.

1. Introduction

The value of abstraction is well-known in artificial intelligence. The basic idea is that in order to efficiently solve a complex problem, a problem solver should at first ignore low level details and concentrate on the essential features of the problem, filling in the details later. The idea readily generalizes to multiple hierarchical levels of abstraction, each focused on a different level of detail. Empirically, the technique has proven to be very effective in reducing the complexity of large problems.

Like many ideas in AI, the value of abstraction in human problem solving was pointed out by George Polya in *How to Solve It* [2]. The first explicit use of abstraction in an AI program was in the planning version of the General Problem Solver (GPS) developed by Newell and Simon [1]. The most thorough exploration of abstraction to date is Sacerdoti's work on the ABSTRIPS system [3].

This paper presents a quantitative analysis of abstraction in problem solving. Our goal is to provide an analytic explanation for the empirical observation that abstraction reduces complexity. The questions we address are: how much search

---

This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165, and by the National Science Foundation Division of Information Science and Technology under grant IST-84-18879.

efficiency is gained by the use of abstraction, and what is the optimum level of detail for each level of abstraction. To do so, we first formalize a model of abstraction. Next, we consider the special case of a single level of abstraction. Finally, we turn our attention to the general case of multiple abstraction levels. The analysis is done in the average case. The main result is that an abstraction hierarchy can reduce the amount of search to solve a problem from linear in the size of the problem space to logarithmic. A practical result of the analysis is that many levels of abstraction, with only small differences between them, reduce search the most.

2. A Model of Abstraction

Our model of abstraction in problem solving is based on the *problem space* model of Newell and Simon [1]. A problem space consists of a set of states and a set of operators which are partial functions from states to states. A *problem* is a problem space together with an initial state and a goal state. The task is to find a sequence of operators that map the initial state to the goal state. For example, if the domain were transportation, the states might be towns and the operators might be all direct means of transportation between towns.

We model the states of an abstract space as a subset of the states in the original problem space (the base space). Continuing our example, a suitable set of abstract states for the transportation problem would be the set of major cities. An alternative model is for *each* state of the abstract space to correspond to a subset of the states in the base space. In the transportation example, a state of the abstract space would then correspond to a region in the base space. In many cases, however, operators only apply to particular states as opposed to sets of states. For example, while an airline route may serve an entire area, it departs from a particular point in that area. In any case, the main difference between the subset and region models is that in the region model, no effort is required to get into the abstract space, since any node in a region is already a member of the region as a whole. This reduces the search complexity by a constant factor, and for this reason we adopt the more restricted subset model.

The operators of the abstraction map states in the abstract space to other states in the abstract space. In the case of GPS and Sacerdoti's work, the abstract operators were that subset of the primitive operator set that mapped abstract states to abstract states. In the transportation example, the operators of the abstract space would be direct means of transportation between major cities, such as airline routes. Alternatively, the abstract operators may be macro-operators or sequences of primitive operators that go between abstract states. A necessary property of the macro-operators is that they be stored or otherwise known to the system and do not require search to find. If we restricted our example to road navigation, the abstract operators might be driving routes between major cities, which in general

are sequences of different roads marked by signs. Regardless of which type of abstract operators we choose, the assumption of known operators between abstract states is an important aspect of our model. An abstract space requires both a set of abstract states and a set of abstract operators between those states. If paths between the abstract states must be found by search among the operators of the base space, then abstraction by itself makes no improvement in search efficiency. For example, if we have to search for routes between nearby major cities, then the abstraction of major cities is of no use in routefinding. Note that we do not require an abstract operator between every pair of abstract states, but simply that the set of abstract states be connected by abstract operators.

Problem solving using an abstraction space involves three steps. First, a path must be found from the initial state to the nearest state in the abstract space using the operators of the base space. Next, a path from the initial abstract state to the abstract state nearest the goal state must be found using the operators of the abstract space. Finally, a path must be found from the abstract goal state to the actual goal state using the operators of the base space. For example, in the transportation domain, the problem of getting from a starting point to a destination point is solved by finding a route from the starting point to the nearest major city, finding a route from that city to the major city nearest the destination, and finally finding a route from there to the actual destination. One way of recognizing the abstract state which is closest to the goal state is to search backward from the goal state to the nearest abstract state.

For purposes of this analysis, we assume that the relative distribution of abstract states over the base space, while not necessarily uniform, is the same for all levels of abstraction. In our example, this corresponds to the observation that while there are more towns in the eastern U.S. than in the west, the relative distribution of large cities is roughly the same as that for small cities. An average case analysis can be done independent of the actual distribution, as long as the distribution remains constant over different levels of abstraction.

3. Single Level of Abstraction

We begin our analysis with the special case of a single level of abstraction. Let the number of states in the base space be n and the number of states in the abstract space be n/k, with k being the ratio of the size of the base space to the size of the abstract space. The first issue we address is the expected amount of search required to find a solution to an average problem, using the abstraction. By average problem, we mean one where the initial and goal states are randomly selected from all states with equal probability.

We assume that the amount of time to search for a solution is proportional to the number of nodes visited in the search. The expected number of nodes visited is the

sum of the expected number of nodes visited in the three phases of the search: getting into the abstract space, solving in the abstract space, and getting to the goal in the base space. Since we assume that the distribution of states in the abstract space is the same as the base space, the probability that any randomly selected node is a member of the abstract space is $1/k$. Thus, the expected number of nodes that must be generated in a search before a node in the abstract space is encountered is k. This is both the expected number of nodes to be searched in going from the initial state to the abstract space and also in going from the abstract space to the goal state.

Since the total number of abstract states is n/k, to find a path between an arbitrary pair of abstract states, we would expect to have to examine $(1/2)n/k$ nodes on the average. Thus, the total expected number of nodes expanded is $t=2k+(1/2)n/k$. Note that this is actually an upper bound on the expected number of nodes because it ignores the slight possibility that the goal state will be found before the abstract space is encountered when searching from the initial state.

What value of k minimizes this total search time? The minimum occurs at $k=(1/2)n^{1/2}$, or $n/k=2n^{1/2}$, giving a value of $t=2n^{1/2}$. Thus, for a single level of abstraction, if the base space is of size n, the optimum size for the abstract space is on the order of the square root of n. This abstraction reduces total search time from $O(n)$ without any abstraction, to $O(n^{1/2})$.

4. Multiple Levels of Abstraction

We now consider the general case of multiple hierarchical levels of abstraction. In order to solve a problem in a hierarchy of abstraction spaces, we first map the initial state to the nearest state in the first level of abstraction, then map this state to the nearest state in the second level of abstraction, etc., until the highest level of abstraction is reached. At the same time, this process is repeated starting from the goal state and working up through successive levels of abstraction until the paths from the initial and goal states meet at the highest abstraction level. The questions we want to answer are how much do multiple levels of abstraction reduce search, how many levels of abstraction should there be, and what should the ratios between their sizes be in order to minimize the search time to solve an average problem.

Again, let the number of states in the base space be n. Let k_1 be the ratio between the size of the base space and the size of the first level of abstraction, k_2 be the ratio of the sizes of the first and second levels of abstraction, etc. In order to make the search at the top level the same as in the lower levels, let the highest level of abstraction consist of a single node. The task at the top level then is to find paths from both the initial and goal states to this single common state. With this slight simplification, the expected amount of time to find a solution to an

average problem in this hierarchy of abstraction spaces is $2k_1+2k_2+...+2k_m$, where m is the number of levels of abstraction, since two searches must be made at each level to find a node in the next higher level. As in the case of a single abstraction, this is actually an upper bound on the expected time since it ignores the possibility that the actual goal state will be found before the highest level of abstraction is reached.

In order to minimize this expression, we must minimize the sum of the k_is. The constraint on the k_is is that their product must equal n, since they represent the ratios between the number of nodes at each level and n is the total number of nodes. Thus, the problem becomes one of factoring a number such that the sum of the factors is minimized, or in other words, finding a *minimum sum factorization*.

If the factors are constrained to be integers, then the prime factorization of n is a minimum sum factorization. However, in our case the k_is are ratios which need not be integral but rather can be arbitrary rational numbers. In that case, the minimum sum factorization of a number n consists of $ln\ n$ factors, each of which is equal to e. To see this, first note that all the factors must be equal, because two unequal factors could be replaced by two factors between them, such that the sum is reduced without changing the product. This implies that there must be $log_k n$ factors of k, and minimizing their sum, $k\ log_k n$, yields $k=e$.

This implies that the optimum abstraction hierarchy for a problem space with n nodes consists of $ln\ n$ levels of abstraction and the ratio of the sizes of successive levels of abstraction is e. Of course, in a real problem, such an optimum abstraction hierarchy may not be achievable. However, this result suggests that in general a deep abstraction hierarchy, in terms of number of levels, will reduce search more than a shallow one.

The average case time to find a solution given an optimum abstraction hierarchy is proportional to $2k_1+2k_2+...+2k_m$ or $2e(ln\ n)$. Thus, the use of such an abstraction hierarchy can reduce the expected search time from $O(n)$ to $O(log\ n)$. This improvement makes exponential problems tractable.

5. Conclusions

We have presented a quantitative model of the use of abstraction in problem solving. The main result is that for a problem with n states, the abstraction hierarchy which minimizes the amount of search to solve an average problem consists of $log\ n$ levels where the ratio between the number of states in successive levels is a constant. Such an abstraction hierarchy reduces the amount of search from $O(n)$ to $O(log\ n)$. This reduction from linear to logarithmic time explains how abstraction can make exponential problems tractable. Furthermore, it suggests that deep abstraction hierarchies reduce search more than shallow ones.

The only quantitative empirical data in the literature on the use of abstraction in problem solving is Sacerdoti's comparison of STRIPS and ABSTRIPS on five different problems. For these problems, ABSTRIPS provided an approximately logarithmic speedup over STRIPS in time to find a solution. While five data points are too small a sample to support the model, Sacerdoti's data is at least consistent with our theory.

Acknowledgments

The solution to the minimum sum factorization problem was pointed out to me by Herbert Simon in a different context. Several ideas were clarified in discussions with Tom Ellman. In addition, Kathy McKeown, Judea Pearl, and Steve Taylor carefully read an earlier draft of this paper and provided many helpful suggestions.

References

[1] Newell, A. and H. A. Simon.
Human Problem Solving.
Prentice-Hall, Englewood Cliffs, N.J., 1972.

[2] Polya, George.
How to Solve It.
Princeton University Press, Princeton, N.J., 1945.

[3] Sacerdoti, Earl D.
Planning in a Hierarchy of Abstraction Spaces.
Artificial Intelligence 5:115-135, 1974.