OK

Natural Generation of Admissible Heuristics

Dennis Kibler

University of California at Irvine

Information and Computer Science Department

TR-188

15 July 1982

This research was supported by the Naval Ocean Systems Center under contract N00123-81-C-1165.

Abstract

If a problem space can be represented by a relational production system, then many useful heuristics can be generated by appealing to appropriate abstraction spaces. Here we formalize the process by which heuristics can be generated. We show that these heuristics are admissible and monotonic. Finally we give several heuristics from the literature which could have been formed by the process, as well as some which cannot.

Index Terms: Heuristic generation, admissible heuristics, problem transformation, quotient spaces, problem representation.

Introduction

This paper formalizes the following idea: an estimate of the distance between the current state and the goal state can be found by mapping the current problem into an analogous, simpler problem, and counting the number of steps in a solution of the analogous problem. Moreover the analogous problem is defined in a natural way from the original problem. Similar relaxation methods have been explored by Pearl [10] and Valtorta [12] but they do not as precisely define the problem representation nor as exactly define the problem transformations.

Changing the representation of a problem may provide insights into its solution. Amarel [1] shows how to transform the representation of a generalized missionaries and cannibals problem. Korf [5] discusses the heuristic search through a class of graphical representations. In both papers a problem is transformed and a solution to the transformed

problem provides a solution to the original problem. In our work, the transformed problem provides a means for gaining insight into the original problem. This insight takes the form of an admissible heuristic.

We begin by formally defining natural transformations, quotient maps, and counting functions. We then apply the concepts to the eight-tile puzzle, the cube slicing problem, the blocks world, and the mutilated checkerboard problem. Lastly, we discuss the technique's limitations.

Natural Transformations of Problem Spaces

In this section we formally define problem domains and natural transformations. We alter slightly the concept of a relational production system as defined by Vere [13]. This modification was introduced by Morris [4] where its merits are more fully discussed.

A problem space or problem domain consists of an initial state, which is a finite bag or multiset of assertions, and a finite collection of relational productions or operators. In contrast to Vere, we use multisets of assertions rather than sets of assertions. We define a relational operator by specifying two multisets of conditions, one called the preconditions and the other the postconditions. An operator is applicable in a given state if each of its preconditions is satisfied in the state. The preconditions of an operator are satisfied if there exists a substitution for the variables such that the instantiated preconditions are contained, as a multiset, in the state description. One applies an applicable operator by deleting each of the instantiated

preconditions from the state and adding each of the instantiated postconditions.

A problem is a list of goals to be satisfied. Note that a problem may only partially specify a final state. A solution to a problem in a domain is a sequence of productions which when applied to the initial state, results in a state which satisfies each goal listed in the problem.

A natural transformation T from one domain into another is a mapping of states and operators of the source domain into the states and operators of the image domain such that the following equation holds, for all states and applicable operators:

apply(T(Op),T(S))=T(apply(Op,S)) (equation 1)

where apply(Op,S) is the new state arrived at by applying Op to the state S. This is similar to the definition of a homomorphism given by Banerji [2] for games. Given a source domain D with initial state I and problem P and a natural transformation T into some image domain, we note a number of interesting properties.

The solution of the problem in the source domain maps into a solution of the image problem T(P) with initial state T(I).

The length of a solution in the image domain is not greater than the length of a solution in the source domain.

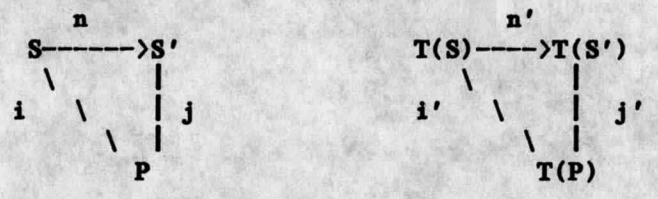
The length of the minimum solution in the image domain is a lower bound on the length of the solution in the source domain and consequently provides an admissible heuristic.

Nilsson [7] defines an admissible search algorithm as one which is guaranteed to find the optimal solution, whenever one exists. We say a heuristic h is admissible if it underestimates the cost from the node to the goal. Nilsson proves that the evaluation function f(n)=g(n)+h(n), where g(n) is the current cost to the node n, and h is admissible, defines an admissible algorithm. Pearl [8] analyzes the cost effectiveness of A^* in terms of the accuracy of the heuristic h. In a further work, he and Kim [9] anaylze three extensions of A^* which weaken the assumption of admissibility. Valtorta [12] analyzes the cost of A^* where he includes the cost of computing the heuristic h assuming that h is computed by blind search rather than some optimized procedure.

We have the following theorem:

> **Theorem 1:** If T is a natural transformation from domain D with initial state I and problem P, and h(S), for a state S of D, is defined as the minimum number of operators required to go from T(S) to T(P), then h is admissible and monotonic.

The admissibility has already been demonstrated. Recall that monotonicity is a form of the triangle inequality. To show monotonicity consider the following diagrams:



where n is the length of a path from S to S', i is h(S), j is h(S'), and i', j', and n' are the minimum lengths between states in the image domain. We must show that $n+j \geq i$. By definition $i=i'$ and $j=j'$. Also $n'+j' \geq i'$ by definition. Since $n \geq n'$, we have $n+j \geq i$, and monotonicity is proven.

Monotonic heuristics eliminate the need for updating the g-values of CLOSED nodes. Whenever the g-value of a node is updated, one must check whether any descendant of the node requires updating. This is an expensive operation within the A^* algorithm.

A simple useful class of natural transformations are quotient maps. A quotient map simply forgets about some of the relations in the source domain, for both operators and states. More precisely a quotient map Q is defined by a set S of relations as follows:

Q(state)={n| member(n,state) and not member(n,S)}
Q(operator)=newop where
preconditions(newop)=Q(precondition(op)) and
postcondition(newop)=Q(postcondition(op)).

Such a map is easily shown to be a natural transformation. Notice that if the operators and states can be described with n relations, then the number of quotient maps is 2^n.

Another useful class of natural transformations are counting maps. A counting map replaces a collection of literals by the number of instances of each type of literal. Intuitively, quotient maps forget about some relations, while counting maps forget about the particular objects. More precisely, if the domain is described by n relations, say r_1, r_2, through r_n, then the counting map will map each state into an n-vector of integers, where the integer of the i^{th} component corresponds to the number of assertions of type r_i in the state. Equation 1 determines the definition of the image of an operator. Again this map is clearly a natural tranformation.

(For convenience we denote the multiplicity of a relation by writing multiplicity*relation. Consequently when we write the

preconditions of a counting operators as $n*r_1$, $m*r_2$, etc. we mean that the operator deletes n assertions of type r_1, m assertions of type r_2, etc. Postconditions and states are denoted similarly.)

Notice that natural transformations have the effect of focusing attention on some aspect of the problem. Focusing on the wrong aspect of a problem leads to no insight. Focusing on the right aspect can lead to appropriate heuristics or a quick proof that the problem is unsolvable.

EXAMPLES

Tile puzzle

This puzzle is analyzed by Nilsson [6] where he defines a number of heuristics for its solution. This puzzle can be defined as a relational production in the following way. Let the board size be 3X3 and label the positions as in the diagram below.

position labels	goal state
a b c	1 2 3
d e f	4 5
g h i	6 7 8

A state, such as the goal state depicted above, could be described as:

```
pos(a),pos(b),...,adj(a,b),adj(b,a),adj(a,d),...
on(a,1),on(b,2),on(c,3),on(d,4),blank(e),on(f,6),
on(g,7),on(h,8).
```

where pos stands for position and adj stands for adjacent. There is only one relational operator, defined by

```
preconditions: adj(X,Y),pos(X),pos(Y),blank(Y),on(X,V)
postconditions: adj(X,Y),pos(X),pos(Y),blank(X),on(Y,V)
```

where adj stands for adjacent and pos stands for position.

A number of different quotient spaces can be constructed from this

problem. If we demand that the quotient space maintain the "on" relation, then there are seven (2^3-1) candidate quotient spaces. We will look at three of them. Different quotients spaces sometimes lead to the same heuristics.

If we forget about the requirement that a tile be blank, then we get a new problem space with an operator such that

preconditions: pos(X),pos(Y),adj(X,Y),on(X,V)
postconditions: pos(X),pos(Y),adj(X,Y),on(Y,V).

This corresponds to a tile-puzzle where you are allowed to pile up tiles. In this world it is easy to see that the distance (minimum number of moves to change one configuration to another) is exactly the sum of the city-block distances between each tile's current position and its destination. This is exactly heuristic P(n) [6].

If we forget about both the blank requirement and the adjacency requirement we get a new problem space with an operator whose preconditions are pos(V),pos(X),on(X,Y) and whose postconditions are pos(X),pos(V),on(X,V). This corresponds to a tile-puzzle where you are allowed to pick up any tile and move it to any desired position. In this puzzle, the minimum distance between two states is the number of tiles out of position. Hence the distance of the minimum solution for this quotient problem defines the admissible heuristic W(n) [6].

If we forget about the adjacency requirements we get the new operator with preconditions: blank(Y),on(X,V) and postconditions: blank(X),on(Y,V). This corresponds to allowing a tile to be moved into a blank position regardless of its position. The resulting heuristic is closely related to W(n).

Another heuristic, one that reduces the search more than any of those above, is defined to be P(n)+3*S(n) where S(n) measures the "sequence score" of tile position [6]. This heuristic is not admissible and so cannot be found by any natural transformation.

Cube Slicing

Since admissible heuristics provide lower bounds on the number of operations required to fulfill a goal, they can be used to determine that some problems have no solution. An instance of such a problem is the question of whether a cube can be sliced into 27 equal cubelettes with only five slices. By insight one sees that it is necessary to slice each of the six faces of the inner cube, so the problem is impossible.

This conclusion can also be reached by applying natural transformations. A detailed relational description of the original problem would be tedious. We give only the image domain and the corresponding operators. By applying quotient maps and counting functions, one reduces the problem to one whose initial state is {block<27>}, where the 27 refers to the volume in terms of smaller cubes. The goal state is {27*block<1>}. The slicing operations generate a collection of relational operations. Each relational operator deletes a set of blocks and replaces each block of the subset by two blocks, where the new blocks are of equal size or one is twice as large as the other. Some of the relational operators are:

op1: delete conditions: {block<9>,block<6>}
add conditions: {3*block<3>,block<6>}

op2: delete conditions: {block<9>,block<6>}
add conditions: {block<3>,block<6>,block<2>,block<4>}

Without worrying about the entire search tree that would be generated, but just following the division of the largest block, we get the following chain:

block<27>->block<18>->block<9>->block<6>->
->block<3>->block<2>->block<1>.

This chain has six operations in it so the original problem requires at least six operations.

Blocks World

We will only sketch the application of natural transformations to the blocks world. Using any of the usual relational descriptions of the blocks world [7], one can define a quotient map which forgets about all relations but the "on" relations. The admissible heuristic generated is the number of different "on" instances between the current state and the goal state. Such a heuristic has the effect of making the search somewhat similar to that generated by STRIPS [3], in that the search is directed towards achieving the "on" goals sequentially for each possible ordering of the goals. However, the admissible heuristic approach guarantees finding a solution. We note that goal ordering fails for STRIPS [11] because the goals only partially specify the goal state. If the goals had been "completed" or extended to give a full specification of the final state, then goal ordering (from table upwards) would allow STRIPS to solve all blocks worlds problems easily.

Mutilated checker board

The problem is: if a checkerboard has the white corners removed, can it be covered by a dominoes. We will use a natural transformation to show that the problem is impossible. Below is an abbreviated relational description of the initial state, the single operator, and

```
the goal state:
 initial state: adj(a,b),....
                white(a),...
                red(b),...
                free(a),free(b),...
 operator: Pre: adj(X,Y),red(X),white(Y),free(X),free(Y).
           Post: adj(X,Y),red(X),white(Y),covered(X),covered(Y).
 goal: covered(a),covered(b),...
```

We define a natural tranformation into states described simply by the number of free white (fw) and free red (fr) squares. Consequently we have:

```
 image initial state: 30*fw,32*fr.
 image operator:  preconditions: fw,fr.   {deleted relations}
                  postconditions: empty. {added relations}
 image goal:        empty. {i.e. (0*fw,0*fr)}
```

It is easy to check that this mapping is a natural transformation, although not one that is a combination of quotient or counting maps. Moreover in the new space only one operator is applicable to any state, so one quickly sees that there is no way to reach the state {0*fw,0*fr}.

Limitations

This approach always generates admissible heuristics, but not all admissible heuristics are useful. Applied to the tower-of-hanoi problem, the approach yields no useful heuristic. The quotient map may fail to help because it a) trivializes the problem or b) does not reduce the difficulty sufficiently so that the calculation of the minimal length solution is simple.

Conclusions

We have shown that by applying natural transformations to a multiset representation of a problem, one can generate admissible and monotonic heuristics. These heuristics can be used to guide the search for a solution or to demonstrate the impossibility of finding a

solution. Moreover, two types of natural transformations, quotient maps and counting functions, were defined which are intrinsic to the original problem, i.e. they do not require a "eureka" form of insight to construct the appropriate image domain or transformation. Finally we have used the technique to generate a number of standard heuristics as well as providing a new demonstration of the impossibility of the cube slicing problem.

REFERENCES

[1]
Amarel, S.
On the representations of problems of reasoning about actions.
American Elsevier, 1968.

[2]
Banerji, R.B.
Artificial Intelligence: A Theoretical Approach.
Elsevier, 1980.

[3]
Fikes, R.E., and Nilsson, N.J.
STRIPS: A new approach to the application of theorem proving to problem solving.
AI,1971.

[4]
Kibler, D.F, and Morris, P.H.
Plan Variants and a Plan Refinement Algorithm.
Technical Report 178a, Univeristy of California, Irvine, 1982.

[5]
Korf, R.E.
Toward a Model of Representation Changes.
AI,1980.

[6]
Nilsson, N.J.
Problem-Solving Methods in Artificial Intelligence.
McGraw-Hill, 1971.

[7]
Nilsson, N.J.
Principles of Artificial Intelligence.
Tioga, 1980.

[8]
Pearl, J.
Knowledge versus Search: A Quantitative Analysis Using A*.
Technical Report UCLA-ENG-CSL-8065, Univ. of California, Los Angeles, 1981.

[9]
Pearl, J. and Kim, J.H.
Studies in Semi-Admissible Heuristics.
IEEE Transactions on Pattern Analysis and Machine Intelligence,1982.

[10]
Pearl, J.
On the Discovery and Generation of Certain Heuristics.
Technical Report UCLA-ENG-CSL-8234, Univ. of California, Los Angeles, 1982.

[11]
Sacerdoti, E.D.
Planning in a hierarchy of abstraction spaces.
AI,1974.

[12]
Valtorta, M.
A Result on the Computational Complexity of Heuristic Estimates for the A* Algorithm.
Technical Report, University of North Carolina, 1981.

[13]
Vere, S.A.
Relational Production Systems.
AI, 1977.