CHARACTERIZING SEARCH SPACES

Roy Rada
Department of Computer Science
Wayne State University
Detroit, Michigan 48202

ABSTRACT

A heuristic is good on a search space to the extent that it allows the prediction of which states are near to the goal. In this paper heuristics are investigated on several different search spaces. A measure is proposed for assessing the predictive accuracy of a given heuristic on a given search space. The measure sheds light on characteristics of the Traveling Salesman Problem that make it computationally more difficult to solve than the Minimum Spanning-Tree Problem.

INTRODUCTION

A search space 55 is a quadruple (5, O, /, G), where 5 is a set of states, *O* is a set of operators for moving from state to state, / is a set of initial states, and G is a set of goal states (see *Handbook AI* '81 and Banerji, '82). The problem is to find the sequence of operators that connects some initial state to a goal state. To avoid exhaustive search, the search algorithm employs, as a guide, a heuristic evaluation function /(Nilsson, '80).

To characterize the utility of / on 55 a measure N1V is created. The author hypothesizes that difficult problems can be distinguished from easy problems with NIV. Difficulty is traditionally related to computational complexity, but NIV directly characterizes relationships within the search space.

FORMALIZATION

NIV on (55, f*)* tells the degree to which similar states have similar f values. Two states are similar, if they are connected by an operator. To handle "similar" for / values the following terms are introduced: equivalence classes of states, heuristic-predictions, and inversions. The equivalence class 5 i is defined to be { *s* | *s e* 5 and *s* can be reached from / by a sequence of i operations } The heuristic-prediction function *h* on *s e* 5 is *h(s)* = max *{f[w)* \ *w e* S_{1+}, and 1 move connects *s* to w }. The ordering imposed by *h* on S_1 is compared to the ordering imposed by *f* on S_1, and to quantify this comparison, inversions are used. Let $a_1, \ldots, a_n$ be a permutation of the set $\{1, \ldots, n\}$. If $i < j$ and $a_i > a_j$, then the pair (a_i, a_j) is called an inversion of the permutation (Knuth, '73). For example, the permutation 2, 4, 1, 3 has the three inversions (2,1), (4,1), (4,3).

To determine $NIV_k(S,f)$ the following 3 steps are performed:

i) Order all $s \in S$, so that $f(s_1) \leq f(s_2) \leq \cdots f(s_m)$.

ii) Compute *h* for each s and create the sequence $h(s_1), h(s_2), \ldots, h(s_m)$, where $f(s_j) \leq f(s_{(j+1)})$. If $f(s_i) = f(s_{i+1})$, then $h(s_i) \leq h(s_{i+1})$.

iii) $NIV_k(S,f)$ = number of inversions in the sequence $h(s_1), \ldots, h(s_k)$.

Consider an ordering $A = a_1, \ldots, a_n$ and a permutation $A' = a_{i_1}, \ldots, a_{i_n}$. Define $NIV_j(A')$ to be the total number of inversions in $a_{i_1}, \ldots, a_{i_j}$. In other words, $NIV_j(A')$ is the number of inversions in the first j elements of A'.

Theorem:
Given A' and $NIV_2(A'), \ldots, NIV_n(A')$,
the list A can be constructed.

Proof:
Basis:
If $NIV_2 = 1$, then $a_{i_1}, a_{i_2} \rightarrow a_{i_2}, a_{i_1}$,
else $a_{i_1}, a_{i_2} \rightarrow a_{i_1}, a_{i_2}$.
This makes the first two elements correctly ordered.
Inductive Step:
Assume the first k elements are correctly ordered.
Move $a_{i_{(k+1)}}$ to the left $NIV_{(k+1)}(A') - NIV_k(A')$ positions.
Now the first k + 1 elements are correctly ordered.
∴

Thus, one deals at each stage with just the total number of inversions--a scalar--and not the actual inversions which form a set of ordered pairs. Yet, from these scalars and A', A can be reconstructed.

RESULTS

MST and TS

Two important graph problems are the Minimum-cost Spanning Tree Problem (MST) and the Traveling Salesman Problem (TS). Both involve weighted graphs and the search for a subgraph with certain properties. In the MST a subgraph is desired that spans the graph and has minimal cost. In the TS a cycle is desired that crosses every vertex and has minimal cost. For a graph with *e* edges, the solution to the MST requires

This work was supported in part by a Wayne State University Faculty Development Grant.

O(eloge) time steps. The only known algorithms to solve the TS require more than a polynomial amount of time (Garey and Johnson, '79) Why are these two problems of such different time complexity?

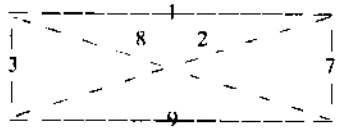
For many combinatorial problems *S* is part of a power set *P* on some primitives. Consider, for example, the complete 4-node graph with edges a, b, . . ., f and weights w(a), . . ., w(f). S_{MSI} = *{s\st* P({(a,w(a)), . . ., (f,w(f))}) and \s\ < 3 }. In order that $S_{MST} = S_{TS}$, a modified TS is henceforth considered in this paper, where the goal on an n-node graph has n-l edges. This modified TS is also NP-complete (Garey and Johnson, '79). A reasonable /for TS follows:

/TS({{X1W(X,))——, $(X_J,W(X_J))$})

(w(x,) + ■ • • w(x,)) 1 if x,, ... x, contains no cycles or trisections

= 0, otherwise.

Illustration of NIV for TS



S_2	$s_{1,2}$	$s_{1,3}$	$s_{2,3}$	$s_{1,7}$	· ·
f_{TS}	3	4	5	8	· ·
h_{TS}	11	11	12	11	· ·

Figure 1. Data from which NIV is calculated.

The reader is welcome to verify that for the weighted-graph of Figure 1 $NIV_3(S_2, f_{TS}) = 0$ and $NIV_4(S_2, f_{TS}) = 1$.

A computer program was written to determine NIV on MST and TS. The program generated complete graphs over 5 nodes. Weights were assigned to the edges after having been randomly chosen from a uniform distribution over {1, 2, . . ., 1000}. Eighteen graphs selected in this way had an average $NIV_j(S_2, f_{TS}) > NIV_j(S_2, f_{MST})$ for all j in S_2 (see figure 2).

NIV versus Position

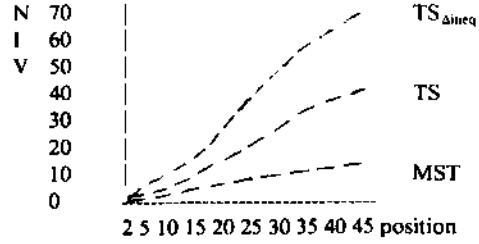


Figure 2. NIVj(5_2X) for j from 2 to 45 and X *e* {TS_{Aincq}, TS, MST}.

For 5_3 the relationship between NIV j(S_{3jfTS}) and NIVj($S3/_{MST}$) is similar to that for S_2. This evidence supports the hypothesis that more difficult problems have more NIV. However, at any given position j there exists a graph such that NIV (Sif_{TS}) < NIVj(Sif_{MST}), where i e {2,3}. Just the average over the 18 graphs is consistently, significantly higher for TS than for MST.

Graphs were also studied whose edge weights satisfy the triangle inequality (A ineq). Approximation algorithms for TS_{Aineq} can propose a solution whose length is guaranteed to not be greater than 1.5 times the length of an optimal tour (Reingold, et al, '77). No such algorithms exist for the unconstrained TS. Thus one suspects that by some standard of search space "difficulty" D that D(MST) < $D(TS_{Aineq})$ < D(TS). This author hypothesized that N1V would order these 3 problems as D did. To investigate this hypothesis the computer program was amended to generate 18 graphs whose edge weights satisfied the A ineq. For each graph the program chose 5 nodes whose coordinates were randomly chosen from the set {1, . . ., 707}. The edge weights thus ranged from 1 to 1000.

The results did not support the hypothesis. The average NIV over the 18 graphs revealed: $NIV_j(S_2, TS_{\Delta ineq}) > NIV_j(S_2, TS)$, j e {3,...,45} (see Figure 2). The author does not know how to explain this data. Three possibilities are:

1) The data showed $NIV_2(S_2, TS_{\Delta ineq}) < NIV_2(S_2, TS)$, which supports the hypothesis. The first two positions, as measured by NIV_2, have great bearing on problem difficulty.
2) The weights in the $TS_{\Delta ineq}$ tests were not uniformly distributed over {1,...,1000} but clustered into a smaller range. This may make the problem more difficult.
3) $TS_{\Delta ineq}$ may be in important ways more difficult than TS.

Function Optimization

The perceptron learning algorithm can be viewed as a function optimizer. Minsky and Papert ('72) showed that the search space of the perceptron is smooth and unimodal. The learning component of Samuel's checker player is similar to a perceptron. A number of developments in the history of AI can be viewed as developments in function optimization. A simple example shows how *SS, f,* and NIV apply to optimization of discrete functions. Two functions were studied: one had 1 peak and the other had 4 peaks. One expects that the multimodal function is harder to optimize, and the author's experiments produced a larger NIV for the multimodal function. *S* was the power set over all (x,y) pairs defining the function to be optimized. A set of initial (x,y) pairs formed /. *O* took a state S , = {(x,,y. . . . . (x„y,)} to {(x,,y,),...,(xi,y1), (xi ,,y,,,)}, such that X_{i+1}, was "near" an x in s,. *G* was the (x,y) pair whose y was maximal. The heuristic was f(*s)* = max{y | (x,y) *e s}.*

Rule-based Planning

Rule-based planning systems, like STRIPS and DCOMP (Nilsson, '80), fit the framework of this paper. For a simple example, denote the registers of a computer by r, and the contents of r; by Cj. The *S* of *SS* is {(r_1,c,), (r_2,c_2), (r_3,c_3), (r_4,c_4)}, where c_1 e {0, a, b}. The initial state is {(r_1,a), (r_2,b), (r_3,0), (r_4,0)}. One exchange problem E_1, has the goal of {(r ,,0), (r_2,0), (r_3,a), (r_4,b)}. Another exchange problem E_2 has the goal of {(r_1 ,b), (r_2,a), (r_3,0), (r_4,0)}. The operator is the assignment rule in Nilsson ('80):

rule:	assign (rj,a,r_j,b)
precondition:	(r_ha), (r_jb)
delete:	(rj,a)
assert:	(r_j,b)

The heuristic f(*s)* tells the number of registers in *s* whose contents are the same as the corresponding registers in the goal. Both E, and E2 can be solved in 4 steps. E2 seems harder, since for it a step from s, to sI+1 must be made such that h(s,) > *h(sI+1).* Application of NIV to E, and E2 reveals that NIV is much higher for E2 than E,. This supports the hypothesis that NIV distinguishes problems by their difficulty.

Extensions: Backward-chaining and Search Algorithms

The definition of heuristic-prediction *h* involves looking from 5, to 5i+1 and thus is related to forward-chaining. A small amendment to *h* requires the new function *bh* to look backward from 5, to S,.,. For *s e* S„ *bh(s)* = max{ *f(w)* \ *we* SI-1 and *s* and w are connected by O}. For all of the previously mentioned experiments, *bh,* as well as *h,* was used. When *h* is replaced by *bh* in the definition of NIV, a new function, called BNIV, is produced. For the function optimization and rule-based problems BNIV showed the same tendencies as NIV did-namely, the values of BNIV on the harder problems were larger than on the easier problems. BNIV for TS is not everywhere bigger than BNIV for MST. If step 2 in the definition of BNIV (and NIV) is amended so that f(*s*,)=/(.sJ +1) - *bh(sj+1* > *bh(sI+1*,), then BNIV (and NIV) of TS is everywhere greater than BNIV (and NIV) of MST.

Another direction in which to extend the applicability of NIV is that of relating the performance of search algorithms to the NIV of search spaces. Consider a class of search algorithms A = {A,, A2, . . .}. Each A, proceeds through an *(SSf)* for which the state space *S* has been divided into equivalence classes Sj, as defined earlier. Furthermore, A, goes from S, to *S:* to • • • Sk by emphasizing the states in each equivalence class whose / value places those states in the top i positions. One can show that over all possible *SS* and/those on which A , performs well have, on the average, smaller NIV values than those on which A l+1 performs well. Details of this argument can be found in (Rada, '81).

DISCUSSION

An efficient search algorithm examines only a relatively small number of the states in a search space before finding a solution. Heuristics permit a search to explore efficiently, but a given heuristic is good for some problems and not for others. How can one characterize a heuristic's power vis-a-vis a particular class of search spaces? A heuristic must, in some sense, predict which states in the search space are close to the goals. Often, easily-solved problems are those for which a heuristic / exists such that similar states have similar / values (see the continuity argument of Lenat, '82). This "similar-state –> similar-value" notion has been formalized in this paper by the introduction of predictions and a measure NIV, where NIV quantifies the accuracy of the predictions.

The hypothesis of this paper is that NIV can distinguish problems by their difficulty. Theoretical considerations reveal that for a search space and heuristic on which depth-first, best-first search finds the solution, NIV *2(Sif)* = 0, for all j. An easy problem, like MST, has the property that NIV *2(Stf)* = 0, for all j. To further test the validity of the hypothesis that NIV varies in proportion to problem difficulty, a large number of experiments have been performed. Three famous combinatorial graph problems, MST, TSAineq and TS, were analyzed on the computer. The computational complexity of MST is less than that of TS, and NIV for MST proves experimentally to be less, on the average, than NIV for TS. The experiments also demonstrate several phenomena which require more study before they can be considered evidence for or against the hypothesis. To demonstrate the applicability of this paper's methodology on other than graph problems, function optimization and rule-based planning have been tested; the results support the hypothesis.

Characterizing the utility of heuristics is an important step in the development of a theory of heuristics. Constraints in a search space must be advantageously utilized by good search algorithms (Pearl, '83). By noting what a heuristic says about the search spaces on which it is useful (as NIV docs), one may begin to understand how a heuristic benefits from constraints. One next step towards a theory of heuristics and search spaces is to characterize hard and easy problems by their NIV values.

Acknowledgements: Thanks to the referees and graduate student Ching-Chio Sheu.

REFERENCES

Banerji, R B (1982) "Theory of Problem Solving: A Branch of Artificial Intelligence" *Proc IEEE, 70:12* p 1428-1448.

Garey, M and Johnson, D (1979) *Computers and Intractability* Freeman & Co: San Francisco.

Handbook of Artificial Intelligence, Vol I (1981) ed A Barr and E Feigenbaum, William Kaufmann, Inc: Los Altos, Calif.

Knuth, D E (1973) *The Art of Computer Programming, Vol 3, Sorting and Searching,* Addison-Wcsley: Reading, Mass.

Lenat, Douglas (1982) "The Nature of Heuristics" *Artificial Intelligence, 19* 189-249.

Minsky, Marvin and Papert, Seymour (1972) *Perceptrons* , MIT Press: Massachusetts.

Pearl, Judea (1983) "On the Discovery and Generation of Certain Heuristics" *The AI Magazine, 4:1* p 23-33.

Nilsson, Nils (1980) *Principles of Artificial Intelligence* Tioga: Palo Alto, Calif.

Rada, Roy (1981) "Searching and Gradualness" *BioSystems, 14* p 219-226.

Reingold, E, Nievergelt, J, and Deo, N (1977) *Combinatorial Algorithms* Prentice-Hall: Englewood Cliffs, New Jersey.