THE AVOIDANCE OF (RELATIVE) CATASTROPHE, HEURISTIC COMPETENCE, GENUINE DYNAMIC WEIGHTING AND COMPUTATIONAL ISSUES IN HEURISTIC PROBLEM SOLVING

Assistant Professor Ira Pohl
Information Sciences, University of California
Santa Cruz, California

Abstract

To solve difficult problems heuristically, requires detailed attention to computational efficiency. This paper describes how a heuristic problem solving system, HPA, attempts to find a near optimal solution to the traveling salesman problem. A critical innovation over previous search algorithms is an explicit dynamic weighting of the heuristic information. The heuristic information is weighted inversely proportional to its depth in the search tree--in consequence it produces a narrower depth first search than traditional weightings. At the same time, dynamic weighting retains the catastrophe protection of ordinary branch and bound algori thms.

Key Words. Heuristic Search, Branch and Bound, HPA, Traveling Salesman Problem, Binary Tree Sort, Dynamic Weighting, preconditioning, Heuristic Combinatorics, Artificial Intelligence, Pruning.

Artificial intelligence can be defined as the art of computational compromise. Problem spaces in artificial intelligence are sizeable and are non-analytic in the sense that smooth convergence to a solution is unavailable. Hence computational methods must attempt to use their resources--space and time--cleverly. This paper discusses a heuristic search procedure for the Travel Salesman problem--n! in size, emphasizing computational efficiency and search organization.

The problem (symmetric costs)

There is a complete graph of n nodes. Each edge of the graph has a non-negative length. A tour is a path in the graph visiting every node once and only once and returning to the initial node. The length of a tour is the sum of the lengths of the edges of the tour. A tour is called optimal, if among all possible tours, it is the shortest. Since tours are cyclic, the initial node can be fixed as node 1. The space of all tours are the (n-1)! permutations of the nodes 2 through n.

It is well known (Lawler and Wood, 1966) that ordinary operations research algorithms require exponential time to solve the problem. Thus large (n>15) problems are either solved by branch-and-bound methods or by local-refinement methods (Shen Lin 1965). Branch-and-bound methods guarantee an optimal solution, if they terminate within their allotted time (of course the method is guaranteed to terminate in exponential time). Local-refinement methods generate solutions in polynomial time which are not guaranteed to be optimal.

Heuristic path Algorithm (HPA)

HPA is a general graph search algorithm that searches or-graphs using a merit ordering. It was developed by this author (Pohl 1969, 1970) to generalize the methods of Doran and Michie (1966) and Hart, Nilsson and Raphael (1968). The critical difference from the latter's algorithm A* is the use of a weighted merit function $f(x) = (1-w)g(x) + wh(x)$, $0 < w < 1$ (see Table 1). Suitable generalizations of HPA have been used by Kowalski (1969) and Michie (1972) for resolution theorem proving. In this domain, Kowalski's criticisms (Kowalski 1972) are especially astute. He notes that "diagonal search strategies" (his term for $w \sim 0.5$) have a critical defect. They investigate all equally meritorious alternative paths. In difficult problem spaces where any solution path is desired, this procedure is inappropriately breadth first.

Computational Catastrophes

When a computational procedure is forced to terminate having used up its time and space resources without finding a solution--this will be called a type 1 catastrophe.*

When a computationaT procedure terminates with an insufficiently good solution—this will be called a type 2 catastrophe. This type 2 catastrophe can only occur in problems that have an ordered set of feasible solutions.

Examples: Type 1 catastrophe occurs all the time in artificial intelligence research. An early example was SAINT's failure on 2 integration problems (Slagle 1961). A typical Type 1 catastrophe is the failure of a resolution search procedure to find a proof within a preset level bound.

Type 2 catastrophe can only occur in problems that require "best" solutions. The simplest local refinement strategy for the Traveling-Salesman problem is the nearest-neighbor rule (1-opt in Shen Lin's terminology). The rule is to select the nearest node not yet included in the sub-tour. Using this rule on the graph in figure 1 leads to a very bad tour, a type 2 catastrophe.

Other examples of type 2 catastrophe are not generating the only winning or drawing move in a game playing program, or remaining on a "heuristic plateau" (Minsky 1961), which is distinctly sub-optimal, in a problem space.

Over-relaxation, admissibility and dynamic weighting

When using HPA to solve various 15-puzzle (see figure 2) problems, it was found that values of $w > 0.5$ were most efficient. This weighting of the heuristic function no longer guaranteed "admissibility", i.e. a minimum length solution. The overreliance on the heuristic term is called overrelaxation because of the analogous technique in the

*Space and time catastrophe will be distinguished as separate types in a later paper.

computational solution of partial differential equation problems. The weight w was set to a constant using previous experience to obtain an efficient setting. The 15 puzzle experiments demonstrated the computational utility of overrelaxation. However, these solutions were rarely near the minimum length solution. In the traveling salesman problem this type of search would result in type 2 catastrophe.

To avoid a type 2 catastrophe, HPA would like to retain admissibility or at least be within a known percentage of error (Pohl 1970). This can be accomplished by using a technique to be called "dynamic weighting". Dynamic weighting makes w a function of the state. HPA now uses f(x) = g(x) + w(x)h(x) where 1 < w(x) * ∞.

Let x be a state in the search for a tour. Let the depth (x) = the number of edges included in the subtour represented by x and define

$$w(x) = (1 + e - \frac{e * \text{depth}(x)}{n}).$$

s = 1, depth (s) = 0
w(s) = 1 + e
if x is in Γ (s) then
x = (1, i), depth (x) = 1
$w(s) = 1 + e - \frac{e}{n}$

g(x) = t_{1i} {length of edge (1, i)}

The effect of having w decrease with depth is to insure a depth first search, if and only if the added cost decreases within ever tighter bounds. If at some point in the depth first search a type 2 catastrophe were to occur, the search would revert to ordinary branch-and-bound behavior. The dynamic weighting provides incentive to proceed in a depth first manner provided no unforseen increase in cost occurs. The resulting solution is within (1 + e) of the optimal solution (see appendix for proof).

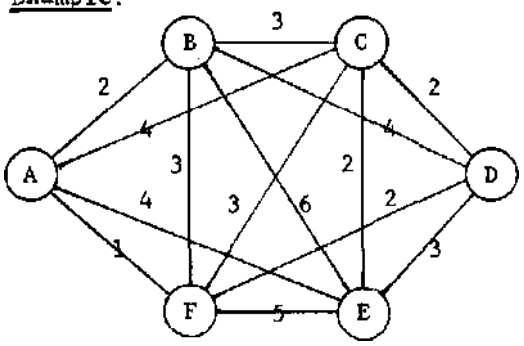
Heuristic Estimators for the Traveling Salesman Problem

There are several known bounds on the Traveling Salesman Tour. One is the Assignment Problem (Lawler and Wood 1966), and another is the Minimum Spanning One-tree (Held and Karp 1971). A third estimator used, in my experiments on symmetric distance problems (i.e. undirected graphs) is a variant of the Little estimator (Little, et al 1963), to be called the in-out estimator.

The Traveling Salesman Tour is the minimum weight connected graph, whose nodes are all of degree 2. The assignment problem is the minimum weight graph whose nodes are all of degree 2. The minimum spanning tree is the minimum weight connected graph. A 1-tree is a minimum spanning tree over nodes [2,...,n] with the addition of the two shortest edges out of node 1, In the case of the assignment problem a solution graph may be a set of cycles. In the case of the 1-tree a solution may contain many nodes of degree not equal to 2. Each relaxes a constraint of the Traveling Salesman Problem, hence each contains many nodes of degree not equal to 2. Each relaxes a constraint of the Traveling Salesman Problem, hence each contains the solution of the Traveling Salesman Problem as a feasible solution. (see Figure 3)

The in-out estimator, due to the author, is particularly simple to calculate. It is a lower bound on the Assignment Problem, but where an assignment problem is calculated in $0(n^3)$ steps, this bound ia calculable in $0(n^2)$ steps given that the edges are in sorted order. The in-out estimator is calculated by summing the 2 shortest edges out of each node not already included in the subtour and such that these edges are not connected to interior nodes of the subtour. This sum is added to the shortest edge out of each end node of the subtour, and the total divided by 2. The in-out estimator is easily seen to be a lower bound on the remaining length of a tour from the given subtour.

Example:



let x be the subtour
ABF
then A and F are end nodes and B is an internal node with
g(x) = 2 + 3 = 5
in-out (x) = 0.5 (2 + 2 + 2 + 2 + 2 + 3 + 1 + 1) = 7.5
(C, D, E, F, A)

FDCEA is the best remaining order for a true cost of 10. The in-out heuristic like the one-tree bound in computed in linear time when the edge lengths are previously sorted.

Experiments, at this time, have been carried out using a hybrid heuristic function and dynamic weighting:

f(x) = g(x) + w(x) *
if h_1 (x) > h_2 (x) then h_1 (x) else h_2(x);
where h_1 is the in-out estimator and h_2 is the one-tree estimator. The experiments on a 20 node graph of known difficulty (Croes 1958) showed clearly that dynamic weighting found near optimal solutions relatively cheaply when compared to ordinary branch-and-bound using the same heuristic estimator (see table 2).

Additional Computational Techniques for Increasing Search Efficiency

The candidate set is kept as a sorted binary tree, where merit (left son) < merit (parent) < merit (right son). The next node to be expanded is always the left-most node of this sorted candidate set. New nodes are entered into the tree in approximately $\log_2$ (m) time, where m is the size of the candidate set.

The bound on the tour is initially the best nearest-neighbor tour. Such a nearest-neighbor tour is generated using each node as a starting subtour. During the execution of HPA this may be updated by a tour found in the course of expanding depth - n- 5 search nodes

The most important savings are obtained from preconditioning the edge lengths. The edges are sorted by length which requires

0 (n2 $\log_2$ n) steps. Furthermore, the edges are stored in completely sorted order and in sorted order for each node.

1 rsortj points at a triple (j, k, m) and means the edge (j, k) of length m is the ith smallest edge in the graph

2 order (j, i) is an integer k where edge (j, k) is the ith smallest edge originating at node j.

The one-tree and in-out computations only require linear time when the edges are already sorted by length. This reduces these computations from quadratic time; at the expense of a single (n $\log_2$ n) computation and extra storage for the sorted lists. In all cases an overall saving will occur because at least

$0(n^9)$ heuristic functions need to be calculated.

Summary

An important guiding principle in heuristic programming is the avoidance of catastrophe. This principle overshadows its competitor--the pursuit of the optimal solution—as a pragmatic guideline in large search spaces. The whole notion of min-max rationality in game theory is an explicit axiomatization of this conservative rule.

A dynamic weighting approach is proving computationally efficient in obtaining near optimal solutions to the Traveling Salesman Problem. It offers a highly depth first search along with guarantees of avoiding type 2 catastrophes.

References

1. Lawler, E.L. and D.E. Wood (1966): Branch-and-Bound Methods; A Survey, Operations Research v.14, no.4, pp.699-719, July-August 1966.

2. Lin, S. (1965): Computer Solution of the Traveling Salesman Problem, Bell System Technical Journal, v.44 no.10 pp.2245-2269, December 1965.

3. Pohl, I, (1969): Bi-directional and Heuristic Search in Path Problem, SLAC report no.104, May 1969.

4. *F*ohl, 1. (1970): Heuristic Search Viewed as Path Finding in a Graph, Artificial Intelligence v.l pp.193-204, 1970.

5. Hart, P., Nilsson, N. and Raphael, B., (1968): A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Trans. System Sci. Cybernetics v.4 no.2 pp.100-107, July 1968.

6. Kowalski, R. (1969): Search Strategies for Theorem-Proving, in Machine Intelligence v.5 pp.181-202, 1969.

7. Michie, D., Ross, R. and G.J. Shannon (1972) G-deduction, in Machine Intelligence v.7 pp.141-165, 1972.

8. Kowalski, R., (1972): And-or Graphs, Theorem Proving Graphs and Bi-directional Search, Edinburgh School of Artificial Intelligence Memo no.56, July 1972.

9. Slagle, J. (1961): A Computer Program for Solving Problems in Freshman Calculus (SAINT), Lincoln Laboratory Report 5G-001, May 1961.

IO. Minsky, M. (1961): Steps Toward Artificial Intelligence, Proc. IRE 49, pp.8-30, January 1961.

11. Held, M. and Karp, R. (1971): The Traveling Salesman Problem and Minimum Spanning Trees, Operations Research v.19, 1971.

12. Little, J., Murty, K., Sweeney, D. and C. Karel (1963): An Algorithm for the Traveling Salesman Problem, Operations Research, v.ll pp.972-989, 1963.

13. Croes, G. (1958): A Method for Solving Traveling Salesman Problems, Operation Research v.6 pp.791-812 (1958).

14. Camerini, P.M., Fratta L., F. Maffioli (1973): A Heuristically Guided Algorithm for the Traveling Salesman Problem, Instituto de Elettrotechnica *ed* Elettronica del Politecnico de Milano, Memo 73-1, 1973.

Appendix

The Heuristic Path Algorithm - HPA -

A problem space is a locally finite directed graph G.

G: X - (x, , x2, x3, . .), X is the set of nodes and can be infinite

E: ^(x_1, Xj) I *xi* ,Xj ϵ X, Xj ϵ T (x_i)},

E is the set of edges and can be infinite but locally must be finite, i.e. | Γ (x.)

I ϵ N - the integers.

Γ is the successor mapping

f : X -> 2^X, the mapping of X into its power set.

In using directed graphs to specify problem domains, the nodes are interpreted as problem states and the edges are operators for transforming states. In the traveling salesman problem, a node may specify a sub-tour of k cities, with the operators being the addition of a city not already contained in the sub-tour. In game playing the edges are legal moves, and in theorem proving allowable inferences.

A problem consists of finding a terminal (solution) node given some starting node. The solution would be the explicit terminal node together with the path from the starting node. In certain cases the complete search tree generated in finding the solution path is a proof that solution path is optimal. An algorithm conducting such a search succeeds, if within the limits of its computational resources, It reaches a terminal node.

Heuristic Path Algorithm - HPA

s = start node, t = terminal node, x = any node

g: x -> R^+ (non-negative reals), the cost-to-date of expanding a node x from s

h: x -> R , an estimate of the remaining cost to a terminal node t - heuristic term

f(x) - (l-w)g(x)+w-h(x) , O w l

ϵ ≤

- evaluation function

For each edge in the graph, there is an associated non-negative real, its cost or length c_{ij}.

S = set of nodes already visited and expanded.

$\tilde{S}$ - set of nodes one edge away from nodes in S, but not themselves in S--the candidate set.

1. Place s in S and calculate Γ (s), placing them in S. If $x \in \Gamma(s)$, then $g(x) = C_{sx}$, i.e. the cost of going from s to x.

2. Select $n \in \tilde{S}$ such that $f(n)$ is a minimum

3. Place n in S and Γ (n) in S, discarding any nodes already in $S \cup \tilde{S}$. Calculate f for each new member of $\tilde{S}$. If $x \in \Gamma(n)$, then $g(x) = g(n) + C_{nx}$.*

4. If n is the goal state, then halt otherwise go to step 2.

Theorem: HPA with dynamic weighting function, $f(x) = g(x) + w(x)h(x)$, where 1) h(x) is a lower bound on remaining cost, 2) $w(x) = (1 + e - e * \frac{depth\ (x)}{n})$, $0 \leq e < 1$, is guaranteed to find a solution within (1 + e) of the optimum.

Proof: Note, when e = 0 we have the Hart, Nilsson, Raphael theorem.

Let m be the tour length found by HPA using a given e. Let L be the length of an optimal tour. We must show $L(1 + e) > m$.

Assume m is not the optimal length and that node x is the leaf node, i.e. in set S, which represented a subtour of the optimal tour. Then $f(x) \leq m$, otherwise HPA would not have quit.

$f(x) = g(x) + (1 + e - e * \frac{depth(x)}{n})\ h(x)$

$f(x) < g(x) + (1+e) * h(x)$

$h(x) \leq h_p(x)$ where h_p is the true remaining cost

$f(x) < g(x) + (1+e) * h_p(x)$

$g(x) + h_p(x) + e * h_p(x) = L + e * h_p(x)$*

$L + e * h_p(x) < L(1+e)$

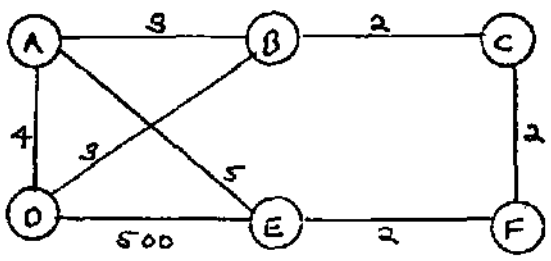
so $m \leq f(x) < L(1+e)$, hence m is within (1+e) of the optimum.

*Admissibility is retained only when $g(n) = g_p(n)$, otherwise one must retain the minimum length path from s to n.

*$g(x) + h_p(x) = L$ (see theorem 6, in reference 4

FIGURE 1.

Type 2 Catastrophe



Nearest neighbor rule gives the tour A B C F E D A, length 513 instead of A E F C B D A, length 18.

FIGURE 2.

15 Puzzle Searches by HPA

A7 (designation in Pohl 1969)

1	4	2	3
6	5	8	11
14	9	12	15
10	13	7	

starting configuration

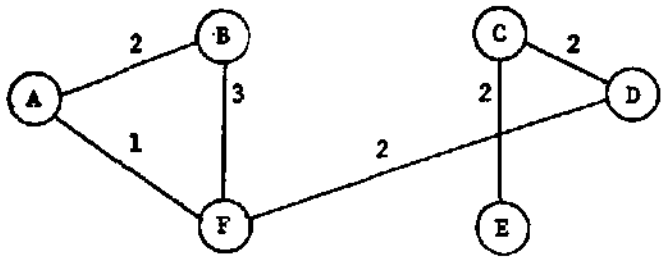
w	nodes expanded	path length	
0.5	1000	--	gives up
0.75	317	38	
0.89	431	62	
0.94	279	62	
1.00	514	80	

h (x) is the "P (x)" function described in Doran and Michie 1966 or Nilsson 1971 pg. 66.

The minimal length solution is 32.

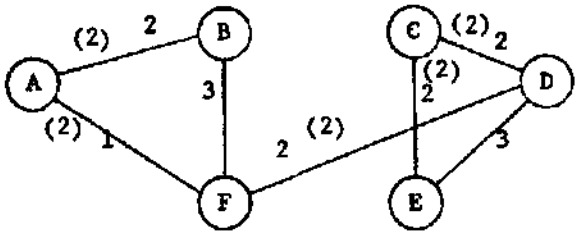
FIGURE 3.

Various bounds to the graph presented in the text.



minimum 1-tree solution

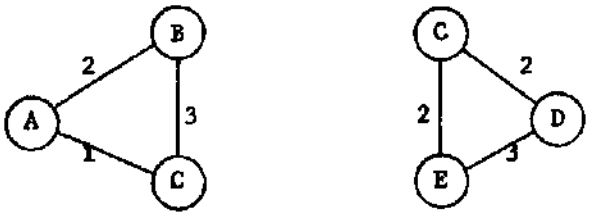
cost 12



minimum in-out solution

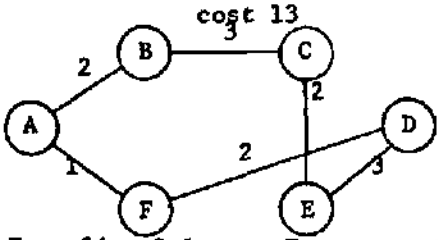
cost 12

(numbers in parentheses refer to how many times an edge is chosen by this estimator).



assignment problem solution

cost 13



Traveling Salesman Tour - cost 13

TABLE I

Various Algorithms Classified in HPA Terms

Weight in HPA terms	Algorithm
$w = 0$	Moore uniform edge lengths Dijkstra arbitrary edge lengths
$w = 1$	Doran-Michie Graph Traverser
$w = \frac{1}{2}$	H. N. R. A Branch-and-Bound algorithm - when h is a lower bound
$0 \leq w \leq 1$	Poh1 HPA
$\begin{cases} 0 \leq e \leq 1 \\ f(x) = g(x) + (1 + e - e * \frac{depth}{n}) h(x) \end{cases}$	Poh1 Dynamic weighting

TABLE II

Results of HPA

Croes graph (20 nodes)	knowi optimal length 246
Best solution from 20 different nearest-neighbor solutions	length 308
HPA with dynamic weighting e = 0.6	length 260 nodes expanded 53
HPA with dynamic weighting e = 0.4	nodes expanded 474 length 253
HPA without dynamic weighting (same as A* or branch-and-bound)	nodes expanded 500 - gives up
Randomly generated complete graph (32 nodes)	
Best solution from 32 different nearest-neighbor solutions	length 270.09
HPA with dynamic weighting e - 0.6	length 217.34 nodes expanded 71