AN IMPROVED BI-DIRECTIONAL HEURISTIC SEARCH ALGORITHM

Dennis de Champeaux & Lenie Sint
Instituut voor Bedrijfseconomie en Accountancy
Universiteit van Amsterdam/Netherlands
Mirror Cy; 1974 December; Revised 1975 June

0. Abstract and Keywords

A modification of Pohl's bi-directional heuristic search algorithm is desribed together with a simplified implementation. Theorems are proved about conditions yielding shortest paths. The results are given of a worst case analysis of different algorithms suggesting a rankorder of their quality. Results that Pohl had obtained with a uni-directional heuristic search algorithm on the 15-puzzle are compared with the results obtained by the new -simplified- algorithm.

keywords: artificial intelligence, bi-directional neuristic search, front to front guiding, path finding.

1. Introduction

In this paper we limit ourselves to those problems
a) which are representable in the state.-operator-state,-, paradigm;
b) where the goal can be explicitly represented as a state in a set of states;
c) whose problem space can be described as a labelled graph;
d) where it is equally possible to work from the start state to the goal state as the other way round;
e) where there is a heuristic function available which can be used to guide the search process.

As is well known, see e.g. / 1 / , a heuristic function can be used to guide uni-directional search. The property of ending with a shortest path between start state and goal state, as can be found with breadth first search without heuristic function, can even be preserved if the heuristic function is a lower bound *on* the real minimum effort to be made. In */2/* and /3/ an algorithm is described which generalizes to the bi-directional case. This algorithm in fact performs two independent uni-directional searches, a forward search guided to the goal node, and a backward search guided to the start node. The disadvantage of this is, that in a search space where more than one path exists *from* the start node to the goal node, the two searches often proceed along two different paths, so that the two sets of closed nodes (for the terminology of 'closed', 'nodes' and other technical jargon see /1/) grow into nearly complete uni-directional trees before intersecting each other, instead of meeting in the 'middle' of the space.

In the next section we discuss another algorithm that remedies this deficiency and we give a simplification of that algorithm which has been implemented in a Fortran program. In section 3 we compare our results with the results of Pohl's uni-directional heuristic search on the 15-puzzle. In section 4 we mention some possible improvements of the implemented algorithm.

2.0 Bi-directional, Heuristic, Front-to-Front Algorithm (BHFFA)

Before going into a precise description of the BHFFA, we give an intuitive sketch. Suppose we have the situation as in fig.1, where S and T are the sets of closed nodes and 3 and T are the two fronts of open nodes, and we decide to expand a node from S. In Pohl's BHPA algorithm a node was chosen in S which had a minimum value of the function $g_s + h_s$ where g_s was the current mimimum to the start node & and h_s was an estimator of the distance from the node to the goal node t. The h_s we use is different, h_s is a minimum of $H+g_s$ over every node in the opposite front T, where g_s is like g_s but with respect to the goal node t and H is an estimator of the shortest distance between arbitrary pairs of nodes. Immediately the disadvantage of this algorithm with respect to BHPA must be clear, since the calculation of the function h_s in our algorithm is much more complicated than the calculation of the distance to t in BHPA. On the other hand there are gains, but we defer discussing them to the next section.

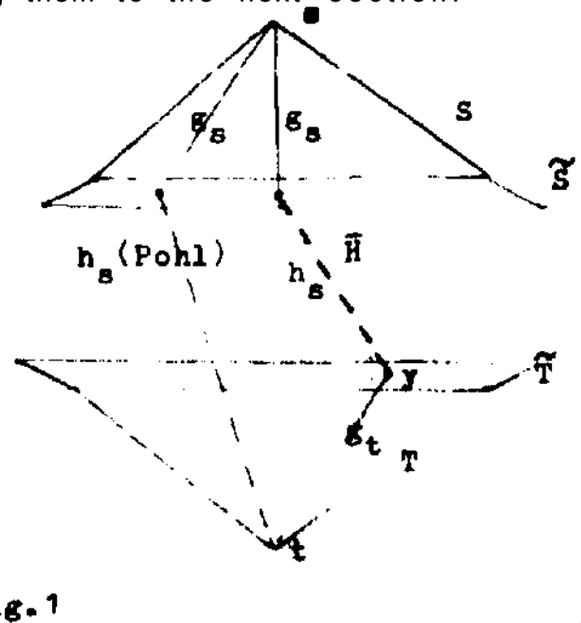

fig.1
The dashed line distances are estimated with a heuristic function.
... $=h_s$(BHFFA).

In order to describe BHFFA we have to give some definitions, where we follow as close as possible the terminology of /3/.
- s is the start node;
- t is the goal node;
- S is the collection of nodes reached from s with known f_s-value;
- T is the collection of nodes reached from t with known f_t-value;
- $\bar{S}$ is the collection of nodes, not in S, but direct successors of nodes in S;
- $\bar{T}$ is the collection of nodes, not in T, but direct successors of nodes in T;

- $H(x,y)$ is the minimum distance between node x and node y;
- $\bar{H}(x,y)$ is an estimator of the distance between x and y with $\bar{H}(x,y)=\bar{H}(y,x)$;
- $g_s(y)$ is the minimum distance between s and y for $y\in S\cup\tilde{S}$, with path in $S\cup\tilde{S}$;
- $g_t(y)$ is like $g_s(y)$ with respect to t and $T\cup\tilde{T}$;
- $h_s(n) = \min_{y\in\tilde{T}} (\bar{H}(n,y) + g_t(y))$;
- $h_t(n) = \min_{y\in\tilde{S}} (\bar{H}(n,y) + g_s(y))$;
- $f_s(x) = g_s(x) + h_s(x)$;
- $f_t(x) = g_t(x) + h_t(x)$;
- $\Gamma(x)$ is the finite set of nodes obtainable by applicable operators on x;
- $\Gamma^{-1}(x)$ is like $\Gamma(x)$, but with inverse operators instead;
- $\ell(n,x)$ is the edge length between n and x.

We now define BHFFA:

1) Place s in $\tilde{S}$, and t in $\tilde{T}$, with $f_s(s):=f_t(t):=\bar{H}(s,t)$.
2) If $\tilde{S}\cup\tilde{T}=\emptyset$ then stop without a solution, else decide to go forward, step 3, or backward, step 10.
3) Select n in $\tilde{S}$ with $f_s(n) = \min_{y\in\tilde{S}} f_s(y)$, remove n from $\tilde{S}$ and put n in S, let descendants(n):=$\Gamma(n)$.
4) If $n\in\tilde{T}$ then halt with a solution path.
5) If descendants(n)=$\emptyset$ then go to step 2.
6) Let $x\in$ descendants(n) and remove x from it.
7) If $x\in\tilde{S}$ then [if $g_s(n)+\ell(n,x)<g_s(x)$ then $g_s(x):=g_s(n)+\ell(n,x)$; if $g_s(x)+h_s(x)<f_s(x)$ then $f_s(x):=g_s(x)+h_s(x)$; go to step 5].
8) If $x\in S$ then [if $g_s(n)+\ell(n,x)<g_s(x)$ then $g_s(x):=g_s(n)+\ell(n,x)$; if $g_s(x)+h_s(x)<f_s(x)$ then {$f_s(x):=g_s(x)+h_s(x)$; remove x from S and put x in $\tilde{S}$}; go to step 5].
9) Put x with its value $f_s(x)$ in $\tilde{S}$ and go to step 5.
10) Do the same as step 3 upto 9 with $(s,S,\tilde{S},\Gamma)$ replaced by $(t,T,\tilde{T},\Gamma^{-1})$.

In step 2 nothing is said about the decision to go forward or backward, As investigated by Pohl, the most promising procedure is to count the number of nodes in S and T and to select the front which has the fewest (but at least one). Benidea the f-valuo the g-value needs to be stored at each node, as the g-value might be updated in step 7 and 6 of BHFFA.

2.1 Some theorems about BHFFA

We give some theorems and proofs about BHFFA which parallel the theorems and proofs about the unidirectional A* algorithms of /1/.
In this section we also formulate another bi-directional algorithm BHFFA2 which is interesting from a theoretical viewpoint and which we compare with BHFFA in section 2.2.

Theorem 1

If $\bar{H}(x,y)\leq H(x,y)$ and all edge-labels are not less than some positive δ then BHFFA halts with a shortest path between s and t (provided there is one).

Proof 1: As in the unidirectional case we first prove a lemma.

Lemma 2 If $\bar{H}(x,y)\leq H(x,y)$ then for every iteration of BHFFA and for every optimal path P from s to t there exist open nodes $n\in\tilde{S}$, $m\in\tilde{T}$, on P with $f_s(n)\leq H(s,t)$ and $f_t(m)\leq H(s,t)$.

Proof 2: Let n be the first node on P, counted from s, with $n\in\tilde{S}$. Let m be the first node on P, counted from t, with $m\in\tilde{T}$ (they exist because otherwise BHFFA had already halted).

$f_s(n) = g_s(n)+h_s(n)$
$= g_s(n)+\bar{H}(n,y)+g_t(y)$ for some $y\in\tilde{T}$
$\leq g_s(n)+\bar{H}(n,m)+g_t(m)$ by definition of h_s
$\leq g_s(n)+H(n,m)+g_t(m)$
$= H(s,t)$ since we are on an optimal path.

$f_t(m)\leq H(s,t)$ is proven in the same way.

Now suppose theorem 1 isn't true. Then we have three cases:
1) BHFFA doesn't halt;
2) BHFFA halts without a solution path;
3) BHFFA halts without a shortest path.

Case 1: Let P be an optimal path from s to t. According to lemma 2 there exists always an open node n in SuT on P with f (n) or $f(n)<H(s,t)$. Therefore the nodes expanded must nave an f-value less or equal to H(s,t). Consequently their g-values are less or equal to H(s,t). Thus BHFFA only expands nodes at most H(s,t)/6 steps away from s or t, and this is a finite number. Let M_s and M_t be the sets of all nodes which are ever generated from s and t respectively. As every node has only a finite number of successors and as the maximum number of steps any node is away from s or t is finite, both M_s and M_t can only contain a finite nuaber of nodes, ana so $M=M_s\cup M_t$ is of finite size. Let the number of nodes in M be v. Let p be the (necessarily finite) number of different paths from s to m if $m\in M_s$, and from t to m if $m\in M_t$, and let p be the maximum over all p. Then p is the maximum number of different times a node can be reopened. After p.v iterations of BHFFA all nodes of M are permanently closed. So SUT=# and BHFFA halts, which produces a contradiction.

Case 2: We have just proved that BHFFA eventually halts, and it can only do so for two reasons: It has found a solution path, or SuT is empty. If the latter is the case, then the last node expanded had no successors at all, (otherwise they would have been placed in 5 or T). But this means that there is no path from s to t, contradictory to assumption. So BHFFA halts because it has found a solution path.

Case 3: Just before ending with a node m, there would, by lemma 2, be a node n in S with $f(n)\leq H(s,t)<f(m)=g_s(m)+g(n)$ and thus n woull be chosen for expansion instead of m. .

q.e.d.l

The next theorem proved in /1/ is the optiraality theorem, which states that if two heuristics, H and H*, are related by $H^*(n,t)<H(n,t)*H(n,t)$ for all n, and if for both heuristics the consistency property holds, (meaning that $H(n,t)+H\{n,x)<$ $H(x,t)$) then every node expanded by H will also be expanded by II . This theorem doesn't hold for BHFFA. The reason for that is, that, unlike in the uni-directional and in Pohl's bi-directional algorithm, the f-value of an open node is not static in BHFFA, and the way in which it changes depends on both the exact form of the heuristic and the real distances between the nodes in the opposite front; and without assuming some detailed information about those, the exact behaviour of any heuristic with BHFFA is hard to predict. We haven't succeeded in finding a general proof that, if one heuristic is better (in the above sense) than an other, it will always finish in less iterations. We don't consider this a great loss, because in most cases it will nevertheless be true. {Something, for example, which can be proved is, that if there are two heuristics, $\bar{H} = H.\delta$ and $\bar{H}^* = H.\varepsilon$ and $0\leq\varepsilon<\delta\leq 1$, then H will finish in less iterations).

Now we describe another bi-directional algorithm, BHFFA2, because it can be easily implemented and seems to be elegant. The worst case error analysis, the results of which are described in the next section, however, suggests that the number of nodes expanded explodes faster. The notation of BHFFA2 follows more closely the terminology of /1/.

BHFFA2

1) Place $\{(s,t,0,0,\bar{H}(s,t))\}$ in OPEN.
2) If OPEN=$\emptyset$ then halt without a solution.
3) Select $n=(x,y,v1,v2,v3)$ in OPEN with minimal $v1+v2+v3$;
   OPEN:=OPEN-{n}; CLOSED:=CLOSED$\cup${n};
   descendants(n):=$\Gamma(x)x\Gamma^{-1}(x)$.
4) If x=y or $y\in\Gamma(x)$ then halt with a solution path.
5) If descendants(n)=$\emptyset$ then go to step 2.
6) Select $m=(m1,m2)$ in descendants(n) and remove m from descendants(n).
7) If $m\in$CLOSED which means $m1=z1$, $m2=z2$, and $(z1,z2,g_s(z1),g_t(z2),\bar{H}(z1,z2))$ is in CLOSED then
   [if $g_s(z1)>g_s(x)+\ell(x,m)$ and/or $g_t(z2)>g_t(y)+\ell(y,m2)$ then
   {update $g_s(z1)$ and/or $g_t(z2)$;
   place $(z1,z2,g_s(z1),g_t(z2),H(z1,z2))$ in OPEN};
   go to step 5].
8) If $m\in$OPEN then
   [do updating of $g_s(z1)$ and/or $g_t(z2)$ as in step 7 if necessary;
   go to step 5].
9) $g_s(m1):=g_s(x)+\ell(m1,x)$; $g_t(m2):=g_t(y)+\ell(m2,y)$;
   OPEN:=OPEN$\cup$ $\{(m1,m2,g_s(m1),g_t(m2),H(m1,m2))\}$;
   go to step 5.

Theorem 3

The lemmas and theorems proved in /1/ for the uni-directional A* algorithm, including the optimality theorem, also hold for BHFFA2.

Proof 3: BHFFA2 is in fact the uni-directional A algorithm in the product space of the nodes with itself. q.e.d.3.

2.2 Worst case analysis

A first order comparison of these algorithms may be done by investigating how they behave in worst case situations. We have derived formulas for the uni-directional, Pohl's bi-directional, BHFFA and BHFFA2 algorithms, assuming that the heuristic function used gives a maximum error between relative bounds. We will summarize the results here, an expanded version of this paper containing their derivations will be sent on request to readers interested in them. Let the search space be a countable collection of nodes, with two nodes, the start and the goal node, having m edges (m>1), while from all the other nodes emanate m+1 edges; and there are no cycles. Let all the edge-lengths be equal to 1, and let there be a path of length K between start node and goal node. From a uni-directional viewpoint this space is a tree with branching-rate m, since the algorithm will not look beyond the goal node. In this space the following results are obtained:

1. Let R be the real distance from some node on the solution path to the goal node in the uni-directional case, to the start or goal node in Pohl's bi-directional case, and to the other front in the BHFFA cases. If for each node on the solution path nodes off the solution path are expanded until some depth j, and if this

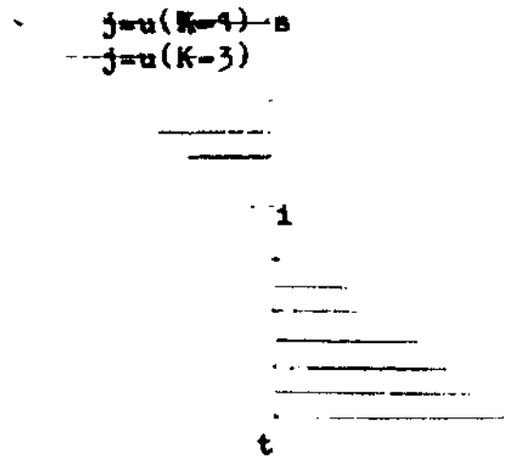

figure 2, BHFFA.
R decreases with steps of 2.

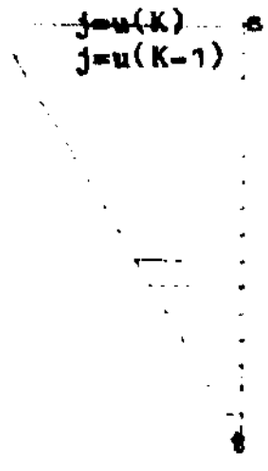

figure 3, uni-directional.
R decreases with steps of 1.

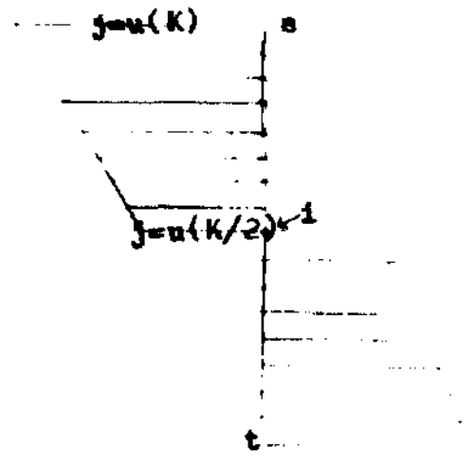

figure 4, bi-directional Pohl.
As at i R is still K/2,
j never gets very small.

j is a monotonically decreasing function of R, then, no matter what the heuristic exactly looks like,

$\mathcal{N}_{BHFFA} < \mathcal{N}_{uni\text{-}directional} < \mathcal{N}_{bi\text{-}directional\ Pohl}$;

where $\mathcal{N}_A$ denotes the number of nodes expanded by algorithm A.

That this indeed must be the case can be understood intuitively by looking at the figures 2, 3 and 4, where the length of a bar represents the depth to which nodes off the solution path are expanded. The total length of all the bars increases from figure 2 to figure 4.

2. Suppose that for two nodes both lying on the solution path holds that $\bar{H}(n,m)=H(n,m).(1+\delta)$ forsome $\delta>0$, and else (at least one of the two lying off the path) $\bar{H}(n,m)=H(n,m)/(1+\delta)$. Then δ is a relative error bound and this is the worst possible case. Now for the BHFFA, the uni-directional algorithm and Pohl's bi-direc,tional algorithm nodes off the solution path will be expanded until a depth $j=6.R + \delta.(\delta+1)/(\delta+2)$.
So here the monotone condition holds with the result stated above. Furthermore we give exact formulas for the number of nodes expanded by the different algorithms in this worst case:

$$\mathcal{N}_{BHFFA} = 2.m^{\delta-c_\delta}.\frac{m^{K\delta}-1}{m^{2\delta}-1};$$

$$\mathcal{N}_{uni\text{-}directional} = m^{\delta-c_\delta}.\frac{m^{K\delta}-1}{m^{\delta}-1};$$

$$\mathcal{N}_{bi\text{-}directional\ Pohl}=2m.m^{(K/2).\delta-c_\delta}.\frac{m^{(K/2).\delta}-1}{m^{\delta}-1};$$

where m is the branching-rate, K is the solution path length, and $c_\delta=\delta.(\delta+1)/(\delta+2)$.

3. The BHFFA2 case is less easy to handle, the way it ranks in the series depends on m and on K. The analysis, however, shows that with increasing m and K BHFFA becomes the better algorithm in the long run. The exact formula for the number of nodes expanded is very complicated and won't be given here, a reasonable estimate is given by

$$BHFFA2 = m^{2(\delta-c_\delta)}.\frac{m^{2K\delta}-1}{m^{2\delta}-1};$$ this is the uni-directional formula with the solution path length halved and the branching rate squared. In general it underestimates the number of nodes expanded by BHFFA2.

3. Experiments with BHFFA

3.1 The program

A modified BHFFA algorithm is implemented as a Fortran program, geared toward solving the 15-puzzle. In this search-space all edgelengths are taken as unity. The most important restriction made is that if, after having expanded 1000 nodes, the program still has not found a solution path, it gives up. Another important modification is that the number of open nodes in a front cannot exceed some maximum m, which is given to the program as an input parameter but must be less 100. Whenever an (m+1)-th node should be added to a front, first the open node with the current worst f—value is deleted from it -unless the node itself has an f-value even worse, in which case it won't be inserted. We call thin operation "pruning", a term also used in /1/. Thin step is mainly necessary to save time, as the number of comparisons needed to calculate h (x) for a node x in S, is equal to the number of nodes in T and vice versa. But this also means that the algorithm is not admissable anymore, (an algorithm satisfying theorem 1 is called admissible), because it is possible that some node on the optimal path will be thrown away because it looked very bad at some iteration. In a search space where only one path exists from start, to goal node, some backtracking mechanism would be required to ascertain that this path is found.
In the case of the 15-puzzle, the actual influence didn't appear to be very large if m was set at 50 (or larger), as will be seen in section 3-2-
Step 8 of BHFF'A, where occurrence of a new node in the collection of closed nodes of it's own front is checked was eliminated. We did this because we thought the time gained in possibly expanding a few nodes less would not balance the loss caused by searching through the set of closed nodes for every new open node, whereas the solution path found will most likely not be influenced. Step 9 was done by estimating all distances to the opposite front, and inserting node x in one of the ordered fronts of open nodes, where the ordering is given by the f-values of the open nodes. A nasty side-effect was that the insertion of a new node in S could imply a reordering of T and vice versa. The ordering was done using a square matrix in which all combinations of the H(x,y)-values of the fronts were stored. Step 4 {the terminating condition) was eliminated and replaced by a test in step 6: 'If x'T'UT then halt with a solution path'. The testing of x being in $\bar{T}\cup T$ instead of just being in T is a necessary consequence *of* the pruning, as it is possible that a descendant of a closed node is deleted from the open front.
Three heuristic functions were implemented in the program:

1. $P(a,b)=\sum_i p_i$, with p_i being the Manhattan distance between the position of tile i in a and in b;
2. $S(a,b)=\sum_i p_i^2 . h_i^{.5}$, where p_i is as in 1 and h_i is the distance in a, of tile i to the empty square;
3. R(a,b) is the number of reversals in a, with respect to b, where a reversal has the meaning that a(i)=b(j) and a(j)=b(i), and i and j are adjacent tiles.

Of these functions the first two originally come from /4/, whereas the third comes from /2/.

3.2 Results, conclusions and remarks

In order to compare our results with the uni-directional case, the program was run with the same 15-puzzle problems as were used by Pohl in /2/, using the same heuristic functions with the same -values:
f1 = g + ω.P with ω= 1,2,3,4,8,16,∞
f2 = g + ω.(P+20.R) withω =1,2,3,4,8,16,∞
f3 = g + ω.S with = .5, .75, 1, 1.5, 2,3,4,16,∞

Table 1.

	shortest path found for each problem.		average pathlength		average number of nodes expanded (only over the solved problems)	
problem	Pohl	our	Pohl	our	Pohl	our
A1	12	12	12	12	12.6	12.9
A2	26	26	42.8	27.3	161.5	54.4
A3	36	34	64.8	44.0	389.1	320.9
A4	20	20	21.8	24.3	90.2	108.6
A5	38	32	56.7	39.7	310.4	250.1
A6	32	32	42.8	37.1	335.7	333.4
A7	36	36	56.6	53.7	365.4	429.3
A8	85	61	132.0	93.0	605.6	485.9
A9	86	88	152.9	118.5	551.2	563.5
A10	64	60	92.3	90.0	609.3	532.3

Table 2.

	number of problems solved.		score for pathlength		score for number of nodes expanded	
	Pohl	our	Pohl	our	Pohl	our
f1	36	47	22	42	20	37
f2	51	60	30	48	33	28
f3	36	45	32	39	31	34
f4	80	88	23	82	31	58
sum	203	240	107	211	115	157

f4 = g + ω.(S + 20R) with ω = .5, .75, 1, 1.5, 2, 3, 4, 16, ∞

(which in fact means $f1(x)=g_s(x) + \min_{y \in T}(P(x,y)+g_t(y))$ for x in S; etc.).

As there were ten *di* frerent 15-puzzles, this amounts to a total of 320 problems, of which our program solved 240, whereas the uni-directional program of Pohl solved 203 of them, it can be seen that in nearly all these cases the heuristic is not a lower bound on the real effort to be made. This is the main reason why many of the solutions found are not optimal, both for Pohl's program and ours.

Tables 1 and *2* summarize the results, the full results on all ten problems may be found in the expanded version of this paper already mentioned in section *P.P.* In table ?, where the performance of the different functions is compared, the score for pathlength was obtained as follows: The program with the shortest path for some problem scored 1 and the other 0, (and any path is counted shorter than no path at all); if the same pathlength was found by both programs they both scored 1; if a problem was not solved by either *of* the programs they both scored 0. The score for the number of nodes expanded was obtained in a similar way. In so far the solution quality is concerned, BHFFA is an improvement over the uni-directional algorithm: it solves more problems, finds in general shorter paths, and expands less nodes on the average, although the last effect is less prominent than we expected. BHFFA performs particularly well with a strong heuristic function; with f4 the total number of nodes expanded by our program was 32% less than that by Pohl's program. The frontlength adequate for the problems was found empirically. Experimental runs were made with frontlengths 25, *32* and 50. An increasing number *of* problems was solved and a higher stability was reached. (By stability we mean the chance that a longer frontlength preserves a solution obtained with a shorter frontlength; pruning tricks are the obstructing force here).

As could be expected, the performance with respect to the frontlength depends on both the solution path length and the heuristic used: The better the estimator, the smaller the frontlength required (a length of 1 would suffice for a perfect heuristic). All problems were run with a frontlength of 50, and the least satisfactory solved were run again with a frontlength of 99, in order to see whether the maximum number of 1000 expanded nodes or the pruning in the fronts was the bottleneck. In general the first seems to bo the case, since no significant improvement was made, with the exception of f1 on A9 where six instead of one out of 3even problems was solved. The main disadvantage of Pohl's bi-directional algorithm, as mentioned in section 1, appeared to be remedied. The fronts now did meet near the middle of the search-space, which we could see by comparing g and g_t of the intersection nodes.

The large disadvantage of our algorithm is the very time-consuming calculation of the distance estimator. With a frontlength of 50, I would expect the BHFFA-program to take in the average about 35 times longer to obtain a solution (for the same problem with the same heuristic) than a uni-directional program, run on the same machine. In general, the loss of efficiency will not be sufficiently set-off by the shorter paths found. Nevertheless, it may pay off in, for example, an ABSTRIPS-like environment (see /5/), where it is crucial to find an optimal path from among many different existing paths, as the number of sub-problem searches depends on the path-length found in the dominating problem-space. There are BHFFA or a similar algorithm (in the next section we will suggest ways to make it cheaper), with a strong heuristic function, may find an optimal path more efficiently than a uni-directional program with a heuristic satisfying the lower bound condition, because this kind of heuristic tends to be rather weak and results in a fast explosion of the number of nodes expanded.

4. Open problems and loose ends.

BHFFA can be simplified by not calculating the heuristic distance to every node in the opposite front but only to the best half or even leas of them. This idea is inspired by the fact that, in the limited number of cases where we checked it, a node realised its minimum nearly always to a node which belonged to the best ten of its own front. A further simplification would be to delete the resequencing of the opposite front as the consequence of adding a node to a front. The sensitivity of the solution quality to these computation time and memory savings should be tested.

BHFFA? needs implementation to be able to compare its performance with that of BHFFA.

Looping as a consequence of the pruning can be recognized but which rescue program should be started then is unclear.

The partly expanded node technique, as suggested in /6/, needs investigating.

A loss technical question however concerns the selection of the most interesting ones among the vast amount of pot.cutinl macro-operator sequences that appear in a solution.path.

But the real A.I.-question is still:
How can the program improve its heuristic function beyond simply optimizing some coefficients.
Heavily related to this is the question: How to find automatically the best representation for a problem to be treated by heuristic algorithms.

Acknowledgements

This research originated from a grant of the Netherlands Organization for the Advancement of Pure Research (Z.W.O.), which enabled the first author to visit U.T. in Austin, Texas, where L. Siklosky introduced him to this field. Discussions with T. Pohl on earlier drafts were highly appreciated.

References

/1/ N.J. Nilsson, Problem-solving Methods in Artificial Intelligence, 1971.
/2/ Ira Pohl, Bi-directional Heuristic Search in Path Problems, 1969.
/3/ Ira Pohl, Bi-directional Search, M.I.6, 1971.
/4/ J. Doran and D. Michie, Experiments with the Graph Traverser Program, 1966.
/5/ E.D. Sacerdoti, Planning in an Hierarchy of Abstraction Spaces, 3IJCAI, 1973.
/6/ R.S. Rosenberg, Look-Ahead and One-Person games, University of British Columbia, 1972.