A Relationship Between Classification Accuracy and Search Quality

Sharad Saxena
Paul E. Utgoff

Department of Computer and Information Science
University of Massachusetts at Amherst
Amherst, MA 01003

COINS Technical Report 88-104
December 21, 1988

Abstract

A self organizing best first search algorithm for state space search problems is presented. The algorithm permits the effective use of an imperfect preference predicate. A perfect preference predicate, given any two states (a, b), returns *true* if and only if state a should be explored before state b. The analysis of the algorithm gives a quantitative relationship between the classification accuracy of the preference predicate, the maximum allowable search effort, and the probability of obtaining a solution within the allowed search effort. The algorithm, its analysis and its applications in the design of machine learning systems, are illustrated using the eight puzzle problem.

1. Introduction

This paper presents two results. The first is an improvement to the best first search algorithm that uses a preference predicate for node selection[4]. The improvement, which is called the *self organizing best first search*, makes it practical to use an imperfect preference predicate for searching, without any increase in the computational effort required to do the search. The second, which follows from the analysis of the improvement, is a quantitative relationship among three factors: the accuracy of the preference predicate, maximum allowable search effort, and the probability of obtaining a solution within the allowed search effort. The relationship is important because the specification of any two factors makes it possible to determine the third. For example, given certain performance requirements in the form of the maximum allowable search effort and the probability of finding a solution, the accuracy required of a preference predicate can be determined. Learning a preference predicate with that accuracy can then be considered as the learning task.

The rest of the paper is organized as follows. In section 2 best first search that uses a preference predicate for node selection is reviewed and analyzed. In section 3 the self organizing best first search algorithm is presented and analyzed. The relationship between the accuracy of a preference predicate and the probability of finding a solution within the maximum allowable search effort is illustrated for the eight puzzle problem in section 4. In section 5 the experimental results obtained from using the self organizing best first search algorithm for the eight puzzle problem are presented.

2. Review of preference predicates

A *preference predicate* is used for node selection in heuristic search[4]. The fundamental principle of heuristic search is that if a state with the greatest promise of being on a path to the goal state is explored first then a solution is likely to be found with less search effort. The standard approach for comparing the promise of states is to build a *numeric* evaluation function that maps each state to a number. The states are then compared by comparing the numbers assigned to them. With a preference predicate it is possible to avoid the indirect step of mapping the states to numbers, and directly compare the relative promise of two states. In particular a perfect preference predicate $P(a,b)$ returns *true* if and only if a node containing the state a should be expanded before a node containing the state b. A heuristic evaluation function is just a particular implementation of a preference predicate, namely, $P(x,y) \iff h(x) < h(y)$, where $h(x)$ is the heuristic estimate of the cost of the minimal cost path that is constrained to go through the state x. The purpose of using the more general notion of preference predicate is that it admits other implementations, in addition to evaluation functions. In particular, it makes it possible to apply concept learning algorithms to the problem of learning state preference. For every observed example, where the state a is preferred over the state b, the ordered pairs (a,b) and (b,a) can be considered as positive and negative examples respectively, of state preference. As the preference predicate being learned becomes more accurate, the search effort required can be expected to decrease.

```
1. $open \leftarrow start\text{-}state$.
2. $closed \leftarrow empty$.
3. $current\text{-}state \leftarrow best(open)$.
4. while ( $current\text{-}state \neq goal\text{-}state$)
    (a) remove *current-state* from *open* and add it to *closed.*
    (b) generate all successors of the *current-state.*
    (c) add each successor of the *current-state* not present in *closed* to *open*, anywhere.
    (d) $current\text{-}state \leftarrow best(open)$.
```

Figure 1: Best first search.

The standard best first search algorithm is shown in figure 1. Step 4(d) indicates that the node selected for expansion is identified by *best*(*open*). If heuristic evaluation functions are used to compare states then *best* is determined by repeated pair-wise comparison of h values for the nodes, always retaining the least so far. This is the process of finding a minimum of a list of numbers. In the best first search that uses a preference predicate, *best* is determined in the same manner, $P(x, y)$ is used to determine whether to retain the state x or the state y as the best.

An important issue when using a preference predicate for best first search is: what is the impact of an imperfect preference predicate on the amount of search required to find a goal state? A perfect preference predicate is always transitive and therefore *best* is not sensitive to the order in which the nodes in *open* are compared. However, an imperfect preference predicate may not be transitive, in which case the order in which the nodes in *open* are compared becomes crucial. Notice that if numeric evaluation functions are used, because the numeric comparison operator is transitive, the order in which the nodes in *open* are compared has no effect on *best.* In such a situation the problem addressed here is equivalent to doing node selection with a intransitive numeric comparison operator.

Suppose the probability that the preference predicate is correct in any comparison is at least p. Call p the *classification accuracy* of the preference predicate. Assume that making an error in a particular comparison is independent of making an error in any other comparison. These assumptions model the case where the preference predicate has independent and random errors. If there are m nodes in the *open* list then the probability of choosing the best state is the lowest when the best state is the first element of *open.* In that case there are $m - 1$ comparisons to be made, so the probability of choosing the best node is p^{m-1}. As the best first search progresses, the size of *open* increases. As a result, the probability of selecting the best node from a frontier keeps decreasing. In the next section, a best first search algorithm is given in which the probability of having selected the best node from a frontier after each step cannot be less than a value that is independent of the

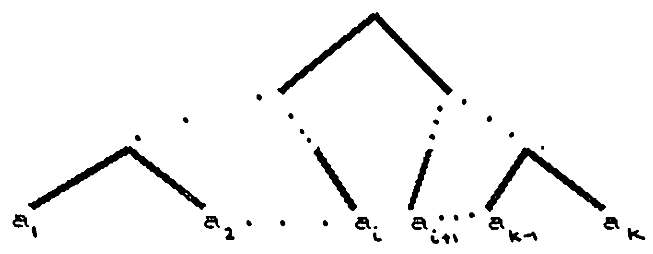

Figure 2: A typical search frontier.

number of nodes in *open*.

3. Self organizing best first search

In *self organizing linear search* after the element being searched for has been found, the elements in the search list are permuted according to a given rule. The purpose of permuting the elements of the search list is to facilitate future searches[1]. The self organizing rule used here is called *move-to-end.* If b is the best state in the open list then the successors of b are added to the *end* of *open.* The best first search algorithm using this self organizing rule is identical to the standard best first search algorithm, except that the step 4(c) in figure 1 now becomes:

4(c) add each successor of the current state not present in *closed* to the *end* of *open.*

The effect of this change on best first search is that the probability of having found the best node after each step is greater than or equal to a value that can be determined from the classification accuracy of the preference predicate and the maximum branching factor of the search.

3.1 Analysis

Suppose p is the classification accuracy of a preference predicate. As above, assume that the errors made by the preference predicate are random and independent. Let b be the maximum number of successors of any node. Recall that if the search frontier is not reorganized then, in the worst case, as the size of the search frontier grows, the probability of selecting the best node keeps decreasing. With the move-to-end reorganizing rule this does not happen. Every time the best node happens to be correctly chosen from a search frontier, the rest of the nodes in that frontier contribute only one node to the future selections of the best node. This is because the best state in the future steps of the search is one of the successors of the best state in the current step of the search. As a result, the best state in the future steps of the search occurs among the last few nodes of the search frontier.

More precisely, suppose that the current search frontier is as in figure 2. Suppose that a_i is the best node in this frontier and it happens to be correctly chosen as the best node. Let the successors of a_i be $s_1, s_2, \ldots, s_b$. By definition of best, a_i must be on an optimal

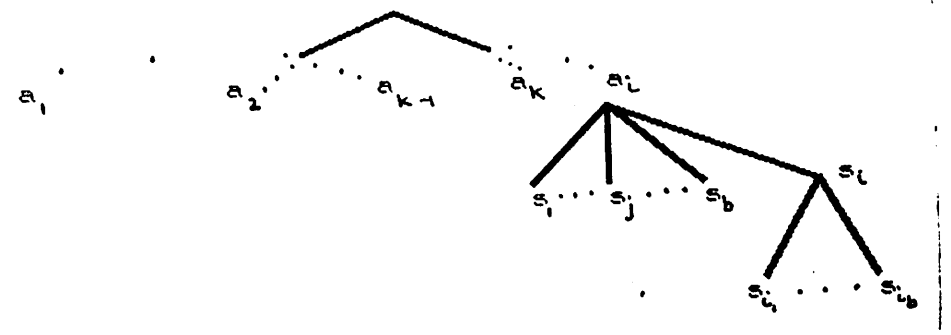

Figure 3: Frontier after successors of s_i are added to the end.

cost path to the goal state. Therefore the best state in the next step must be in the set $\{s_1, s_2, \dots, s_b\}$. The probability of choosing the best state in the next step is the lowest when s_1 is the best state. In that case, s_1 has to be chosen over $\{s_2, s_3, \dots, s_b\}$, that is, $b-1$ comparisons, and the best state from among $\{a_1, a_2, \dots, a_{i-1}, a_{i+1}, \dots, a_k\}$, for a total of b comparisons.

The analysis above shows that, after each step in which the best state happens to be chosen, the preceding nodes in that frontier contribute only one node to the future selections of the best node. But it could happen that a best state is never chosen. Notice however that for each step in which the best state happens to be chosen— because the number of nodes with which the best state is compared in the next step is small— the probability that the best is not chosen in the next step is small. The size of the set with which the best state is compared increases only if the best state is not chosen in the next step. A situation where i node expansions have occurred and the best state has not yet been chosen, happens only if the best state was not chosen in each of the $1, \dots, i-1$ steps. Denote the probability of this event by p_i. Notice that $p_i \leq p_1$.

To be precise, let A_i denote the event that i node expansions have occurred and the best node has not yet been chosen. Then $\overline{A_i}$ denotes the event that i node expansions have occurred and the best state was expanded in at least one of them. Let $P(X)$ denote the probability that an outcome represented by the event X occurs. From above observe that $P(\overline{A_1}) = p^b$ and therefore $P(A_1) = 1 - p^b$. Suppose that among $\{s_1, s_2, \dots, s_b\}$, s_1 is the best state, but s_i is chosen. In the worst case s_i can have b children. After these children are added to the end of *open* the search frontier given in figure 3 is obtained. The probability of choosing the best node from this frontier is the lowest when s_1 is the best node. In that case, s_1 has to be compared with $2b-1$ nodes, and so the probability of selecting s_1 is p^{2b-1}. The probability of not choosing s_1 from this frontier is $1 - p^{2b-1}$. In general, the probability of not having chosen a best node 2 steps after a step in which a best node was chosen is $P(A_2) = P(A_1) \times (1 - p^{2b-1})$. Therefore $P(A_2) < P(A_1)$. It follows that $P(A_2) < 1 - p^b$, and $P(\overline{A_2}) \geq p^b$. Similar analysis at each step shows that in general, $P(\overline{A_i}) \geq p^b$. That is, the probability of having chosen the best state after any comparison is at least p^b, independent of what happens in the other steps.

Consider a search problem in which the starting state is k steps away from the goal. Call k the *size* of the problem. Suppose that m steps are allowed to solve the problem. If $m = d \times k$, where d is an integer, then d is called the dilation of the search. To find the goal state, selecting the best state in any k or more of the m steps is sufficient. This is because every time a best state is chosen, the distance to the goal is reduced by 1 step. Denote the event that a problem of size k is solved within m steps by $Success$. A lower bound on $P(Success)$ can be obtained by assuming that the probability of having chosen the best state after each step is $r = p^b$. With this approximation, the search process can be modeled by m independent trials of an experiment, in which each trial of the experiment consists of searching for a best state in *open*. The outcome of each trial of the experiment is choosing a best state or failing to do so. $P(Success)$ can be estimated by the probability of the event that in k or more trials the best state is chosen. The probability that an outcome representing this event occurs is given by the tail of a binomial distribution, in which the probability of choosing the best state in each trial is r. That is,

$$P(Success) \geq \sum_{i=k}^{m} \binom{m}{i} r^i (1-r)^{m-i}. \qquad (1)$$

Each term of the above sum gives the probability of choosing the best state in exactly i of the m steps. The next section illustrates the applications of the above relationship. The eight puzzle problem is used as an example.

4. Illustrations of the relationship for the eight puzzle

Table 1 shows the probability of solving a problem of size k within m steps when the classification accuracy is p. For the eight puzzle problem b, the maximum branching factor, is 4. Table 1 has been computed by letting $m = d \times k$, where d is the dilation of the search.

Table 1 can be used by a designer of learning systems in many ways. First, for a given learning algorithm and for a known value of p— the classification accuracy of the preference predicate produced by the algorithm— the performance guarantees available from such a preference predicate can be determined from a table similar to Table 1. For example, for the eight puzzle problem, if it is known that the preference predicate produced by a given learning algorithm has the classification accuracy 0.8, and that the errors are independent and random, then the probability of solving a problem of size 23 within dilation 3 is at least 0.92.

A second potential use of table 1 is to determine the classification accuracy required of a preference predicate, given certain performance requirements. Suppose it is desired to solve any randomly chosen eight puzzle problem within dilation d, with a expected probability of at least P_n. A worst case estimate of the probability of solving a problem within a given dilation can be obtained. For example, if it is sufficient to solve all eight puzzle problems with a probability of at least 0.75, within a dilation of 3, then a preference predicate with a classification accuracy of 0.8 would suffice. This is because from table 1 with classification accuracy 0.8 and dilation 3, the probability of finding a solution is at least 0.78. Learning a preference predicate with this classification accuracy can then be considered as a learning

k	$p = 0.8$		$p = 0.9$	
	$m = 2k$	$m = 3k$	$m = 2k$	$m = 3k$
0	1.00	1.00	1.00	1.00
1	0.65	0.79	0.88	0.96
2	0.54	0.78	0.88	0.98
3	0.48	0.78	0.89	0.99
4	0.43	0.79	0.90	1.00
5	0.39	0.80	0.91	1.00
6	0.36	0.81	0.92	1.00
7	0.33	0.82	0.93	1.00
8	0.31	0.83	0.94	1.00
9	0.29	0.84	0.95	1.00
10	0.27	0.85	0.95	1.00
11	0.26	0.86	0.96	1.00
12	0.24	0.86	0.96	1.00
13	0.23	0.87	0.97	1.00
14	0.22	0.88	0.97	1.00
15	0.20	0.88	0.97	1.00
16	0.19	0.89	0.98	1.00
17	0.18	0.90	0.98	1.00
18	0.17	0.90	0.98	1.00
19	0.17	0.90	0.98	1.00
20	0.16	0.91	0.98	1.00
21	0.15	0.91	0.99	1.00
22	0.14	0.92	0.99	1.00
23	0.14	0.92	0.99	1.00
24	0.13	0.92	0.99	1.00
25	0.12	0.93	0.99	1.00
26	0.12	0.93	0.99	1.00
27	0.11	0.94	0.99	1.00
28	0.11	0.94	0.99	1.00
29	0.10	0.94	1.00	1.00
30	0.10	0.94	1.00	1.00
31	0.09	0.95	1.00	1.00

Table 1: Probability of solving a problem of size k within m steps.

	dilation	
p	$d=2$	$d=3$
0.8	0.1462	0.9169
0.9	0.9878	1.0

Table 2: Probability of solving a eight puzzle problem within dilation d.

task. If the distribution of the number of states of different problem sizes can be obtained, then better estimates of the required classification accuracy can be obtained. For example, the exact distribution of the number of states of different problem sizes for the eight puzzle problem was found in [5]. Using this distribution, table 2 was constructed by weighting the probability of obtaining a solution for a problem of given size, by the probability of selecting a problem of that size, assuming that every state of eight puzzle is equally likely to be a start state. In particular, let $p_s(k)$ be the probability of selecting a problem that is k steps away form the goal state. Let $p^n(k)$ be the probability of solving a problem of size k within dilation d. Then,

$$P_n = \sum_{k=0}^{31} p_s(k)p^n(k).$$

For example, from table 2 observe that a preference predicate of classification accuracy 0.8 is needed to solve with probability at least 0.9 any eight puzzle problem within dilation 3.

A third potential use is to determine the dilation of the search that must be tolerated, given a preference predicate of known classification accuracy, and the probability with which a solution is desired. For example, observe from table 1, that with a preference predicate of classification accuracy 0.9, dilation of 2 is sufficient if the probability of finding a solution is required to be at least 0.88. However, with the same classification accuracy, the dilation must be 3 if a solution is required with a probability of at least 0.96. Again, if the exact distribution of the number of states of various problem sizes is available, then better estimates for the dilation required can be obtained. For example, observe from table 2 that with a classification accuracy of 0.9, dilation of 2 is sufficient if the expected probability of finding the solution is required to be at least 0.91, with a dilation of 3 however, a solution is found with expected probability 1.00.

Observe in table 1 that, if the classification accuracy and the dilation are above specific values, then there is a problem size, such that for all problems of greater size, the probability of finding a solution increases as problem size increases. For example, with a dilation 3, when the classification accuracy is 0.9, the probability of finding a solution for a problem of size 3 is 0.88. The probability of finding a solution for a problem of size 31 is 1.00 for the same classification accuracy and dilation. Though this can be totally explained by substituting the values of classification accuracy, maximum branching factor and dilation in equation 1 and obtaining the appropriate probabilities, an explanation in terms of what is happening in the search is not clear. The next section reports experiments in which this theoretically predicted phenomenon was observed. Another counter-intuitive phenomenon has recently been reported in [3]. Korf observed that if the computational effort is kept fixed then the depth to which search could be conducted increases with increasing branching factor for certain branch and bound problems.

5. Experimental results from eight puzzle simulation

In this section the experimental results obtained from using the self organizing best first search for the eight puzzle are reported. The objective of the experiments was to find a sequence of eight puzzle moves that will take a given starting state to the goal state within a fixed dilation. The results for dilation 3 are reported here.

5.1 Experimental setup

A look-up table that maps each of the $\frac{9!}{2}$ possible states of the eight puzzle to its size was used [5]. Recall that the size of a problem is the minimum number of steps to the goal state. Ten problems of each possible size were selected at random to form a test set. Each problem in the set was solved by best first search using a preference predicate, without self organization, and by self organizing best first search. A preference predicate of classification accuracy p was simulated by accepting the answer of a comparison using a prefect evaluation function[5], with probability p. This technique models independent and random errors. The number of problems of each size solved within the given dilation was recorded. The process was repeated 5 times and the number of problems of each size solved within the given dilation was averaged for these 5 trials.

5.2 Results

Figure 4 gives the frequency of solving eight puzzle problems of different sizes, within dilation 3, using best first search without self organization The graph shows the results for varying classification accuracy of the preference predicate. Figure 5 gives the frequency when self organizing best first search is used. With dilation 3 and classification accuracy 0.8, self organizing best first search could solve any eight puzzle problem with probability ≥ 0.85. If the classification accuracy was 0.9, self organizing best first search could solve all eight puzzle problems in the sample within dilation 3. However without self organization most problems could not be solved with classification accuracy 0.9 and dilation 3.

The simulations showed that, as predicted, without self organization the probability of selecting the best node from a frontier keeps decreasing as the search progresses. In particular, in the simulations none of the problems of size ≥ 20 could be solved within dilation 3, for a classification accuracy of 0.9. Simulations of self organizing best first search gave the probability of solution that is higher than the theoretically predicted value. This is because the analysis gives only a lower bound on the probability of solving a problem of particular size. The analysis, which is a worst case analysis, is based on maximum branching at each step. It is also assumed that the best node at each step is present in *open* in a position where the probability of its being chosen is the lowest possible. Also,

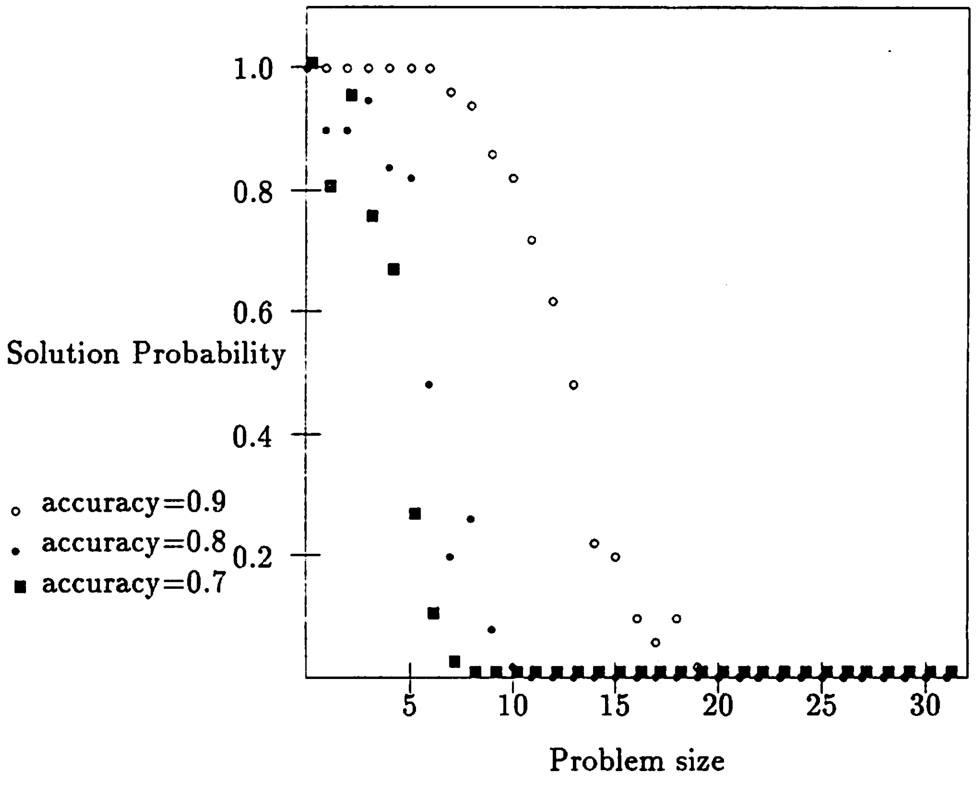

Figure 4: Solution probability versus problem size without self organization.

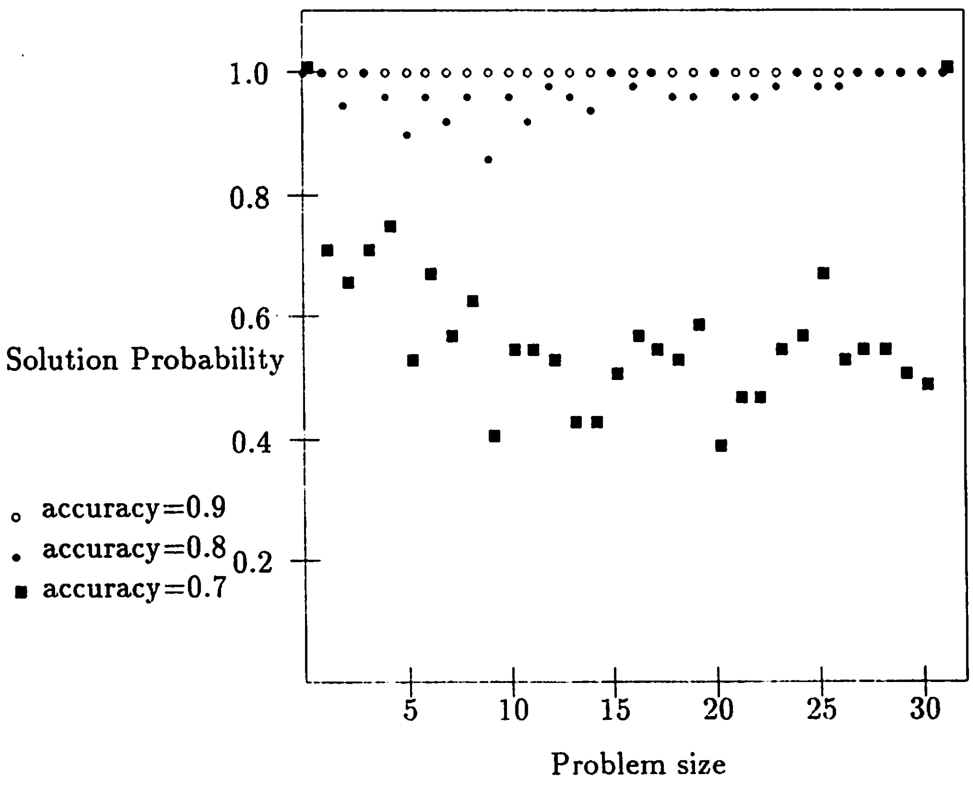

Figure 5: Solution probability versus problem size with self organization.

the modeling of the search process by a binomial distribution is an approximation to the actual search process. At each step in the search, the probability of having found the best node after that step, is greater than or equal to a fixed value, independent of the size of *open*. That is, $P(\overline{A_i}) \geq p^b$, but in estimating the probability of finding a solution by tail of the binomial distribution it was assumed that $P(\overline{A_i}) = p^b$. Finally, in figure 5 observe that, as discussed in the previous section, if the classification accuracy is 0.8 and the dilation is 3, then the frequency of finding a solution increases as the problem size increases.

6. Conclusions

This paper illustrates a way to analyze the performance component of a learning system. Probabilistic analysis of algorithms for combinatorial optimization is well known[2]. This paper demonstrates how such probabilistic analysis may be useful for learning systems. The requirement on the performance component considered here was that the performance component produce "good" approximations, measured by the dilation of the search, "most of the time", measured by the probability of finding a solution within the given dilation. This is in the spirit of the Valiant framework of learning[6], in which the learned function is required to be a close approximation to the target function most of the time. The analysis here was facilitated by assuming that the errors in the preference predicate were independent and random. The effect on the performance system of errors that are not independent and/or that are systematic remains to be investigated.

This paper has two contributions. The first is an improvement to best first search using preference predicates. The self organizing best first search algorithm permits the effective use of an imperfectly learned preference predicate, without any increase in the computational effort required to do the search. This was illustrated in the paper for a random sample of eight puzzle problems. A preference predicate with a classification accuracy of 0.9 was able to solve all problems in the sample within a dilation of 3. Best first search without self organization, on the same sample, could not solve, within a dilation of 3, any problem of size ≥ 20. For self organizing best first search, if the classification accuracy of the preference predicate and the dilation of the search are above certain values, then there is a problem size, such that for all problems of a greater size, the probability of finding the solution within a fixed dilation increases as the problem size increases.

The second contribution is the quantitative relationship between the classification accuracy of the preference predicate and the probability of a solution within a given dilation. This relationship makes it possible to determine one of the factors given the other two. In particular, this relationship makes it possible to determine the effort that must be spent on learning a preference predicate— measured by classification accuracy that must be achieved by the preference predicate— given the performance requirements in the form of the dilation of the search and the minimum probability of obtaining a solution within the given dilation.

7. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IRI-8619107 and by the Office of Naval Research through a Univerisity Research Initiative Program, under contract number N00014-86-K-0764. Helpful comments have been provided by James Callan, Victor Coleman, Margret Connell, Tom Fawcett and David Haines.

8. References

[1] Hester, J. H. and Hirschberg, D. S., "Self organizing linear search", *ACM Computing Surveys*, vol. 17, No. 3, September 1985.

[2] Karp, R. M., "Combinatorics, Complexity, Randomness.", ACM Turing Award Lecture, *Communications of the ACM*, vol 29, no 2, pp. 98-109, 1986.

[3] Korf, R. E., "Real-Time heuristic search: new results.", *Proceedings of the Seventh National Conference on Artificial Intelligence*, Morgan Kaufman, pp. 139 - 144, 1988.

[4] Utgoff, P. E. and Saxena, S., "Learning a preference predicate", *Proceedings of the Fourth International Workshop on Machine Learning*, Morgan Kaufman, pp. 36 - 40, 1987.

[5] Utgoff, P. E. and Saxena, S., "A perfect lookup table evaluation function for the eight-puzzle", COINS Technical Report 87-71, University of Massachusetts at Amherst, 1987.

[6] Valiant, L., "A theory of the learnable", Communications of the ACM, 27(11), pp. 1134-1142, 1984.