A METHOD FOR AUTOMATIC GENERATION OF HEURISTICS FOR STATE-SPACE PROBLEMS

Larry Rendell
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

CS-76-10

February 1976

A METHOD FOR AUTOMATIC GENERATION OF HEURISTICS FOR STATE-SPACE PROBLEMS

by

Larry Rendell

ABSTRACT

A method is described for automatic generation of a dynamic evaluation function for a given state-space problem class. The evaluation function is a nonlinear composition of more elementary functions (features) which directly utilizes statistics of previously encountered problems and which incorporates clustering to reduce stored information. The method includes continuing revision of the initiatory evaluation function and also allows subsequent consideration of added features.

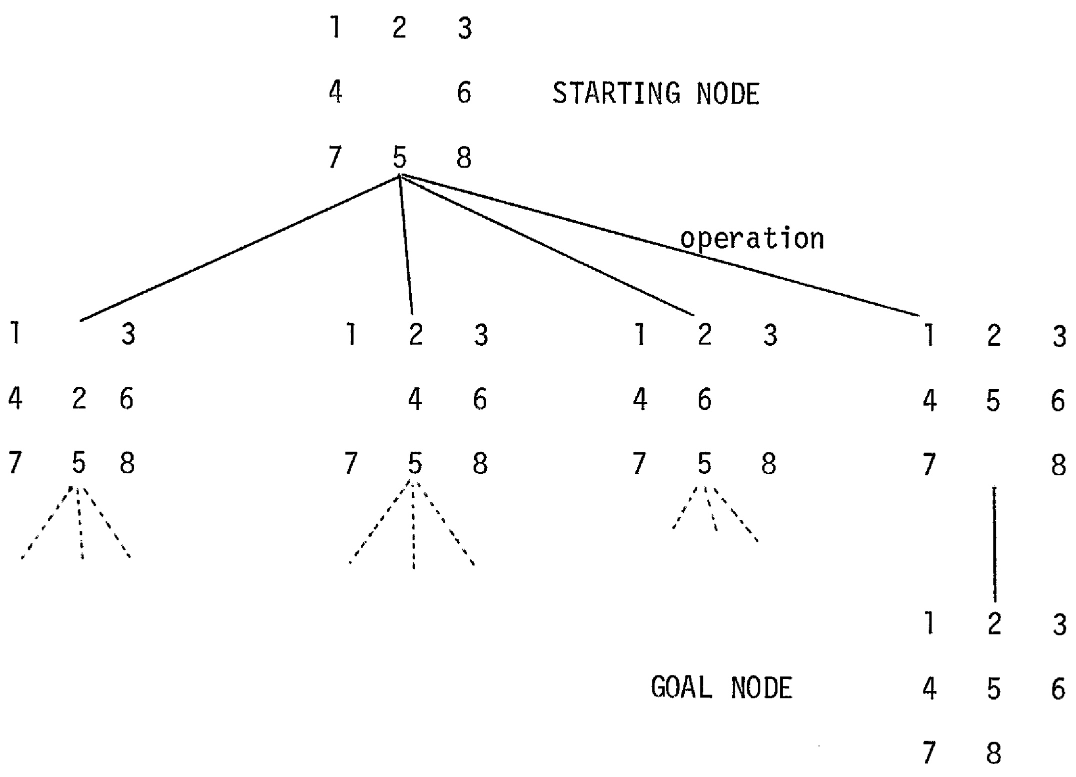
CONTENTS

1. INTRODUCTION

There is a large class of problems which can be analysed as state-space problems. A state-space problem is one whose solution can be given as a sequence of distinct states. Usually, each state has more than one successor each of which can be considered to be generated by some operator. Thus, a state can be represented as a node on a tree whose arcs correspond to operators, a state tree. A solution sequence is given by a path through the tree from a starting node or state to a goal.

A common example is the 8-puzzle, consisting of eight numbered squares in a 3×3 configuration with one position empty. Here, a representation for the state of the problem might be a 3×3 array, each element representing a square. An operator creates a successor state by a "move" of an adjacent square into the empty position:

PART of the STATE TREE for an 8-puzzle problem



A more complex example of a state-space problem is the following theorem proving formulation. Suppose we have a set of axioms, along with a negated theorem, all in clause form. The clauses form a basis set of nodes, and new nodes are created by resolution [Nilsson (1971)] of pairs of clause nodes. The complete (often infinite) result would be an exhaustive deduction graph. Any subgraph which includes the same basis set of nodes, we can call a deduction graph. A deduction graph which has exactly one null clause is a final deduction graph (corresponding to a proof).

In this problem class, a state is not a clause node, but rather a deduction graph. And the operators linking states are resolution along with some clause pair selector.

In this report, we shall be concerned with any arbitrarily chosen state-space problem class, and with some graph (or tree) which is continually being extended as a solution is attempted. We shall call this graph a (state) description graph (or tree). For the 8-puzzle, a description tree is the current state tree. For the resolution example, one description graph is the current deduction graph. In general, the state description graph can be the state tree, but it can also be something else, as long as its structure and nodes somehow reflect the current state of the problem, and as long as the continuing solution attempt results in constant addition of description nodes, one for each operator application.

The state description graph which corresponds to a solution, we shall call a final (description) graph (or tree).

One approach to the efficient solution of state-space problems is to consider the state tree, and to try to develop heuristic search procedures which generally result in the creation of few "useless" nodes, and lead more directly to the goal. And one kind of search procedure incorporates an evaluation function [Doran and Michie (1966), Doran (1967)] to rank description nodes. The value of this function is supposed to reflect the likelihood that the node will participate in a solution to the problem, so higher ranking nodes are expanded first. The system described in this paper utilizes such a function, the "utility estimate".

An evaluation function can be statically defined by the programmer, but a more sophisticated approach is to develop a system which composes its own, from a set of "features" (more fundamental functions defined over the nodes). Implicit in this approach is the hypothesis that each feature is an abstraction which generally correlates with node usefulness. The feature set-utility estimate correlation for one set of problems (in a given domain) is used to induce utility estimates for nodes generated in solution attempts of (other) problems which are in the same domain.

There is an obstacle, however: It is difficult to unite the features properly to create this utility estimate or evaluation function. Multiple linear regression has been tried, successfully [Slagle and Bursky (1968), Slagle and Farrell (1971)]; another (this one a nonlinear) method is given in Samuel (1959, 1967). A system which has the capability of forming a utility estimate as a general (nonlinear) functional composition of individual features has the obvious advantage of potentially greater accuracy. But such a system also can employ more primitive, lower level features, with the machinery taking care of any relationships. For example, if the problem domain is the 8-puzzle, the feature set might simply be the nine functions which indicate which numbered square is in what position. Or, for resolution theorem proving, the computer could be "told" how to count and also be informed that the things to count in a clause (count all, also count discrete) are symbols; further that symbols are predicate, functional, connective, constant, or variable; predicate symbols are "=", "<", ...; etc.

This paper presents an approach to this problem of automatic composition of features. Rather than using a presupposed functional relationship between the utility estimate and the features, the system works directly with discrete numerical values derived from the statistics. An overview of the method follows.

Features have values which are constrained to be integer, only. Each feature which is active, or "current", becomes a dimension in a "feature space". Within a region, R, in the "current" feature space--R may or may not be a point--a "utility estimate" is defined, which is the ratio of the number of "good" nodes (nodes that have participated in a solution), which map into R; to the total number of nodes generated, again which map into R. Now for a high dimensional space, the number of regions, if they were point regions, would become prohibitively large. This situation is precluded by allowing adjoining regions to coalesce. This clustering also smooths out random statistical variations. To decrease further the storage information required, the regions are constrained to be rectangular. In the section which follows this one, rectangular regions and specific "cluster functions" are detailed, and the third section specifies the clustering algorithm.

The system also allows the addition of new features (feature space extension) after a utility estimate has already been created for a current feature space. Section four describes the algorithm which extends the old clusters into the proposed space extension, and which checks whether the new features will help to "discriminate" good nodes. Finally, section five relates a procedure for adding the statistical results for successive problem solutions, in order to form a cumulative utility estimate. This continual revision sometimes necessitates "splitting" of the clusters as the estimates become more refined.

2. FEATURES AND CLUSTER FUNCTIONS

Suppose that we are considering a given state-space problem, and a particular description graph representation. A feature (of a description node) for this representation is any function which maps the set of nodes into the set of natural numbers, Π . For example, if our problem is resolution theorem proving and the description graph is the deduction graph, then one feature of a clause node might be the total number of symbols in the clause. Another feature could be the depth of the node in the deduction graph (distance from the basis clause level). This second example involves the deduction graph as well as the clause per se , while the first involves only the clause node.

If $S = \{f_1, f_2, \ldots, f_n\}$ is an ordered set of features and π is a node, then $(f_1(\pi), f_2(\pi), \ldots, f_n(\pi))$ can be thought of as a point in an n dimensional feature space, $\underline{F}$.

Suppose that we have such a feature space, F. Consider a rectangular region or cluster in F which is aligned with the axes. Any such cluster of points may be completely specified by just two extreme corner points, $\underline{A} = (a_1, a_2, \ldots, a_n)$, $\underline{B} = (b_1, b_2, \ldots, b_n)$ ($a_i \le b_i$, $1 \le i \le n$, or, abbreviated, $\underline{A} \le \underline{B}$). A point, $\underline{X} = (x_1, x_2, \ldots, x_n)$ lies within the rectangle if $a_i \le x_i \le b_i$, $1 \le i \le n$, i.e. if $\underline{A} \le \underline{X} \le \underline{B}$.

Now imagine that we have a finite portion of F which is partitioned into m such rectangular clusters, given by the m corner point pairs, $\underline{A}^{(j)} = (a_1^{(j)}, a_2^{(j)}, \ldots, a_n^{(j)})$ and $\underline{B}^{(j)} = (b_1^{(j)}, b_2^{(j)}, \ldots, b_n^{(j)})$, $1 \le j \le m$. Notice that, to define the set of m clusters, just 2mn integers are required.

Later on, we shall see how these clusters are formed. For now, we shall define a class of functions, which we can call "elementary utility functions", over a set of clusters. These functions not only will be used to construct evaluation functions, but will also participate in the clustering process. Both of these aspects will become clear by the end of the next section.

Before we discuss these functions, we need to define two auxiliary <u>counting functions</u>, τ and γ. Suppose we have the following:

(1) A state-space representation of a problem, and a final state description graph, $\mathscr{S}$, which corresponds to a solution of this problem.

(2) A feature space, F, defined by the ordered set of features $\{f_1, f_2, \ldots, f_n\}$.

(3) A partition, $\mathscr{P}$, of a portion of F, into m rectangular clusters, $\{R_1, R_1, \ldots, R_m\}$ whose lower and upper extreme corner points are given by:

$A^{(j)} = (a_1^{(j)}, a_2^{(j)}, \ldots, a_n^{(j)})$, and

$B^{(j)} = (b_1^{(j)}, b_2^{(j)}, \ldots, b_n^{(j)})$, respectively, for $1 \le j \le m$.

Then, the total <u>counting function</u>, $\tau(\mathscr{S}, F, R_j) \underset{\text{def}}{\equiv}$

the total number of nodes, π, in $\mathscr{S}$, such that

$a_i^{(j)} \le f_i\ (\pi) \le b_i^{(j)},\ 1 \le i \le n.$

(I.e. the number of nodes, π, in $\mathscr{S}$, such that the feature space map of π falls in cluster R_j).

And the "<u>good</u>" <u>node counting function</u>,

$\gamma(\mathscr{D},F,R_j) \; \underline{\underline{\text{def}}}$

the number of nodes π, in $\mathscr{D}$, such that

(1) $a_i^{(j)} \le f_i(\pi) \le b_i^{(j)}$, $1 \le i \le n$, and

(2) π participates in the solution.

If a particular $\mathscr{D}$ and F are understood, we can abbreviate the counting functions to $\tau(R_j)$ and $\gamma(R_j)$.

What we want to do now is to use these statistics for $\mathscr{D}$ to estimate the probability that nodes of description graphs for <u>other</u> problems might be used in a solution. We shall hypothesize that, in general, if the feature space map of a node falls in R_j, then the probability that the node is "good" relates to $\tau(R_j)$ and $\lambda(R_j)$.

Suppose that we have a feature space, rectangular partition, and counting functions, all denoted as above. Suppose, also, that π is a node of a description graph for <u>another</u> problem, both the problem and the graph being of the same type that F, etc. apply to.

Assume that $a_i^{(j)} \le f_i(\pi) \le b_i^{(j)}$, $1 \le i \le n$, for some $j \le m$. If we denote $(f_1(\pi), f_2(\pi),\ldots,f_n(\pi))$ by $F(\pi)$, then this statement is equivalent to $F(\pi) \in R_j$. Let G represent the event that π participates in a solution, and ω_j be the event $F(\pi) \in R_j$. Then the probability of G, given ω_j,

$$P(G/\omega_j) = \frac{P(G \cap \omega_j)}{P(\omega_j)}.$$

This is, we hypothesize, estimated by

$\gamma(\mathcal{A},F,R_j)/\tau(\mathcal{A},F,R_j)$; i.e.

$$P(G/\omega_j) \doteq \frac{\gamma(\mathcal{A},F,R_j)}{\tau(\mathcal{A},F,R_j)} = \frac{\gamma(R_j)}{\tau(R_j)} .$$

To repeat, we have hypothesized that the statistics for one problem solution can be generalized to another.

We now define the <u>elementary utility function</u> for $\mathcal{A}$, F, $\mathcal{P}$ to be

$$U_{\mathcal{A},F,\mathcal{P}}(R_j) = U(R_j) \underset{\text{def}}{\equiv} (\gamma(R_j) , \tau(R_j)) \underset{\text{def}}{\equiv} (\gamma,\tau)(R_j) .$$

For any utility function, $U = (\gamma,\tau)$, we define an associated function, the <u>utility estimate</u>,

$$u(R_j) \underset{\text{def}}{\equiv} \gamma(R_j)/\tau(R_j) \quad , \text{ if } \tau(R_j) \neq 0 .$$

This is the estimator of $P(G/\omega_j)$; i.e. if $F(\pi) \in R_j$, then the estimate of the probability that π might participate in a solution is $u(R_j)$. A later section will describe a method for constant revision of U (and thus u), using statistics from successive problem solutions.

Consider two clusters, R_i and R_j, and suppose that $u(R_i) \leq u(R_j)$. We shall say that R_i is <u>similar to</u> R_j (<u>under</u> $\underline{U_{\mathcal{A},F,\mathcal{P}}}$) if $\tau(R_i) = 0$ or $\tau(R_j) = 0$ or

$$\frac{\gamma(R_i) + \sqrt{\gamma(R_i)}}{\tau(R_i) - \sqrt{\tau(R_i)}} \geq \frac{\gamma(R_j) - \sqrt{\gamma(R_j)}}{\tau(R_j) + \sqrt{\tau(R_j)}}$$

The reasoning behind this is as follows. Let us imagine that we have a large number of problems which all have the same representation schema as the one above. Consider the entire set, S, of description nodes for all these problems and the mappings of all these nodes into F. We have already hypothesized that $P(G/\omega_j) = \gamma(R_j)/\tau(R_j)$ $(j \le m)$ for any problem will apply to ther problems. What we want now is a measure of the "reliability" of this ratio. To find a rough estimate, let us make the gross assumption that the nodes which counted in $\tau(R_j)$ and $\gamma(R_j)$, as calculated for $\mathscr{S}$, actually were chosen randomly from S, rather than by the selection mechanism for the particular problem solution attempt. Then, for the total counting function, the variance estimate for this hypergeometric (equivalent to binomial) distribution is

$$\sqrt{N(R_j)\,\frac{\tau(R_j)}{N(R_j)}\left(1 - \frac{\tau(R_j)}{N(R_j)}\right)} \quad \text{, where}$$

$N(R_j)$ is the number of nodes of S whose F maps fall in R_j. If $N(R_j)$ is large compared with $\tau(R_j)$, the expression becomes approximately $\sqrt{\tau(R_j)}$. Similarly, the variance estimate for the "good" counting function is $\sqrt{\gamma(R_j)}$. Thus, our definition for similarity of two regions is equivalent to the statement that two regions are similar if their utility estimates overlap when the component counting functions are altered by up to one variance estimate.

3. CLUSTERING

Let us recursively define a utility function to be either an elementary utility function (of the preceding section) or else the result of operating on a utility function by any of the algorithms in the remainder of this report.

This section describes the process of clustering the regions of a feature space. The process is similar to some well known algorithms [Hartigan (1975), Duda and Hart (1973)]; joining, or agglomeration, of "most similar" neighbours is used. However, our algorithm is different from many others, in some respects. One is that the distance, or similarity function is a property of "experience" of the model, not just of the feature space, namely through utility functions. Another significant feature of our algorithm is that the final number of clusters is not fixed a priori, but rather is determined by the data. The precise meaning of these comments will soon become clearer.

Suppose we are given the following:

(1) A feature space, F (for some state space representation), defined by the ordered feature set $\{f_1,f_2,\ldots,f_n\}$.

(2) A partition, $\mathcal{P}$, of a portion of F, into m clusters $\{R_1,R_2,\ldots,R_m\}$, assumed to be rectangular.

(3) A utility function, $U = (\gamma,\tau)$, over $\mathcal{P}$.

We shall now define a non-metric distance function, d, for $(F,\mathcal{P},U)$, which maps the cartesian product of the cluster set with itself into $[0,\infty]$: Consider two clusters, R_i,R_j $(i,j \leq m)$.

$$d(R_i,R_j) \overset{\text{def}}{=} \begin{cases} \infty, & \text{if } R_i, R_j \text{ are not similar (see previous section)} \\ \max\,(u(R_i)/u(R_j),\; u(R_j)/u(R_i)) - 1, & \text{if } R_i, R_j \text{ are similar.} \end{cases}$$

Now the clustering algorithm can be stated:

CLUSTERING ALGORITHM

For each of the m clusters, R_i ($i \le m$), call another cluster, R_j, an <u>amalgamation candidate</u> if:

There are two points, $\underline{X} = (x_1,x_2,\dots,x_n) \in R_i$, and $\underline{Y} = (y_1,y_2,\dots,y_n) \in R_j$ such that

(i) $\underline{X} = \underline{Y} \pm c.b_k$

where $c \in \mathbb{I}$ and b_k is one of the n ($k \le n$) unit coordinate basis vectors

and

(ii) R_j borders R_i, or else there is a subset of the clusters, $\{z_1,z_2,\dots,z_p\}$, with R_i bordering z_1, z_1 bordering $z_2,\dots,$etc., and z_p bordering R_j, such that a line joining $\underline{X}$ and $\underline{Y}$ passes through all the z_i, and such that $U(z_k) = (0,0)$ for all $k \le p$.

(1) For each cluster, R_i, calculate the distance function, $d(R_i,R_j)$ for each of its amalgamation candidates, R_j. Choose R_k and R_l such that

$d(R_k,R_l)$ is a minimum of all these distances (choose one pair if there is a tie). If $d(R_k,R_l) = \infty$, stop. Otherwise, there are two cases. R_l is an amalgamation candidate for R_k of the above type (i) or of type (ii). If the former (R_k and R_l directly adjacent), then replace R_k and R_l with $R_{kl} = R_k \cup R_l$, and decrement m by one. If the cluster set $\{z_1,z_2,\ldots,z_p\}$ lies between R_k and R_l then replace all of these p+2 clusters by $R_{kl} = R_k \cup R_l \cup Z_1 \cup Z_2 \cup \ldots \cup Z_p$ and decrement m by p+1. In each case, set

$$U(R_{kl}) = U(R_k) + U(R_l)$$

Go to (1).

This is the basic clustering algorithm, but there are two modifications we shall make. The first is to restrict the process of clustering to just some dimensions of F, which are specified by a set of input _clustering dimensions_; these form a _clustering subspace_. Because of the way in which the algorithm will be used, this restriction is not severe. This fact will be appreciated later on.

The other modification we shall make to the clustering algorithm is an addition. As it stands, the algorithm outputs clusters which are not rectangular in general. But for storage reasons, it is important, sometimes essential to minimize the information required to specify a feature space region. And, as we saw in the previous section, rectangular regions, aligned with the axes, require little information to specify. The following addendum rectangularizes the clusters.

CLUSTERING ALGORITHM ADDENDUM

Let the original regions which were input to the clustering algorithm be denoted by $\mathscr{R}_0 = \{r_1,r_2,\ldots,r_m\}$ and let the utility function for

$\mathcal{P}_0$ be U_0. Suppose that the algorithm has output the cluster set $\mathcal{P} = \{R_1, R_2, \ldots, R_m\}$, having operated in the $c \le n$ dimensional clustering subspace $F_c \subseteq F$.

Consider the rectangular surfaces, $\mathcal{S}$, in F which have the following properties:

(1) They are n-1 dimensional, and defined by $x_i = k$, where x_i is the i^{th} coordinate of F, and $k \in K_i$; a set defined as follows: Let $(y_1, \ldots, y_i, \ldots, y_n)$ be a point within any of the clusters, R_j. Then $y_i \in K_i$ if there is no other point, $(y_1, \ldots, y_{i-1}, z_i, y_{i+1}, \ldots, y_n) \in R_j$ such that $z_i < y_i$. Also, $y_i \in K_i$ if there is no other such point such that $z_i > y_i$.

(2) Each surface does not intersect any of the original clusters, r_j, except at a boundary only.

Generate a new, rectangular, partition, $\mathcal{P}_1 = \{R_1^{(1)}, R_2^{(1)}, \ldots, R_m^{(1)}\}$, so that:

(1) For $R_j \in \mathcal{P}$, there is a possibly different $R_j^{(1)} \in \mathcal{P}_1$.

(2) The boundaries of $R_j^{(1)}$ are selected from the just-defined set of surfaces, $\mathcal{S}$, consistant with the fact that $\mathcal{P}_1$ is a partition.

Repeat this rectangular partitioning, using $\mathcal{P}$ and $\mathcal{S}$, to create as many distinct sets $\mathcal{P}_1$, $\mathcal{P}_2, \ldots$ as possible. Say there are p such partitions, with the regions for $\mathcal{P}_i$ represented by $R_1^{(i)}, R_2^{(i)}, \ldots, R_m^{(i)}$ $(i \le p)$. We shall select the partition which is the "best", as follows.

For each $\mathcal{P}_i$, calculate an "error" or "total distance" function, using the "unrectangular" partition, $\mathcal{P}$, as well as $\mathcal{P}_i$:

$$D(R_j^{(i)}) \ \overline{\overline{\text{def}}} \ \sum_{R \in R_j^{(i)}} d(R_j, R)$$

and sum these over all regions of $\mathcal{P}_i$:

$$D_i \quad \underset{\text{def}}{=} \quad \sum_{j \le m} D\,(R_j^{(i)}) \; .$$

Now, choose the partition, $\mathcal{P}_k$, for which D_k is a minimum. $\mathcal{P}_k$ is the final rectangular partition output by the clustering algorithm, and its utility function,

$$U(R_j^{(k)}) \quad \underset{\text{def}}{=} \quad \sum_{R \,\in\, R_j^{(k)}} U\,(R)$$

In order that the rectangularizing section of the algorithm be reasonably fast, either the original set $\mathcal{P}_0$ or else the number of clustering dimensions must be fairly small.

The following is an example from resolution theorem proving.

$n = 1$

f_1 = depth of clause node in deduction graph

$F = \{f_1\}$

Clustering subspace: $F_c = F$.

f_1	U
0	7,7
1	1,2
2	1,6
3	1,20
4	1,66
5	1,181
6	1,143

$\longrightarrow$

f_1	U
0-1	8,9
2-3	2,26
4-6	2,390

4. EXTENDING THE FEATURE SPACE

Suppose we have a feature space, F, defined by $\{f_1,f_2,\ldots,f_n\}$; a partition, $\mathcal{P}$, of a part of F, into m clusters $\{R_1,R_2,\ldots,R_m\}$; and a utility function, U, over $\mathcal{P}$. Let $\{f_{n+1}, f_{n-2},\ldots,f_{n+\ell}\}$ be an ordered set of distinct new features, and define an extension, F', of F which has dimensions given by $\{f_1,f_2,\ldots,f_{n+\ell}\}$. Next, for each n-dimensional cluster, R_j $(j \le m)$, form a set of clusters, such that each new $n+\ell$ dimensional region, $R_j^{(k)}$ projects into R_j, in F; but into a point, in F'-F.

Let us suppose that we also have a final state description graph, $\mathcal{G}$. We are now in a position to define the

DISCRIMINATION ALGORITHM

Calculate the counting functions, γ and τ for $\mathcal{G}$, F', and extended clusters $R_j^{(k)}$. Apply the clustering algorithm to these extended clusters, using the elementary utility function defined by γ and τ. Cluster only in the ℓ dimensions of F'-F. Let the resulting partition of F' be represented by $\mathcal{P}'$, with m' clusters; $\mathcal{P}' = \{R_1^{(1)} \ldots R_1^{(k_1)}, R_2^{(1)}, \ldots, R_2^{(k_2)},\ldots,\ldots,R_m^{(1)},\ldots,R_m^{(k_m)}\}$, where the regions of like subscripts project into the corresponding original regions in F, i.e., $\{R_j^{(1)}, R_2^{(2)},\ldots,R_j^{(k_j)}\} = \mathcal{P}_j$ is a "cylinder" for R_j, (and $\mathcal{P}' = \mathcal{P}_1 \cup \mathcal{P}_2 \cup \ldots \cup \mathcal{P}_m$). Also, let the utility function over $\mathcal{P}'$ which the algorithm outputs be denoted by U'.

Consider f_i $(n < i \le n+\ell)$. If, for some $\mathcal{P}_j$ $(j \le m)$, the f_i - projection of all the clusters of $\mathcal{P}_j$ forms more than just one line segment, then we can say that f_i discriminates among state description nodes (of $\mathcal{G}$, over F, and for $\mathcal{P}$). Any f_i $(n < i \le n+\ell)$ which does not discriminate is removed from the ordered set of features, and from the feature space, F'.

Suppose we begin with F the null space, and successively select new feature sets, and apply the discrimination algorithm. Any feature which discriminates is retained in F, and such a feature we shall call a <u>current feature</u>. The ordered set of current features defines the <u>current feature space</u>, F. Furthermore, there is a <u>current utility function</u>, U, associated with F.

Let us consider an example of feature space extension and discrimination algorithm application. As the current feature space, we choose the one dimensional one of the previous section. The associated current region function is the three region one produced in that example.

Current feature space is defined by $\{f_1\}$.

f_1 = depth of clause in deduction graph

f_2 = number of symbols in clause

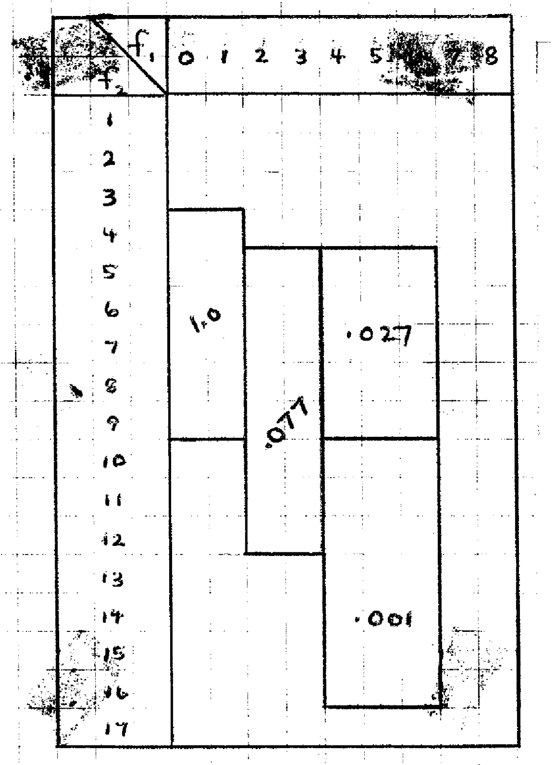
Extended feature space $\{f_1,f_2\}$

Clustering subspace is $\{f_2\}$

f_2 \ f_1	0-1	2-3	4-6
1			
2			
3	1,2		
4			
5	2,2	1,8	1,4
6	2,2	1,3	
7	1,1		0,24
8		0,4	1,39
9	1,1	0,8	0,6
10		0,1	0,48
11			0,99
12		0,2	0,40
13			0,17
14			0,55
15			
16			0,18
17			

The result of extension and clustering is:

f_1	f_2	U	u
0 - 1	3 - 9	7,7	1.0
2 - 3	5 - 12	2,26	.077
4 - 6	5 - 9	2,73	.027
4 - 6	10 - 16	0,277	.001



Utility Estimate values, u, within regions

The new feature, f_2, was found to discriminate among description nodes, so the current feature space is now given by $\{f_1,f_2\}$. The new current utility function does not retain any values from the old one, but the projection of its regions into the old feature space, $\{f_1\}$ gives exactly the regions of the former utility function.

5. REVISING UTILITY FUNCTIONS

Suppose we have a current utility function, $U = (\gamma,\tau)$, over a current feature space, F, given by $\{f_1,f_2,\ldots,f_n\}$. Suppose, also, that the clusters associated with U are denoted by $\mathcal{P} = \{R_1,R_2,\ldots,R_m\}$. Now, imagine that a problem solution has just been reached, whose final description graph is $\mathcal{D}$.

Use $\mathcal{P}$ to calculate the elementary utility function, $U_{\mathcal{D},F,\mathcal{P}} = U \quad =$ $(\gamma_{\mathcal{D}},\tau_{\mathcal{D}})$. Now we shall use $U_{\mathcal{D}}$ to update U.

ADDITION PROCEDURE

Consider U and $U_{\mathcal{D}}$ region by region. If R_j under $U_{\mathcal{D}}$ is similar to R_j under U (refer to section 2) then U and $U_{\mathcal{D}}$ are <u>consistant</u> <u>within</u> <u>R_j</u>. In this case, add $U_{\mathcal{D}}$ to U:

$$U(R_j) := U(R_j) + U_{\mathcal{D}}(R_j) \ .$$

If, however, U and $U_{\mathcal{D}}$ are not consistant within R_j, then attempt to split R_j into two (rectangular) clusters, $R_j^{(1)}$ and $R_j^{(2)}$, in such a way that $\tau_{\mathcal{D}}(R_j^{(1)}) \neq 0 \neq \tau_{\mathcal{D}}(R_j^{(2)})$ and R_j under U is similar to $R_j^{(1)}$ under $U_{\mathcal{D}}$ but not to $R_j^{(2)}$ under $U_{\mathcal{D}}$. If this attempt is successful, then remove R_j from $\mathcal{P}$, replacing it with $R_j^{(1)}$ and $R_j^{(2)}$ (and incrementing m by one), and set

$$U(R_j^{(1)}) := U(R_j) + U_{\mathcal{D}}(R_j^{(1)}),$$

$$U(R_j^{(2)}) := U_{\mathcal{D}}(R_j^{(2)}) \ .$$

If no such splitting can be found, then this procedure fails. This failure conceivably might be used to signal an attempt to find a discriminating new feature (section 4).

If the procedure succeeds to this point, it is not yet complete, because it can happen that some of the nodes of $\mathcal{G}$ fall outside of $\mathcal{P}$ in F. Let $\mathcal{S} = \{R_{m+1}, R_{m+2}, \ldots, R_{m+p}\}$ be the set of point clusters external to $\mathcal{P}$. With the clustering subspace all of F, apply the clustering algorithm to $\mathcal{P} \cup \mathcal{S}$.

The current region function is now U as altered by this addition procedure.

6. SOME REMARKS

The system has some interesting properties. Early on in its experience, there tend to be few clusters, since similarity depends on the square root of the number of nodes encountered. Later on, as experience is gained, the discrimination becomes more refined; clusters are subdivided as inconsistancies arise (section 5).

Wherever discriminatory power becomes greater through feature space extension, however, there is greater uncertainty, until the system gains familiarity with the new feature(s), since probability values for the predecessor space are discarded (section 4).

Our approach to automatic heuristic generation can be thought of in terms of concept and hypothesis formation [Hunt (1962)]. For the example in section 3, the clauses on levels 0 and 1 are "early", those on levels 2, 3 are "middle", and those on levels 4, 5, 6 are "late". The hypothesis includes that early clauses have likelihood 8/9 of being useful, while late clauses have a chance of only 2/390.

Referring to the example in section 4, the "late" clauses are subdivided into "short" (4-9 symbols) and "long" (10-16 symbols). The rectangular regions correspond to concepts linked by conjunction (c.f. Brunner (1956)).

BIBLIOGRAPHY

Bruner, J.S., Goodnow, J.J., and Austin, G.A.: A Study of Thinking, Wiley, 1956.

Doran, J.: "An Approach to Automatic Problem Solving", in N. Collins and D. Michie (eds.), Machine Intelligence 1, pp. 105-123, American Elsevier, 1967.

Doran, J. and Michie, D.: "Experiments with the Graph Traverser Program", Proc. Roy. Soc., A, vol. 294, pp. 235-259, 1966.

Duda, R.O. and Hart, P.E.: Pattern Classification and Scene Analysis, Wiley, 1975.

Hartigan, J.A.: Clustering Algorithms, Wiley, 1975.

Hunt, E.B.: Concept Learning, an Information Processing Problem, Wiley, 1962.

Nilsson, N.J.: Problem Solving Methods in Artificial Intelligence, McGraw-Hill, 1971.

Samuel, A. (1959): "Some Studies in Machine Learning Using the Game of Checkers", IBM J. Res. Develop, vol. 3, pp. 211-229, 1959. Reprinted in E. Feigenbaum and J. Feldman (eds.), Computer and Thought, pp. 71-105, McGraw-Hill, 1963.

Samuel, A. (1967): "Some Studies in Machine Learning Using the Game of Checkers II. Recent Progress", IBM J. Res. and Develop., vol. 11, no. 6, pp. 601-617, 1967.

Slagle, J. and Bursky, P.: "Experiments with a Multipurpose, Theorem Proving Heuristic Program", J. ACM, vol. 15, no. 1, pp. 85-99, 1968.

Slagle, J. and Farrell, C.D.: "Experiments in Automatic Learning for a Multipurpose Heuristic Program", C. ACM, vol. 14, no. 2, pp. 91-99.