85-20

Report No. UIUCDCS-R-85-1217

GENETIC PLANS AND THE PROBABILISTIC LEARNING SYSTEM: SYNTHESIS AND RESULTS

by

Larry Rendell

July 1985

Department of Computer Science
University of Illinois at Urbana-Champaign
1304 W. Springfield Avenue
Urbana, Illinois 61801

This work was supported in part by the National Science Foundation under Grant No. DCR 84-06801 and by an operating grant from the Natural Sciences and Engineering Research Council of Canada.

This paper will appear in the Proceedings of the International Conference on Genetic Algorithms and their Applications, July 24-26, 1985, Carnegie-Mellon University.

GENETIC PLANS AND THE PROBABILISTIC LEARNING SYSTEM: SYNTHESIS AND RESULTS

Larry Rendell
Department of Computer Science,
University of Illinois at Urbana-Champaign,
1304 West Springfield Avenue, Urbana, Illinois 61801

ABSTRACT

This paper describes new conceptual and experimental results using the probabilistic learning system *PLS2*. PLS2 is designed for any task in which overall performance can be measured, and in which choice of task objects or operators influences performance. The system can manage incremental learning and noisy domains.

PLS2 learns in two ways. Its lower "perceptual" layer clusters data into economical cells or *regions* in augmented feature space. The upper "genetic" level of PLS2 selects successful regions (compressed *genes*) from multiple, parallel cases. Intermediate between performance data and task control structures, regions promote efficient and effective learning.

Novel aspects of PLS2 include compressed genotypes, credit localization and "population performance". Incipient principles of efficiency and effectiveness are suggested. Analysis of the system is confirmed by experiments demonstrating stability, efficiency, and effectiveness.

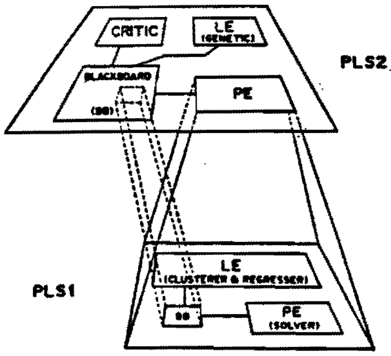

Figure 1. Layered learning system PLS2. The *perceptual* learning system PLS1 serves as the performance element (PE) of the *genetic* system PLS2. The PE of PLS1 is some task. PLS2 activates PLS1 with different knowledge structures ("cumulative region sets") which PLS2 continually improves. The basis for improvement is competition and credit localization.

1. INTRODUCTION

The author's *probabilistic learning system PLS* is capable of efficient and effective generalization learning in many domains [Re 83a, Re 83d, Re 85a]. Unlike other systems [La 83, Mit 83, Mic 83a], PLS can manage noise, and learn incrementally. While it can be used for "single concept" learning, like the systems described in [Di 82], PLS has been developed and tested in the difficult domain of heuristic search, which requires not only noise management and incremental learning, but also removal of bias acquired during task performance [Re 83a]. The system can discover optimal evaluation functions (see Fig. 2). PLS has introduced some novel approaches, such as new kinds of clustering.[1]

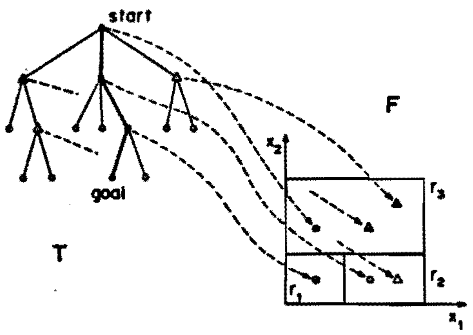

Figure 2. One use of PLS. In heuristic search, an object is a state, and its utility might be the probability of contributing to success (appearing on a solution path). E.g., for r_3, this probability is 1/3. Here the pair (r_3, p_3) is one of three *regions* which may be used to create a heuristic evaluation function. Region characteristics are determined by clustering.

1. See [Re 83a] for details and [Re 85a, Re 85b] for discussion of PLS's "conceptual clustering" [Mic 83b] which began in [Re 76, Re 77]. PLS "utility" of domain objects provides "category cohesiveness" [Me 85]. [Re 85c] introduces "higher dimensional" clustering which permits creation of structure. Appendix A summarizes some of these terms, which will be expanded in later sections of this paper.

Another successful approach to adaptation is *genetic algorithms* (GA's). Aside from their ability to discover global optima, GA's have several other important characteristics, including stability, efficiency, flexibility, and extensibility [Ho 75, Ho 81]. While the full behavior of genetic algorithms is not yet known in detail, certain characteristics have been established, and this approach compares very favorably with other methods of optimization [Be 80, Br 81, De 80]. Because of their performance and potential, GA's have been applied to various AI learning tasks [Re 83c, Sm 80, Sm 83].

In [Re 83c] a combination of the above two approaches was described: the doubly layered learning system *PLS2* (see Fig. 1).[2] PLS1, the lower level of PLS2, could be considered "perceptual"; it compresses goal-oriented information (task "utility") into a generalized, economical, and useful form ("regions" — see Figs. 2, 4). The upper layer is genetic, a competition of parallel knowledge structures. In [Re 83c], each of these components was argued to improve efficacy and efficiency.[3]

This paper extends and substantiates these claims, conceptually and empirically. The next section gives an example of a genetic algorithm which is oriented toward the current context. Section 3 describes the knowledge structure (regions) from two points of view: PLS1 and PLS2. Section 4 examines the synthesis of these two systems and considers some reasons for their efficiency. Sections 5 and 6 present and analyze the experimental results, which show the system to be stable, accurate, and efficient. The paper closes with a brief summary and a glossary of terms used in machine learning and genetic systems.

2. For the reader unfamiliar with learning system and other terminology, Appendix B provides brief explanations.

3. PLS2 is applicable to any domain for which features and "usefulness" or *utility* of objects can be defined [Re 83d]. An object can represent a physical entity or an operator over the set of entities. Domains can be simple (e.g. "single concept" learning), or complex (e.g. expert systems). State-space problems and games have been tested in [Re 83a, Re 83d]. The PLS approach is uniform and can be deterministic or probabilistic. The only real difficulty with a new domain is in constructing features which bear a smooth relationship to the utility (the system can evaluate and screen features presented to it).

2. GENETIC SYSTEMS: AN EXAMPLE

This section describes a simple GA, to introduce terminology and concepts, and to provide a basis for comparison with the more complex PLS2. The reader already familiar with GA's may wish to omit all but the last part of this section.

2.1. Optimization

Many problems can be regarded as function optimization. In an AI application, this may mean discovery of a good control structure for executing some task. The function to be optimized is then some measure of task success which we may call the *performance* μ. In the terminology of optimization, μ is the *objective function.* In the context of genetic systems, μ is the *fitness, payoff,* or *merit.*[4]

The merit μ depends on some control structure, the simplest example of which is a vector of *weights* $\mathbf{b} = (b_1, b_2, \ldots, b_n)$. Frequently the analytic form of $\mu(\mathbf{b})$ is not known, so exact methods cannot be used to optimize it (this is the case with most AI problems). But what often is available (at some cost) is the value of μ for a given control structure. In our example, let us suppose that μ can be obtained for any desired value of $\mathbf{b}$, by testing system performance. If μ is a well behaved, smooth function of $\mathbf{b}$, and if there is just one peak in the μ surface, then this *local optimum* is also a *global optimum*, which can be efficiently discovered using hill climbing techniques. However, the behavior of μ is often unknown, and μ may have numerous optima; in these cases a genetic adaptive algorithm is appropriate.

2.2. Genetic Algorithms

In a GA, a structure of interest, such as a weight vector $\mathbf{b}$, is called a *phenotype.* Fig. 3 shows a simple example with just two weights, b_1 and b_2. The phenotype is normally coded as a string of digits (usually bits) called the *genotype* B. A single digit is a *gene;* gene values are *alleles.* The position of a gene within the genotype is given by an index called the *locus.* Depending on the resolution desired, we might choose a greater or lesser number of sequential genes to code each b_i. If we consider 5 bits to be

4. μ might also be called the "utility", but we reserve this term for another kind of quality measure used by PLS1.

sufficient, the length of the genotype B will be L = 5n bits (see Fig. 3).

Instead of searching weight space directly for an optimal vector **b**, a GA searches gene space, which has dimensionality L (gene space is Hamming space if alleles are binary). A GA conducts this search in parallel, using a set of *individual* genotypes called a *population* or *gene pool.* By comparing the relative merits μ of individuals in a population, and by mating only the better individuals, a GA performs an informed search of gene space. This search is conducted iteratively, over repeated *generations.* In each new generation, there are three basic operations performed: (1) selection of parents, (2) generation of offspring, and (3) replacement of individuals. (1) and (2) have been given more attention. *Parent selection* is usually stochastic, weighted in favor of individuals having higher μ values. *Offspring generation* relies on *genetic operators* which modify parent genotypes. Two natural examples are mutation (which alters one allele), and crossover (which slices two genotypes at a common locus and exchanges segments—see Fig. 3).

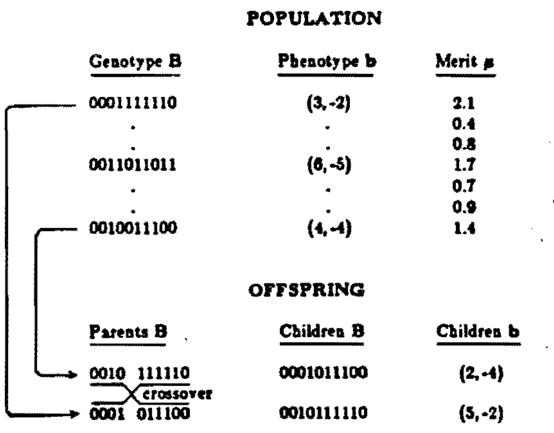

Figure 3. Simple genetic system. The upper part of this diagram shows a small population of just seven individuals. Here the set of characteristics (the *phenotype*) is a simple two element vector b. This is coded by the *genotype* B. Each individual is associated with its measured *merit* μ. On the basis of their μ values, pairs of individuals are stochastically chosen as parents for genetic recombination. Their genotypes are modified by crossover to produce two new offspring.

Because the more successful parents are selected for mating, and because limited operations are performed on them to produce offspring, the effect is a combination of knowledge retention and controlled search. Holland proved that, using binary alleles, the crossover operator, and parent selection proportional to μ, a GA is K^3 times more efficient than exhaustive search of gene space, where K is the population size [Ho75, Ho81]. Several empirical studies have verified the computational efficiency of GA's compared with alterative procedures for global optimization, and have discovered interesting properties of GA's, such as effects of varying K. For example, populations smaller than 50 can cause problems [Br81, De80].

2.3. Application in Heuristic Search

One AI use is search for solutions to problems, or for wins in games [Ni80].[5] Here we wish to learn an evaluation function H as a combination of variables $x_1, x_2, \ldots, x_n$ called *attributes* or *features* (features are often used to describe states in search). In the simplest case, H is expressed as the linear combination $b_1x_1 + b_2x_2 + \ldots + b_nx_n = \mathbf{b}.\mathbf{x}$, where the b_i are weights to be learned. We want to optimize the weight vector **b** according to some measure of the *performance* μ when H is used to control search.

A rational way to define μ (which we shall use throughout this paper) is related to the average number D of states or nodes developed in solving a set of problems. Suppose D is observed for a population of K heuristic functions H_i defined by weight vectors b_i. Since the performance improves with lower values of D, a good definition of the merit of H_i (i.e. of b_i) is the relative performance measure $\mu_i = \bar{D} / D_i$, where $\bar{D}$ is the average over the population, i.e. $\bar{D} = \sum D_j / K$. This expression of merit could be used to assess genotypes B representing weight vectors b_i, as depicted in Fig. 3.

Instead of this simple genetic approach, however, PLS2 employs unusual genotypes and operators, some of which relate to PLS1. In the remaining sections of this paper, we shall examine the advantages of the GA resulting from the combination of PLS1 with PLS2.

5. Notice that search takes place both at the level of the task domain (for good problem solutions), and at the level of the learning element (for a good control structure H).

3. PLS INFORMATION STRUCTURING: DUAL VIEWPOINT

The connection between PLS1 perceptual learning and PLS2 genetic adaptation is subtle and indirect. Basically PLS1 deals with *objects* **x** (which can be just about anything), and their relationships to task performance. Let us call the usefulness of an object **x** in some task domain its *utility* u(**x**).

Since the number of objects is typically immense, even vast observation is incomplete, and generalization is required for prediction of u, given a previously unencountered **x**. A significant step in generalization is usually the expression of **x** as a vector of high-level, abstract features $x_1, x_2, \ldots, x_n$, so that **x** really represents not just one object, but rather a large number of similar objects (e.g. in a board game, **x** might be a vector of features such as piece advantage, center control, etc.). A further step in generalization is to *classify* or *categorize* **x**'s which are similar for current purposes.[6] Since the purpose is to succeed well in a task, PLS1 classifies **x**'s having similar utilities u.

Class formation can be accomplished in several ways, depending on the *model* assumed. If the task domain and features permit, objects having similar utilities may be *clustered* in feature space, as illustrated in Figs. 2 & 4, giving a "region set" R.[7] Another model is the linear combination H = **b.f** of §2.

It is at this point that a GA like PLS2 can aid the learning process. Well performing b's or R's may be selected according to their merit μ. Note that merit μ is an overall measure of the task performance, while utility u is a quality measure localized to individual objects.

The question now is what information structures to choose for representing knowledge about task utility. For many reasons, PLS incorporates the "region set" (Fig. 4), which represents domain knowledge by associating an object with its utility. We examine the region set from two points of view: as a PLS1 knowledge structure, and as a PLS2 genetic structure.

6. Here *to classify* means *to form* classes, categories, or concepts. This is difficult to automate.

7. PLS1 initiated what has become known as conceptual clustering — where not just feature values are considered, but also predetermined forms of classes (e.g. rectangles), and the whole data environment (e.g. utility). See [Re 76, Re 77, Re 83a, Re 85a, Re 85b], and also Appendix A.

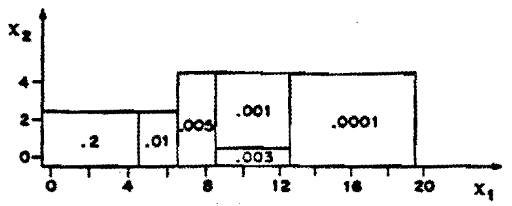

Figure 4. Dual interpretation of a region set R. A region set is a partition of feature space (here there are 6 regions). Points are clustered into regions according to their utility u in some task domain (e.g. u = probability of contributing to task success — see Fig. 2). Here the u values are shown inside the rectangles. A *region* R is the triple (r, u, e), where e is the error in u. The region set **R** = {R} serves both as the PLS1 knowledge structure and as the PLS2 genotype. In PLS1, **R** is a discrete (step) function expressing variation of utility u with features x_i. In PLS2, **R** is a compressed version of the detailed genotype illustrated in Fig. 5.

3.1. The Region as PLS1 Knowledge Structure

In a *feature space* representation, an object is a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$.[8] In a problem or game, the basic object is the *state*, frequently expressed as a vector of features such as piece advantage, center control, mobility, etc.[9] Observations made during the course of even many problems or games normally cover just a fraction of feature space, and generalization is required for prediction.

In *generalization learning*, objects are abstracted to form *classes, categories,* or *concepts*. This may take the form of a partition of feature space, i.e. a set of mutually exhaustive local neighborhoods called *clusters* or *cells* [An 73, Di 82]. Since the goal of clustering in PLS is to aid task performance, the basis for generalization is some measure of the worth, quality, or

8. Feature spaces are sometimes avoided because they cannot easily express structure. However, alternative representations, as normally used, are also deficient for realistic generalization learning. A new scheme mechanizes of a very difficult inductive problem: feature *formation* [Re 83d, Re 85c].

9. The object or event could just as well be an operator to be applied to a state, or a state-operator pair. See [Re 83d].

utility of a state or cell, relative to the task. One measure of utility is the probability of contributing to a solution or win. In Figs. 2, 4, probability classes are rectangular cells (for economy). The leftmost rectangle r has probability $u = 0.2$.[10] The rectangle r is a category generalizing the conditions under which the utility u applies.

In PLS, a rectangle is associated not just with its utility u, but also with the utility *error* e. This expression e of uncertainty in u allows quantification of the effect of noise and provides an informed and concise means for weighting various contributions to the value of u during learning. The triple $R = (r,u,e)$, called a *region*, is the main knowledge structure for PLS1. A set $\mathbf{R} = \{R\}$ of regions defines a partition in *augmented* feature space.

R may be used directly as a (discrete) *evaluation* or *heuristic* function $H = u(r)$ to assess state $x \in r$ in search. For example, in Fig. 4, there are six regions, which differentiate states into six utility classes. Instead of forming a discrete heuristic, **R** may be used indirectly, as data for determining the *weight vector* **b** in a smooth evaluation function $H = \mathbf{b}.\mathbf{x}$ (employing curve fitting techniques). We shall return to these algorithmic aspects of PLS in §4.

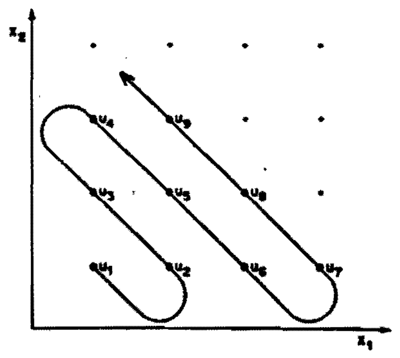

Figure 5. Definition of maximally detailed genotype U. If the number of points in feature space is finite and a value of the utility is associated with each point, complete information can be captured in a detailed genotype U of concatenated utilities $u_1 u_2 \dots u_L$. Coordinates could be linearly ordered as shown here for the two dimensional case. U is an fully expanded genotype corresponding to the compressed version of Fig. 4.

10. This could be expressed in other ways. The production rule form is $r \rightarrow u$. Using logic, r is represented: $(0 \leq x_1 \leq 4) \cap (0 \leq x_2 \leq 2)$.

3.2. The Region Set as Compressed and Unrestricted PLS2 Genotype

Now let us examine these information structures from the genetic viewpoint. The weight vector **b** of evaluation function H could be considered a GA phenotype. What might the genotype be? One choice, a simple one, was described in §2 and illustrated in Fig. 3: here the genotype B is just a binary coding of **b**. A different possibility is one that captures exhaustive information about the relationship between utility u and feature vector **x** (see Fig. 5). In this case, the gene would be $(\mathbf{x},u)$. If the number of genes is finite, they can be indexed and concatenated, to give a very detailed genotype U, which becomes a string of values $u_1 u_2 \dots u_L$ coding the entire utility surface in augmented feature space.

This genotype U is unusual in some important ways. Let us compare it with the earlier example B of §2 (Fig. 3). B is simply a binary form of weight vector **b**. One obvious difference between B and U is that U is more verbose than B . This redundancy aspect will be considered shortly. The other important difference between B and U is that alleles within B may well interact (to express feature nonlinearity), but alleles u_i within U *cannot* interact (since the u_i express an absolute property of feature vector **x**, i.e. its utility for some task). As explained in the next section, this freedom from gene interdependence permits localization of credit.[11]

The detailed genotype U codes the utility surface, which may be very irregular at worst, or very smooth at best. This surface may be locally well behaved (it may vary slowly in some volumes of feature space). In cases of local regularity, portions of U are redundant. As shown in Fig. 5, PLS2 compresses the genotype U, into the region set **R** (examined in §3.1 from the PLS1 viewpoint). In PLS2, a single region $R = (r,u,e)$ is a *set* of genes, the whole having just one allele u (we disregard the genetic coding of e). Unlike standard genotypes, which have a stationary locus for each gene and a fixed number of genes, the region set has no explicit loci, but rather a

11. While one of the strengths of a GA is its ability to manage interaction of variables (by "co-adapting" alleles), PLS2 achieves efficient and concise knowledge representation and acquisition by flexible gene compression, and by certain other methods examined later in this paper.

variable number of elements (regions), each representing a variable number of genes. A region compresses gene sets having similar utility according to current knowledge.

4. KNOWLEDGE ACQUISITION: SYNERGIC LEARNING ALGORITHMS

In this section we examine how R is used to provide barely adequate information about the utility surface. This compact representation results in economy of both space and time, and in effective learning. Some reasons for this power are considered.

The ultimate purpose of PLS is to discover utility classes in the form of a region set R. This knowledge structure controls the primary task: for example, in heuristic search, $\mathbf{R} = \{R\} = \{(r,u,e)\}$ defines a discrete evaluation function $H(r) = u$.

The ideal R would be perfectly *accurate* and maximally *compressed.* Accuracy of utility u determines the quality of task performance. Appropriate compression of R characterizes the task domain concisely but adequately (see Figs. 3, 4), saving storage and time, both during task performance and during learning.

These goals of accuracy and economy are approached by the doubly layered learning system PLS2 (Fig. 1). PLS1 and PLS2 combine to become effective rules for generalization (induction), specialization (differentiation), and reorganization. The two layers support each other in various ways: for example PLS2 stabilizes the perceptual system PLS1, and PLS1 maintains genotype diversity of the genetic system PLS2. In the following we consider details, first from the standpoint of PLS1, then from the perspective of PLS2.

4.1. PLS1 Revision and Differentiation

Even without a genetic component, PLS1 is a flexible learning system which can be employed in noisy domains requiring incremental learning. It can be used for simple concept learning like the systems in [Di 82], but most experiments have involved state space problem solving and game playing.[12] Here we examine PLS in the context of these difficult tasks.

12. These experiments have led to unique results such as discovery of locally optimal evaluation functions (see [Re 83a, Re 83d]).

As described in §3.1, the main PLS1 knowledge structure is the region set $\mathbf{R} = \{(r,u,e)\}$. Intermediate between basic data obtained during search, and a general heuristic used to control search, R defines a feature space augmented and partitioned by u and e. Because R is improved incrementally, it is called the *cumulative region set.* PLS1 repeatedly performs two basic operations on R. One operation is correction or *revision* (of utility u and error e), and the other is specialization, differentiation, or *refinement* (of feature space cells r). These operators are detailed in [Re 83a, Re 83d]; here we simply outline their effects and note their limitations.

Revision of u and e. For an established region $R = (r,u,e) \in \mathbf{R}$, PLS1 is able to modify u and to decrease e by using new data. This is accomplished in a rough fashion, by comparing established values within all rectangles r with fresh values within the same r. It is difficult or impossible to learn the "true" values of u, since data are acquired during performance of hard tasks, and these data are biased in unknown ways because of nontrivial search.

Refinement of R. Alternately performing then learning, PLS1 acquires more and more detail about the nature of variation of utility u with features. This information accumulates in the region set $\mathbf{R} = \{R\} = \{(r,u,e)\}$, where the primary effect of clustering u is increasing resolution of R. The number, sizes, and shapes of rectangles in R reflect current knowledge resolution. As this differentiation continues in successive iterations of PLS1, attention focuses on more useful parts of feature space, and heuristic power improves.

Unfortunately, so does the likelihood of error. Further, errors are difficult to quantify and hard to localize to individual regions.

In brief, while the incremental learning of PLS1 is powerful enough to learn locally optimal heuristics under certain conditions, and while PLS1 feedback is good enough to control and correct mild errors, the feedback can become unstable in unfavorable situations: instead of being corrected, errors can become more pronounced. Moreover, PLS1 is sensitive to parameter settings (see Appendix B). The system needs support.

4.2. PLS2 Genetic Operators

Qualities missing in PLS1 can be provided by PLS2. As §4.1 concluded, PLS1, with its single region set, cannot discover accurate values of utilities u. PLS2, however, maintains an entire population of region sets, which means that several regions in all cover any given feature space volume. The availability of comparable regions ultimately permits greater accuracy in u, and brings other benefits.

As §3.2 explained, a PLS2 genotype is the region set $\mathbf{R} = \{R\}$, and each region $R = (r, u, e) \in \mathbf{R}$ is a compressed gene whose allele is the utility u. Details of an early version of PLS2 are given in [Re 83c]. Those algorithms have been improved; the time complexity of the operators in recent program implementations is linear with population size K. The following discussion outlines the overall effects and properties of the various genetic operators (compare to the more usual GA of §2).

K-sexual mating is the operator analogous to crossover. Consider a population $\{\mathbf{R}\}$ of K different region sets **R**. Each set is composed of a number of regions R which together cover feature space. A new region set **R'** is formed by selecting individual regions (one at a time) from parents **R**, with probability proportional to merit μ (merit is the performance of **R** defined at the end of §2). Selection of regions from the whole population of region sets continues until the feature space cover is approximately the average cover of the parents. This creates the offspring region set **R'** which is generally not a partition.

Gene reorganization. For economy of storage and time, the offspring region set **R'** is repartitioned so that regions do not overlap in feature space.

Controlled mutation. Standard mutation operators alter an allele randomly. In contrast, the PLS2 operator analogous to mutation changes an allele according to evidence arising in the task domain. The controlled mutation operator for a region set $\mathbf{R} = \{(r, u, e)\}$ is the utility revision operator of PLS1. As described in §4.1, PLS1 modifies the utility u for each feature space cell r.

Genotype expansion. This operator is also provided by PLS1. Recall the discussion of §3.2 about the economy resulting from compressing genes (utility-feature vectors) into a region set **R**. The refinement operator was described in §4.1. This feature space refinement amounts to an expansion of the genotype **R**, and is carried out when data warrant the discrimination.

Both controlled mutation and genotype expansion promote genotype diversity. Thus PLS1 helps PLS2 to avoid premature convergence, a typical GA problem [Br 81, Ma 84].

4.3. Effectiveness and Efficiency

The power of PLS2 may be traced to certain aspects of the perceptual and genetic algorithms just outlined. Some existing and emerging principles of effective and efficient learning are briefly discussed below (see also [Re 85a, Re 85b, Re 85c]).

Credit localization. The selection of regions for K-sexual mating may use a single merit value μ for each region R within a given set **R**. However, the value of μ can just as well be localized to single regions within **R**, by comparing R with similar regions in other sets. Since regions estimate an absolute quantity (task-related utility) in their own volume of feature space, they are independent of each other. Thus credit and blame may be assigned to feature space cells (i.e. to gene sequences).

Assignment of credit to individual regions within a cumulative set **R** is straightforward, but it would be difficult to do directly in the final evaluation function H, since the components of H, while appropriate for performance, omit information relevant to learning (compare Figs. 2, 4).[13]

Knowledge mediation. Successful systems tend to employ information structures which *mediate* data objects and the ultimate knowledge form. These mediating structures include means to record growing assurance of tentative hypotheses.

When used in heuristic search, the PLS region set mediates large numbers of states and a

13. There are various possibilities for the evaluation function H, but all contain less useful information than their determinant, the region set **R**. The simplest heuristic used in [Re 83a, Re 83d] is $H = \mathbf{b}.\mathbf{f}$, where **b** is a vector of weights for the feature vector **f**. (This linear combination is used exclusively in experiments to be described in §5.) The value of **b** is found using regions as data in linear regression [Re 83a, Re 83b].

very concise evaluation function H. Retention and continual improvement of this mediating structure relieves the credit assignment problem. This view is unlike that of [Di81, p.14, Di82]: learning systems often attempt to improve the control structure itself, whereas PLS acquires knowledge efficiently in an appropriate structure, and utilizes this knowledge by compressing it only temporarily for performance. In other words, PLS does not directly search rule space for a good H, but rather searches for good cumulative regions from which H is constructed.

Full but controlled use of every datum. Samuel's checker player permitted each state encountered to influence the heuristic H, and at the same time no one datum could overwhelm the system. The learning was stochastic: both conservative and economic. In this respect PLS2 is similar (although more automated).

Schemata in learning systems and genetic algorithms. A related efficiency in both Samuel's systems and PLS is like the schemata concept in a GA. In a GA, a single individual, coded as a genotype (a string of digits), supports not only itself, but also all its substrings. Similarly, a single state arising in heuristic search contains information about every feature used to describe it. Thus each state can be used to appraise and weight each feature. (The effect is more pronounced when a state is described in more elementary terms, and combinations of primitive descriptors are assessed — see [Re85c]).

5. EXPERIMENT AND ANALYSIS

PLS2 is designed to work in a changing environment of increasingly difficult problems. This section describes experimental evidence of effective and efficient learning.

5.1. Experimental Conditions

Tame features. The features used for these experiments were the four of [Re83a]. The relationship between utility and these features is fairly smooth, so the full capability of a GA is not tested, although the environment was dynamic.

Definition of merit μ. As §4 described, PLS2 choses regions from successful cumulative sets and recombines them into improved sets. For the experiments reported here, the selection criterion was the global merit μ, i.e. the performance of a whole region *set*, without localization of credit to individual regions. This measure μ was the average number of nodes developed D in a training sample of 8 fifteen puzzles, divided into the mean of all such averages in a population of K sets, i.e. $\mu = \bar{D}/D$, where $\bar{D}$ is the average over the population ($\bar{D} = \sum D_j / K$).

Changing environment. For these experiments, successive rounds of training were repeated in incremental learning over several iterations or *generations*. The environment was altered in successive generations; it was specified as problem *difficulty* or depth d (defined as the number of moves from the goal in sample problems). As a sequence of specifications of problem difficulty, this becomes a training *difficulty vector* $\mathbf{d} = (d_1, d_2, \ldots, d_n)$.

Here **d** was static, one known to be a good progression, based on previous experience with user training [Co84].[14] In these experiments, **d** was always (8, 14, 22, 50, #, #, ...). An integer means random production of training problems subject to this difficulty constraint, while "#" demands production of fully random training instances.

5.2. Discussion

Before we examine the experiments themselves let us consider potential differences between PLS1 and PLS2 in terms of their effectiveness and efficiency. We also need a criterion for assessing differences between the two systems.

Vulnerability of PLS1. With population size K = 1, PLS2 degenerates to the simpler PLS1. In this case, static training can result in utter failure, since the process is stochastic and various things can go wrong (see Appendix B). The worst is failure to solve any problems in some generation, and consequent absence of any new information. If the control structure H is this poor, it will not improve unless the fact is

14. PLS and similar systems for problems and games are sometimes neither fully supervised nor fully unsupervised. The original PLS1 was intermediate in this respect. Training problems were selected by a human, but from each training instance, a multitude of individual nodes for learning are generated by the system. Each node can be considered a separate example for concept learning [Re83d]. [Co84] describes experiments with an automated trainer.

detected and problem difficulty is reduced (i.e. dynamic training is needed).

Even without this catastrophe, PLS1 performs with varying degrees of success depending on the sophistication of its training and other factors (explained in Appendix B). With minimal human guidance, PLS1 always achieves a good evaluation function H, although not always an optimal one. With static training, PLS1 succeeds reasonably about half the time.

Stability of PLS2. In contrast, one would expect PLS2 to have a much better success rate. Since PLS1 is here being run in parallel (Fig. 1), and since PLS2 should reject hopeless cases (their μ's are small), a complete catastrophe (*all* H's failing) should occur with probability $p \leq q^K$, where q is the probability of PLS1 failure and K is population size. If q is even as large as one half, but K is 7 or more, the probability p of catastrophe is less than 0.01.

Cost versus benefit: a measure. Failure plays a part in costs, so PLS2 may have an advantage. The ultimate criterion for system quality is cost effectiveness: is PLS2 worth its extra complexity? Since the main cost is in task performance (here solving), the number of nodes developed D to attain some performance is a good measure of the expense.

If training results in catastrophic failure, however, all effort is wasted, so a better measure is the expected cost D/p, where p is the probability of success. For example, if D = 500 for viable control structures, but the probability of finding solutions is only ½, then the average cost of *useful* information is 500 / ½ = 1000.

To extend this argument, probability p depends on what is considered a success. Is success the discovery of a perfect evaluation function H, or is performance satisfactory if D departs from optimal by no more than 25%?

5.3. Results

Table I shows performances and costs with various values of K. Here p is estimated using roughly 36 trials of PLS1 in a PLS2 context (if K = 1, 36 distinct runs; if K = 2, 18 runs; etc.). Since variances in D are high, performance tests were made over a random sample of 50 puzzles. This typically gives 95% confidence of D ± 40.

Accuracy of learning. Let us first compare results of PLS1 versus PLS2 for four different success criteria. We consider the learning to be successful if the resulting heuristic H approaches optimal quality within a given margin (of 100%, 50%, 25%, and 10%).

Columns two to five in the table (the second group) show the proportion of H's giving performance D within a specified percentage of the best known D (the best D is around 350 nodes for the four features used). For example, the last row of the table shows that, of the 36 individual control structures H tested in (two different) populations of size 19, all 36 were within 100% of optimal D (column two). This means that all developed no more than 700 nodes before a solution was found. Similarly, column five in the last row shows that 0.21 of the 36 H's, or 8 of them, were within 10%, i.e. required no more than 385 nodes developed.

Cost of accuracy. Columns ten and eleven (the two rightmost columns of the fourth group) show the estimated costs of achieving performance within 100% and within 10% of optimum, respectively. The values are based on the expected total number of nodes required (i.e. D/p), with adjustments in favor of PLS1 for extra PLS2 overhead. (The unit is one thousand nodes developed.) As K increases, the cost of a given accuracy first increases. Nevertheless, with just moderate K values, the genetic system becomes cheaper, particularly for an accuracy of 10%.

TABLE 1. COSTS and PERFORMANCES at GENERATION 5.

Pop. Size	Proportion Satisfying Success Criterion (proximity to optimal D)				Mean Nodes Developed (Random Sample of 50)			Cost Per Individual H	Expected Cost of One H Within		% Performance Rough Estimate
K	100%	50%	25%	10%	Avg $\bar{D}$	Best D_b	Pop. D_p	(10^3 nodes)	100%	10%	(for cost=4×10^3)
1	.47	.28	.12	.04	>>654	375	-	12.5	26.6	312	31%
4	.69	.31	.19	.03	573	381	406	17.4	25.2	580	30%
7	.82	.43	.31	.08	531	377	395	18.1	22.7	226	20%
12	1.00	.53	.39	.11	507	384	397	18.7	18.7	170	12%
15	1.00	.69	.61	.14	450	370	390	19.2	19.2	137	9%
19	1.00	.71	.63	.21	453	348	367	19.7	19.7	94	7%

The expected cost benefit is not the only advantage of PLS2.

Average performance, best performance, and population performance. Consider now the third group of columns in Table I. The sixth column gives the *average* $\bar{D}$ for all H's in the sample (of 36). The seventh column gives D_b for the *best* H_b in the sample. These two measures, *average* and *best* performance, are often used in assessing genetic systems [Br 81]. The eighth column, however, is unusual; it indicates the *population performance* D_p resulting when all regions from every set in the population are used together in a regression to determine H_p. This is sensible because regions are independent and estimate the utility, an absolute quantity ([Re 83c], cf [Br 81, Ho 75]).

Several trends are apparent in Table I. First, whether the criterion is 100%, 50%, 25%, or 10% of optimum (columns 2-5), the proportion of good H's increases steadily as population size K rises. Similarly, average, best, and population performance measures $\bar{D}$, D_b and D_p (columns 6-8) also improve with K. Perhaps most important is that the population performance D_p is so reliably close to best, even with these low K values. This means that the whole population of regions can be used (for H_p) without independent verification of performance. In contrast, individual H's would require additional testing to discover the best (column 7), and the other alternative, any H, is likely not as good as H_p (columns 6 and 8). Furthermore, the entire population of regions can become an accurate source of massive data for determining an evaluation function capturing feature interaction [Re 83b].

This accuracy advantage of PLS2 is illustrated in the final column of the table, where, for a constant cost, rough estimates are given, of the expected error in population performance D_p relative to the optimal value.

It is interesting that such small populations improve performance markedly; usually population sizes are 50 or more.

6. EFFICIENCY AND CAPABILITY

Based on these empirical observations for PLS2, on other comparisons for PLS1, and on various conceptual differences, general properties of three competing methods can be compared: PLS1, PLS2, and traditional optimization. In [Re 81], PLS1 was found considerably more efficient than standard optimization, and the suggestion was made that PLS1 made better use of available information. By studying such behaviors and underlying reasons, we should eventually identify principles of efficient learning. Some aspects are considered below.

Traditional optimization versus PLS1. First, let us consider efficiency of search for an optimal weight vector **b** in the evaluation function $H = \mathbf{b}.\mathbf{f}$. One good optimization method is *response surface fitting* (RSF). It can discover a local optimum in weight space by measuring and regressing the response (here number of nodes developed D) for various values of **b**. RSF utilizes just a single quantity (i.e. D) for every problem solved. This seems like a small amount of information to extract from an entire search, since a typical one may develop hundreds or thousands of nodes, each possibly containing relevant information. In contrast to this traditional statistical approach, PLS1, like [Sa 63, Sa 67], uncovers knowledge about *every* feature from *every* node (see §4.3). PLS1, then, might be expected to be more efficient than RSF. Experiments verify this [Re 81].

Traditional optimization versus PLS2. As shown in §5, PLS2 is more efficient still. We can compare it, too, with RSF. The accuracy of RSF is known to improve with $\sqrt{N}$, where N is the number of data (here the number of of problems solved). As a first approximation, a parallel method like PLS2 should also cause accuracy to increase with the square root of the number of data, although the data are now regions instead of D values. If roughly the same number of regions is present in each individual set R of a population of size K, accuracy must therefore improve as $\sqrt{K}$. Since each of these K structures requires N problems in training, the accuracy of PLS2 should increase as $\sqrt{N}$, like RSF.

Obviously, though, PLS2 involves much more than blind parallelism: a genetic algorithm extracts accurate knowledge and dismisses incorrect (unfit) information. While it is impossible for PLS1 alone, PLS2 can refine merit by localizing credit to individual regions [Re 83c]. Planned experiments with this should show further increases in efficiency since the additional cost is small. Another inexpensive improvement will attempt to reward good regions by decreasing their estimated errors. Even without these refinements, PLS2 retains meritorious regions (§4), and should exhibit accuracy improvement better than $\sqrt{N}$. Table I suggests this.

PLS2 versus PLS2. As discussed in §4.1 and Appendix B, PLS1 is limited, necessitating human tuning for optimum performance. In contrast, the second layer learning system PLS2 requires little human intervention. The main reason is that PLS2 stabilizes knowledge automatically, by comparing region sets and dismissing aberrant ones. Accurate cumulative sets have a longer lifetime.

This ability to discriminate merit and retain successful data will likely be accentuated with the localization of credit to individual regions (see §4.2). Another improvement is to alter dynamically the error of a region (estimated by PLS1) as a function of its merit (found by PLS2). This will have the effect of protecting a good region from imperfect PLS1 utility revision; once some parallel PLS1 has succeeded in discovering an accurate value, it will be more immune to damage. A *fit* region will have a very long lifespan.

Inherent differences in capability. RSF, PLS1, and PLS2 can be characterized differently. From the standpoint of time costs: given a challenging requirement such as the location of a local optimum within 10%, the ordering of these methods in terms of efficiency is RSF $\leq$ PLS1 $\leq$ PLS2. In terms of *capability*, the same relationship holds. RSF cannot handle feature interactions without a more complex model (which would increase its cost drastically). PLS1, on the other hand, can provide some performance improvement using piecewise linearity, with little additional cost [Re 83b]. PLS2 is more robust than PLS1. While the original system is somewhat sensitive to training and parameters, PLS2 provides stability using competition to overcome deficiencies, obviate tuning, and increase accuracy, all at once. PLS2 buffers inadequacies inherent in PLS1. Moreover, PLS2, being genetically based, may be able to handle highly interacting features, and discover *global* optima [Re 83c]. This is very costly with RSF and seems infeasible with PLS1 alone.

7. SUMMARY AND CONCLUSIONS

PLS2 is a general learning system [Re 83a, Re 83d]. Given a set of user-defined features and some measure of the *utility* (e.g. probability of success in task performance), PLS2 forms and refines an appropriate knowledge structure, the *cumulative region set* R, relating utility to feature values, and permitting noise management. This economical and flexible structure *mediates* data objects and abstract heuristic knowledge.

Since individual regions of the cumulative set R are independent of one another, both credit localization and feature interaction are possible simultaneously. Separating the task control structure H from the main store of knowledge R allows straightforward credit assignment to this *determinant* R of H, while H itself may incorporate feature nonlinearities without being responsible for them.

A concise and adequate embodiment of current heuristic knowledge, the cumulative region set R was originally used in the learning system PLS1 [Re 83a]. PLS1 is the only system shown to discover locally optimal evaluation functions in an AI context. Clearly superior to PLS1, its genetic successor PLS2 has been shown to be more stable, more accurate, more efficient, and more convenient. PLS2 employs an unusual genetic algorithm having the cumulative set R as a compressed genotype. PLS2 extends PLS1's limited operations of revision (controlled mutation) and differentiation (genotype expansion), to include generalization and other rules (K-sexual mating and genotype reorganization). Credit may be localized to individual gene sequences.

These improvements may be viewed as effecting greater efficiency or as allowing greater capability. Compared with a traditional method of optimization, PLS1 is more efficient [Re 85a], but PLS2 does even better. Given a required accuracy, PLS2 locates an optimum with lower expected cost. In terms of capability, PLS2 insulates the system from inherent inadequacies and sensitivities of PLS1. PLS2 is much more stable and can use the whole population of regions reliably to create a highly informed heuristic (this *population performance* is not meaningful in standard genetic systems). This availability of massive data has important implications for feature interaction [Re 83b].

Additional refinements of PLS2 may further increase efficiency and power. These include rewarding meritorious regions so they become immune to damage. Future experiments will investigate nonlinear capability, ability to discover global optima, and efficiency and effectiveness of localized credit assignment.

This paper has quantitatively affirmed some principles believed to improve efficiency and effectiveness of learning (e.g. credit localization). The paper has also considered some simple but little explored ideas for realizing these capabilities (e.g. full but controlled use of each datum).

REFERENCES

[An 73] Anderberg, M.R., *Cluster Analysis for Applications*, Academic Press, 1973.

[Be 80] Bethke, A.D., *Genetic algorithms as function optimizers*, Ph.D. Thesis, University of Michigan, 1980.

[Br 81] Brindle, A., Genetic algorithms for function optimization, C.S. Department Report TR81-2 (PhD Dissertation), University of Alberta, 1981.

[Bu 78] Buchanan, B.G., Johnson, C.R., Mitchell, T.M., and Smith, R.G., Models of learning systems, in Belzer, J. (Ed.), *Encyclopedia of Computer Science and Technology 11* (1978), 24-51.

[Co 84] Coles, D. and Rendell, L.A., Some issues in training learning systems and an autonomous design, *Proc. Fifth Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, 1984.

[De 80] DeJong, K.A., Adaptive system design: A genetic approach, *IEEE Transactions on Systems, Man, and Cybernetics SMC-10*, (1980), 566-574.

[Di 81] Dietterich, T.G. and Buchanan, B.G., The role of the critic in learning systems, Stanford University Report STAN-CS-81-891, 1981.

[Di 82] Dietterich, T.G., London, B., Clarkson, K., and Dromey, G., Learning and inductive inference, STAN-CS-82-913, Stanford University, also Chapter XIV of *The Handbook of Artificial Intelligence*, Cohen, P.R., and Feigenbaum, E.A. (Ed.), Kaufmann, 1982.

[Ho 75] Holland, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[Ho 80] Holland, J.H., Adaptive algorithms for discovering and using general patterns in growing knowledge bases, *Intl. Journal on Policy Analysis and Information Systems 4*, 2 (1980), 217-240.

[Ho 81] Holland, J.H., Genetic algorithms and adaptation, *Proc. NATO Adv. Res. Inst. Adaptive Control of Ill-defined Systems*, 1981.

[Ho 83] Holland, J.H., Escaping brittleness, *Proc. Second International Machine Learning Workshop*, 1983, 92-95.

[La 83] Langley, P., Bradshaw, G.L., and Simon, H.A., Rediscovering chemistry with the Bacon system, in Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Ed.), *Machine Learning: An Artificial Intelligence Approach*, Tioga, 1983, 307-329.

[Ma 84] Mauldin, M.L., Maintaining diversity in genetic search, *Proc. Fourth National Conference on Artificial Intelligence*, 1984, 247-250.

[Me 85] Medin, D.L. and Wattenmaker, W.D., Category cohesiveness, theories, and cognitive archeology (as yet unpublished manuscript), Dept. of Psychology, University of Illinois at Urbana Champaign, 1985.

[Mic 83a] Michalski, R.S., A theory and methodology of inductive learning, *Artificial Intelligence 20*, 2 (1983), 111-161; reprinted in Michalski, R.S. et al (Ed.), *Machine Learning: An Artificial Intelligence Approach*, Tioga, 1983, 83-134.

[Mic 83b] Michalski, R.S. and Stepp, R.E., Learning from observation: Conceptual clustering, in Michalski, R.S. et al (Ed.), *Machine Learning: An Artificial Intelligence Approach*, Tioga, 1983, 331-363.

[Mit 83] Mitchell, T.M., Learning and problem solving, *Proc. Eighth International Joint Conference on Artificial Intelligence*, 1983, 1139-1151.

[Ni 80] Nilsson, N.J., *Principles of Artificial Intelligence*, Tioga, 1980.

[Re 76] Rendell, L.A., A method for automatic generation of heuristics for state-space problems, Dept of Computer Science CS-76-10, University of Waterloo, 1976.

[Re 77] Rendell, L.A., A locally optimal solution of the fifteen puzzle produced by an automatic evaluation function generator, Dept of Computer Science CS-77-36, University of Waterloo, 1977.

[Re 81] Rendell, L.A., An adaptive plan for state-space problems, Dept of Computer Science CS-81-13, (PhD thesis), University of Waterloo, 1981.

[Re 83a] Rendell, L.A., A new basis for state-space learning systems and a successful implementation, *Artificial Intelligence 20* (1983), 4, 369-392.

[Re 83b] Rendell, L.A., A learning system which accommodates feature interactions, *Proc. Eighth International Joint Conference on Artificial Intelligence*, 1983, 469-472.

[Re 83c] Rendell, L.A., A doubly layered, genetic penetrance learning system, *Proc. Third National Conference on Artificial Intelligence,* 1983, 343-347.

[Re 83d] Rendell, L.A., Conceptual knowledge acquisition in search, University of Guelph Report CIS-83-15, Dept. of Computing and Information Science, Guelph, Canada, 1983 (to appear in Bolc, L. (ed.), *Knowledge Based Learning Systems,* Springer-Verlag).

[Re 85a] Rendell, L.A., Utility patterns as criteria for efficient generalization learning, *Proc. 1985 Conference on Intelligent Systems and Machines,* (to appear), 1985.

[Re 85b] Rendell, L.A., A scientific approach to applied induction, Proc. 1985 International Machine Learning Workshop, Rutgers University (to appear), 1985.

[Re 85c] Rendell, L.A., Substantial constructive induction using layered information compression: Tractable feature formation in search, *Proc. Ninth International Joint Conference on Artificial Intelligence,* (to appear), 1985.

[Sa 63] Samuel, A.L., Some studies in machine learning using the game of checkers, in Feigenbaum, E.A. and Feldman, J. (Ed.), *Computers and Thought,* McGraw-Hill, 1963, 71-105.

[Sa 67] Samuel, A.L., Some studies in machine learning using the game of checkers II — recent progress, *IBM J. Res. and Develop. 11* (1967) 601-617.

[Sm 80] Smith, S.F., A learning system based on genetic adaptive algorithms, PhD Dissertation, University of Pittsburgh, 1980.

[Sm 83] Smith, S.F., Flexible learning of problem solving heuristics through adaptive search, *Proc. Eighth International Joint Conference on Artificial Intelligence,* 1983, 422-425.

APPENDIX A. GLOSSARY OF TERMS

Clustering. *Cluster analysis* has long been used as a tool for induction in statistics and pattern recognition [An 73]. (See "induction".) Improvements to basic clustering techniques generally use more than just the features of a datum ([An 73, p. 194] suggests "external criteria"). External criteria in [Mi 83, Re 76, Re 83a, Re 85b] involve prior specification of the forms clusters may take (this has been called "conceptual clustering" [Mi 83]). Criteria in [Re 76, Re 83a, Re 85b] are based on the data environment (see "utility") below).[15] This paper uses clustering to create economical, compressed genetic structures *(genotypes).*

Feature. A *feature* is an attribute or property of an object. Features are usually quite abstract (e.g. "center control" or "mobility") in a board game. The *utility* (see below) varies smoothly with a feature.

Genetic algorithm. In a GA, a the character of an *individual* of a *population* is called a *phenotype.* The phenotype is coded as a string of digits called the *genotype.* A single digit is a *gene.* Instead of searching rule space directly (compare "learning system"), a GA searches gene space (i.e. a GA searches for good genes in the population of genotypes). This search uses the merit μ of individual genotypes, selecting the more successful individuals to undergo genetic operations for the production of offspring. See §2 and Fig. 3.

Induction. *Induction or generalization learning* is an important means for knowledge acquisition. Information is actually *created,* as data are compressed into *classes* or *categories* in order to predict future events efficiently and effectively. Induction may create feature space neighborhoods or *clusters.* See "clustering" and §4.1.

Learning System. Buchanan et al. present a general model which distinguishes components of a learning system [Bu 78]. The *performance* element *PE* is guided by a *control structure* H. Based on observation of the PE, the *critic* assesses H, possibly localizing credit to parts of H [Bu 78, Di 81]. The *learning* element *LE* uses this information to improve H, for the next round of task performance. *Layered* systems have multiple PE's, critics, and LE's (e.g. PLS2 uses *PLS1* as its PE — see Fig. 1). Just as a PE searches for its goal in problem space, the LE searches in *rule space* [Di 82] for an optimal H to control the PE.

To facilitate this higher goal, PLS2 uses an intermediate knowledge structure which divides feature space into *regions* relating feature values to object *utility* [Re 83d] and discovering a useful subset of features (cf [Sa 63]). In this paper, the control structure H is a linear evaluation func-

15. A new learning system [Re 85c] introduces higher-dimensional clustering for creation of structure.

tion [Ni80], and the "rules" are feature weights for H. Search for accurate regions replaces direct search of rule space; i.e. regions *mediate* data and H. As explained in §3, sets of regions become compressed GA "genotypes". See also "genetic algorithms", "PLS", "region", and Fig. 1.

Merit μ. Also called *payoff* or *fitness*, this is the measure used by a genetic algorithm to select parent genotypes for preferential reproduction of successful individuals. Compare "utility", also see "genetic algorithms".

Object. Objects are any data to be generalized into categories. Relationships usually depend on task domain. See "utility".

PLS. The *probabilistic learning system* can learn what are sometimes called "single concepts" [Di82], but PLS is capable of much more difficult tasks, involving noise management, incremental learning, and normalization of biased data. PLS1 uniquely discovered locally optimal heuristics in search [Re83a], and PLS2 is the effective and efficient extension examined in this paper. PLS manipulates "regions" (see below), using various inductive operations described in §4.

Region or Cell. Depending on one's viewpoint, the *region* is PLS's basic structure for clustering or for the genetic algorithm. The region is a compressed representation of a utility surface in *augmented* feature space; it is also a compressed genotype representing a utility function to be optimized. As explained in [Re83d], the region representation is fully expressive, providing the features are. See §3 and Figs. 3 & 4.

Utility u. This is any measure of the usefulness of an object in the performance of some task. The *utility* provides a link between the task domain and PLS generalization algorithms. Utility can be a probability, as in Fig. 2. Compare *merit*. See §1, 3.

APPENDIX B. PLS1 LIMITATIONS

PLS1 alone is inherently limited. The problems relate to modification of the main knowledge structure, the cumulative region set $R = \{(r, u, e)\}$. As mentioned in §4.1, R undergoes two basic alterations. PLS1 gradually changes the *meaning* of an established feature space rectangle r by updating its associated utility u (along with u's error e). PLS1 also incrementally *refines* the feature space, as rectangles r are continually split.

Both of these modifications (utility revision and region refinement) are largely directed by search data, but the degree to which newer information affects R depends on various choices of system parameters [Re83a]. System parameters influence estimates of the error e, and determine the degree of region refinement. These, in turn, affect the relative importance of new versus established knowledge.

Consequently, values of these parameters influence task performance. For example, there is a tradeoff between utility revision and region refinement. If regions are refined too quickly, accuracy suffers (this is theoretically predictable). If, instead, utility revision predominates, regions become inert (their estimated errors decline), but sometimes incorrectly.[16]

There are several other problems, including difficulties in training, limitation in the utility revision algorithm, and inaccurate estimation of various errors. As a result, utility estimations are imperfect, and biased in unknown ways.

Together, the above uncertainties and sensitivities explain the failure of PLS1 always to locate an optimum with static training (Table I). The net effect is that PLS1 works fairly well with no parameter tuning and unsophisticated training, and close to optimal with mild tuning and informed training [Co84], as long as the features are well behaved.

By nature, however, PLS1 requires features exhibiting no worse than mild interactions. This is a serious restriction, since feature nonlinearity is prevalent. On its own, then, PLS1 is inherently limited. There is simply *no way* to learn utility accurately unless the effects of differing heuristic functions H are compared, as in PLS2.

ACKNOWLEDGEMENTS

I would like to thank Dave Coles for his lasting enthusiasm during the development, implementation and testing of this system. I appreciate the helpful suggestions from Chris Matheus, Mike Mauldin, and the Conference Reviewers.

16. Although system parameters are given by domain-independent statistical analysis, tuning these parameters nevertheless improves performance in some cases. (This is not required in PLS2.)

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-85-1217	2.	3. Recipient's Accession No.

4. Title and Subtitle GENETIC PLANS AND THE PROBABILISTIC LEARNING SYSTEM: SYNTHESIS AND RESULTS	5. Report Date July 1985 6.
7. Author(s) Larry Rendell	8. Performing Organization Rept. No.
9. Performing Organization Name and Address Dept. of Computer Science University of Illinois 1304 W. Springfield Urbana, IL 61801	10. Project/Task/Work Unit No. 11. Contract/Grant No.
12. Sponsoring Organization Name and Address Natural Sciences and Engineering Research Council of Canada 200 Kent Street Ottawa, Canada K1A 1H5 National Science Foundation Washington, DC	13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts

This paper describes new conceptual and experimental results using the probabilistic learning system PLS2. PLS2 is designed for any task in which overall performance can be measured, and in which choice of task objects or operators influences performance. The system can manage incremental learning and noisy domains.

PLS2 learns in two ways. Its lower "perceptual" layer clusters data into economical cells or regions in augmented feature space. The upper "genetic" level of PLS2 selects successful regions (compressed genes) from multiple, parallel cases. Intermediate between performance data and task control structures, regions promote efficient and effective learning.

Novel aspects of PLS2 include compressed genotypes, credit localization and "population performance." Incipient principles of efficiency and effectiveness are suggested. Analysis of the system is confirmed by experiments demonstrating stability, efficiency, and effectiveness.

17. Key Words

machine learning
inductive inference
conceptual clustering
genetic algorithms

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 15
	20. Security Class (This Page UNCLASSIFIED	22. Price