CHAPTER III

THE EVOLUTIONARY WEAK METHOD AND EMERGENT INTELLIGENCE

3.0 The Evolutionary Weak Method and Emergent Intelligence

The commonalities of evolutionary algorithms combined with their lack of task-specific knowledge indicates their status as a weak method which this dissertation calls the *evolutionary weak method*. The major difference between standard weak methods and the evolutionary weak is the adaptive behavior of evolutionary algorithms. Over time, an evolutionary weak method garners information about the search environment and uses this information to constrain future search. Thus evolutionary weak methods begin without any informedness and become increasingly informed during the search. This is unique among the collection of weak methods discussed in the previous chapters.

In this chapter, I show that the evolutionary weak method's technique of reproducing with probabilistic variations defines a weak, i.e. task-independent, approach to credit assignment that has an empirical nature. This form of credit assignment allows the natural constraints of the task environment to emerge as structures in the population without introducing explicit knowledge into the evolutionary weak method. Based on this method of credit assignment, *emergent intelligence*, a new computational approach to intelligence and problem solving, is defined. The goal of emergent intelligence is to reduce or remove AI's reliance on explicit knowledge in a problem solver by favoring instead the *emergence* of problem specific constraints through direct interaction with the task environment.

3.1 Empirical Credit Assignment

Evolutionary weak methods are distinct from standard weak methods because they use past search points to increase their level of informedness during the search. The mechanism that gathers this information during the search, which I call *empirical credit assignment*, works at a coarse grained, task-independent level. It relies on the empirical aspects of evolutionary weak methods and the finite resource of the population for its implementa-

tion. Because it is task-independent, empirical credit assignment is a weak form of credit assignment. The features of empirical credit assignment that distinguish it from strong forms of credit assignment are its representation specific operators, its ability to propagate abstract features and the inherent strength of the empirical process at its core.

3.1.1 Representation Specific Operators

As discussed in the last chapter, standard weak methods are augmented by task-specific knowledge in order to direct their problem solving ability. Task-specific knowledge typically requires an understanding of the task environment to justify its presence in the problem solving algorithm. In other words, task-specific knowledge encodes some aspect of the task environment that is generally invalid outside the task environment. This often includes the definition of task-specific operators that transform a given solution into one better able to solve the problem.

In an evolutionary weak method, the reproduction operators are defined to manipulate a specific representation independent of its interpretation by a fitness function. For instance, genetic algorithms define crossover and point mutation specifically for the manipulation of bit strings. The operators in evolution strategies are similarly defined for the real valued features that are typical in those weak methods. Evolutionary programming also uses representation specific operators to manipulate whatever representation is being used. The distinctions between these methods lie in the amount of representation-specific knowledge they place into their respective reproduction operators. GAs are in a sense the weakest in representation-specific knowledge since the fixed-length bit string is often reinterpreted as a new structure before being evaluated.[1] Manipulations of the bit string by crossover and point mutation can violate the representational constraints of the interpretation. In genetic and evolutionary programming, more knowledge of the representation is provided to the operators to discourage unwarented manipulation. Genetic programming's crossover parses the program and manipulates the representation at the subtree level. Likewise, evolutionary programs also mainpulate at a high representational level. For instance, Angeline, Saunders and Pollack (1994) provide separate reproduction heuristics for manipulating the structural and parametric features of recurrent neural networks. Thus, GAs are representationally weaker than GP and EP and, as with standard weak and strong

1. See Chapter 4 of this dissertation for an example and discussion of this point.

methods, perform less efficient searches and require longer search times for the same tasks. Such a perspective suggests that Holland's (1975) conclusion that binary representations should be preferred in genetic algorithms since they allow maximal implicit parallelism is erroneous and that genetic algorithms using multi-valued string representations should perform quicker but at the expense of generality (cf. Antonisse 1989).

Some GA researchers have investigated the augmentation of a genetic algorithm's operators by task-specific knowledge (Grefensttete 1987; Michalewicz 1993). Given GA's status as an evolutionary weak method, the associated increases in performance demonstrated in these studies is not surprising. Designing operators with a priori knowledge of the task is the basic philosophy of weak method and the subsequent increase in performance should be expected. However, while a short term benefit often results from the injection of knowledge into an evolutionary algorithm's operators, this practice eventually leads to the same difficulties experienced by standard AI strong methods. From the perspective of this dissertation, the addition of representation-specific knowledge should be similarly beneficial without overly narrowing the generality of the technique.

3.1.2 Abstract Feature Propagation

Empirical credit assignment assigns credit at a coarse level to the abstract features of population members. These points were first made about genetic algorithms in Holland (1975). However, the form of the theory in Holland (1975) is too specific to the bit string representations he assumed. In this subsection I generalize Holland's schema so it is applicable to all evolutionary weak methods. First a review of schema is presented.

3.1.2.1 Schema Theory

Holland (1975) hypothesizes that genetic algorithms preserve and recombine at the representational level of schema. A *schema* designates a set of bit string patterns across the assumed fixed-width binary string representation of the population. Designation of schemata take the form $(1, 0, \#)^n$ where n is the length of the binary string representation. A "1" or a "0" at position i in a schema signifies that each string in the set represented by the schema contains that value at that position. A "#" at position i designates strings that contain either a "1" or a "0" at position i are in the set. Notice that the possible schemata for a given length n do not cover all possible subsets of binary strings of length n. For

instance, there is no schema of length four that represents the set {0101, 1010}. The concept of a schema encodes an assumption of sets of features. This is natural given the typical binary feature vector interpretation usually used in genetic algorithms.

Holland (1975) argues that a genetic algorithm assigns credit not to the strings in the population but to schemata of the population, i.e. to sets of features. He observes that assuming schema are assigned credit, feedback to a single population member provides information about the various schemata it is a member in:

> Given l [features], a single structure A ... is an instance of 2^l distinct schemata, as can be easily affirmed by noting that A is an instance of any schema ξ defined by substituting [#'s] for one or more of the l attribute values in [the structure's] representation. (p. 69)

In effect, Holland (1975) argues that while only a single population member is rated by a fitness function, information is garnered about all 2^l schemata of which that string is a member. Holland (1975) calls this property *intrinsic parallelism* and argues that credit assignment (which he called "apportionment of credit") was performed on schema present in the population.

Holland's (1975) *schema theorem* provides the following lower bound for schema growth in a standard genetic algorithm:

$$m(H, t+1) \geq m(H, t) \frac{f(H)}{\bar{f}} [1 - \varepsilon(H, t)] \qquad \text{(EQ 2)}$$

where *m(H, t)* is the number of population members with schema H at time t, *f(H)* is the average fitness of the population members with schema H, $\bar{f}$ is the average fitness of the population and ε*(H,t)* is the probability that schema H is disrupted by an operator. Equation 2 can be rewritten in the following form:

$$m(H, t+1) \geq m(H, t) \frac{f(H)}{\sum_{i \in P_t} f(i)} [1 - \varepsilon(H, t)] \qquad \text{(EQ 3)}$$

where P_t is the population at time t and $f(i)$ is the fitness of the ith population member. The population then becomes a reservoir of useful schema which are recombined into new combinations with assumably higher and higher fitness. This constructive progression is called the *building block hypothesis* (Goldberg 1989a).

While some have questioned various aspects of Holland's theory (Antonisse 1989; Vignaux and Michalewicz 1992; Fogel 1992) it is still the most generally held characterization of processing in genetic algorithms. Schema theory has even been extended to a few more general, fixed-length representational schemes (Goldberg and Lingle 1985; Goldberg 1989; Radcliff 1991). However, schema theory is not general across all evolutionary weak methods since the concept of a schema, and its extentions is linked to fixed-length string representations.

3.1.2.2 Abstract Features

Given that the only feedback to evolutionary weak methods is the fitness values of each population member, components of a particular representation can never be explicitly rated. Further, there is nothing inside the evolutionary weak method shown in Figure 6 that can perform a component level credit assignment. Instead, probabilistic applications of the representation specific operators create new points in the problem space from the current population members. Each new population member represents an experiment testing the features preserved by the applied operator against those that are modified. Successful experiments, i.e. ones that construct comparatively fit individuals, are added to the population and used as the basis of future experimentation. Such an empirical approach leads to the selection of features that are both beneficial for solving the problem and conducive to manipulation by the operators being used.

Because evolutionary weak methods only have information about the performance of the complete individual and not its representational components, evolutionary weak methods propagate any abstract feature, μ, of an individual that is preserved by the reproduction operators. An abstract feature is any aspect of a population member that is preserved during reproduction. For instance, abstract features include any subset of possible values for particular positions in the representations, similar to schemata. However, abstract features are not limited only by the explicit content of the representation. Abstract features

may also be some property of the individual when interpreted in the task environment, for instance, a particular strategy or substrategy for performing the task.

Abstract features differ sharply from Holland's schema theory (Holland 1975) for genetic algorithms. Schemata, and the associated theory, assume that specific organizations of representational components are selected by the genetic algorithm and preserved in the population (Holland 1975). This is the intuition behind the building block hypothesis. But there is nothing in the dynamics of a genetic algorithm, or any evolutionary weak method, that permits it to distinguish between two individuals with distinct structures but identical fitness ratings. Such confusions cause significant problems for genetic algorithms (Angeline, Saunders and Pollack 1994; Schaffer, Whitley and Eshelman 1992; Belew, McInerney and Schraudolph 1992). Radcliff (1991) suggests that there should be "minial redundency" in genetic algorithm representations.

Thus it cannot be the organizations of representational components that are selected by evolutionary weak methods but the more abstract *phenotypic* features, i.e., the qualities the representation possesses when applied to the task. Interestingly enough, when a positional encoding is used, such as is often the case in genetic algorithm, there is no distinction between representational features and phenotypic features. But when the interpretation of the representation is not tied to a particular position of the representation, such as in genetic programming or evolutionary programs, the mapping between structure and behavior is more complex. Selection for representational components in evolutionary algorithms using these more complex interpretation functions cannot occur because the representational features, as well as organizations of them such as schemas, are inaccessible to the credit assignment mechanism.

3.1.3 The Strength of Reproduction

The key to understanding how evolutionary weak methods become stronger during the course of the search is not in the operation of the algorithm over the landscape, but in how an evolutionary weak method manages the population of objects as a resource of the search. Consider that the next set of points investigated by an evolutionary weak method is dependent on the composition of the current population. If the population contains structures that are all dissimilar from the solution, then there is little or no chance that the solution is discovered on the next generation. If instead the population contains a single

member that is very similar to the goal solution, then there is potential for the solution to be constructed as an offspring. As the number population members that are close to a solution increases, the probability of finding the solution in the next generation also increases. This characterization is also consistent with standard weak methods that use a "population" such as beam search and best-first search.

Given the population is a resource for the search, then evolutionary weak methods must allocate the resources of the population so that search effort is concentrated in profitable regions of the space. Assuming that a particular object in the *t*th population, o_i, has a probability of $\sigma(o_i,t)$ to be selected, the expected number of objects in the subsequent generation that are derived from it is:

$$E(o_i) = n\sigma(o_i, t) \qquad \text{(EQ 4)}$$

where n is the population size. Thus, when the chance that an object is selected is better than $1/n$, then the number of objects in the ensuing population that are derived from it is likely to be greater than one, thus allocating more resources to objects that are in the search neighborhood of o_i. Likewise, if the probability of selection is less than $1/n$ the expected number is less than one and less of the search resources is expended in the next generation. The question is what determines the probability of selection for an object.

This result can be refined in the following way. Let an abstract feature of the representation be denoted as μ and let the objects in the *t*th population that have this feature be denoted by $\Lambda(\mu, t)$. Then the number of objects in the subsequent population with this feature, $m(\mu, t+1)$ is:

$$m(\mu, t+1) = n\,[1 - \varepsilon(\mu, t)] \sum_{i \in \Lambda(\mu, t)} \sigma(i, t) \qquad \text{(EQ 5)}$$

where $\varepsilon(\mu,t)$ is the probability that the feature is disrupted by the reproduction operators when constructing the offspring. Note that $m(\mu, t) = |\Lambda(\mu, t)|$. Equation 5 shows that when the cumulative selection probability of all objects in the population with the feature exceeds $m(\mu,t)/n$ and the chance of destroying the feature with the operators is small then the number of objects in the population with the feature is expected to increase. And if the

cumulative selection probability is below $m(\mu,t)/n$ then the number of objects in the population with the feature decreases.

The application of probabilistic reproductive operators modifies features of objects in the population with dynamics as described in Equation 5. If modifications produce an improvement in terms of an increased selection of the object with a feature over other objects in the population, then more objects with that feature are created in future generations. This suggests that the modification is beneficial in the current context of the search and should be allotted more of the resource reserves. If a crucial feature of the object is removed by the reproduction operators, then the modified object has less of a chance to propagate its features to other components of the population and the share of resources for those features declines.

The manipulation of the population as a resource for specific features by the evolutionary weak method highlights a weak *empirical* approach to credit assignment. Consider that when no information is present about what features of an object are important, information is gained by selecting some aspect of the object, modifying it and observing a result in terms of the fitness function. Such an experiment is not a sufficient basis for a complete induction of the global worth of the feature. It is important then to gather sufficient statistics of the worth of specific features in several contexts. Features found to be beneficial in several contexts may be indicative of a necessary feature of the solution.

In essence, the number of objects in the population with a given feature can be construed as an approximation of the importance of the abstract feature over the previous empirical contexts. Thus, the population itself serves as a reservoir of features that have previously been found to associate with better solutions. This leads to the conclusion that the size of the population determines the accuracy of this method of credit assignment. Stronger methods for empirically determining credit assignment could be constructed by simply allowing a larger amount of space for storing information about the search.

Recall from Equation 5, the expected number of population members at time $t+1$ which contain feature μ is given by:

$$m(\mu, t+1) = n\,[1 - \varepsilon(\mu, t)] \sum_{i \in \Lambda(\mu, t)} \sigma\,(i, t) \qquad \text{(EQ 6)}$$

where n is the population size and $\varepsilon(\mu, t)$ is the probability that the feature is disrupted during reproduction. Equation 6 can be rewritten as follows:

$$m(\mu, t+1) \quad = \; n\,[\,1 - \varepsilon(\mu, t)]\,\frac{m\,(\mu, t)}{m\,(\mu, t)} \sum_{i \in \Lambda(\mu,\, t)} \sigma\,(i, t) \qquad \text{(EQ 7)}$$

$$m(\mu, t+1) \;=\; n[\,1 - \varepsilon(\mu, t)]\;\; m(\mu, t)\;\; \bar{\sigma}(\mu, t) \qquad \text{(EQ 8)}$$

where $\bar{\sigma}(\mu, t)$ is the average probability of selection for a member of $\Lambda(\mu, t)$. Notice that when μ is a schema, $\bar{\sigma}$ is defined in accordance with fitness proportionate reproduction and ε is defined to adjust for crossover and point mutation then the lower bound expression for the schema theorem of Holland (1975) shown in Equation 3 is recovered.

The interest in Equation 8 for abstract feature propagation stems from its characterization of the properties of reproduction as the relative fitness of the population members with and without μ change. As long as $\bar{\sigma}(\mu, t) > 1/(n - n\varepsilon(\mu, t))$, the number of population members with the feature is likely to be larger in the next generation. On the other hand, if $\bar{\sigma}(\mu, t) < 1/(n - n\varepsilon(\mu, t))$ then the number of population members containing μ is likely to decrease. In other words, as long as μ presents a sufficient selection advantage to the subpopulation that contains it, additional population members tend to acquire the feature. When μ is no longer an advantage, the feature is removed from the population automatically by the natural dynamics of the evolutionary weak method.

Equation 8 characterizes the empirical power of the reproductive process used in all evolutionary weak methods. It is this strength that separates evolutionary algorithms from other search and optimization techniques and provides the adaptive capacity that separate them from standard weak methods. Of equal importance is the generality and exploitability of this reproductive process. For example, Davis (1991) and Schwefel (1981) describe different methods for evolving the parameters for manipulating population members for two different evolutionary weak methods.

The necessity of a population for empirical credit assignment does not imply that standard weak methods that use populations, such as best-first search and beam search, embody empirical credit assignment: a population is necessary but not sufficient. In

empirical credit assignment, the reproduction operators must map between members of the problem space independent of task-specific knowledge. Evolutionary weak methods accomplish this by defining representation-specific operators for manipulation rather than task-specific operators as in standard weak/strong methods. To the extent that best first or beam search algorithms use representation specific operators and not task-specific operators, then these weak methods use a form of empirical credit assignment. However, if the operators that create new population members from old employ task-specific knowledge, as is intended in beam search and best-first search, then the assignment of credit is strong.

In short, evolutionary weak methods gather a degree of information about the search space by modifying the content of the population so that it contains abstract features that consistently associate with objects of better than average fitness. Evolutionary weak methods become informed about the search space and the features important for solving the problem during the run by manipulating the contents of the population. Informedness is a quality consistently missing from the standard weak methods described above. The types of features preserved and passed on to subsequent populations are determined chiefly by the reproduction operators and the method of selecting which objects in a population are used to create the next.

3.2 Emergent Intelligence

Empirical credit assignment uses the fitness function as an integral part of its computation. Unlike stronger forms of credit assignment, empirical credit assignment does not deduce the validity of representational components but empirically determines an abstract feature's validity though modification and retesting. This sets up a strong interaction between the evolutionary weak method and the task environment. Through this interaction, solution constraints that are implicit within the task environment become manifested in the population as they are discovered. Viewing the system at the knowledge-level (Newell 1982), an observer would deduce that the evolutionary weak method possesses task-specific knowledge that lead to the solution's structure. Given that there is no task-specific knowledge in the problem solver, the interaction of the evolutionary weak method with the task environment allows knowledge to emerge at the knowledge-level that is not available to the problem solver as explicit knowledge.

3.2.1 Emergence and Emergent Computation

Emergent intelligence is a direct generalization from the properties of empirical credit assignment and evolutionary weak methods that exploits the task environment during problem solving. Emergent intelligence suggests that interaction of a simple problem solver with a task environment can remove the need for task-specific knowledge internal to the problem solver. Before a more adequate explanation of emergent intelligence is given, this section provides a discussion of the term *emergence* and its implications for computation.

3.2.1.1 Emergence and Emergent Properties

Emergence and emergent phenomena are generally associated with a compositional process that results in epiphenomena that are apparently beyond the composition of components. In standard compositions, the content of a complex structure is exactly the sum of its component parts. Emergence suggests a phenomena where the global attributes are more than the sum of their individual parts. Consider, for instance, the English phrase "kick the bucket." The meaning of the component "kick" would be "to exert a force using the foot and leg" and the meaning of the component "the bucket" would be "a medium sized vessel used for carrying liquids." The compositional meaning for this phrase is just the juxtaposition of the individual meanings giving something like "exert a force using the foot and leg on a medium sized vessel used for carrying liquids" which is the literal translation. However, this phrase has an alternative meaning, one that stands for dying. This alternative meaning may have arisen from the interaction of humans using the phrase in a particular social context, possibly one of farmers being seriously injured while milking a cow. Such phrases with alternative meanings that have arisen through human interaction warrant their own class of phrases called *idioms*. Their alternative meanings emerge from the interaction of the humans in the social settings and bear little or no resemblance to their literal compositional meanings.

A similar emergent progression can be seen in our own biological environment, the original proponent of the concept of emergence. Consider the progression from physics, to a chemistry that sustains structure, to a biology that sustains life, to a social organization that sustains culture. At each of these distinct levels of organization, new properties that

are specific to that level and exploited by the level above arise out of simple interactions of lower levels.

In a more cognitive science vein, Searle (1992) theorizes that consciousness is a higher-order emergent phenomena that results from the interaction of neurons in the brain. He defines emergent properties of a system in the following way:

> Suppose we have a system, *S*, made up of elements *a,b,c*... In general, there will be features of *S* that are not, or not necessarily, features of *a,b,c*... Let us call such features "system features."... Some system features can be deduced or figured out or calculated from the features of *a,b,c*... just from the way these are composed and arranged (and sometimes from their relations to the rest of the environment). But some other system features cannot be figured out just from the composition of the elements and environmental relations; they have to be explained in terms of the causal interactions among elements. Let's call these "causally emergent system features." (p 111)

Searle (1992) gives examples for standard systems features and "causally emergent system features." Standard system features of a stone include its weight and shape. Both of these qualities of a stone can be deduced or calculated from the properties of the molecules that comprise the stone. Searle lists solidity, liquidity and transparency as examples of causally emergent system features. Searle continues:

> ... consciousness is a causally emergent property of systems. It is an emergent feature of certain systems of neurons in the same way that solidity and liquidity are emergent features of systems of molecules. The existence of consciousness can be explained by the causal interactions between elements of the brain at the micro level, but consciousness cannot itself be deduced from or calculated from the sheer physical structure of the neurons without some additional account of the causal relationships between them. (p 112)

Consciousness for Searle (1992) arises as an epiphenomenon of the interactions of the neurons. Rather than labeling the effect as epiphenomenal, Searle prefers to call it "caus-

ally supervenient" and argues that the mapping from neurophysiological states to mental states is a many-to-one causal mapping. Regardless, the base effect is that of an epiphenomenon. If we consider the interactions of the elements of the systems to be contained in a single dimension, Searle's causally emergent systems phenomena comes out of this line at a right angle creating a new dimension. Searle's causal supervenience suggests that features of this new dimension are dependent but not necessarily uniquely dependent on certain interactions in the original line. Thus, a phenomena can emerge but it may be indeterminate which low level interactions produced it.

To further distinguish causally emergent system phenomena, Searle describes a second type of emergent property:

> [The above definition] of causal emergence, call it "emergent1," has to be distinguished from a much more adventurous conception, call it "emergent2." A feature *F* is emergent2 iff *F* is emergent1 and *F* has causal powers that cannot be explained by the causal interactions of *a,b,c...* If consciousness were emergent2 then consciousness could cause things that could not be explained by causal behavior of the neurons. (p 112)

Searle states that he does not know of anything that is emergent2. However, phenomena such as telekinesis, telepathy and extra-sensory perception (ESP), if they existed, are described well by such a definition.

For Searle (1992), causality plays an important role in emergent phenomena. While it is difficult to pin down exactly what Searle means by a causal relation, it is safe to say he feels emergent phenomena result from certain types of interactions between lower level components.

3.2.1.2 Emergent Computation

The reliance on emergence as a computational feature of systems is relatively uninvestigated. The very concept of harnessing epiphenomena for computational gains seems counter-intuitive to the logical progressions of a computer. However, systems from fields such as connectionist models (Rumelhart and McClelland 1986; McClelland and Rumelhart 1986a), classifier systems (Holland et al. 1986), cellular automata (Langton 1992;

Wolfram 1984), artificial life models (Langton et al. 1992; Langton 1989), and the study of social cooperation (Axelrod 1984) do exactly this.

Summarizing the features of these systems, Forrest (1991) defines *emergent computation* as having three parts:

1. A collection of agents, each following explicit instructions;

2. Interactions among agents (according to the instructions), which form implicit global patterns at the macroscopic level i.e., epiphenomena;

3. A natural interpretation of the epiphenomena as computations.

In other words, the object of the computation is not a single functional result but a global state of the system after processing. These global states are achieved by local computations. What is most interesting in Forrest's definition is the understated role of an observer. The third part of the definition specifies that the emergent property has a "natural interpretation" as a computation. This implies an observer is making an interpretation of the epiphenomena. In spite of the difficulties of including an observer as an interpreter of the phenomena (Kolen and Pollack 1994), it is imperative for an observer to be included in the definition since the emergent properties may be computationally expensive to identify.

Forrest (1991) describes several potential advantages for evolutionary computations over standard computational mechanisms:

> First, implementing computations indirectly as emergent patterns may provide implementation efficiencies because of the need for less control over the different components.... Second, flexibility is important for systems that must interact with complex and dynamic environments, e.g. intelligent systems. For these systems, it is impossible to get enough flexibility from explicit instructions; for realistic environments it is just not possible to program in all contingencies ahead of time. Therefore, the flexibility must appear at the emergent level.... Third, the advantage in representation arises

> in systems for which it is difficult to articulate a formal description of the emergent level. (p 3)

Because little theoretical research is available on these methods, the limitations of emergent computations in terms of computational complexity are not known. Forrest (1991) suggests that:

> ... emergent computations are almost certainly [languages more expressive than] context free since they arise from interactions among components. However, the low-level instructions that generate emergent computations may well be context free. (p 5)

Forrest (1991) draws an analogy between the literal compositional semantics of programming languages and the superposition principle in physics. She argues that the standard compositional approaches that do not interact result in a straight-forward, i.e., linear composition of the constituents. However, with interactions, non-linear organizations can arise that take advantage of properties such as chaos, solitons and self-organization.

3.2.2 Emergent Intelligence: The Very Idea

Emergent intelligence (EI) is any problem solving algorithm that uses emergence as an integral component of the problem solving process such that the amount of explicit knowledge required to implement the same problem solving ability is reduced. The intention is that appropriate knowledge emerges at the knowledge-level as an epiphenomenon of the problem solving process. Rather than providing all necessary knowledge as explicit knowledge to the problem solver, EI algorithms allow knowledge that is pertinent to the specific problem solving instance to emerge from the natural dynamics of the problem solver interacting with the task environment.

Emergent intelligence relies on the following two assumptions about computational problem solving generalized from the properties of evolutionary weak methods:

1. The task environment itself is a more concise representation for knowledge specific to the task than explicit knowledge.

2. Direct interaction of a simple problem solver with the task environment permits the task environment's inherent constraints to be expressed naturally during the problem solving process. As a result, pertinent task-specific knowledge *emerges* from the interaction of the problem solver with the constraints of the task environment.

The task environment implicitly contains all knowledge associated with the task. This is similar to how the rules of the game implicitly define equivalence classes of strategies without explicitly describing any single strategy. Knowledge pertaining to any abstract feature that is distinguishable by the task environment can emerge through interaction with an evolutionary weak method as long as the representation-specific operators in use do not discourage the abstract feature's propagation.

In short, emergent intelligence uses the above assumptions to remove explicit knowledge that is a natural consequence of the problem solving process interacting with the task environment. Emergent intelligent techniques work "as if" they had explicit knowledge because the original source of the constraints, the problem specification itself, is available to the problem solving algorithm.

3.2.2.1 Necessity of Interaction with the Task Environment

From Forrest's (1991) definition above, there must be a local interaction between agents for emergent computation to result. In emergent intelligence, the task environment itself must be available for interaction with the other agents. By allowing the task environment to be separate from the problem solving algorithm, all the natural constraints, including those too subtle for a knowledge engineer to extract, are available to the algorithm and emerge while solving problems.

An interesting consequence of relying on the task environment as a source of available knowledge during problem solving is that if the task is modified, no modification of the problem solver is necessary. For instance, if in an evolutionary weak method a more accurate solution is required, then only a task environment that distinguishes between finer differences in the solutions is needed. This is in direct contrast to knowledge-based AI techniques where the knowledge-base must be augmented.

In a sense task-specific knowledge represents the "common sense" of a task environment necessary to solve the task. AI traditionally views common sense as "given" internal knowledge, to the point that a sort of common sense knowledge-base is being built (Lenat, Prakash and Shepard 1986). However, at least one psychological theory considers certain portions of common sense to be directly available to an agent by virtue of being in the environment (Gibson 1966; 1977; 1986).

3.2.2.2 Task-Specific versus Representation-Specific Knowledge

Emergent intelligent systems encode a general approach to problem solving by relying on representaion-specific knowledge rather than task-specific knowledge. *Representation-specific knowledge* is knowledge about the representation, i.e., only that it is a tree or array or a simple bit string, but falls short of using the interpretation of the representation as a motivation for processing.

In task-specific knowledge, components of a representation are often designated as representing some significant feature of the task environment. For instance, "IS-A" links in some representations are used to denote a membership relation that is task-specific. Without the task environment, the link between the member and its superset denoted by an "IS-A" link is arbitrary. Representation-specific knowledge and processing avoids relying on such arbitrary associations.

3.2.2.3 The Role of the Observer in Emergent Intelligence

The interaction of a problem solver with the problem environment allows appropriate knowledge to "emerge" as a by-product. From the outside, a program that has a large reserve of knowledge and knows how to locate knowledge pertinent to the current problem is no different than one that allows interaction with the task to limit the computation in a similar manner. Both methods are equally intelligent, since both solve the problem. Both are also equivalent at the knowledge-level (Newell 1982) since they have access to exactly the same knowledge. The difference lies only in the representation of the pertinent task-specific knowledge.

As in Forrest's (1991) definition of emergent computation, the knowledge-level relies on an observer to provide an analysis of the behavior of a program. Generally, the fact that behavior must be observed is ignored in AI resulting in a knowledge-level reductionist

approach. In systems such as GPS (Newell and Simon 1969) observations of humans at the knowledge-level were codified and "animated" by a computer program. The weak methods are abstractions of this reductionist approach. Even today in systems such as SOAR (Laird, Rosenbloom and Newell 1986a; Laird, Rosenbloom and Newell 1986b; Newell1990), this reductionist approach persists. But merely animating a knowledge-level description only demonstrates the consistency of the description and has no explanatory power for the necessary properties of the lower representation and implementation levels, beyond the need for universality. The results of Kolen and Pollack (1993; 1994) imply that a knowledge-level description can actually make a simple lower level system appear overly complex.

3.2.2.4 Non-referential Symbols and Representations

One of the implications of emergent intelligence is that the constructed results may not be consistent with a human problem solver's pre-conceptions about many aspects of intelligence. As in the distinction between apparent and actual knowledge suggested in the first chapter, the symbols and representations within an emergent intelligent program, or its products, may not refer to easily identifiable aspects of the task environment. They instead encode important task environment features in a more functionalist manner. This distinction is similar to the concept of distributed representations of connectionist networks (Hinton, McClelland and Rumelhart 1986; Smolensky 1988) and the differences of first-person and third person symbols discussed by Clancey (1993).

3.2.2.5 A Comparison Between EI and AI

The distinction between knowledge-based AI techniques and emergent intelligence are best illustrated by an example. Consider the pattern of a snowflake. Each snowflake has at its center a crumb of dust that forms the snowflake's skeleton. Ice crystals begin forming on the dust particle and successive layers are deposited with certain symmetrical patterns. The fluctuation of humidity, pressure, and temperature at the exposed surfaces of the flake at the instant of the next layer's crystallization determine the exact shape of the patterns. The uniqueness of a snowflake's shape is due to the sensitivity of the physical process constructing the flake to these variations. The snowflake's symmetry emerges as a natural constraint of the current structure interacting with the environment, i.e. snowflakes form "as if" they had knowledge of symmetry. This is the sense that emergent intelligence

allows task-specific knowledge to emerge "as if" the algorithm had the knowledge explicitly represented. Because the constraint is a natural consequence of the interaction of the snowflake's structure with the environment there is no need to represent it explicitly. It emerges naturally as a consequence of the interaction. In contrast, the standard AI strong method for determining the shape of a snowflake would be to place a rule in the knowledge base that all snowflakes must be symmetrical along with other rules for deducing a snowflake's final shape directly from the myriad of variations of humidity, pressure and temperature.

A summary of the comparison between EI and AI, discussed throughout this chapter, is shown in Table 1. Properties such as the nature of operators used in the techniques and

Table 1: *Differences between artificial intelligence and emergent intelligence.*

Property	*Artificial Intelligence*	*Emergent Intelligence*
task-specific knowledge	integrated into problem solver	separate from problem solver
operators	task-specific	representation-specific
processing	centralized and global	local and opportunistic
knowledge representation	explicit	emergent
knowledge content	static	dynamic and adaptive
credit assignment	task-specific and static	representation specific, empirical and dynamic

credit assignment are taken from the discussion of evolutionary weak methods while the others are due to the generalization to emergent intelligence.

3.2.3 Review of Related Approaches

There are several alternative approaches in the artificial intelligence and psychology communities that are related to the theory offered here. This section describes the similarities and differences between some of these approaches and emergent intelligence.

3.2.3.1 Society of Mind

Minsky (1985) and Papert (1980) both offer a model of mind as a society. However, these computational societies are ordered where each agent has a particular job in the overall computation. As a result the connections between components in the society are hard coded.

Minsky's (1985) and Papert's (1980) theories of intelligence are only simple decomposition of explicit knowledge and therefore are subject to the same criticisms as given in Chapter 1 for standard AI.

3.2.3.2 Embodiment

Brooks (1986; 1991) takes the radical view that nothing short of an actual physical environment is sufficient to investigate intelligence on the level of humans. He claims that only by immersing our experiments into the same physical environment can information be garnered about human intelligence. He also claims that intelligence can arise from systems that have no actual representation. Further, Brooks decomposes the interaction of his "creatures" with the environment along functional terms but abandon's this approach when actually constructing his "creatures" (Saunders et. al. 1992).

Verela, Thompson and Rosch (1992) concur with Brooks' approach and provide the term *embodiment* as a label for the perspective. While their view of embodiment investigates the questions of mind and consciousness rather than that of motion it involves the same commitment to immersion into an environment.

Brook's radical view of investigating intelligence, while along the same lines as emergent intelligence, is a bit too extreme. His emphasis on including the environment is exactly correct, but there is much to be learned from simulated environments that apply to instantiating intelligence in actual physical environments. Furthermore, Brooks' view of representation being unnecessary for intelligent behavior is also consistent with emergent

intelligence, however, emergent intelligence leaves open the possibility for augmentation of intelligence through additional internal representations. Lastly, in Brook's methodology, only behavior is emergent, although he does not describe it as emergent. Emergent intelligence suggests that more than just appropriate behavior in an environment can emerge from a system.

3.2.3.3 Situated Action

Work in planning and problem solving performed by Agre and Chapman (1987), Chapman (1989), Suchman (1987), Winograd and Flores (1986), and Greeno (1989) has come to be termed situated action (SA). Situated action concentrates on the emergence of appropriate behavior from an actor interacting with a complex environment. The proponents of SA are sympathetic to Gibson's theory of ecological perception (Gibson 1966; 1986) and his theory of "affordances" (Gibson 1977).

Situated action researchers tend to hold the following assumptions (adapted from Vera and Simon 1993):

1. SA actors require little or no internal state or planning apparatus to implement complex behaviors.

2. SA actors interact with the environment on purely functionalist principles.

3. Goals are properties of the environment rather than the actor.

There are significant similarities between SA and the work of Brooks (1986; 1991). In fact, Vera and Simon (1993) include Brooks (1991) as an instance of situated action. Situated action shares a commitment to a separate task representation with Brooks as well as the perspective that knowledge need not be explicitly represented. However, SA researchers are not as radical about immersion into the physical environment as Brooks. They are content to use simulated environments for their experiments. In addition, SA researchers are content to use explicit knowledge within the agent, as evidenced by Agre and Chapman (1987 and Chapman (1989). SA agents merely address the necessity of explicit goals in literal knowledge-level models. While plan knowledge can be said to emerge from SA agents, this is the only type of knowledge that emerges.

Emergent intelligence takes no position on the necessity of internal representations. It merely suggests that there is a class of knowledge that need not be represented internally because it is a natural consequence of interacting with the environment. Whether or not the knowledge should be explicitly represented is a consequence of the properties of the problem solver and the task environment. Emergent intelligence also makes no commitment on the type of interaction between the actor and the environment, only that they must be separate and interaction must occur. Finally, EI admits that goals can be a property of the environment but that an agent can also have goals that augment processing.

As in Brooks' work, SA stops at the supposition that appropriate complex behavior can emerge from the interaction of a simple actor with a complex environment. No consideration is given to EI's stronger claim of inducing modifications in the actor using emergent techniques. This is perhaps the most fundamental difference between SA methods and those described in this dissertation.

3.3 Innate Emergent Intelligence in Genetic Programming

The standard definition of genetic programming provides innate emergent properties that naturally assist in the acquisition of solutions. These properties emerge not because they were designed into the genetic programming process but because the dynamics of the genetic program determine them to be useful or necessary for success. I describe them here as an example undesigned emergence.

3.3.1 Emergence of Introns

One of the most misinterpreted aspects of genetic programming (Koza 1992), and one that shows the tension between human programming preconceptions and the dynamics of GP very clearly, is the redundancy in evolved genetic programs. Several solutions evolved by Koza (1992) contain semantically extraneous components. These components are unnecessary since they can be removed from the program without altering the solution the program represents. Such superfluous constructions appear in the resulting expression trees so often that Koza uses an automatic editor to remove them at the conclusion of a run.

The ubiquity of redundant constructs suggests a more basic truth about their nature. There is an advantage, from the viewpoint of the evolving genetic program, for syntacti-

cally redundant constructions that are semantically transparent. Because the redundant functions do not effect the semantics of the sub-expression, if a crossover operation is performed such that it splits the redundant components, the semantics of the removed subtree are still preserved.

These redundant functions serve as *introns* for the evolving subexpression giving the blind crossover operator a higher probability of capturing the complete subexpression intact. An intron is a portion of a chromosome that is never expressed that provides spacing between the genes. Levenick (1991) shows that introducing introns into a genetic algorithm's representation improves the success rate as much as a factor of 10. Work by Forrest and Mitchell (1992), investigating a class of fitness functions, called the *royal road functions*, showed the standard genetic algorithm could only solve their problem when introns were introduced into the representation. Separating the expressed components of the bit string with padding, crossover has a better chance of transferring a substring intact. The situation is the same in genetic programs. By introducing redundant syntactic structures, which are semantically unexpressed, crossover in GP has a better chance of transferring a complete subtree rather than a partial one. The advantage of GP introns is that genetic programming introns appear dynamically, i.e., they emerge naturally as a consequence of the dynamics in the algorithm rather than being designed into the bit string. It is important then to not impede this emergent property as it may be crucial to the successful development of genetic programs. In fact, encouraging redundancy by supplying primitive functions that are not completely orthogonal may actually assist the evolutionary process.

3.3.2 Emergent Diploidy and Dominance

Another interesting form of emergence occurs when the primitive language for a genetic program includes a conditional statement with semantics of the form *if* <*condition*> *then* <*action1*> *else* <*action2*>. When this statement is available, the otherwise haploid structure of the genetic program can create a task dependent diploidic representation with some interesting dominance characteristics. Consider that when the conditional is executed, the <*condition*> is first evaluated and based on its result either <*action1*> or <*action2*> is evaluated. In the event that the condition is independent of the state of the environment, then the conditional represents a static form of diploidy with the dominant action determined by the condition. So, for instance, with a condition that always computes to false, <*action2*> would be the dominant subtree in the individual. The "reces-

sive" subtree is never executed but, in the matter of a few generations, could become dominate in an offspring if required. This is no different than the form of diploidy and dominance explicitly designed into some genetic algorithm representations (Goldberg 1989a).

Unlike standard forms of dominance computation, dominance in these genetic programs is not designed into the representation but emerges dynamically. The <*condition*> is local to the conditional statement and thus is itself mutable by the evolutionary process. Consequently, the dominance of the actions is specific to the individual. Beneficial dominance relations emerge from the interaction of fit individuals with the task. In addition, the dominance calculation need not be static but can be dependent on the context of the situation. When a <*condition*> uses elements of the environment's state, then the dominance of the various actions is determined dynamically by the context of the environment. This allows a more flexible, environment specific form of dominance.

3.4 Demonstrations of Emergent Intelligence

The remainder of this dissertation is devoted to demonstrating various aspects of emergent intelligence on various problems using evolutionary weak methods as the computational substrate. Each of these experiments describes an evolutionary weak method that is designed to acquire solutions that would require explicit knowledge in a knowledge-based AI technique. The first experiment, in Chapter 4, demonstrates how an evolutionary program, using only representation specific operators, can induce both the weights and architecture for recurrent neural networks. Chapter 5 shows that a small, representation-specific modification to the program of Chapter 4 allows a more directed search. In Chapter 6, modular LISP programs are evolved using a variant of genetic programming. In this chapter the knowledge to determine an appropriate LISP model for the task emerges. Finally, the experiments of Chapter 7 demonstrate that goal-directed behavior can emerge without goals being represented.