CHAPTER I

INTRODUCTION: SEARCH AND EXPLICIT KNOWLEDGE

1.0 Introduction: Search and Explicit Knowledge

Two broad goals cover most of the subtopics within artificial intelligence (AI) research (Schank 1987). One goal is the study of human intelligence and problem solving by creating computational models (e.g. Newell and Simon 1963; Anderson 1976, 1983; Minsky 1985; McClelland and Rumelhart 1986a; Newell 1990; and many others). Such cognitive modeling work is concerned mainly with issues of human psychological performance, but not necessarily making programs intelligent, although this is occasionally assumed. AI's alternative goal is the creation of techniques that allow computers to perform well at tasks that humans are currently more proficient (e.g. Samuel 1959; Shortliffe 1976; Berliner 1977; Lindsay et al 1980; Erman et al. 1980; and many others). Like its cognitive modeling alter-ego, this AI methodology often involves considering human intelligence for computational inspiration. However, it is not necessary to recapitulate human cognitive abilities for a computer to perform intelligently.

A commonality in both forms of AI is *search*. Search is the topic from which most other AI subtopics specialize, including learning, discovery, pattern recognition, classification, diagnosis and design. Search methods for such problems fall into two categories, strong and weak. *Strong methods* are rich in task-specific knowledge that a programmer or knowledge engineer places explicitly into the search mechanism. Strong methods tend to be narrowly focused but fairly efficient in their ability to identify domain specific solutions. *Weak methods* encode search strategies that are task independent and consequently less efficient. Strong methods often use one or more weak methods working underneath the task-specific knowledge. In spite of the central role of search in AI, the focus of AI is to solve problems. Given this fact, this dissertation often refers to a particular search task as an instance of *problem solving*. Likewise, an algorithm that performs problem solving is referred to as a *problem solver.*

Often a task is too difficult to engineer out sufficient knowledge for its solution, or the goal is to discover solutions too complex for humans to construct. In these cases, there may be no strong method to rely on and the augmentation of a weak method by task-specific knowledge is infeasible. Such complex problems still fit well within the AI charter, but by virtue of their complexity traditional AI techniques can not adequately solve them. Thus, the strong vs. weak dichotomy presents a constrained methodology for creating intelligent programs in AI; either explicitly engineer out task knowledge or analytical solutions for a strong method or be doomed to the inefficient search of weak methods.

The focus of the following work is twofold, as reflected in the title of this dissertation. The primary focus is to investigate algorithms that do not require *explicitly engineered task knowledge* for search but still perform as strong methods. These methods fit into a class of techniques that this dissertation defines as *emergent intelligence*. Of particular interest is the investigation of emergent intelligence using evolutionary algorithms, algorithms that use an analogy to natural evolution as their base computational engine. Before discussing these points and the content of this dissertation further, the following sections discuss the approach of knowledge-based artificial intelligence, its reliance on explicit task knowledge and the classic problems which result.

1.1 Literal Knowledge Level Models

Newell (1981, 1982) defines two levels of observation that can be used to describe the intelligence contained by any agent, including a computer. The first, the *symbol level*, describes the representations and the program used by the agent. This level roughly corresponds with Marr's (1981, 1982) representation and algorithm level. The *knowledge level* is a level of description above the symbol level. A description of an intelligent agent at the knowledge level has four components. The first is a set of *actions* that can be executed by the agent. Second is a collection of *goals* for the agent that provide a context for its actions. The third component is *knowledge* that allows the agent to achieve the goals through the execution of actions. The final component is what Newell (1981, 1982) calls a *principle of rationality.* This principle dictates that if the agent contains knowledge that is applicable in the current context and will satisfy one of its goals, then a rational agent will apply the knowledge to achieve the goal.

The simplest manner in which to model observed intelligent behavior is to take the observations at the knowledge level at face value and implement them directly at the symbol level. All that is needed is a representation for knowledge and goals and an algorithmic version of the principle of rationality. Knowledge and goals are extracted through analysis of the task environment. After the knowledge and goals are extracted, weak methods can serve as an appropriate principle of rationality, assuming that control knowledge appropriate for the task is added for efficiency. Newell echoes this modeling approach by suggesting that the knowledge level can be reduced to the symbol level. In this reduction, the knowledge and goals at the knowledge level correspond simply to data structures in the program:

> When [AI researchers] say, as we often do in explaining an action of a program, that the "program knows K" ... we mean that there is some structure in the program that we view as holding K... (Newell 1981, p. 15)
>
> The discovery, development and elaboration of [the symbol level] to predicting the behavior of an intelligent agent has been what AI has been all about... (Newell 1981, p. 12)

I label such models of intelligence, where there is a direct correspondence between the knowledge described at the knowledge level and the content of the representations at the symbol level, as *literal knowledge level models* of intelligent behavior. Such models represent the majority of knowledge-based AI systems.

It is important to note that what is observed at the knowledge level may not accurately reflect the process that gave rise to the observations. In other words, there may be a disparity between the *apparent* knowledge of the agent as seen at the knowledge level and the *actual* knowledge contained in the implementation of the agent at the symbol level. Knowledge-based AI has long confused the implementation of intelligence within an agent with its knowledge level description, choosing to equate the two to simplify modeling (Chandrasekaran, Goel and Allemang 1989). Situated activists and connectionists have noticed a similar confusion in the terms like *plan*, *symbol* and *representation* as used in knowledge-based AI systems (Winograd and Flores 1986; Hinton, McClelland and Rumelhart 1986; Smolensky 1988; Greeno and Moore 1993; Clancey 1993). Brooks

(1986, 1991) makes similar points in his approach to modeling intelligent behavior by constructing robots devoid of explicit representations. All of these confusions and criticisms, in my opinion, can be traced to knowledge-based AI's dependence on literal knowledge level models of intelligence.

1.2 The Problems of Explicit Knowledge

In a literal knowledge level model, the knowledge to solve the problem is *explicitly* represented within the problem solving agent in a knowledge-base using some representation. This dissertation defines *explicit knowledge* to be any manner of domain theory, domain knowledge, world model, heuristics, knowledge-base, or other form of information, *regardless of representation*, that is specific to the task being solved and included *explicitly within* the problem solver. Explicit knowledge is the basis of strong methods and is usually assumed as a given for AI tasks. It often serves as the representation of the associated "world" knowledge about the task. In other words, explicit knowledge is a representation of the task environment that is internal to the problem solver. The quality of the problem solving and search methods that are standard in AI are chiefly determined by the quality of explicit knowledge they contain. It is no wonder that the phrase "in the knowledge lies the power," coined by Edward Feigenbaum, was a popular catch phrase for knowledge-based AI through the 1980s.

Every action of an agent using a literal knowledge level model is explicitly accountable to some component within the agent's knowledge-base. This reduction from knowledge-level observations to complete accountability within the representations of the agent leads to classic problems in artificial intelligence. In order for problem solving ability to improve on a given task when using knowledge-based AI techniques, the quality of representation for the task environment within the problem solver must improve. For a more accurate problem solver, more accurate knowledge must be provided. If knowledge of the task environment is incomplete then the ability of the problem solver is also incomplete. This direct relationship between quantity of knowledge and quality of ability in strong methods leads to several classic difficulties in AI: scaling of method, credit assignment, the knowledge acquisition bottleneck, the knowledge indexing problem and the selection of appropriate representations. The next few subsections review each of these problems.

1.2.1 Credit Assignment Problem

The *credit assignment problem* (Minsky 1963) highlights the issue of how to convert feedback about problem solving into information about how to manipulate a knowledge structure internal to the problem solver. There are two forms of the credit assignment problem: global and local. The *global credit assignment problem* is the determination that there is in fact an error in the internal structure. Typically, explicit goals or evaluation functions within the problem solver determine the global correctness of its internal structure. In the case of explicit knowledge, global credit assignment is determined by its inability to correctly solve the problem at hand. The *local credit assignment problem* is the identification of individual components of the internal structure that are in fact erroneous. Often, it is the local credit assignment problem that poses the most difficulty in constructing a problem solver.

Knowledge-based AI's answer to the local credit assignment problem is to add explicit knowledge to the problem solver that describes how to identify the faulty structure (e.g. Winston 1975; Minton et al. 1989; Schank and Leake 1989). This knowledge is often task-specific and relates the feedback from the task environment directly to faulty components.

1.2.2 The Knowledge Acquisition Bottleneck

The knowledge acquisition problem is typically cast as the difficulty in determining how a program should interact with an expert so that the expert's knowledge can be incorporated into the problem solver (Hayes-Roth et al. 1983). This creates a bottleneck in the populating the method with the knowledge required to solve the task. The knowledge acquisition bottleneck can be generalized to the problem of extracting knowledge from any task representation external to the problem solver and incorporating it into the problem solver. The root of this problem lies in the transduction from the external task representation into the format of the internal environment. As in the local credit assignment problem above, knowledge-based AI techniques simply supply explicit knowledge internal to the problem solver that describes how to acquire knowledge from the given domain (e.g. Davis 1979).

1.2.3 Memory (Knowledge) Indexing Problem

In methods such as Case-Based Reasoning (CBR) (Schank 1982; Kolodner 1989) and Explanation-Based Learning (EBL) (De Jong and Mooney 1986; Minton et al. 1989; Schank and Leake 1989) knowledge is stored in memory as "experience" and retrieved when the situation suggests its appropriateness. This can be generalized to the problem of indexing any large knowledge-base where only a small percentage of the knowledge is relevant at any one time (Schank 1987). To search a large knowledge-base each time knowledge is required for a problem would be prohibitive. CBR and EBL researchers often assume an indexing scheme for the knowledge that uses features salient to the task. The identification of these features is task dependent to allow all similar knowledge to be retrieved with a specific index. Also, the similarity metric is often task-specific, forcing an experience-base to be reindexed when the task changes.

1.2.4 The Problem of Scaling

A simple relation of AI systems is that as the accuracy of the explicit knowledge improves, the accuracy of the problem solver also improves. However, to become an order of magnitude more accurate, the quantity of explicit knowledge or processing required by the problem solver often increases exponentially or worse. Such a discrepancy of scaling between the accuracy of the problem solver and the algorithmic effort is the chief difficulty faced in AI techniques.

For a complex problem the quantity of internal knowledge required to represent the task environment sufficiently may be prohibitive to extract and/or provide explicitly. The popularity of blocksworld tasks (e.g. Winograd 1972; Waltz 1975) was chiefly due to their concisely represented task environment. The finiteness of the situations that could arise allowed a complete encoding as explicit knowledge. This allows a *closed-world assumption* to restrict the knowledge needed to solve the problems. The extreme of the blocksworld approach requires a complete and hence enormous encyclopedic knowledge-base of "common sense" internal to a general problem solver (Lenat, Prakash and Shepard 1986).

1.2.5 Representation Design

The volume of information required to solve some complex problems often forces researchers to spend a significant amount of time determining an appropriate representation that reduces the explicit knowledge to a manageable level. Often this is done to compensate for inadequacies inherent in the technique itself (e.g. McClelland nd Rumelhart 1986b). Given that a particular task has a specific computational ability, the game becomes can the task be represented in such a way that the limited technique can solve it. This has lead to a number of representations to be proposed for problem solving including production systems (Newel and Simon 1981), predicate calculus (Nilson 1980), fuzzy logic (Zadeh 1965), connectionist networks (Rumelhart and McClelland 1986), semantic networks (Brachman 1979), frames (Minsky 1975), conceptual structures (Sowa 1984), scripts (Schank and Ableson 1977), functional representations (Sembugamoorthy and Chandrasekaran 1986), the multiple representations of generic tasks (Chandrasekaran 1986) and others. This representationalist perspective to AI problem solving adheres to the philosophy "Given the right task representation, magic happens."

There is a significant problem with manipulating the task representation for maximal computational benefit. The directed design of the problem solver's representation assumes knowledge of the task, the algorithm to solve it and how the task is best represented given the algorithm to solve it. This forces the human programmer to remain in the problem solving loop in order to determine an adequate representation before problem solving. However, it is often said that the true intelligence for solving problem comes in the determination of an appropriate representation for problem solving (Newell 1981, 1982; Reeke and Edelman 1988).

1.3 The Root of the Problems

All of the limitations for AI methods that use explicit knowledge stem from the fact that the task environment is represented internally in the problem solver. The explicit knowledge, by virtue of its task-specific nature, is a representation or model of the external task environment that is *directly* available to the problem solver. Because this model is internal, it is manipulatable, can be rearranged and augmented, and can be employed using only weak methods if necessary. But once the commitment to include explicit knowledge is made, then the classic problems of AI arise. Because the task environment is external

and thus is not directly accessible to the problem solver, all knowledge required for a task must be provided a priori. If the explicit knowledge is incomplete the system is brittle. When the explicit knowledge grows too large, it must be indexed by task-specific features for quick retrieval. If the problem solver is to learn, a method of credit assignment for the chosen representation of the task must be provided.

In order to avoid the above problems of AI techniques, the reliance on explicit knowledge must be reduced. This involves the identification of new techniques that can be fully defined *independently* of the task. In other words, in order to remove the task-specific knowledge, a problem solver must use *task-independent* representations and operations for problem solving and must acquire whatever task-specific knowledge is necessary to solve the problem *while* problem solving.

Clearly, traditional weak methods are inapplicable for the definition of a task-independent problem solver since they invariably assume task-specific knowledge and operators to complete their definition. Further they provide no mechanism to gather knowledge during the problem solving process. Explanation-based learning methods (EBL) (De Jong and Mooney 1986; Minton et al. 1989; Schank and Leake 1989) are a contemporary learning paradigm for the identification of task-specific knowledge during problem solving. These systems require a task-specific domain theory from which to construct explanations about single examples of problem solving and deduce appropriate rules. Like most other learning methods, the objective of EBL is to merely automate the acquisition of explicit task-specific knowledge. In other words, the broad goals of learning methods is to merely automate the transduction of knowledge from the task environment into the problem solver. While this reduces the brittleness of the knowledge-base, it does not address the other problems associated with explicit knowledge described above. In fact, simply appealing to learning as a solution to the problems of explicit knowledge, as is generally done, is inconsequential if the methods of learning rely on task-specific knowledge to perform.

The hypothesis of this dissertation is that the problems that arise from the usage of explicit knowledge indicate that literal knowledge level models may be an inappropriate level to describe a general computational intelligence and as a result AI must investigate techniques that do not rely on internal task models to perform general problem solving. Such methods must be generally applicable, avoid relying on explicit knowledge, and

allow the same sorts of processing as in strong methods, but without an internal task model. Evolutionary algorithms provide such features.

1.4 Evolutionary Algorithms

Evolutionary algorithms are techniques for search and optimization inspired by natural evolution. These techniques include genetic algorithms (Holland 1975; Goldberg 1989a), evolutionary programming (Fogel, Owens and Walsh 1966; Fogel 1992a) and evolution strategies (Rechenberg 1973; Schwefel 1981; Bäck, Hoffmeister and Schwefel 1991) and are distinguished by their parallel investigation of several positions of a search space by manipulating a *population* of problem solutions simultaneously. In this dissertation, these techniques are generalized into a novel weak method, termed the *evolutionary weak method*, that bears comparison to standard weak methods but encodes some interesting differences.

Evolutionary weak methods provide several beneficial attributes for exploring task-independent problem solving. First, they model the task environment as a separate evaluation function, called a *fitness function*, that maps an individual of the population into a real number. The problem solver only sees the result of this function as feedback for the members of the population. Such minimal feedback provides a strong separation between the task environment and the problem solver. This problem set-up is akin to system identification or reinforcement learning. Second, the operators in evolutionary algorithms are representation specific rather than task-specific. In other words, the operators are defined for a specific representation, such as bit strings or LISP expressions, independently of how the task environment interprets that representation. The simplicity and task-independent nature of these algorithms ensures their general applicability across many problem types.

The most important distinction between standard weak methods and evolutionary algorithms is a task independent empirical approach to local credit assignment, which is termed *empirical credit assignment*. Empirical credit assignment works by creating novel structural variations in the population through probabilistic application of the representation-specific operators. Subsequent interaction between the population and the task environment, represented as the fitness function, determines the viability of the modifications and the abstract features they encode. While the search progresses, the population preserves the best solutions found and attempts additional manipulations. As the population

becomes filled with progressively better members, the search is constrained into areas of the search space that are dense with features previously found to be applicable to the task. Thus, empirical credit assignment allows evolutionary algorithms to *adapt* its search in the problem space dynamically. As a result, an evolutionary algorithm that solves the task appears to an observer to contain explicit task knowledge. Because no explicit knowledge exists in the evolutionary algorithm, this knowledge must emerge from the interaction of the simple problem solver and the task environment.

1.5 Emergent Intelligence

Based on the abilities of empirical credit assignment, this dissertation offers an alternative to the explicit representation of task knowledge in a problem solver that has potential to impact the central problems of knowledge-based AI. *Emergent intelligence* (EI) relies on the following two assumptions about computational problem solving:

1. The task environment itself is often a more concise representation for knowledge specific to the task than any internal representation of 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 in the problem solver during the problem solving process. As a result, pertinent task-specific knowledge *emerges* from the interaction of the problem solver with the innate constraints of the task environment.

Emergent Intelligence avoids each of the problems associated with explicit knowledge described above by removing the source of the problems, the explicit knowledge. Since knowledge pertinent to the specific problem solving situation emerges from interaction with the task environment, there is no need for formulating any task-specific indexing scheme, credit assignment or acquisition method. The knowledge that emerges is dependent on the particular task being solved, since it emerges from direct interaction with the task environment, so small changes in the task results in small changes in the knowledge that emerges rather than brittleness. To paraphrase an old saying, one task environment is worth a thousand heuristics (or more).

1.6 Demonstrations of Emergent Intelligence

The experiments within this dissertation illustrate four distinct examples of emergent intelligence illustrated by evolutionary algorithms. The examples cover the acquisition of task-specific connectionist network architectures, task-directed component manipulation while inducing finite-state machines, the acquisition of modular computer programs that create appropriate high-level abstractions and implicit goal directed progression. Each of these effects are shown to emerge from the interaction of a simple evolutionary algorithm and the task environment without the assistance of explicit knowledge. This section describes how each of these problems are addressed as instances of emergent intelligence.

Currently, there is no known general solution for inducing both an appropriate architecture and parametric values for a neural network. Current methods either assume an architecture as given and learn only the weights or use approximate heuristics that force a task into a restricted architecture rather than fit an architecture to the task. The difficulty with inducing neural network architectures is that the they are highly specific to the task being solved. This is especially true in the case of recurrent networks. Chapter 4 of this dissertation describes a program, GNARL, originally described in Angeline, Saunders and Pollack (1994), that induces both the topology and parametric values for recurrent neural networks. This is done without explicit knowledge that describes what constitutes an appropriate architecture for the task. GNARL begins with only a population of randomly generated networks and an evaluation function that rates a network's performance on a pre-specified task. The results show GNARL is able to induce networks in comparable time and with better generalization than a method that assumes a fixed topology.

Chapter 5 describes an instance of emergent intelligence where the interaction of the evolutionary algorithm with the task environment determines which components of finite state machines (FSMs) should be manipulated while solving a simple control task. The program is similar to GNARL except it also employs two additional operators that preserve randomly selected components from further manipulation. The appropriateness of the protected component is determined through future interactions of the population with the task environment. In essence, the knowledge of which components to protect and which to manipulate emerges naturally as a result of the interaction with the simple control task. An interesting side-effect of the program is that it induces the FSMs more quickly

with the added random operators than without. The experiments of this chapter also appear in Angeline and Pollack (1993c).

Another program, GLiB, described in Chapter 6, induces modular computer programs without explicit knowledge of what makes a beneficial module in general or for a specific task. Angeline and Pollack (1993a; 1992) describe initial explorations with this program. GLiB uses operators that randomly select portions of a program to be defined as a new module. As in GNARL's determination of appropriate architectures, the validity of a module is induced naturally through future interactions with the task environment. The resulting modules reflect the presence of knowledge that is a natural constraint of the task environment. GLiB's modules are thus highly dependent on the task. An interesting side effect of this program is that the modules represent task-specific abstractions from the initial general programming language. This automatic process aids in the discovery of larger structures by abstracting not only the representation but the search operators as well.

Given the task environment is a crucial component of a problem solver using emergent intelligence, Chapter 7 explores the implications of differing task environments on problem solving. Here, it is shown, again using GLiB, that introducing direct competition in the evaluation of population members leads to a natural progression towards complex solutions. This is interpreted as emergent goal-directed behavior since explicit knowledge of the goal and associated subgoals are absent from GLiB. This chapter is an expansion of the work originally published in Angeline and Pollack (1993b).

1.7 Scope of this Dissertation

This work is concerned with the latter of the two goals attributed to AI above, namely creating programs that perform as well or better than humans on tasks but not necessarily using humans as a model. I will be content to demonstrate that explicit knowledge, while sufficient, is not necessary for a problem solver to perform intelligently on a task. The determination that the reduced reliance on explicit knowledge actually alleviates the various problems described above is left for future work.

Because of the insistence of no explicit task-specific knowledge, the techniques presented in this dissertation do not perform as well as other techniques that use task knowledge when available. This is not a limitation of the technique but a direct result of the assumed knowledge-poor experimental paradigm. Further, there is no a priori reason why

the techniques investigated here can not be integrated with explicit task knowledge. This has not been done in this work since such knowledge would only obfuscate the effects of the techniques under investigation and their ability to address problems where no explicit knowledge is available.

Other limitations of the following work stem from its opportunistic nature. The basic evolutionary mechanism is extremely opportunistic, and typically exploits organizations that are challenging to analyze. This is similar to the difficulty of analyzing the distributed representations (Hinton, McClelland and Rumelhart 1986; Sejnowski and Rosenberg 1987; Smolensky 1988) of some connectionist networks trained by simple gradient decent methods such as backprop (Rumelhart, Hinton and Williams 1986). As a result, the decompositions or organizations acquired by these techniques do not relate well to the normative solutions preferred by humans. In addition, indirect routes of verification are often needed to demonstrate an effect.

While the opportunistic nature of these methods presents difficulties in fully interpreting the results of an experiment, it may ultimately pose an advantage for automatic problem solving. By shedding the preconceptions of humans and becoming more opportunistic the resulting structures may provide computational advantages that our human-centric preconceptions otherwise obfuscate.

1.8 Contributions of this Work

Instead of relying on the analysis of a task to supply explicit knowledge for its solution with an AI technique, this work demonstrates that some programs can work "as if" they have appropriate explicit knowledge. Such phenomena have the potential to greatly extend the domain of tasks solvable by AI mechanisms to those problems that are to difficult to analyze by humans. This is the first step to a dynamic, self-sufficient intelligent system.

An additional contribution of this work is the identification of the evolutionary weak method and its relation to standard weak and strong methods. By clarifying this association, techniques developed for evolutionary weak methods can suggest analogous mechanisms for standard weak methods. Also, once the commonalities between these techniques are seen, recent research on the "strengthening" of evolutionary weak methods can be put into perspective as a consequence of their status as weak methods. This serves to illuminate how these methods fit into the toolbox of artificial intelligence techniques.

1.9 A Guide to Reading this Dissertation

The next chapter reviews the standard weak methods and their traditional usage in artificial intelligence. It then describes the basics of evolutionary algorithms, what makes them distinct and how they fit into the traditional notions of weak methods. Chapter 3 defines emergent intelligence and discusses it as a general computational theory as well as the implications of the concept of emergence on computation and cognitive science. Chapters 4 through 7 demonstrates specific examples of emergent intelligence as reviewed above. Finally, chapter 8 concludes this dissertation with a summary and review of the main points of this work.