International Journal on Artificial Intelligence Tools, Vol. 7, No. 3 (1998) 239–291

CAEP: AN EVOLUTION-BASED TOOL FOR REAL-VALUED FUNCTION OPTIMIZATION USING CULTURAL ALGORITHMS

Chan-Jin Chung
Department of Mathematics and Computer Science
Lawrence Technological University, Southfield, MI 48075-1058
CHUNG@ltu.edu

Robert G. Reynolds
Department of Computer Science
Wayne State University, Detroit, MI 48202
reynolds@cs.wayne.edu

Received 10 June 1998
Revised 4 September 1998

ABSTRACT

Cultural Algorithms are computational self-adaptive models which consist of a population and a belief space. The problem-solving experience of individuals selected from the population space by the acceptance function is generalized and stored in the belief space. This knowledge can then control the evolution of the population component by means of the influence function. Here, we examine the role that different forms of knowledge can play in the self-adaptation process within cultural systems. In particular, we compare various approaches that use normative and situational knowledge in different ways to guide the function optimization process.

The results in this study demonstrate that Cultural Algorithms are a naturally useful framework for self-adaptation and that the use of a cultural framework to support self-adaptation in Evolutionary Programming can produce substantial performance improvements over population-only systems as expressed in terms of (1) system success ratio, (2) execution CPU time, and (3) convergence (mean best solution) for a given set of 34 function minimization problems. The nature of these improvements and the type of knowledge that is most effective in producing them depend on the problem's functional landscape. In addition, it was found that the same held true for the population-only self-adaptive EP systems. Each level of self-adaptation (component, individual, and population) outperformed the others for problems with particular landscape features.

Keywords: Evolutionary Computation, Evolutionary Algorithms, Cultural Algorithms, Evolutionary Programming, Function Optimization, and Self-adaptation

1. Introduction

1.1. *Culture as a paradigm for problem solving*

Evolutionary Computation (EC) methods have been successful in solving many diverse problems in search and optimization due to the unbiased nature of their operations which can still perform well in situations with little or no domain knowledge [Fogel, 1995]. However, there can be considerable improvement in EC's performance when problem solving knowledge acquired during evolution is used to bias the problem solving process in order to identify patterns in the performance environment [Reynolds, 1993, 1996; Chung 1996]. These patterns are used to influence the generation of candidate solutions, to promote more instances of desirable candidates, and to reduce the number of less desirable candidates in the population.

Culture can be viewed as a vehicle for the storage of information that can be globally accessible to all members of the society and that can be useful in guiding their problem solving activities. As such, groups that are able to support a cultural

tradition can use their cultural heritage as a mechanism by which to bias the generation of individuals on at least a phenotypic level by facilitating the production of phenotypes that are promising in a given environment on the one hand, and deterring the production of phenotypes that are less likely to be productive on the other. Cultural Algorithms have been developed [Reynolds 1978, 1979, 1986, 1992, 1993, 1994] in order to model the evolution of the cultural component of an evolutionary computational system over time as it accumulates experience. As a result, Cultural Algorithms can provide an explicit mechanism for global knowledge and a useful framework within which to model self-adaptation in an EC system [Angeline, 1995].

Cultural Algorithms consist of a social population and a belief space [Reynolds, 1978, 1979, 1986, 1992, 1994]. The experience of individuals selected from the population space by the *acceptance function* is used to generate problem solving knowledge that resides in the *belief space*. The belief space stores and manipulates the knowledge acquired from the experience of individuals in the population space. This knowledge can control the evolution of the population component by means of the *influence* function. As a result, Cultural Algorithms provide an explicit mechanism for the acquisition of global knowledge.

The two basic categories of knowledge that we will consider in the cultural evolution model here are: *situational* and *normative*. Situational knowledge is a "snapshot" of the state of the world. The world can be viewed as a sequence of situations linked by behaviors [Russell, 1995]. Situational knowledge provides a set of exemplar cases that are useful for the interpretation of specific individual experience. Normative knowledge describes how a rational agent should act [Russell, 1995]. In other words, it provides standards for individual behavior as well as guidelines within which individual adjustments can be made. Also, normative knowledge defines a standard (an ideal) that can be used to judge which behavior is desirable or undesirable [Valente, 1994]. While there are other types of knowledge that can be contained within a cultural system, these two types were selected because they are viewed to be fundamental to the operation of cultural systems.

The knowledge acquired in the belief space can be used to guide the adaptation process within a population. Holland [1975] developed a formal framework for any generic adaptive system. His framework for adaptation concerns a system that is able to alter its structure and/or behavior based on experience in some set of performance environments based upon a set of adaptive strategies and associated parameters [Reynolds, 1979]. More recently, Hinterding [1997] distinguishes between adaptation and self-adaptation in the following way.

> *Adaptation takes place if there is some form of feedback from the Evolutionary Algorithm that is used to determine the strategy parameter. In self-adaptive systems, the parameters to be adapted are encoded onto the chromosome(s) of the individual and undergo mutation and recombination.*

The concept of self-adaptation was developed first in Evolution Strategies by Schwefel in order to adapt mutation parameters during the problem solving process [Schewefel, 1981]. Because self-adaptation is particularly important in complex performance environments, many forms of self-adaptations have been suggested since then. For example, Angeline [1995] suggested that self-adaptation can take place at three different levels: the population level, the individual level, and the component level. At the population level, aspects of a system's strategies that control a subset of individuals in the

population can be modified. Individual-level adaptive methods associate a strategy with each member of the population that determines how to manipulate the individual. Component-level adaptations associate adaptive parameters with each distinct component of an individual in an evolving system in order to determine how that component will be modified.

In Cultural Algorithms, self-adaptation can take place in all of the forms described above. Thus, knowledge contained in the belief space can be used to affect self-adaptation at any of the lower levels. In addition, self-adaptation can take place at the level of the belief space. In this paper, knowledge in the belief space is used to control self-adaptation within the population space by controlling both the step size and direction of mutation.

The objective of this study will be to investigate how the nature of the belief space self-adaptation process supported in a Cultural Algorithm impacts the system's performance on a given class of function optimization problems. That is, how does the cultural system's belief structure reflect the structure of the problem that it is solving? The basic aspects of a Cultural Algorithm's structure are: (1) the level of self-adaptation, (2) type of knowledge in the belief space used for self-adaptation, (3) the methods used to acquire the knowledge (acceptance function), and (4) the methods used to exploit the knowledge to adjust the population (influence function). The performance measures we are interested in are: (a) the probability of finding the global optimum solutions (success ratio), (b) the accuracy of the solution, (c) the rate of convergence, and (d) the execution speed.

Specifically we investigate how different strategies for self-adaptation using knowledge in the belief space impact the Cultural Algorithms performance for a given population model. Here we use Evolutionary Programming (EP) for the population model [Fogel, 1962; Fogel 1995a].

1.2. *The optimization problems used*

In this study, the class of problems to be solved involves real-valued function optimization, which has long been regarded as a benchmark problem in Evolutionary Computation [De Jong, 1975; Hoffmeister, 1992; Kim, 1995; Fogel, 1995; Koon, 1995; Salomon, 1996; Michalewicz, 1994; Schwefel, 1995; Yao, 1996; Bäck, 1996].

Since $\max\{f(x)\} = \min\{-f(x)\}$, without loss of generality, we consider only minimization problems here. Unconstrained real-valued global minimization problems can be formalized as a pair $<S,f>$, where $S \subseteq \Re^n$ is a bounded set on $\Re^n$ and $f: S \mapsto \Re$ is an n-dimensional real-valued function. The problem is to find a vector $\mathbf{x}^* \in S$ such that

$$\forall \mathbf{x} \in S: f(\mathbf{x}^*) \leq f(\mathbf{x}).$$

Note that the objective function f can be continuous or discontinuous, convex or nonconvex, unimodal or multimodal, quadratic or nonquadratic, low-dimensionality or high-dimensionality, deterministic or stochastic, decomposable or nondecomposable. We took care to include functions with various structural characteristics in order to span a wide range of problem structures [Whitley, 1996].

1.3. *Organization of the paper*

The organization of this paper is as follows. Section 2 introduces Cultural Algorithms and their basic framework. Section 3 describes how Cultural Algorithms can be used as a

framework within which to support the self-adaptation process within an evolving Evolutionary Programming (EP) population component. The resultant shell is called "Cultural Algorithms with Evolutionary Programming", CAEP. The basic design choices for configuring a CAEP system are then described. Section 4 describes the domain of problems to be solved in this study by introducing a set of 34 function minimization problems. Section 5 compares the performance of the various CAEP configurations with standalone models of EP for each of 34 problems. In section 6, a search heuristic for the design of an efficient CAEP configuration for a given problem is presented. In section 7, the impact of a problem's functional landscape on the system performance is discussed. This is addressed from the perspective of how the functional landscape of a problem impacts design of a cultural configuration to solve it. In section 8, the conclusions are given and further research directions are outlined.

2. Cultural Algorithms

In human societies, culture can be viewed as a vehicle for the storage of information that is potentially globally accessible to all members of the society and that can be useful in guiding their problem solving activities. Note that this information is stored independently of the individual descriptions. As such, groups that are able to support a cultural tradition can use their cultural heritage as a mechanism by which to bias the generation of new individuals on at least a phenotypic level. This is done by facilitating the production of phenotypes that are promising in a given environment on the one hand, and deterring the production of phenotypes that are less likely to be productive on the other. Cultural Algorithms were initially developed by Reynolds [1978, 1979] in order to model the evolution of the cultural component of an evolutionary computational system over time as it accumulates experience.

From an anthropological point of view, Durham [1994] defines culture as: *a system of symbolically encoded conceptual phenomena that are socially and historically transmitted within and between populations.* This definition assumes the existence of intellectual, generalized knowledge acquired within a cultural context. Thus, culture can be regarded as the highest, as well as the most general, infrastructure in complex knowledge-based adaptive systems. As a result, Cultural Algorithms can provide an explicit mechanism for global knowledge and a useful framework within which to model any level of self-adaptation in an EC system.

Cultural Algorithms (CAs) are a class of computational models derived from models of the cultural evolution process. These algorithms support the basic mechanisms for cultural change described in the anthropological and archaeological literature [Durham, 1994; Boyd, 1985; Cavalli-Sforza, 1981]. The Cultural Algorithm framework offers a mechanism to support a dual inheritance system consisting of a cultural component at the macro-evolutionary level, and a population component at the micro-evolutionary level. It also provides a framework in which knowledge acquired at the cultural level is isolated and exploited in order to accelerate the problem solving process. It has been suggested that cultural evolution can proceed at a faster rate than biological evolution [Reynolds, 1992] due to the guiding influence of the cultural knowledge.

At the micro-evolutionary level, there is a society or population of individuals. Each individual is described in terms of a set of behavioral traits or phenotype. Each individual also has an internal model of the world as they see it. This model is derived as a generalization of the individual's experiences. Note that Schwefel also used the term "internal model" for strategy parameters that undergo self-adaptation [Bäck, 1997]. These

traits, similar to genetic material, can be transmitted among individuals and modified, through the use of socially motivated operators via the belief space.

At the macro evolutionary level, individual experiences are collected, merged, generalized, and specialized in the belief space. New traits can also be introduced in the population and others may be eliminated over time, as new knowledge accumulates. This knowledge in the belief space can serve to direct the future actions of the population and its individuals. Therefore, Cultural Algorithms may be useful in exploring large search spaces by accumulating global knowledge about the problem space.

We can also view the cultural evolution process as a vehicle for amplifying individual or group behavior and building consensus. In other words, during cultural evolution, "conceptual beacons" that symbolize acceptable and unacceptable behavior for individuals in a population are accumulated. Figure 2.1 gives a pseudo code description of the Cultural Algorithm.

Individuals are first *evaluated* using a performance function. The performance information represents the problem solving experience of an individual. An *acceptance function* determines which individuals in the current population are able to impact, or to be *voted* to contribute, to the current beliefs. The experience of these selected individuals is used to *adjust* the current group beliefs. These group beliefs are then used to guide and *influence* the evolution of the population at the next step, where parameters for self-adaptation can be determined from the belief space. Information that is stored in the belief space can pertain to any of the lower levels, e.g. population, individual, or component. As a result, the belief space can be used to control self-adaptation at any or all of these levels.

```
Begin
  t = 0;
  Initialize P^t
  Initialize B^t
  repeat
    Evaluate P^t
    Adjust(B^t, Accept (P^t))
    Variation(P^t, Influence(B^t))
    t = t + 1;
    Select P^t from P^(t-1)
  until (termination condition achieved)
End
```

Figure 2.1 Cultural Algorithms Pseudo Code

The Cultural Algorithm is a dual inheritance system with evolution taking place at the population level and at the belief level. The two components interact through a communications protocol. The protocol determines the set of "acceptable" individuals that are able to update the belief space. Likewise the protocol determines how the updated beliefs are able to impact and "influence" the adaptation of the population component.

The belief space in Cultural Algorithms is then a natural repository for information concerning the self-adaptation process. Information stored in the belief space can relate to various global parameters that can be used to direct the population through the specified communications protocol. A Cultural Algorithm framework is denoted as an 8-tuple:

$$CA = \langle P, S, V_c, f, B, Accept, Adjust, Influence \rangle,$$

where: P is a population; S is a selection operator; V_c is a variation operator; and f is the performance function; B is the belief space; *Accept* is the acceptance function; *Adjust* is a belief space operator that adjusts or updates the belief space knowledge, B; and *Influence* is a set of influence functions to be used to influence the variation operator V_c. *Accept* and *Influence* together represent the communication protocol for a Cultural Algorithm. Each basic CA component and its relationships are depicted in figure 2.2.

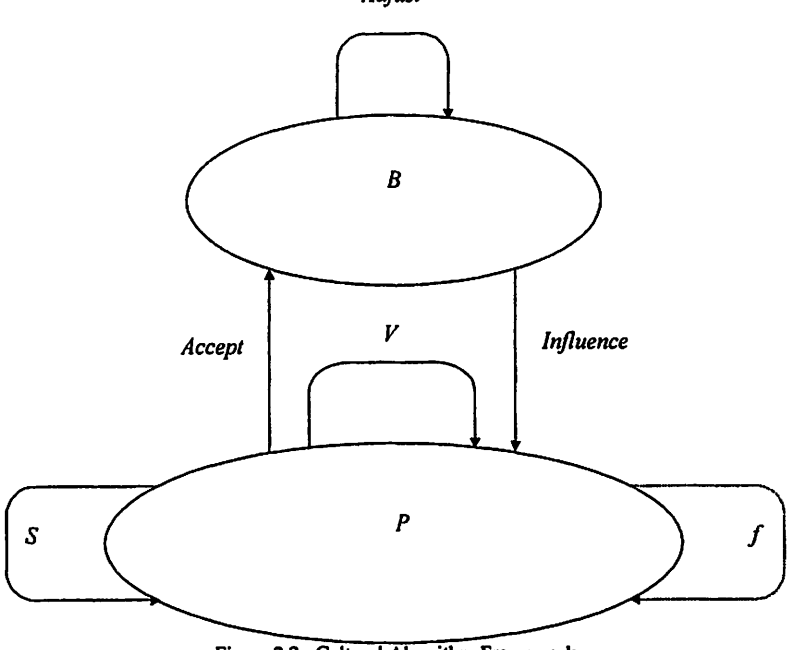

Figure 2.2 Cultural Algorithm Framework

3. Cultural Algorithms with Evolutionary Programming

In this section we discuss the set of possible configurations used here to embed the Evolutionary Programming population model into a Cultural Algorithm framework. We begin by introducing a general evolutionary computation framework in 3.1. The different forms of self-adaptation that will be used with the Evolutionary Programming population model are then described in 3.2. They include component, individual, and population levels of self-adaptation. In section 3.3 we give the description of the CAEP (Cultural Algorithms with EP) shell which will support self-adaptation at the level of the belief space. The performance of these self-adaptive systems will be compared relative to a set of 34 function minimization problems.

3.1. *Evolutionary computation framework*

An Evolutionary Algorithm (EA) can be denoted as a 4-tuple:

$$EA=\langle P,S,V,f\rangle$$

where P is a model for a population, S is a selection operator, V is a variation operator, and f is performance function. The evolution of population P can be represented as [Fogel, 1996]:

$$P^{t+1}=S(V(P^t))$$

where t is the generation number, and P^t denotes a population of m individuals under a specified representation at generation t. $V(\ )$ is the variation operator used to generate new solutions, and $S(\ )$ is the selection operator that determines which candidate solutions will survive in the next population P^{t+1}. The population, P, consists of a vector of individual structures and a vector to represent the corresponding individual's performance score. This can be defined as:

$$P=\left\langle x_{[1..m,1..n]}, score_{[1..m]}\right\rangle$$

where m equals $2p$, the p individuals and p offspring. p will be used to represent the number of surviving individuals in a population. n is the number of parameters that describe each individual. $score_i$ is a performance score for each individual i, that is, $score_i = f(\mathbf{x}_i) \in \Re$. Throughout this paper, the index i is used to denote an individual in a population, and j is used as an index for a specific parameter.

Cultural Algorithms can be represented in terms of this framework as follows:

$$P^{t+1} = S(V_c(P^t, Influence(Adjust(B^t, Accept(P^t))))),$$

where V_c represent all of the cultural variation operators. Note that the influence function can influence any or all of the self-adaptive strategies, depending on the type of self-adaptation in the population, P.

3.2. *Evolutionary programming*

In Evolutionary Programming (EP), attention is given to the phenotypes of an entire reproductive population (species). The components in an evolving population are abstracted as vectors of behavioral traits [Fogel, 1995a]. Each vector of behavioral traits is evaluated in the light of a performance function. A selection operator, S, acts to eliminate those individuals having relatively poor performance. The tournament selection method is often used for S. New vectors are created by applying operators, V, which mimic the functional variance in phenotypic traits in individuals. However, no recombination operators are used since recombination can be viewed from the perspective of an entire population as a specialized form of the functional variance or mutation operator [Fogel 1995a, Angeline, 1995]. Thus, only mutation is used since it is a more generalized mechanism for the variation operator, V. The basic EP algorithm skeleton, which will be used for all other EP versions developed here, is shown below.

(1) Generate at random an initial population of p candidate solutions for n parameters into $x_{[1..p,1..n]}$.

(2) Assess the performance for each parent solution using the given objective (evaluation, or performance) function f.

(3) Generate p new offspring solutions from $x_{[1..p,1..n]}$ by applying the variation operator, V. Now there are $m=2p$ solutions in the population, $x_{[1..m,1..n]}$.

(4) Assess the performance score of each offspring using the given objective function f.

(5) For each individual, select c competitors at random from the population of $2p$ individuals. Next, conduct pairwise competitions between the individual and the competitors.

(6) Select the p solutions that have the greatest number of wins (w_i) to be the

parents for the next generation.

(7) Return to step 3 unless the available execution time is exhausted or an acceptable solution has been discovered.

3.2.1. *EP using Schwefel's self-adaptation mechanism*

Schwefel [1981] devised a method for Evolution Strategies for self-adaptation of the component level parameter σ which is used to calculate changes in mutation step sizes for each of the n parameters. Thus, the population structure for this mechanism can be represented as:

$$P = \left\langle x_{[1..m,1..n]}, \sigma_{[1..m,1..n]}, score_{[1..m]} \right\rangle.$$

According to Angeline's [1995] classification, this mechanism can be considered as a component level self-adaptation, since it has adaptive strategies for each distinct component parameter of an evolving individual. The strategies determine how parameters will be modified. The variation operator in this method was defined as:

$$\forall i \in \{1,2,\cdots,p\} \text{ and } \forall j \in \{1,2,\cdots,n\}:$$
$$x_{p+i,j} = x_{i,j} + \sigma_{i,j} \cdot N_{i,j}(0,1)$$
$$\sigma_{p+i,j} = \sigma_{i,j} \cdot \exp(\tau' \cdot N_{i,j}(0,1) + \tau \cdot N_i(0,1)),$$

where $x_{i,j}$ is the jth parameter of an ith individual. $N_{i,j}(0,1)$ is a realization of a zero mean Gaussian normal deviate with a standard deviation of 1. $x_{p+i,j}$ is the new offspring produced by this mutation. Note that updating the self-adaptive parameter, σ, is done after the mutation of x. This method is called the σ-last method. The offspring's step size is a lognormal perturbation of the parent's step size. Bäck and Schwefel [1993] recommended setting the exogenous parameters τ and τ' in the following way:

$$\tau = \left(\sqrt{2n}\right)^{-1} \qquad \tau' = \left(\sqrt{2\sqrt{n}}\right)^{-1}$$

Saravanan and Fogel [1994] applied this σ-last method to EP and recently Gehlhaar and Fogel [Gehlhaar, 1996] reported that σ-first method demonstrated better convergence than the σ-last method for a given set of function optimization problems. In the σ-first method, the self-adaptive parameter, σ, is updated before the mutation of x. In this study, we use the σ-first method.

3.2.2. *EP using Fogel's self-adaptation mechanism*

Fogel at al. [1991] offered another method for component level self-adaptation. This mechanism uses the same population structure, P, as defined in 3.2.1. The variation operator is shown below in σ-first form [Gehlhaar, 1996].

$$\forall i \in \{1,2,\cdots,p\} \text{ and } \forall j \in \{1,2,\cdots,n\}:$$
$$\sigma_{p+i,j} = \sigma_{i,j} + \beta \cdot \sigma_{i,j} \cdot N_{i,j}(0,1)$$
$$x_{p+i,j} = x_{i,j} + \sigma_{p+i,j} \cdot N_{i,j}(0,1)$$

The step size is a Gaussian perturbation of σ with a scaling factor β. If any $\sigma_{i,j}$ becomes negative, it is reset to a small value ε.

3.2.3. *EP using individual level self-adaptation based upon age information*

An individual level self-adaptive EP is developed here using age information. The intuition for this method derives from the work of [Kim, 1995]. The population structure for this EP version can be represented as:

$$P = \left\langle x_{[1..m,1..n]}, \sigma_{[1..m,1..n]}, \alpha_{[1..m]}, score_{[1..m]} \right\rangle,$$

where α_i means the *age* since birth for the individual *i*. This represents the number of generations that the individual has been a parent. When an individual is generated by a perturbation, the age is reset to zero. After the tournament selection process, if an individual survives to be a parent for the next generation, its age is increased by one. If the age reaches the size of the population, *p*, the counter is re-initialized to one. In order to prevent older individuals from getting stuck on false peaks, age is used to increase the step size over time. If we incorporate this age information into Fogel's self-adaptive method, the variation operator will become:

$$\forall i \in \{1,2,\cdots,p\} \text{ and } \forall j \in \{1,2,\cdots,n\}:$$
$$\sigma_{p+i,j} = \sigma_{i,j} + \beta \cdot \alpha_i \cdot \sigma_{i,j} \cdot N_{i,j}(0,1)$$
$$x_{p+i,j} = x_{i,j} + \sigma_{p+i,j} \cdot N_{i,j}(0,1)$$

3.2.4. *EP using population level self-adaptation mechanism*

Rechenberg postulated his 1/5 success rule for Evolution Strategies [Hoffmeister, 1990; Schwefel,1995] as follows:

> *From time to time during the evolution process check the ratio of the number of successes to the total number of trials (mutations). If the ratio is greater than 1/5, increase the variance; if it is less than 1/5, decrease the variance.*

This can be regarded as a population level self-adaptive mechanism [Angeline, 1995], since it modifies the global mutation variance based on the population level ratio, $\varphi(k)$. $\varphi(k)$ is the ratio of successful mutations to unsuccessful mutations over the past *k* generations. The population structure of an EP version that uses this method can be represented as:

$$P = \left\langle x_{[1..m,1..n]}, \omega^t, score_{[1..m]} \right\rangle$$

where ω^t is the global mutation variance at time *t*, updated by the following rule:

$$\omega^{t+1} = \begin{cases} c_d \cdot \omega^t & \text{if } \varphi(k) < 1/5, \\ c_i \cdot \omega^t & \text{if } \varphi(k) > 1/5, \\ \omega^t & \text{if } \varphi(k) = 1/5. \end{cases}$$

Schwefel [1981] recommended $c_d = 0.82$, $c_i = 1.0/0.82$. The variation operator for EP systems employing the 1/5 success rule will be:

$$\forall i \in \{1,2,\cdots,p\} \text{ and } \forall j \in \{1,2,\cdots,n\},$$
$$x_{p+i,j} = x_{i,j} + \omega^t \cdot N_{i,j}(0,1)$$

3.3. *An approach to embedding EP into a cultural algorithm framework*

"Growth and development of civilization is usually achieved by a *creative minority*" [Toynbee, 1934]. Here, we select a "creative minority" from the population using the acceptance function and then generalize their problem solving experience in the form of

intervals [Reynolds, 1995; Chung, 1996] for each variable in the belief space. Other representations that have been used for beliefs include version spaces, algebraic expressions, and semantic networks [Reynolds, 1993, 1994]. The size of the interval for a given parameter is regarded as normative knowledge and will be used to decide the mutation step size. Also, since culture is often learned through imitation [Minkoff, 1984], an set of exemplar individuals are kept. These can be both positive and negative examples, but only positive individuals are used here. This exemplar information can be viewed as situational knowledge.

Individuals in the population have a tendency to follow global indexes in a society. For example, if a global economic index is interpreted as favorable, then they may invest more actively than if it is not. Here the range information for a given parameter can be viewed as analogous to a global economic index. The individuals in the population space can adjust their parameter values relative to the range of acceptable values for each of their parameters. Also, since individuals in the population are apt to imitate the top performers, the direction of the mutation for a given parameter can be decided by the current value of that parameter for the current elite individuals [Chung, 1996, 1996b, 1996c]. These ideas are depicted in Figure 3.1.

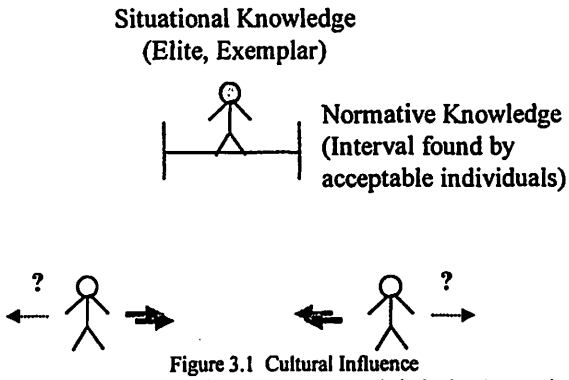

Figure 3.1 Cultural Influence
Belief space (upper part), population space (lower part), imitation (arrows in lower part)

This belief space knowledge can influence the variation operator V, in two ways: (1) by determining the step size of the changes, and (2) by determining the direction of change, e.g. increase or decrease the current value.

3.4. *The basic CAEP shell*

The following is a basic "cultured" EP algorithm skeleton. Steps (3), (4) and (8), shown in bold characters, are the modifications made to the basic EP algorithm presented in 3.2.

(1) Select an initial population of p candidate solutions.
(2) Assess the performance score of each of the parent solutions using the objective function f.
(3) Initialize the belief space. The structure of the belief space is given in 3.5.
(4) Generate new p offspring solutions by applying the variation operator, V as modified by the influence function. Now there are $2p$ solutions in the population.
(5) Assess the performance score of each of the p offspring solutions by the given objective function f.

(6) For each individual, select c competitors at random from the population of parents and offspring. Conduct pair-wise competitions between the individual and the competitors.
(7) Select the p solutions that have the greatest number of wins to be parents for the next generation.
(8) **Update the belief space by accepting individuals using the acceptance function in 3.6. The belief space update rule is described in 3.7.**
(9) The process proceeds to step 4 unless the available execution time is exhausted or an acceptable solution has been discovered.

The following sections will discuss the design options available for each CAEP shell developed in this study. We start by discussing the belief space.

3.5. *Knowledge representations and adjustment functions in the belief space*

There are two basic categories of belief space knowledge that we will consider here: *normative* and *situational*. Normative knowledge provides standards for individual behavior and guidelines within which individual adjustments can be made. Situational knowledge provides a set of exemplar cases that are useful for the interpretation of specific individual experience. Both kinds of knowledge can be used to explicitly represent desirable and undesirable behavior. However in this study, only knowledge about the desirable or optimum behavior is considered. The formal syntax for the belief space, B, used in this study is:

$$B=\langle S, N\rangle,$$

where S denotes structures for situational knowledge and N denotes structures for normative knowledge.

3.5.1. *Situational knowledge for CAEP*

The situational knowledge component, S, in this study consists of a sorted list of exemplar individuals. It is represented as a pair:

$$S=\langle <\mathbf{E}_1,\mathbf{E}_2,\cdots,\mathbf{E}_e>, adjust_E\rangle,$$

where e is a system constant which represents the number of exemplar individuals in the belief space and each $\mathbf{E}_i$ represents the ith best exemplar individual in the evolution history. De Jong [1975] demonstrated that for Genetic Algorithms the current best or *elite* individual, needed to be kept in the population in order to guarantee the production of an optimum solution for some problems. Here we will keep the e best individuals in the belief space by using a belief space operator $adjust_E$. In this study, either $e = 1$ or $e = 2$ is implemented. The exemplar list is initialized with the best individuals from the initial population and contains no duplicates.

3.5.2. *Normative knowledge for CAEP*

The normative knowledge component, N, consists of interval information for each of the n parameters. The interval describes the current range of acceptable values of each parameter based upon a sampling of individuals as performed by the acceptance function. It can be described as:

$$N=\langle \mathbf{n}_{[1..n]}, adjust_N\rangle, \text{ where } \mathbf{n}_j=\langle I_j, L_j, U_j\rangle.$$

I_j denotes the closed *interval* for variable j defined over the continuous set of real

numbers as an ordered pair, l_j and u_j, representing the lower and upper bound respectively:

$$I_j = [l_j, u_j] = \{x | l_j \le x \le u_j, x \in \Re\}.$$

Both the *l*(lower bound) and *u* (upper bound) are initialized by the given boundary values for each parameter. L_j represents the performance score of the individual which contributed to the update of the lower bound l for parameter j. U_j represents the performance score of the individual which contributed to the update of the upper bound u for parameter j. The structures of N are depicted in Figure 3.2.

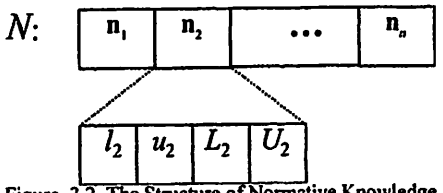

Figure 3.2 The Structure of Normative Knowledge

The normative knowledge component in the belief space, N, is updated as follows. The parameter values for individuals selected by the acceptance function are used to calculate the current acceptable interval for each parameter in the belief space. The idea is to be conservative when narrowing the interval and be progressive when widening the interval. Therefore, we can formulate the interval update rule, $adjust_N$, as the following. All of the individuals selected by the acceptance function are involved in updating both the upper and lower bounds.

For all $j = 1,\ldots,$ n and for all $k = 1, \ldots,$ *Number-of-Accepted-Individuals*:

For the left boundary and its score for parameter j, where l_j^t represents the lower bound for parameter j at generation t and L_j^t denotes the performance score for the lower bound:

$$l_j^{t+1} = \begin{cases} x_{i,j}^t & \text{if } x_{i,j}^t \le l_j^t \text{ or } f(\mathbf{x}_i^t) < L_j^t \\ l_j^t & \text{otherwise} \end{cases}$$

$$L_j^{t+1} = \begin{cases} f(\mathbf{x}_i) & \text{if } x_{i,j} \le l_j^t \text{ or } f(\mathbf{x}_i) < L_j^t \\ L_j^t & \text{otherwise} \end{cases}$$

For the right boundary and its score for parameter j, where u_j^t represents the upper bound for parameter j at generation t and U_j^t denotes the performance score for the upper bound:

$$u_j^{t+1} = \begin{cases} x_{k,j} & \text{if } x_{k,j} \ge u_j^t \text{ or } f(\mathbf{x}_k) < U_j^t \\ u_j^t & \text{otherwise} \end{cases}$$

$$U_j^{t+1} = \begin{cases} f(\mathbf{x}_k) & \text{if } x_{k,j} \ge u_j^t \text{ or } f(\mathbf{x}_k) < U_j^t \\ U_j^t & \text{otherwise} \end{cases}$$

3.6. *Acceptance function*

The acceptance function selects individuals who will impact the formation of the current belief space. A number of different classes of acceptance functions can be employed. The set of acceptance functions used in this study can be specified as:

$$Accept = top10\% | top20\% | top20\%' | top30\% | top40\% | top50\% | above\_avg$$

Most of theses functions are static in nature and use only information about the absolute ranking (top 10%, top 20%, top 30%, top 40%, and top 50%-median of individuals in the population, respectively) or relative ranking (above average individuals) to select the individuals that are used to update the belief space. For example, *top*10% *Accept* function is defined as:

$$top10\% = x_{[1..\alpha][1..n]},\ where\ \alpha = 10 \cdot p / 100 .$$

The following acceptance function is also static in nature but uses temporal information about the number of generations to determine the number of individuals to be used:

$$top20\%' = x_{[1..\alpha][1..n]},\ where\ \alpha = p \cdot \beta + \lfloor p \cdot \beta / t \rfloor,$$

where p is the size of the population and t is the current generation number. Since 0.2 is used for β, this is similar to selecting the top 20%. However, it will select more than that in the beginning of the search process and gradually reduce that amount as the process proceeds. As in simulated annealing, more information is pumped into the belief space at the onset than at the end.

3.7. *Design choices for the CAEP influence function*

The nature of the influence function will depend on the type of knowledge maintained in the belief space - whether it is normative, situational, or both. In addition, the influence functions can be categorized in terms of how they affect the variation operator, V_c. The possible actions considered here concern only changing mutation direction and mutation step size. Each of the influence functions that are used will now be discussed. The functions differ in terms of the type of knowledge employed and the extent to which that knowledge influences the variation operator.

3.7.1. *CAEP(Ns): Cultured EP using normative knowledge for step size*

This approach uses only normative knowledge in order to decide the step size of mutation. The population space for this case can be represented as:

$$P = \left\langle x_{[1..m,1..n]}, score_{[1..m]} \right\rangle, \text{ where } m=2p.$$

The variation is achieved by:

$$\forall i \in \{1,2,\cdots,p\}\ and\ \forall j \in \{1,2,\cdots,n\},$$
$$x_{p+i,j} = x_{i,j} + \left|u_j - l_j\right| \cdot N_{i,j}(0,1),$$

where $\left|u_j - l_j\right|$ represents the size of belief interval for the parameter j. The larger the belief interval is, the larger the step size used.

3.7.2. *CAEP(Sd): Using situational knowledge to influence mutation direction*

This approach uses only situational knowledge about the current best exemplar to decide the direction of mutation. The population structure for this case can be represented as:

$$P = \left\langle x_{[1..m,1..n]}, \sigma_{[1..m,1..n]}, score_{[1..m]} \right\rangle, \text{ where } m=2p.$$

The influence function used is:

$$\forall i \in \{1,2,\cdots,p\} \text{ and } \forall j \in \{1,2,\cdots,n\},$$

$$x_{p+i,j} = \begin{cases} x_{i,j} + \left|\sigma_{i,j} \cdot N_{i,j}(0,1)\right| & \text{if } x_{i,j} < E_j \\ x_{i,j} - \left|\sigma_{i,j} \cdot N_{i,j}(0,1)\right| & \text{if } x_{i,j} > E_j \\ x_{i,j} + \sigma_{i,j} \cdot N_{i,j}(0,1) & \text{otherwise} \end{cases}$$

where $\sigma_{i,j}$ represents the individual level step size for the ith variable of the jth individual. E_j is the parameter value for variable j in the current best performer contained in the situational knowledge component. This rule is based on the idea in Figure 3.1: If an individual's parameter value is less than that of the current best, then the absolute value of the calculated step size, $\left|\sigma_{i,j} \cdot N_{i,j}(0,1)\right|$, is added to the current parameter value. If an individual's parameter value is greater than that of the current best, then the absolute value of the step size, $\left|\sigma_{i,j} \cdot N_{i,j}(0,1)\right|$, is subtracted from the current parameter value. If an individual's parameter value is equal to that of the current best then $\sigma_{i,j} \cdot N_{i,j}(0,1)$ is just added to the current parameter value. In this case the direction of mutation will be random. Here, the step size, $\sigma_{i,j}$, is a constant or can be adjusted in a traditional way e.g. via Fogel's or Schwefel's method.

3.7.3. *CAEP(Ssd): Using situational knowledge to influence both mutation direction and step size*

The direction of mutation is determined in the same way as CAEP(Sd) in 3.7.2. However, the step size is decided by the distance between the parent and the current best. The larger the distance between the parent and the current best for parameter j is, the larger the step size used to modify the parameter. The variation operator for this CAEP(Ssd) in this case is:

$$\forall i \in \{1,2,\cdots,p\} \text{ and } \forall j \in \{1,2,\cdots,n\},$$

$$x_{p+i,j} = \begin{cases} x_{i,j} + \left|(x_{i,j} - E_j) \cdot N_{i,j}(0,1)\right| & \text{if } x_{i,j} < E_j \\ x_{i,j} - \left|(x_{i,j} - E_j) \cdot N_{i,j}(0,1)\right| & \text{if } x_{i,j} > E_j \\ x_{i,j} + \beta \cdot \left|x_{i,j} - E_j\right| \cdot N_{i,j}(0,1) & \text{otherwise} \end{cases}$$

where $\left|x_{i,j} - E_j\right|$ represents the distance between the parent and the current best individual. β is set to 0.3 here.

3.7.4. *CAEP(Ns+S_d): Cultured EP using normative knowledge to influence mutation step size and situational knowledge to influence mutation direction*

This approach uses normative knowledge to influence the step size, and situational

knowledge to determine the direction. The population space of this case can be represented as:

$$P = \langle x_{[1..m,1..n]}, score_{[1..m]} \rangle, \text{ where } m=2p.$$

The variation is achieved by:

$$\forall i \in \{1,2,\cdots,p\} \text{ and } \forall j \in \{1,2,\cdots,n\},$$

$$x_{p+i,j} = \begin{cases} x_{i,j} + |(u_j - l_j) \cdot N_{i,j}(0,1)| & \text{if } x_{i,j} < E_j \\ x_{i,j} - |(u_j - l_j) \cdot N_{i,j}(0,1)| & \text{if } x_{i,j} > E_j \\ x_{i,j} + |(u_j - l_j)| \cdot N_{i,j}(0,1) & \text{otherwise} \end{cases}$$

where E_j is the parameter value for the best exemplar. This is similar to CAEP(Sd) in 3.7.2 where the only difference is the use of $|u_j - l_j|$ in the belief space instead of $\sigma_{i,j}$ to determine step size.

3.7.5. *CAEP(Nsd): Cultured EP using normative knowledge to influence both mutation step size and direction*

This approach uses only normative knowledge to influence both mutation step size and direction. The basic idea is to produce a small random perturbation when the parameter value of a parent is within the current acceptable range. Otherwise perturb the parameter towards the closest current boundary (left or right) of the parameter. The variation is achieved by:

$$\forall i \in \{1,2,\cdots,p\} \text{ and } \forall j \in \{1,2,\cdots,n\},$$

$$x_{p+i,j} = \begin{cases} x_{i,j} + |(u_j - l_j) \cdot N_{i,j}(0,1)| & \text{if } x_{i,j} < l_j \\ x_{i,j} - |(u_j - l_j) \cdot N_{i,j}(0,1)| & \text{if } x_{i,j} > u_j \\ x_{i,j} + \beta \cdot |(u_j - l_j)| \cdot N_{i,j}(0,1) & \text{otherwise} \end{cases}$$

where l_j and u_j means the current lower limit and upper limit in the belief space for the parameter j, respectively. β is set to 0.2 here.

3.7.6. *CAEP(Ns+Sd'): Using normative knowledge to influence mutation step size and using modified situational knowledge to influence mutation direction*

In this approach we mutate *small* if the parent is *near* the current best, otherwise, mutate *large* using the current range size for the parameter in the belief space as the step size. This system can be represented as:

$$\forall i \in \{1,2,\cdots,p\} \text{ and } \forall j \in \{1,2,\cdots,n\},$$

$$x_{p+i,j} = \begin{cases} x_{i,j} + |(u_j - l_j) \cdot N_{i,j}(0,1)| & \text{if } x_{i,j} < (E_j - |(u_j - l_j)|/d) \\ x_{i,j} - |(u_j - l_j) \cdot N_{i,j}(0,1)| & \text{if } x_{i,j} > (E_j + |(u_j - l_j)|/d) \\ x_{i,j} + |(u_j - l_j)| \cdot N_{i,j}(0,1) & \text{otherwise} \end{cases}$$

where 2 is used for d.

3.7.7. *CAEP(Ns+Sd2): Using normative knowledge to influence mutation step size and using both the current and past situational knowledge to influence mutation direction*

Here, a second exemplar case representing the previous best is also used to guide mutation. We mutate *small* if the parent is between the previous best and current new best. Otherwise, we mutate *large* using the current range for the parameter in the belief space as the step size. This system can be represented as:

$\forall i \in \{1,2,\cdots,p\}$ *and* $\forall j \in \{1,2,\cdots,n\}$,

$$x_{p+i,j} = \begin{cases} x_{i,j} + \left| size(I_j) \cdot N_{i,j}(0,1) \right| & if\,(E_j^t < E_j^{t-1} \text{ and } x_{i,j} < E_j^t) or (E_j^t > E_j^{t-1} \text{ and } x_{i,j} < E_j^{t-1}) \\ x_{i,j} - \left| size(I_j) \cdot N_{i,j}(0,1) \right| & if\,(E_j^t < E_j^{t-1} \text{ and } x_{i,j} > E_j^t) or (E_j^t > E_j^{t-1} \text{ and } x_{i,j} > E_j^{t-1}) \\ x_{i,j} + size(I_j) \cdot N_{i,j}(0,1) & otherwise \end{cases}$$

where E_j^t is the best exemplar parameter value in the belief space at current time t, and E_j^{t-1} is the previous best exemplar parameter value at time t-1 for variable j.

3.8. *An example of one generation of the CAEP system*

As an example of the CAEP process works, consider a function $f(\mathbf{x}) = x_1^2 + x_2^2$ with the domain constraints $-1 < x_1 < 1$ and $-1 < x_2 < 1$. When we assume p = 5, the initial population consists of five candidate solutions would look like the figure 3.3.

	1	2	$f(\mathbf{x}_i)$
1	-0.1	0.2	0.05
2	0.0	0.1	0.01
3	-0.8	0.9	1.45
4	-0.3	0.7	0.58
5	-0.2	0.6	0.4

Figure 3.3 An example of initial population

The situational knowledge component, S, of the Belief Space will consist of the best individual from the initial population as shown in figure 3.4. The normative knowledge component, N, will be initialized with the range constraints for each of the problem parameters as shown in Figure 3.5.

Figure 3.6 shows the population structure after generating the p offspring solutions by applying the mutation operators with an influence function such as Nsd. Now, *there are* $2*p$ solutions. Figure 3.7 shows the result of tournament selection.

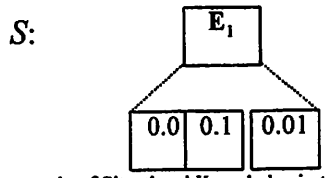

Figure 3.4 An example of Situational Knowledge in the Belief Space

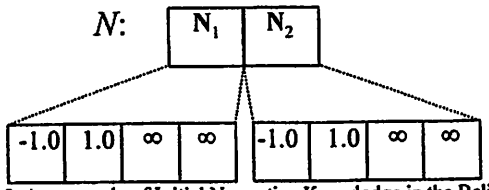

Figure 3.5 An example of Initial Normative Knowledge in the Belief Space

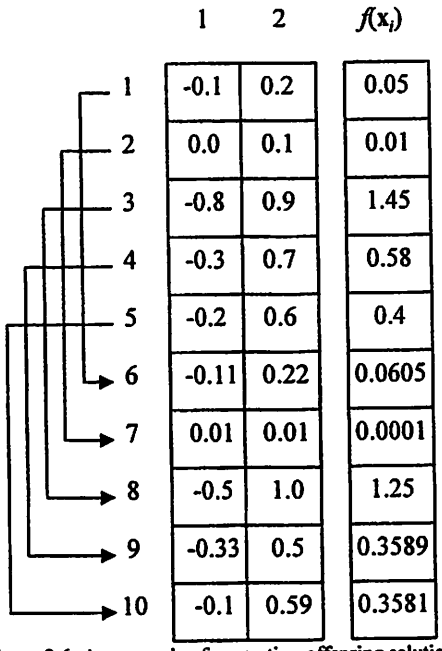

Figure 3.6 An example of generating offspring solutions

Assuming a top 40% acceptance function, the 2 best performers are accepted into the belief space as shown in the *accept* column in figure 3.8. The better of the two is a candidate for replacing the current best (the *update*E column). Both of them can contribute to updating the normative range of values for x_1 and x_2 as shown in the *update*N column in figure 3.8.

	1	2	$f(x_i)$
1	0.01	0.01	0.0001
2	0.0	0.1	0.01
3	-0.1	0.2	0.05
4	-0.11	0.22	0.0605
5	-0.1	0.59	0.3581

Figure 3.7 An example of tournament selection

	1	2	$f(x_i)$	*accept*	*update*E	*update*N
1	0.01	0.01	0.0001	1	1	1
2	0.0	0.1	0.01	1	0	1
3	-0.1	0.2	0.05	0	0	0
4	-0.11	0.22	0.0605	0	0	0
5	-0.1	0.59	0.3581	0	0	0

Figure 3.8 Individuals in a population for updating Belief Space

Figure 3.9 shows a result of adjusting situational knowledge from the population in the figure 3.8. Since the best individual has better performance value (0.0001) than that of the current exemplar, the current exemplar is replaced with the current best, <0.01, 0.01>, in the population space.

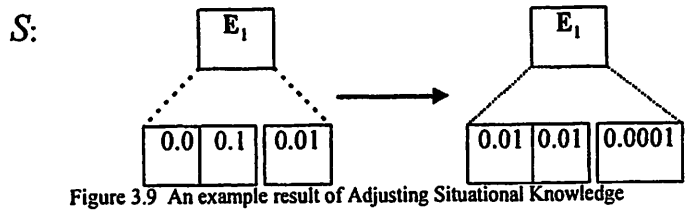

Figure 3.9 An example result of Adjusting Situational Knowledge

Figure 3.10 shows a result of adjusting normative knowledge according to the adjustment rules from the population in the figure 3.8. The top 2 individuals, <0.01, 0.01> with performance score 0.0001 and < 0.0, 0.1> with performance score 0.01 are used to adjust the current normative knowledge from the population.

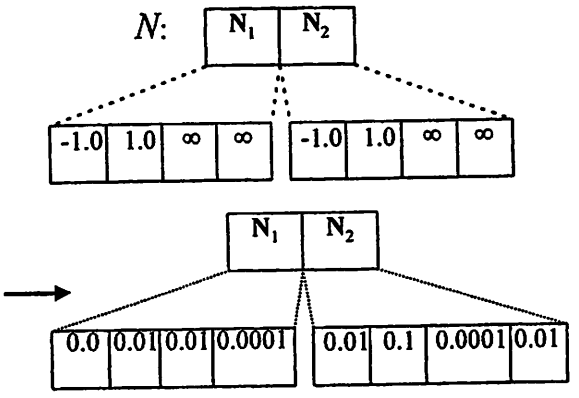

Figure 3.10 An example result of Adjusting Normative Knowledge

The individuals in figure 3.8 are then become the parents for the next generation of the CAEP system and the process begins anew.

4. Metrics to Measure the Complexity of Function Optimization Problems

The "No Free Lunch Theorem for Search" says that "if algorithm *A* outperforms algorithm *B* on some objective functions, then there must exist exactly as many other functions where *B* outperforms *A*" [Wolpert 1995]. This theorem implies that no single search algorithm, or in this case Cultural Algorithm configuration, is best for all optimization problems. Thus, we apply our various configurations to optimization problems taken from a number of different classes in order to discern what type of knowledge is most suitable for a given problem type. The selected functions represent a broad spectrum of function optimization problems and reflect different degrees of complexity [Bohachevsky 1986; De Jong 1975; Hoffmeister 1992; Kim 1995; Fogel 1995; Koon 1995; Salomon 1996; Michalewicz 1994; Schwefel 1995; Whitley 1995; Whitley 1996; Yao 1996]. Some generalized versions of functions (F9, F15, F16) are tested by varying the number of parameters in order to evaluate the impact of dimensionality on the solution process. The functions are listed in Figure 4.1.

4.1. *Characterizing the difficulty of function in terms of its fitness landscape*

What aspects of the fitness landscape for a function make the optimization process difficult? And, what type of belief space knowledge is most effective for searching particular types of fitness landscapes? In order to answer these questions, we have classified each of the functions in Figure 4.1 in terms of landscape features that have observed by AI researchers to affect the search process. Winston [1992] identified basic features of a fitness landscape that have been shown to impact search effort. They include modality, basins, valleys, decomposability, and dimensionality. Each of this will now briefly be discussed.

4.1.1. *Modality*

The modality of a function corresponds to the number of false peaks in its fitness

Sphere: $f_1(x) = \sum_{i=1}^{n} x_i^2$

Rosenbrock: $f_2(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$

Step: $f_3(x) = \sum_{i=1}^{5} \lfloor x_i \rfloor$

De Jong f4 w/o noise: $f_4(x) = \sum_{i=1}^{n} i \cdot x_i^4$

De Jong f4 with noise: $f_4'(x) = \sum_{i=1}^{n} [i \cdot x_i^4 + N(0,1)]$

Shekel's foxholes: $f_5(x) = 1 \Big/ 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}$

Bohachevsky: $f_6(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$

Rastrigin: $f_7(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$

Colville: $f_8(x) = 100(x_1^2 - x_2)^2 + (1-x_1)^2 + 90(x_4 - x_3^2)^2 + (1-x_3)^2 + 0.1\left((x_2 - 1)^2 + (x_4 - 1)^2\right) + 19.8(x_2 - 1)(x_4 - 1)$

Ackley: $f_9(x) = -20\exp(-0.2\sqrt{1/n \cdot \sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos 2\pi x_i) + 20 + e$

Goldstein-Price: $f_{10}(x) = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \cdot \left[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$

6 hump camel back: $f_{11}(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$

Easom: $f_{12}(x) = -\cos(x_1)\cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2}$

Floudas: $f_{13}(x) = -5\sin(x_1)\sin(x_2)\sin(x_3)\sin(x_4)\sin(x_5) - \sin(5x_1)\sin(5x_2)\sin(5x_3)\sin(5x_4)\sin(5x_5)$

Watson: $f_{14}(x) = \sum_{i=1}^{30}\left(\sum_{j=1}^{5}(ja_i^{j-1}x_{j+1}) - \left[\sum_{j=1}^{6}(a_i^{j-1}x_j)\right]^2 - 1\right)^2 + x_1^2$, where $a_i = \frac{i-1}{29}$

Koon's F3: $f_{15}(x) = \frac{1}{2}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i)$

Griewank: $f_{16}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^4 - \prod_{i=1}^{n} \cos(x_i \Big/ \sqrt{i}) + 1$

Kowalik: $f_{17}(x) = \sum_{i=1}^{11}\left(a_i - \frac{x_1(b_i^2 + b_ix_2)}{b_i^2 + b_ix_3 + x_4}\right)^2$

Matyas: $f_{18}(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$

Zettl: $f_{19}(x) = (x_1^2 + x_2^2 - 2x_1)^2 + 0.25x_1$

Michalewicz: $f_{20}(x) = -\sum_{i=1}^{n} \sin(x_i) \cdot \sin^{2m}(i \cdot x_i^2 \Big/ \pi)$, $m = 20$

Beale: $f_{21}(x) = \left[1.5 - x_1(1 - x_2)\right]^2 + \left[2.25 - x_1(1 - x_2^2)\right]^2 + \left[2.625 - x_1(1 - x_2^3)\right]^2$

Langerman: $f_{22}(x) = -\sum_{i=1}^{m} c_i\left(e^{-\|\bar{x} - A(i)\|^2/\pi} \cos(\pi \cdot \|\bar{x} - A(i)\|^2)\right)$, $m = 5$

Ellipsoid: $f_{23}(x) = \sum_{i=1}^{n} 10^{i-1}x_i^2$

Quadratic: $f_{24}(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$

Schwefel 2.22: $f_{25}(x) = \sum_{i=1}^{n}|x_i| + \prod_{i=1}^{n}|x_i|$

Schwefel 2.21: $f_{26}(x) = \max_i\{|x_i|, 1 \le i \le n\}$

Box: $f_{27}(x) = \sum_{j=1}^{10}\left(e^{(-0.1jx_1)} - e^{(-0.1jx_2)} - x_3\left[e^{(-0.1j)} - e^{(-j)}\right]\right)^2$

F28, Subert: $f_{28}(x) = \sum_{i=1}^{5} i\cos[(i+1)x_1 + i] \cdot \sum_{i=1}^{5} i\cos[(i+1)x_2 + i]$

F29, Branin RCOS: $f_{29}(x) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)\cos(x_1) + e$

F30, Rastrigin2D: $f_{30}(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$

Figure 4.1: Function Test Suite

landscape. Each false peak is a possible attractor for problem solvers. Once a problem solver reaches a false peak, there is a tendency to stay there. This can result in directing the search away from better ones. In terms of the belief space, it can slow down the specialization process (narrowing of parameter intervals), and/or decrease the reliability of the belief information.

4.1.2. *Basins*

A basin occurs when a function has a relatively steep decline surrounding a large region where little progress is achieved via the changing of parameters. Problem solvers are easily attracted to these regions, but make little progress once they get there. A basin corresponds to a plateau for maximization problems. The presence of basins in a function can slow down the specialization of parameter ranges in the belief space.

4.1.3. *Valleys*

A valley occurs when a narrow area of little change is surrounded by regions of steep descent. As with basins, minimizers are initially attracted to this region. However, once on the valley floor progress can be slowed considerably. This feature corresponds to a ridge in maximization problems. Valleys can cause specialization to remove desirable subranges, or to proceed slowly within desirable subrange or both.

4.1.4. *Decomposability*

If each parameter is independent of the other parameters in a function, the function is regarded as relatively easy to optimize, since the landscape can be decomposed into simpler components. For example, if all parameters are independent, then the optimization process can be performed in a sequence of n independent optimization processes, where each parameter is optimized independently. As an example, consider the minimization problem: $z = x^2 + y^2$. Regardless of the value of y, we can find the solution, x=0, which minimizes z, and vice versa for y. This objective function is referred to as decomposable or separable [Hadley 1969]. However, many functions such as $f(\mathbf{x}) = (x_1^2 + x_2^2)^4$ are not decomposable in this manner. But such functions are very easy to optimize, since their first derivative is a product, such as $\partial f(\mathbf{x})/\partial x_1 = 8x_1(x_1^2 + x_2^2)^3$. Such a product yields a solution for $x_1 = 0$ that is independent of x_2. From this observation, the following general condition was developed by Salomon in order to determine whether certain functions are easy to optimize or not [Salomon 1996].

$$\frac{\partial f(\vec{x})}{\partial x_i} = g(x_i)h(\vec{x}) \tag{4.1}$$

where $g(x_i)$ means any function of only x_i and $h(\vec{x})$ means any function of any $\vec{x}$. If this condition is satisfied, the function is said to be easy to optimize as a partially decomposable function, because solutions for each x_i can still be obtained independently of all other parameters. For example, F1 with 2 parameters is easy to optimize since F1 satisfies the condition (4.1), as shown below.

$$\frac{\partial f_1(x_1,x_2)}{\partial x_1} = 2x_1 \qquad \frac{\partial f_1(x_1,x_2)}{\partial x_2} = 2x_2$$

Fn	Fn Name	Search Space	Value of optimum: $f(x^*)$	n	M	B	V	D	*Number of iterations by (4.4)*
1	Sphere	$[-5.12, 5.12]^n$	0	30	0	0	0	0	650
2	Rosenbrock	$[-6.0, 6.0]^n$	0	2	0	0	2	1	440
3	Step	$[-5.12, 5.12]^n$	-3.00000E+01	5	0	0	0	0	275
4	De Jong f4 w/o noise	$[-1.28, 1.28]^n$	0	30	0	0	0	0	650
4'	De Jong f4 with noise	$[-1.28, 1.28]^n$	0	30	0	0	0	0	650
5	Shekel's foxholes	$[-65.536, 65.536]^n$	9.98004E-01	2	2	0	0	1	440
6	Bohachevsky	$[-6.0, 6.0]^n$	0	2	1	0	0	1	320
7	Rastrigin	$[-5.12, 5.12]^n$	0	5	3	0	0	0	800
8	Colville	$[-10.0, 10.0]^n$	0	4	0	2	2	1	2120
9	Ackley	$[-32.0, 32.0]^n$	0	3	2	0	0	0	380
9'	Ackley'	$[-32.0, 32.0]^n$	0	30	2	0	0	0	2000
10	Goldstein-Price	$[-2.0, 2.0]^n$	3.00000E+00	2	0	1	0	1	320
11	6 hump camel back	$[-3.0, 3.0]^n$	-1.03160E+00	2	0	0	1	1	320
12	Easom	$[-100.0, 100.0]^n$	-1.00000E+00	2	2	0	0	0	320
13	Floudas	$[0.0, \pi]^n$	-6.00000E+00	5	2	0	0	1	800
14	Watson	$[-2.0, 2.0]^n$	2.28800E-03	6	0	2	2	1	3080
15	Koon's F3	$[-5.0, 5.0]^n$	-1.174985E+02	3	1	0	0	0	290
15'	Koon's F3	$[-5.0, 5.0]^n$	-1132.575	30	1	0	0	0	1100
16	Griewank	$[-600.0, 600.0]^n$	0	3	1	0	0	1	380
16'	Griewank	$[-600.0, 600.0]^n$	0	30	1	0	0	1	2000
17	Kowalik	$[-1.0, 1.0]^n$	3.0750E-04	4	0	2	2	1	2120
18	Matyas	$[-10.0, 10.0]^n$	0	2	0	1	0	1	320
19	Zettl	$[-2.0, 2.0]^n$	-3.7910E-03	2	0	1	0	1	320
20	Michalewicz	$[0.0, \pi]^n$	-4.68700E+00	5	2	0	0	1	800
21	Beale	$[-5.0, 5.0]^n$	0	2	0	1	0	1	320
22	Langerman	$[0.0, 10.0]^n$	-1.40000E+00	5	2	0	0	1	800
23	Ellipsoid	$[-5.12, 5.12]^n$	0	30	0	0	1	0	1100
24	Quadratic	$[-5.12, 5.12]^n$	0	30	0	0	1	0	1100
25	Schwefel 2.22	$[-10.0, 10.0]^n$	0	30	0	0	0	0	650
26	Schwefel 2.21	$[-100.0, 100.0]^n$	0	30	0	0	0	0	650
27	Box	$[-10.0, 10.0]^n$	0	3	0	2	0	1	560
28	Subert	$[-10.0, 10.0]^n$	-186.73	2	3	1	0	1	1160
29	Branin RCOS	$[-5,10][0,15]$	0.397887	2	0	2	2	1	1160
30	Rastrigin2D	$[-3.0, 3.0]^n$	-2.0	2	3	0	0	0	440

Table 4.1 Characteristics of Test Functions Used

In this example, $h(x)$ is regarded as 1. As an another example, consider the Rosenbrock function, $f_2(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$. This is not easy to optimize, since the equations below do not satisfy condition (4.1).

$$\frac{\partial f_1(x_1, x_2)}{\partial x_1} = 400(x_1^2 - x_2)x_1 - 2x_1 - 2 \text{ and } \frac{\partial f_1(x_1, x_2)}{\partial x_2} = -200(x_1^2 - x_2)$$

4.1.5. *Dimensionality*

As the number of parameters or dimensions increase, the search space also increases exponentially in size [Winston 1992; Michalewicz 1992; Yao 1996]. In terms of the belief space, more problem solving knowledge must be kept and the update function can become more complicated, especially if the parameters are of heterogeneous types.

4.2. *Measuring the effort required to search a given functional landscape*

Kauffman developed a framework for modeling the complexity of a functional landscape for functions defined over binary strings of length n, where each bit corresponds to a single parameter [Kauffman 1993]. The fitness contribution of the each of the n parameters can be affected by k other parameters on the average. The ***nk*** model is used to classify fitness landscapes in terms of these two basic parameters. Families of fitness landscapes can then be described using the ***nk*** model. For example, if k=0, then the landscape possesses a single optimum genotype or n bit binary sequence. If k=n-1, then the landscape is said to be completely random with a large number of local optima. It has also been shown that decision problem for the function is *np*-complete when $k \geq 3$ for functions where the positions of the interacting loci on the chromosome are random [Weinberger 1996].

Here we are interested in predicting the amount of effort taken by the population, in terms of generations, to find the optimum point in an n dimensional search space whose landscape is characterized by the features described in 4.1. This will allow us to predict the extent to which systems are able to consistently solve the targeted problems in a time that is proportional to the basic features of the problem's functional landscape. Thus, the time allowed for all systems on all problems will be limited by the effort estimator. The more general purpose a system is, the more frequently it should be able to solve the problems within the prescribed guidelines. This will allow us to more effectively isolate those systems whose performance is most affected by the landscape parameters, one of the major goals of this paper.

The approach to effort prediction taken here corresponds to macro-models of effort developed within software engineering. These models estimate the amount of time taken by a group to deliver a solution to a given problem. Since effort prediction is inherently a non-linear function of a number of project variables, the majority of successful models are non-linear in nature. So, we will focus on these non-linear effort estimation models [Pressman 1997]. Most of the non-linear estimation models proposed in the software engineering literature take the following form:

$$E = \left(a + bS^c\right) \cdot m(\chi), \tag{4.2}$$

where E is the effort to be predicted. a, b, and c are constants usually produced via regression analysis using example problems in a given domain [Conte 1986]. S is the size of the project, and $m(\chi)$ is an adjustment multiplier that depends on one or more cost-driver attributes. For software projects S is typically the lines of code for the problem.

$m(\chi)$ can be a complicated non-linear function of these cost drivers. Cost drivers are those aspects of a problem that function to drive up the effort required to solve it. The cost drivers used here correspond to the features of the functional landscape viewed by Winston to complicate the problem solving process. S corresponds to the dimensionality of the search space n as in Kauffman's approach. The basic formulation based upon (4.2) is then:

$$Effort=\left(a+b\cdot n^{c}\right)\cdot m(M,B,V,D). \tag{4.3}$$

The cost driver multiplier function is viewed as being inherently non-linear, $2^{(M+B+V+D)}$, where the value for M, B, V, and D are given below:

M: Modality
- 0 - Uni-modal (the number of local optima is 0)
- 1 - Medium-modal (the number of local optima is 1, when n=2)
- 2 - Multi-modal (1 < the number of local optima < 50, when n=2)
- 3 - Highly-multi-modal (the number of local optima ≥ 50, when n=2)

B: Basin
- 0 - No basin
- 1 - Medium flat basin
- 2 - Large flat basin

V: Valley
- 0 - No Valley
- 1 - Narrow Valley
- 2 - Wide Valley

D: Decomposability
- 0 - Decomposable, *or* Partially-decomposable, when condition (4.1) is met
- 1 - Not-decomposable, when condition (4.1) is not met.

The weights for each cost driver are initially equal although the variables differ in terms of their individual values. The more quantifiable variables such as modality contribute slightly more since they possess larger ranges of attribute values. If it turns out that some attributes need to contribute more to the projected cost, then many systems will not be able to solve the problem within the time constraint for the problems with those attributes. In that case, we can adjust the weights in future versions in order to allow them more influence. However, the current comparisons will be based upon equal weights for the cost-driver variables.

Given the above effort estimator, a statistical analysis of example problems from our set of 34 optimization problems leads to values for the constants a, b, and c of 200, 15, and 1 respectively. The overall equation for effort prediction used is:

$$Effort(n,M,B,V,D)=\left(200+15\cdot n\right)\cdot 2^{(M+B+V+D)}. \tag{4.4}$$

The values for the predicted number of generations for each of the 34 function are given in the table 4.1. The first column gives the identification number of the function in this study, and the second column gives the name of the function. The third column specifies the domains and $S\subset\Re^{n}$. The fourth column gives the best known solution to each problem. The fifth column, n, provides the numbers of parameters or dimensionality used. Columns 6 through 9 give the level for each of the landscape features, M, P, V, and D. The last column (column 10) gives the effort estimator value for the function which is used to limit the number of generations used to solve the problem by each of the candidate solutions.

4.3. *Effort estimation and problem solving diversity*

Cultural systems have been successful because of their ability to solve a diverse number of problems. Given the CAEP framework described here, we suggest that there are two basic aspects to this. First, we need to predict the amount of effort required to solve the problem. Second, we need to select the appropriate configuration for the CAEP that can solve the problem within those guidelines.

Here we have developed a non-linear effort estimation function based upon a set of landscape features as cost drivers. This model was then used as the basis for allocating problem solving effort. If it is an appropriate measure then we should be able to produce an acceptable solution within the produced time frame. We then will determine which configurations of CAEP are able to perform acceptably within those guidelines. The best CAEP configuration that is able to solve the problem within these guidelines is then stored and reused when a similar problem reappears in the future. This is the topic of sections 5, 6, and 7. If no system is able to produce acceptable performance within the predicted effort constraints for a given problem, the cost driver parameters associated with the problem's structure may need to be adjusted in the future. Here, it turns out that only 7 out of the 34 function cannot be solved within the predicted effort guidelines. This information will then be used to update the cost drivers in future work.

5. Performance Comparison Between Self-Adaptive EPs and Example CAEP Systems

In this section we compare the relative performance of various self-adaptive population-only EP approaches with a selected group of CAEP systems. The four EP systems tested are given in table 5.1. For each system, the level of self-adaptation exploited and the parameters values used for the self-adaptation process are given.

Self-adaptive level	System Name	Parameters Used
Component Level	Schwefel-like	Initial sigma(σ) = 5.0
	Fogel	Initial sigma(σ) = 2.0, β = 0.2, ε = 0.001
Individual Level	Age	Based on Fogel's sigma update rule
Population Level	1/5 rule	Initial omega(ω)=3.0. $c_d = 0.82$, $c_i = 1.0/0.82$ as recommended by Schwefel [1981]. If ω becomes greater than 3.0, it is reset to 3.0; If ω becomes zero, it is set to 0.002. $k = 1$

Table 5.1 Population-only Systems

Seven CAEP configurations were tested. Each system employed the same belief and population component but differed in how the two components interacted via the communication protocol. The various possible protocols that can be produced by combining the acceptance and influence function described earlier are given in table 5.2. The seven CAEP systems tested in this section represent the row of the table associated with the top 20% acceptance function. Other rows of the table will be investigated in later sections.

For each of the 34 functions, 15 runs were performed for each of the EP and CAEP systems tested. The goal of this research is to determine how certain features of a problem's functional landscape affect the optimal level of self-adaptation needed and the nature of the knowledge used to guide self-adaptation with the belief space. Thus, a number of functions with a variety of different landscape features were selected for use here so that each category of landscape feature has at least two functions that support it. As a result, each feature value category has at least 30 runs for any given system configuration. This facilitated the performance comparisons between systems in terms of each landscape feature to be addressed in section 7.

	Sd	Ssd	Ns	Nsd	Ns+Sd	Ns+Sd_2	Ns+Sd'
top10%	✓	✓	✓	✓	✓	✓	✓
top20%	✓	✓	✓	✓	✓	✓	✓
mod. top20%	✓	✓	✓	✓	✓	✓	✓
top30%	✓	✓	✓	✓	✓	✓	✓
top40%	✓	✓	✓	✓	✓	✓	✓
above median	✓	✓	✓	✓	✓	✓	✓
above avg.	✓	✓	✓	✓	✓	✓	✓

Table 5.2 Implemented CAEP Systems

For a fair comparison, all of the experiments employ the following:

- the same population size, $p = 40$,
- the same test suite of 34 performance functions,
- the same maximum number of generations as estimated by the effort metric function. As a result, the number of allowed function evaluations will be the same for all systems for a given problem. Note that number of function evaluations will be p, where p is population size, multiplied by the number of maximum generations for each system.
- the same method to handle infeasible individuals, and
- the same number of tournament competitions (c=40).
- In addition, the search process starts with the same random initial population for each system, and for every problem.

The method used to handle infeasible solutions is a type of repair method. In this method, if a mutation violates the boundary for the parameter, the new parameter value will be forced to fall stochastically within the domain range. This is achieved by the following repair rule.

$$x_{p+1,j} = \begin{cases} dl_j + |N_{i,j}(0,0.1)| & \text{if } x_{p+1,j} < dl_j \\ du_j - |N_{i,j}(0,0.1)| & \text{if } x_{p+1,j} > du_j \\ x_{p+1,j} & \text{otherwise} \end{cases}$$

where dl_j and du_j represent the lower and upper limits respectively of the domain constraint for parameter j.

5.1. *Experimental results*

Tables 5.3.1 through table 5.3.34 show the test results for the population only self-adaptive EPs and CAEP systems. The first value for each system gives the average

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
18078 (0%)	17166 (0%)	16730 (0%)	10016 (100%)	18751 (0%)	12544 (0%)	14248 (0%)	3020 (100%)	12926 (0%)	4233 (100%)	9180 (100%)
3.20010e+00	5.93094e+00	1.90861e+00	3.03608e-08	1.99207e-01	5.53093e+00	1.80092e+02	5.78728e-33	1.08980e+02	2.30567e-25	3.70728e-10

Table 5.3.1 Comparison of EP and CAEP systems for F1 (Sphere with 30 parameters)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
3848 (13%)	3203 (73%)	2543 (73%)	1637 (66%)	2713 (60%)	3634 (0%)	2309 (100%)	1688 (60%)	808 (100%)	879 (86%)	767 (93%)
3.14361e-03	1.25648e-05	7.26822e-06	2.79841e-05	7.45259e-06	1.15479e-01	2.07043e-09	6.66574e-04	1.27236e-25	3.07062e-04	2.45168e-07

Table 5.3.2 Comparison of EP and CAEP systems for F2 (Rosenbrock)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
662 (100%)	634 (100%)	538 (100%)	1054 (66%)	1654 (60%)	2194 (20%)	306 (100%)	248 (100%)	146 (100%)	702 (80%) -2.98000	224 (100%)
-3.00000 e+01	-3.00000 e+01	-3.00000 e+01	-2.96667 e+01	-2.96000 e+01	-2.90000 e+01	-3.00000 e+01	-3.00000 e+01	-3.00000 e+01	e+01	-3.00000 e+01

Table 5.3.3 Comparison of EP and CAEP systems for F3 (Step)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
19980 (0%)	20175 (0%)	17681 (0%)	5982 (100%)	15358 (80%)	12958 (0%)	14616 (0%)	1703 (100%)	13082 (0%)	2693 (100%)	5699 (100%)
4.81895e-01	2.93272e+01	1.35881e+00	2.91640e-15	7.95456e-05	2.32626e-01	9.43682e+01	1.35578e-57	1.98139e+01	9.14804e-40	7.91274e-17

Table 5.3.4 Comparison of EP and CAEP systems for F4 (DeJongF4 without noise)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
12782 (80%)	23618 (6%)	10520 (100%)	1168 (100%)	553 (100%)	1434 (93%)	18180 (0%)	360 (100%)	16725 (0%)	588 (100%)	1470 (100%)
-9.47151 e+00	3.43623 e+01	-8.26850 e+00	-1.32272 e+01	-1.81543 e+01	-1.11804 e+01	9.34552e+01	-2.02443 e+01	1.94380 e+01	-1.70536 e+01	-1.22123 e+01

Table 5.3.5 Comparison of EP and CAEP systems for F4' (DeJongF4 with noise)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
4515 (26%)	5877 (0%) 2.98211e+00	4766 (13%)	4844 (6%) 2.78383e+00	5210 (6%) 2.71782e+00	3442 (33%)	1552 (100%)	2336 (60%)	700 (100%)	3255 (40%)	1709 (73%)
2.05698e+00		2.58555e+00			2.45118e+00	9.98004e-01	1.52763e+00	9.98004e-01	1.79244e+00	1.46109e+00

Table 5.3.6 Comparison of EP and CAEP systems for F5 (Foxholes)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
522 (100%) 0.00000e+00	1304 (93%) 6.44192e-08	827 (100%) 3.25622e-18	502 (100%) 0.00000e+00	428 (93%) 1.45542e-02	439 (87%) 2.91084e-02	331 (100%) 0.00000e+00	158 (100%) 0.00000e+00	170 (100%) 0.00000e+00	155 (100%) 0.00000e+00	158 (100%) 0.00000e+00

Table 5.3.7 Comparison of EP and CAEP systems for F6 (Bohachevsky)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
10372 (0%) 4.11250e+00	9073 (0%) 2.24908e+00	9027 (0%) 4.90847e+00	7996 (0%) 2.98488e+00	9640 (0%) 5.37278e+00	8110 (0%) 6.23507e+00	8447 (0%) 1.10311e+01	4006 (60%) 3.97984e-01	8096 (0%) 4.70164e+00	7300 (13%) 2.38790e+00	7019 (20%) 1.32661e+00

Table 5.3.8 Comparison of EP and CAEP systems for F7 (Rastrigin)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
23152 (0%) 5.17213e-01	21402 (0%) 2.62341e-01	22905 (0%) 1.33899e+00	18597 (6%) 6.51051e-04	22884 (0%) 1.58317e-02	19505 (0%) 5.80782e+00	20934 (0%) 4.26915e-01	18988 (6%) 7.67973e-02	9800 (100%) 3.38135e-12	20004 (0%) 1.04867e+00	19754 (13%) 2.91436e-05

Table 5.3.9 Comparison of EP and CAEP systems for F8 (Colville)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
1462 (100%) -1.43982e-16	3695 (33%) 8.92524e-05	3290 (86%) 4.31225e-07	1419 (100%) -1.43982e-16	858 (100%) -1.43982e-16	516 (93%) 1.41337e-01	1594 (100%) -1.43982e-16	430 (100%) -1.43982e-16	528 (100%) -1.43982e-16	395 (100%) -1.43982e-16	421 (100%) -1.43982e-16

Table 5.3.10 Comparison of EP and CAEP systems for F9 (Ackley)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
64164 (0%) 9.82107e+00	60228 (0%) 8.73339e+00	62732 (0%) 1.68277e+01	44434 (13%) 2.51271e+00	68890 (0%) 1.40379e+01	47214 (0%) 1.23528e+01	57514 (0%) 2.05969e+01	6953 (100%) 3.88243e-15	51560 (0%) 1.99161e+01	17445 (80%) 2.28458e-01	21223 (100%) 1.33563e-14

Table 5.3.11 Comparison of EP and CAEP systems for F9' (Ackley, 30 parameters)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
542 (100%) 3.00000e+00	1530 (100%) 3.00000e+00	1007 (100%) 3.00000e+00	593 (100%) 3.00000e+00	281 (100%) 3.00000e+00	106 (100%) 3.00000e+00	340 (100%) 3.00000e+00	161 (100%) 3.00000e+00	313 (100%) 3.00000e+00	150 (100%) 3.00000e+00	158 (100%) 3.00000e+00

Table 5.3.12 Comparison of EP and CAEP systems for F10 (Goldstein-Price)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd_2)2 0
288	710	381	332	144	70 (100%)	2376	410	234	110	153
(100%)	(100%)	(100%)	(100%)	(100%)	-1.03163	(26%)	(100%)	(100%)	(100%)	(100%)
-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	e+00	-1.03154	-1.03163	-1.03163	-1.03163	-1.03163
e+00	e+00	e+00	e+00	e+00		e+00	e+00	e+00	e+00	e+00

Table 5.3.13 Comparison of EP and CAEP systems for F11 (6 hump Camel Back)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd_2)2 0
760	1599	1636	1977	907	154	2852 (0%)	304	760	264	346
(100%)	(100%)	(93%)	(46%)	(100%)	(100%)	-1.74350	(100%)	(100%)	(100%)	(100%)
-1.00000	-1.00000	-9.33333	-4.66667	-1.00000	-1.00000	e-01	-1.00000	-1.00000	-1.00000	-1.00000
e+00	e+00	e-01	e-01	e+00	e+00		e+00	e+00	e+00	e+00

Table 5.3.14 Comparison of EP and CAEP systems for F12 (Easom)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd_2)2 0
2064	4988	2738	1289	819	300	3160	364	652	363	401
(100%)	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)
-6.00000	-6.00000	-6.00000	-6.00000	-6.00000	-6.00000	-6.00000	-6.00000	-6.00000	-6.00000	-6.00000
e+00	e+00	e+00	e+00	e+00	e+00	e+00	e+00	e+00	e+00	e+00

Table 5.3.15 Comparison of EP and CAEP systems for F13 (Floudas)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd_2)2 0
47331	45401	45240	40274	68122	40768	42179	41384	40216	42343	40086
(0%)	(0%)	(0%)	(0%)	(0%)	(0%)	(0%)	(0%)	(6%)	(0%)	(6%)
1.26697e-02	9.60621e-03	7.39157e-03	6.23111e-03	4.85733e-03	3.04712e-02	5.97963e-02	4.05663e-03	2.63019e-03	2.98548e-03	2.42833e-03

Table 5.3.16 Comparison of EP and CAEP systems for F14 (Watson)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd_2)2 0
677	1911	1029	632	316	124	792	208	248	186	219
(100%)	(93%)	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)
-1.17499	-1.17499	-1.17499	-1.17499	-1.17499	-1.17499	-1.17499	-1.17499	-1.17499	-1.17499	-1.17499
e+02	e+02	e+02	e+02	e+02	e+02	e+02	e+02	e+02	e+02	e+02

Table 5.3.17 Comparison of EP and CAEP systems for F15 (Koon's F3)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd_2)2 0
31332	29367	29077	20139	32458	22065	24759	14416	22892	22910	22398
(0%)	(0%)	(0%)	(0%)	(0%)	(0%)	(0%)	(40%)	(0%)	(0%)	(0%)
-1.02217	-1.01328	-1.02964	-1.04681	-1.03244	-9.88555	-4.94748	-1.12127	-5.04463	-1.06755	-1.07132
e+03	e+03	e+03	e+03	e+03	e+02	e+02	e+03	e+02	e+03	e+03

Table 5.3.18 Comparison of EP and CAEP systems for F15' (Koon's F3, 30 parameters)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
3532 (13%) 1.73105e-01	3746 (0%) 1.26008e+00	3756 (0%) 5.41692e+00	3448 (0%) 2.43707e+00	3783 (6%) 1.70542e+00	3252 (7%) 5.99856e-02	3486 (0%) 6.30901e-02	4947 (46%) 4.29343e-03	3479 (0%) 3.29752e-02	3551 (0%) 1.05176e-02	3164 (26%) 9.80937e-03

Table 5.3.19 Comparison of EP and CAEP systems for F16 (Griewank, 3 parameters)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
66490 (0%) 7.46768e+00	62716 (0%) 8.81806e+01	62728 (0%) 1.59646e+02	40156 (40%) 7.71831e-03	68463 (0%) 2.22544e+01	49331 (0%) 1.58620e+01	55056 (0%) 6.19296e+02	15091 (80%) 1.97097e-03	50048 (0%) 2.76581e+02	33716 (40%) 7.87325e-03	36403 (40%) 1.29595e-02

Table 5.3.20 Comparison of EP and CAEP systems for F16' (Griewank, 30 parameters)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
16762 (73%) 3.31013e-04	11925 (86%) 3.07548e-04	10528 (93%) 3.07495e-04	18319 (13%) 3.22298e-04	7448 (100%) 3.07486e-04	20670 (0%) 3.86840e-04	10334 (100%) 3.07486e-04	26621 (86%) 3.07515e-04	1664 (100%) 3.07486e-04	5503 (100%) 3.07486e-04	4031 (100%) 3.07486e-04

Table 5.3.21 Comparison of EP and CAEP systems for F17 (Kowalik)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
418 (100%) 3.55255e-37	1062 (100%) 4.17039e-11	584 (100%) 1.78986e-16	378 (100%) 1.78872e-39	226 (100%) 1.07046e-71	163 (100%) 2.18554e-39	405 (100%) 7.26990e-38	200 (100%) 2.01219e-16	164 (100%) 7.23829e-95	142 (100%) 2.39657e-104	152 (100%) 1.17797e-72

Table 5.3.22 Comparison of EP and CAEP systems for F18 (Matyas)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
406 (100%) -3.79124 e-03	1044 (100%) -3.79122 e-03	548 (100%) -3.79124 e-03	391 (100%) -3.79124 e-03	182 (100%) -3.79124 e-03	98 (100%) -3.79124 e-03	274 (100%) -3.79124 e-03	120 (100%) -3.79124 e-03	121 (100%) -3.79124 e-03	110 (100%) -3.79124 e-03	111 (100%) -3.79124 e-03

Table 5.3.23 Comparison of EP and CAEP systems for F19 (Zettl)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
8656 (20%) -4.44493 e+00	9468 (6%) -4.54016 e+00	9817 (0%) -4.48755 e+00	8231 (6%) -4.45124 e+00	8444 (20%) -4.50521 e+00	8862 (0%) -4.30399 e+00	9078 (0%) -4.09251 e+00	5569 (40%) -4.66260 e+00	6242 (53%) -4.65582 e+00	7888 (13%) -4.52619 e+00	5308 (46%) -4.62732 e+00

Table 5.3.24 Comparison of EP and CAEP systems for F20 (Michalewicz)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
566 (100%)	1430 (93%)	828 (100%)	403 (100%)	270 (100%)	307 (100%)	483 (100%)	210 (100%)	185 (100%)	195 (100%)	168 (100%)
3.84244e-29	7.24379e-08	6.98197e-14	0.00000e+00	0.00000e+00	2.07901e-23	1.83926e-31	1.10684e-15	0.00000e+00	0.00000e+00	1.41164e-30

Table 5.3.25 Comparison of EP and CAEP systems for F21 (Beale)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
10505 (13%)	5096 (73%)	6415 (53%)	9443 (6%) -7.71788	11048 (6%)	8825 (13%)	8190 (46%)	1666 (86%)	4576 (60%)	8354 (20%)	6200 (46%)
-9.36576 e-01	-1.35486 e+00	-1.20349 e+00	e-01	-7.75305 e-01	-7.62846 e-01	-1.16713 e+00	-1.41373 e+00	-1.27381 e+00	-9.83547 e-01	-1.18125 e+00

Table 5.3.26 Comparison of EP and CAEP systems for F22 (Langerman)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
32698 (0%)	30630 (0%)	30337 (0%)	21333 (0%)	33574 (0%)	23094 (0%)	25726 (0%)	9202 (100%)	23240 (0%)	12189 (100%)	34106 (66%)
1.25032e+20	3.51335e+18	2.21663e+19	1.93792e+25	1.30195e+17	2.29763e+22	4.95481e+21	1.63845e-41	4.51455e+14	6.35539e-30	8.94113e-07

Table 5.3.27 Comparison of EP and CAEP systems for F23 (Ellipsoid)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
33782 (0%)	31714 (0%)	31535 (0%)	22625 (0%)	34852 (0%)	24364 (0%)	27155 (0%)	4218 (100%)	24304 (0%)	5880 (100%)	18110 (100%)
1.69160e+02	1.83020e+02	6.33179e+01	1.36196e+01	9.20518e-01	9.97107e+02	3.91436e+04	9.77266e-56	6.35358e+03	2.13101e-42	6.52891e-17

Table 5.3.28 Comparison of EP and CAEP systems for F24 (Quadratic)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
31025 (0%)	28832 (0%)	28810 (0%)	19946 (0%)	32138 (0%)	23352 (0%)	24888 (0%)	6082 (100%)	22486 (0%)	7844 (100%)	17324 (100%)
9.15488e+00	3.88270e+00	1.12656e+01	1.77003e+00	4.82261e+00	2.18162e+01	6.34185e+09	8.43827e-28	9.32262e+06	1.33607e-23	4.58522e-09

Table 5.3.29 Comparison of EP and CAEP systems for F25 (Schwefel 2.22)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd$_2$)2 0
30641 (0%)	28538 (0%)	28516 (0%)	19648 (0%)	31852 (0%)	21534 (0%)	24386 (0%)	15298 (100%)	22304 (0%)	24009 (0%)	21966 (0%)
3.20302e+01	5.22375e+01	6.31121e+01	3.93317e+01	5.35708e+01	4.82763e+01	8.34362e+01	5.10177e-09	8.34362e+01	7.63826e+00	4.42255e+00

Table 5.3.30 Comparison of EP and CAEP systems for F26 (Schwefel 2.21)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd_2)2 0
793	2377	1026	560	416	1568	558	430	489	720	546
(100%)	(100%)	(100%)	(100%)	(100%)	(93%)	(100%)	(100%)	(100%)	(100%)	(100%)
2.28949e-14	2.56105e-10	4.95900e-15	5.87557e-13	7.88765e-34	9.55306e-06	1.54427e-10	7.30217e-11	6.73064e-11	6.97164e-11	1.32022e-10

Table 5.3.31 Comparison of EP and CAEP systems for F27 (Box)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd_2)2 0
512	1453	656	436	282	858 (93%)	11249	8792	10670	1010	6414
(100%)	(100%)	(100%)	(100%)	(100%)	-1.82521	(0%)	(33%)	(13%)	(100%)	(66%)
-1.86731 e+02	-1.86731 e+02	-1.86731 e+02	-1.86731 e+02	-1.86731 e+02	e+02	-1.86340 e+02	-1.86718 e+02	-1.86175 e+02	-1.86731 e+02	-1.86693 e+02

Table 5.3.32 Comparison of EP and CAEP systems for F28 (Schubert)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd_2)2 0
436	1098	809	470	248	110	2984	3766	556	224	1072
(100%)	(100%)	(100%)	(100%)	(100%)	(100%)	(80%)	(73%)	(100%)	(100%)	(93%)
3.97887e-01	3.97887e-01	3.97887e-01	3.97887e-01	3.97887e-01	3.97887e-01	3.97897e-01	3.97889e-01	3.97887e-01	3.97887e-01	3.97887e-01

Table 5.3.33 Comparison of EP and CAEP systems for F29 (Branin RCOS)

Population-only EP systems				Cultured EP (CAEP) Systems						
Schwefel-like	Fogel	Age	1/5 rule	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd_2)2 0
1979	2621	2086	758 (93%)	1878	2006	1008	238	376	735 (86%)	245
(60%)	(73%)	(66%)	-1.99193	(60%)	(47%)	(100%)	(100%)	(100%)	-1.98385	(100%)
-1.95156 e+00	-1.97686 e+00	-1.95963 e+00	e+00	-1.95156 e+00	-1.92734 e+00	-2.00000 e+00	-2.00000 e+00	-2.00000 e+00	e+00	-2.00000 e+00

Table 5.3.34 Comparison of EP and CAEP systems for F30 (Rastrigin2D)

CPU time, in milliseconds, taken to completion. Completion occurs if either the solution is found or the upper bound on the number of generations, given by the effort metric, has been reached. The solution is said to be reached when 6 significant digits of the optimum are found. Otherwise, the process continues until the maximum number of generations is reached. The second value in parentheses gives the percentage of the 15 runs that found the global minimum. The mean best solution value for the function at the maximum number of time step is given as the third value. The best system for each group is shown in bold characters in the table. The best performer was selected for each group based on the number of successful runs that found the solution. In case of a tie, the first tiebreaker is the mean best solution and the second tiebreaker is the average completion time.

5.2. *Comparing the population-only systems*

For each of the four population-only EP systems tested, the average completion time in milliseconds (millisec.), the % of successful runs for the 34 problems are given (success %), and the number of functions for which it was the best performer are given in table 5.4. In comparing the various population-only self-adaptive EPs, EP systems using higher-level self-adaptation mechanisms exhibit better general-purpose performance than those that support lower level self-adaptation. For example, the EP system with the most wins for the 34 problems used population level self-adaptation with the 1/5 success rule. The system outperformed the other 3 population-only systems 19 times out of 34 and achieved an average success ratio of 54.7% out of 510 runs. On the other hand, each of the other population-only systems that performed self-adaptation at a lower level outperformed their counterparts on at least three different problems. The results suggest that self-adaptation at all levels can be useful in the problem solving process depending upon properties of the fitness landscape for the problem to be solved.

Specifically, those systems exhibiting component level self-adaptation (Fogel, Schwefel-like) won on problems that had little or no basins or valleys but relatively high modality. Of these two, Schwefel-like systems did better on problems that were non-decomposable (6 out of 8 wins) while the Fogel approach was more likely to do best with decomposable problems (2 out of 3). Individual-level self-adaptation (age) was the winner for problems that were unimodal in nature (4 out of 4), non-decomposable (3 out of 4) and that tended to have large valleys and basins (3 out of 4). Thus, each of the systems that supported lower levels of self-adaptation tended to exist in rather complementary niches as defined by the structural properties of the functional landscapes to which they were applied. Population level self-adaptation (1/5 rule) found a different niche than the others. In 10 out of its 19 wins over the other population-only self-adaptive schemes, the problem dimensionality of was 5 or more. In fact, of the 15 problems with a dimensionally ≥ 5, it won 10 of them. Thus, with large dimensionality population-level approaches tend to be more successful in controlling the search as opposed to individual or component levels. It also won in two out of the three cases where the problem had a more than 50 local optima. What is interesting here is that each of these systems appears to be superior to other population-only methods in a distinctly different subset of problems. Component-levels of self-adaptation excelled for problems with high modality but without basins or valleys. Individual level approaches did well with basins and valleys for low modality problems. Population-level self-adaptation worked best on problems with high dimensionality and/or high modality. As such, systems with self-adaptation at the population level were more general purpose than the others. However,

	Population-only EP systems			
	Schwefel-like	Fogel	Age	1/5 rule
ms	14169	13989	13284	9689
success%	49.9%	47.9%	52.3%	54.7%
# wins	8	3	4	19

Table 5.4 Average millisecond and success ratio for all functions tested and total number of wins for all EP systems

	Cultured EP (CAEP) Systems						
	(Sd) 20	(Ssd) 20	(Ns) 20	(Nsd) 20	(Ns+Sd) 20	(Ns+Sd')2 0	(Ns+Sd_2)2 0
ms	14267	10705	12404	5870	10316	6943	8433
success%	53%	43.5%	45.6%	81%	56.8%	72.3%	76%
# wins	2	7	0	14	7	3	1

Table 5.5 Average millisecond and success ratio for all functions tested and total number of wins for all CAEP systems

there were always less complex problems for which lower-level self-adaptive approaches performed better.

5.3. *Comparing the CAEP approaches*

The summary results for the 7 CAEP systems are given in Table 5.5 in the same format as with population-only systems. The CAEP configurations that used knowledge to control only step size (Ns) or direction (Sd) did less well, in general, when compared with systems that coordinated both step size and direction. Of the CAEP systems, CAEP(Nsd) won 14 times, both CAEP(Ns+Sd) and CAEP(Ssd) won 7 times, CAEP(Ns+Sd') won 3 times, CAEP(Sd) won twice, and CAEP(Ns+Sd_2) won once.

Notice that in all of the winning systems except test case 27 (table 5.3.31) and test case 28 (table 5.3.32) both directional and step size modifications were made using the influence function. In addition, normative knowledge played a major role in the winning configurations. The systems that employed some normative knowledge and controlled both step size and direction, Nsd, Ns+Sd, Ns+Sd', Ns+Sd2, won for 25 out of the 34 test cases. Thus, normative knowledge appears to be the dominant knowledge source for directing function optimization here. On the other hand, systems that employed only situational knowledge, such as CAEP(Ssd), were successful primarily when that knowledge was used to control both step size and direction. However, as expected these systems did well for certain specific classes of functions but less well on others. The functions where CAEP(Ssd) performed well were F10, F11, F12, F13, F15, F19, and F29. These functions are low-dimensional ($n \leq 5$) and have an average effort metric value of 395. That means CAEP(Ssd) does well for relatively easy functions. However, CAEP(Ssd) had the lowest overall success ratio, 43.5%, for all 34 functions as shown in table 5.5. The types of functions that require this type of specialized knowledge will be discussed later.

5.4. *Comparing the population-only EP and CAEP performance*

In this section, the performance of the best systems from the population-only and cultured self-adaptive EP systems are compared. In table 5.6, the best EP and CAEP system for each problem is presented. The parameters used to make that selection, mean CPU time (Milliseconds), success ratio, and mean best solution are given as well. In 17 of the 34 problems, the best EP and CAEP system both achieved a 100% success ratio. However in each of these comparisons the best CAEP system won the overall comparison based upon the other tie breaking parameters, mean best solution, and CPU time taken. For these 17 problems the best CAEP system always had a mean best solution that was equal to, or better than, the EP-only solution. Likewise, the mean CPU time of the best CAEP system was always less than that for the best EP system for each of the 17 problems.
The best CAEP configuration outperformed the best population-only configuration of the remaining 17 problems. For these problems, the success ratio for the best CAEP system was better than that for the best EP-only system by 54% on the average. The average CPU time for each of the best CAEP systems was less than that for the corresponding EP-only system except for F16, the Griewank function. There, the best EP-only system took less average CPU than the best CAEP system. However, the success ratio for the best CAEP system was much higher, 46% as opposed to 13%. When the number of parameters for the Griewank function was increased to 30 in Figure F16', the success ratio for both systems increased. The ratio for the best EP-only system rose to 40%,

Fn No.	Best EP			Best CAEP			Best System	time ratio (*)
	millisec	success%	mean best	millisec	success%	mean best		
1	110016	100%	3.03608e-08	3020	100%	5.78728e-33	Nsd	30
2	2543	73%	7.26822e-06	808	100%	1.27236e-25	Ns+Sd	32
3	538	100%	-3.00000e+01	146	100%	-3.00000e+01	Ns+Sd	27
4	5982	100%	2.91640e-15	1703	100%	1.35578e-57	Nsd	28
4'	1168	100%	-1.32272e+01	360	100%	-2.02443e+01	Nsd	30
5	4515	26%	2.05698e+00	700	100%	9.98004e-01	Ns+Sd	16
6	502	100%	0.00000e+00	155	100%	0.00000e+00	Ns+Sd'	30
7	9073	0%	2.24908e+00	4006	60%	3.97984e-01	Nsd	44
8	18597	6%	6.51051e-04	9800	100%	3.38135e-12	Ns+Sd	53
9	1419	100%	-1.43982e-16	395	100%	-1.43982e-16	Ns+Sd'	27
9'	44434	13%	2.51271e+00	6953	100%	3.88243e-15	Nsd	16
10	542	100%	3.00000e+00	106	100%	3.00000e+00	Ssd	19
11	288	100%	-1.03163e+00	70	100%	-1.03163e+00	Ssd	24
12	760	100%	-1.00000e+00	154	100%	-1.00000e+00	Ssd	20
13	1289	100%	-6.00000e+00	300	100%	-6.00000e+00	Ssd	23
14	40274	0%	6.23111e-03	40086	6%	2.42833e-03	$Ns+Sd_2$	99
15	632	100%	-1.17499e+02	124	100%	-1.17499e+02	Ssd	19
15'	20139	0%	-1.04681e+03	14416	40%	-1.12127e+03	Nsd	72
16	3532	13%	1.73105e-01	4947	46%	4.29343e-03	Nsd	140
16'	40156	40%	7.71831e-03	15091	80%	1.97097e-03	Nsd	38
17	10528	93%	3.07495e-04	1664	100%	3.07486e-04	Ns+Sd	16
18	378	100%	1.78872e-39	142	100%	2.39657e-104	Ns+Sd'	38
19	391	100%	-3.79124e-03	98	100%	-3.79124e-03	Ssd	25
20	8656	20%	-4.44493e+00	6242	53%	-4.65582e+00	Ns+Sd	72
21	403	100%	0.00000e+00	185	100%	0.00000e+00	Ns+Sd	46
22	5096	73%	-1.35486e+00	1666	86%	-1.41373e+00	Nsd	33
23	30630	0%	3.51335e+18	9202	100%	1.63845e-41	Nsd	30
24	22625	0%	1.36196e+01	4218	100%	9.77266e-56	Nsd	19
25	19946	0%	1.77003e+00	6082	100%	8.43827e-28	Nsd	30
26	19534	0%	1.86504e+00	15298	100%	5.10177e-09	Nsd	78
27	1026	100%	4.95900e-15	416	100%	7.88765e-34	Sd	40
28	436	100%	-1.86731e+02	282	100%	-1.86731e+02	Sd	64
29	436	100%	3.97887e-01	110	100%	3.97887e-01	Ssd	25
30	758	93%	-1.99193e+00	238	100%	-2.00000e+00	Nsd	31
avg.	12566	63%	-	4388	90%	-	-	39%

(*) (time used by best CAEP / time used by best EP)*100

Table 5.6 Comparison of the "Best EP" and "Best CAEP" systems

which that for the best CAEP system rose to 80%. Insight into this phenomenon can be found in Whitley [1995, 1996]. There, he suggests that the addition of variables produces a smoother landscape for this class of functions which is more conducive to finding the optimum solution.

These 17 problems, in general, possessed higher average dimensionality, frequent valleys, and a higher average effort metric. Seven out of ten problems with valleys were in this group. The average dimensionality of problems in this was 13 as opposed to 9 for the other 17. The average effort for the problems in this group was 1,777 as opposed to 518 for the other problems. The increased effort value reflects the combined impact of the increase in dimensionality and valley size for these problems.

In summary, systems with belief space self-adaptation outperformed lower levels of self-adaptation on the more complex problems, problems characterized by high dimensionality and the presence of valleys. Problems of lower effort can be solved with satisfactory accuracy by the best population-only systems but these systems always took more CPU time and produced less precise results than the best CAEP system. In fact, it has often been conjectured that cultural evolution proceeds at a faster rate than biological evolution for this reason [Reynolds 1992].

5.5. *Belief space adjustment example*

In the previous section, it was observed that the best CAEP system outperformed the best population-only EP system for problems with high dimensionality. In this section, we observe how self-adaptation works at the belief space level for an example problem with high dimensionality, the Griewank function with 30 parameters (F16') mentioned above.

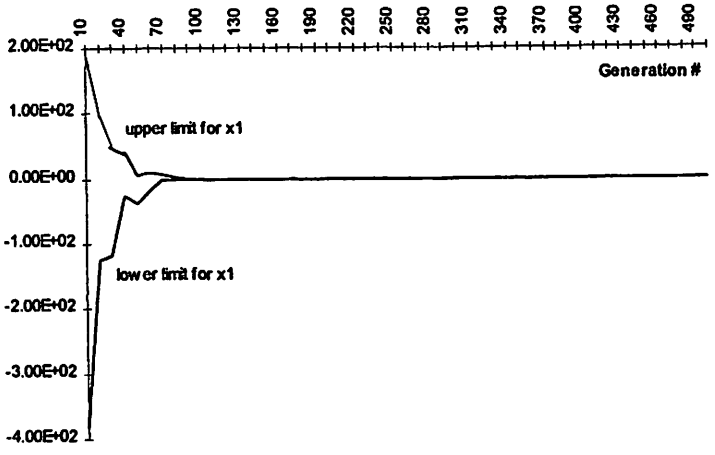

Figure 5.1 Belief space trace by CAEP(Nsd) for f16', x1

Figure 5.1 plots the upper and lower bounds of the belief space range for one of those parameters, x_1 for the best CAEP(Nsd) system as it proceeded to find the optimum. In this system, normative knowledge is used to influence both the step size and direction of mutation. In figure 5.1 the bounds converge quickly. Although it does not occur here, it is possible that the system can open up the bounds later on and reinvestigate the solution if necessary.

Figure 5.2 gives the overall performance of the CAEP(Nsd) system as the belief space range is narrowed for the parameter x_1. The belief space range for x_1 converges around generation 100, whereas the overall solution continues to improve until around 200. This is because the system is still exploiting the other variable ranges that have not yet converged in order to guide the solution. The best population-only EP system took 1000 generations to achieve the solution to 2 significant figures.

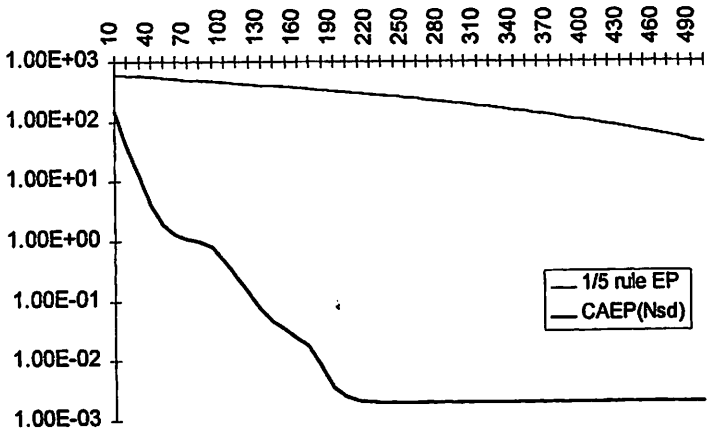

Figure 5.2 Mean best for f16' (with 30 parameters)

6. Heuristics for Configuring CAEP System

Based upon the results of Section 5, it is clear that the addition of a cultural component that exploits the generalized experience of population members produces a number of performance improvements over population-only EP systems for all 34 problems. In all cases the best CAEP self-adaptive system produced a more precise solution in less time that the best population-only system. For problems characterized by high effort and dimensionality the best CAEP system was more accurate as well. However, the CAEP systems investigated were only a subset of the CAEP configurations given in table 5.2. They came from just one row, those systems with a 20% acceptance function.

In this section, our problem solving goal is to find the simplest configuration in the entire table that achieves a given success ratio. The ratio used here will be 100%. We then construct a search heuristic to find the CAEO configuration of minimum complexity that produces the maximum success ratio for a given function; this will allow us to determine the nature of the knowledge needed to efficiently solve a given type of function with a high degree of success.

6.1. *Constructing the configuration search space from the table*

In this section we construct the search space for the configuration problem using table 5.2. The table is organized such that the simplest search configuration is located in the upper left cell (10% acceptance function, Sd influence function). This cell accepts the fewest individuals from the population, and exhibits the simplest influence or control function on the population. As one moves downward row by row, the maximum number

of individuals accepted into the belief space gets progressively larger. Likewise, the column order reflects the relative control complexity of the different influence function based upon a set of ordering rules:

(1) An influence function that uses situational knowledge is less complex than one that uses only normative knowledge. For example, Ns is more complex than Sd and Ssd.
(2) An influence function that controls step size and direction is more complex than one that controls just one of them. For example, Nsd is more complex than Ns.
(3) An influence function that supports heterogeneous knowledge, both situational and normative, is more complex than a system that supports only homogeneous knowledge. For example, Ns+Sd is to the right of Nsd.
(4) The more cases (situational knowledge) an influence function uses, the more complex it is. For example, Ns+Sd2 is more complex and to the right of Ns+Sd.

A portion of the configuration space to be searched is given in figure 6.1. The tree is based upon table 5.2 where the root node is the upper left-hand corner cell. Here the first column of the table 5.2 is excluded since it corresponds to Sd, a control function that can be easily performed with a population-only system. The search proceeds using a best-first [Winston, 1992] search from the root which is Ssd10. The first component is the influence function and the second, 10%, is the acceptance function. The goal is to find the first node that satisfies the given search criteria, Ω. Here, that criteria is expressed in terms of the % of ψ runs that achieved a solution.

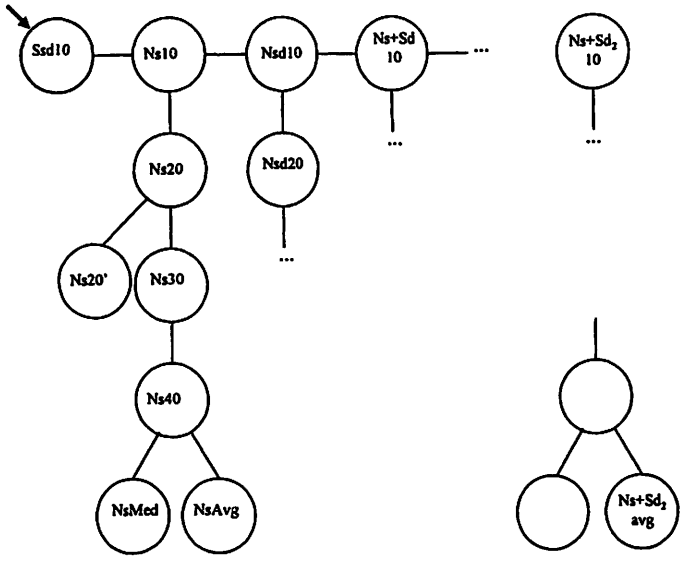

Figure 6.1 CAEP systems organized as a search tree. Each system is described first in terms of its influence and acceptance functions. For instance Ssd10 means CAEP(Ssd) using top 10% acceptance function.

6.2. *Analysis of the expanded search results*

Table 6.1 represents the results of applying the above algorithm to search for the best system in each of the 34 test functions. In this case success ratio or goal, Ω, is set to 100% and the number of runs, ψ, is set to 15. For each of the 34 problems the number of CAEP configuration nodes expanded is given along with the success percentage for the

Function No. (Function Name)	# of CAEP systems tested	Success % of Best	The Best CAEP System	
			Influence function	Acceptance function
F1(Sphere, 30 parameters)	6	100	Nsd	top20%
F2(Rosenbrock)	2	100	Ns	top10%
F3(Step)	2	100	Ns	top10%
F4(DeJongF4 w/o noise)	6	100	Nsd	top20%
F4'(DeJongF4 with noise)	4	100	Nsd	top10%
F5(Foxholes)	2	100	Ns	top10%
F6(Bohachevsky)	2	100	Ns	top10%
F7(Rastrigin)	36	93	Nsd	top40%
F8(Colville)	8	100	Ns+Sd	top20%
F9(Ackley)	2	100	Ns	top10%
F9'(Ackley, 30 parameters)	6	100	Nsd	top20%
F10(Goldstein-Price)	1	100	Ssd	top10%
F10(6 hump Camel Back)	1	100	Ssd	top10%
F10(Easom)	1	100	Ssd	top10%
F10(Floudas)	1	100	Ssd	top10%
F14(Watson)	36	13	Ns+Sd	top10%
F15(Koon's F3)	1	100	Ssd	top10%
F15(Koon's F3, 30 parameters)	10	100	Nsd	above median
F16(Griewank, 3 parameters)	36	46	Nsd	top20%
F16'(Griewank, 30 parameters)	36	93	Nsd	top40%
F17(Kowalik)	2	100	Ns	top10%
F18(Matyas)	1	100	Ssd	top10%
F19(Zettl)	1	100	Ssd	top10%
F20(Michalewicz)	36	53	Ns+Sd	top20%
F21(Beale)	1	100	Ssd	top10%
F22(Langerman)	36	86	Nsd	top20%
F23(Ellipsoid)	6	100	Nsd	top20%
F24(Quadratic)	6	100	Nsd	top20%
F25(Schwefel 2.22)	6	100	Nsd	top20%
F26(Schwefel 2.21)	6	100	Nsd	top20%
F27(Box)	2	100	Ns	top10%
F28(Schubert)	8	100	Ns+Sd'	top10%
F29(Branin RCOS)	1	100	Ssd	top10%
F30(Rastrigin2D)	2	100	Ns	top10%
Average	9.21	94%	-	-

Table 6.1 Number of Systems Tested using the search heuristics

best solution found. If the success ratio is less than the goal, 100%, then the entire space is searched, 36 nodes are expanded in each case. For each expanded node, ψ=15 runs were performed to produce the observed success ratio. The influence and acceptance function for the best configuration found are given in the last two columns of the table.

By expanding the search for best CAEP to the entire table the average success ratio for the best system increased from 90%, for the 20% row, to 94% for all 34 problems. The average number of nodes expanded to find the best solution was 9.21. Comparing these results with the best configuration for the 20% acceptance function given in table 5.6 produces some interesting observations. First, in both searches the majority of the best configurations were homogeneous in nature. That is, only normative or situational knowledge was used to produce changes in the step size and direction. For the table-wide search, the best configuration for 22 of the 34 problems used homogeneous knowledge to control both step size and direction. Thirteen of those systems were Nsd and nine were Ssd, and eight were Ns. The heterogeneous systems that used both normative and situational knowledge won only 4 times in the whole table search as opposed to eleven times in the single row search.

The problems for which the mixed mode approaches performed the best in the table-wide search were F8, F14, F20, and F28. These problems are characterized as decomposable with moderate dimensionality (2-6 parameters), and either multi-modal or with both large basins and valleys. Two of these problems, F14 and F20, were among the 3 hardest systems for the tested CAEP systems to solve with success ratios of 13% and 53% respectively. The suggestion here is that for these classes of problems heterogeneous knowledge is essential for effective solution. However, the low success ratios are achieved for two of them indicate that there is still room for CAEP systems to use their knowledge more effectively in order to improve the success ratio. This will be the subject of future research.

In terms of acceptance functions the winning configurations in table 5.6 all had a 20% acceptance function. For the table wide search, the best system tended to have less than 20% or more. Twenty of the configurations had acceptance functions that used the top 10% while only 10 used 20%. Three systems used the top 40% or more. In general, these were problems with high dimensionality, with an average dimensionality of 22. All three of these best solutions to these problems had Nsd influence functions.

The best table-wide systems with Ssd influence functions (9) employed a 10% acceptance function. On the other hand, the best homogeneous table-wide normative system, Nsd, tended to use acceptance functions of 20% or higher. Only one of the thirteen Nsd systems used an acceptance function of less than 20%. What this suggests is that when situational knowledge is used, fewer contributors are required than in a system controlled by normative knowledge. This makes intuitive sense because in order for norms to be effective they must be representative. Situational knowledge on the other hand benefits from the fact that a small number of unique contributors are used as opposed to a large number of more similar ones.

6.3. *How the number of individuals accepted affects influence function performance*

Results from the previous section suggest that the performance of a given influence function is affected by the number of individuals used to update the belief space. In order to investigate this further, the average success ratio for each of the seven influence functions over the 510 runs for the 34 problems is plotted against the percentage of

individuals used to update the belief space in figure 6.2. Two of the influence functions, Sd and Ssd do not exhibit any change as the number of individuals used to influence the belief space is increased. This is because they both use only the elite, or best, individual to influence mutation. The remaining individuals have no influence, no matter how many are acceptable by the influence function.

On the other hand, the influence function that uses normative knowledge to determine both step size and direction, Nsd, outperforms all other functions when 20% or more individuals are used. When only 10% of the individuals are used Ns + Sd' is the best performer. This suggests that when the number of contributing individuals is small, it is better to use the current best individual seen so far to determine mutation direction rather than normative knowledge.

What is interesting is that the performance of Nsd is relatively unaffected by the participation of more than 20% of the population until approximately 50% of the population is used. At that point, the performance of Nsd begins to degrade markedly. From a social perspective this suggests that in societies where individuals are able to influence the norms or laws, the term "the majority rules" allows an optimal number of individuals to participate in decision-making without major degradation in performance. In other words, it may be that the democratic notion of majority rules has its functional roots in terms of optimal problem solving and decision-making.

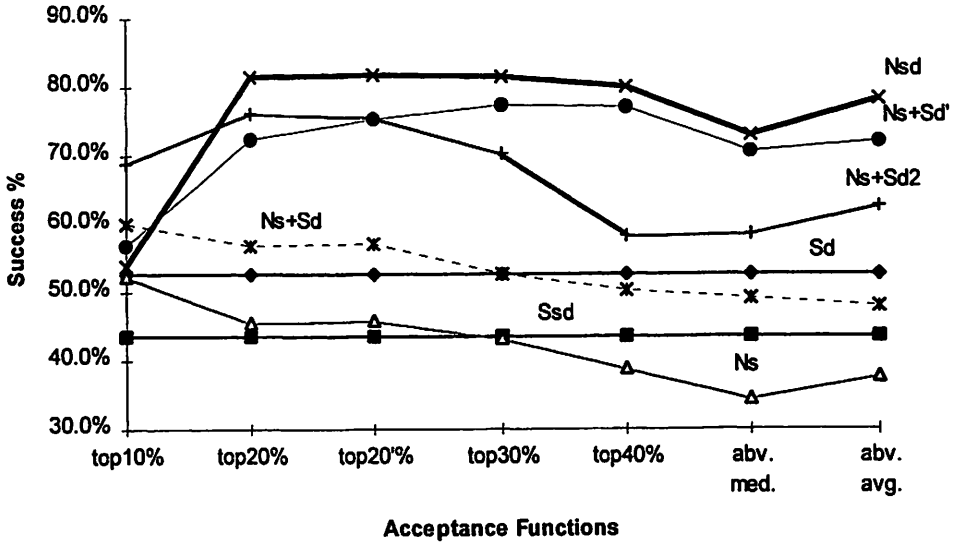

Figure 6.2 Overall Influence Function Performance relative to the number of individuals Used

6.4. *The feasibility of a reconfigurable knowledge-based function optimizer*

If a system is given a set of different optimization problems to solve there are two basic approaches to producing a solution for the entire set. First, one can produce a general purpose system that will be able to solve all the problems within its general framework. Secondly, one can design a system that is able to reconfigure itself in order to solve each specific problem as it comes. Once a particular configuration has been discovered that configuration can be stored and reused when the problem reappears. This latter approach

will be cost effective if simple systems with satisfactory performance can be easily found for problems that are likely to reappear.

For the set of problems investigated here, it appears that a reconfigurable knowledge-based function optimizer would be a feasible solution. For twenty-five of the 34 functions, at most 6 of the 36 configurations needed to be expanded before a successful configuration was produced. The entire table was searched completely in only 5 cases. The average success ratio for the 34 solutions found was 94%. The nature of the systems produced by the heuristic are summarized below:

(1) Homogeneous influence functions were selected more frequently than non-homogeneous ones.
(2) The homogeneous influence functions that were seleted were used situational knowledge used fewer individuals to update the belief space than influence functions that used normative knowledge.
(3) Heterogeneous influence functions were selected for problems that have either a very rough fitness landscape (multimodal) or a very smooth one (wide valleys and/or large basins).
(4) The performance degradation of the most successful normative system, Nsd, at around 50% suggest that the notion, "the majority rules", may reflect a balance between optimal problem solving capability and individual participation. That is, as many people are involved as possible without severely affecting decision-making performance.

7. Influence of Problem Structure on the Knowledge Used to Guide CAEP Search

In section 6, we observed that when designing a special purpose optimizer for a given problem certain configuration tended to be associated with specific landscape features. In this section we compare their performance relative to collections of these 34 functions that exhibit a common landscape feature. A minimum of 30 runs for each CAEP configuration on a given landscape feature was performed. The goal here is to see the extent to which certain types of social knowledge (influence function) affect the performance of a cultural system. The acceptance function is set to 20% as in section 5.

7.1. *CAEP performance as a function of problem dimensionality*

Figure 7.1 plots the success ratio for each of the seven influence functions. For the low dimensional functions, all of the influence functions do relatively well with average success ratios between 58% and 80%. The three non-homogeneous functions that control both step size and direction did better as a group than those which used only one knowledge type.

For high dimensional problems, only three systems exhibited acceptable performance. In fact, the performance for Nsd actually improved to over 90% for these problems. The performance for the two heterogeneous systems that used additional information to control mutation direction (Ns + Sd' and Ns + Sd2) was only slightly reduced for high dimensional problems. However, the heterogeneous system, Ns + Sd, that did not use either additional exemplar knowledge or the distance from the current exemplar performed poorly. This indicates that additional situational knowledge may be needed for the solution of high dimensional problems then for ones of lower dimensionality.

7.2. *The impact of problem modality on the type of social knowledge needed to influence search*

Figure 7.2 gives the relative performance for the 7 influence functions relative to problems with high (number of local minima ≥ 2) and low modality (number of local minima less than two). It is clear that the average performance of the 7 influence functions was less sensitive to changes in modality than to changes in dimensionality. As with dimensionality, the three heterogeneous influence functions outperformed the homogeneous influence functions, on the average, for both high and low modality functions. However, even though one homogeneous function, Nsd, was the best performer overall. In fact, Nsd exhibited an increase in performance with an increase in modality, just as it did for increased dimensionality.

7.3. *The impact of function decomposability on the knowledge used to guide CAEP search*

Figure 7.4 compares the performance of CAEP systems using the top20% acceptance function on decomposable (or partially-decomposable) problems and non-decomposable problems as described in Section 4. The results clearly show that decomposable (or partially-decomposable) problems are easier to solve for all CAEP systems than non-decomposable ones. However, as expected, systems which control both direction and step size performed better than systems controlling only one of them. Problem decomposability appears to impact performance more than modality but less than dimensionality. Again, the heterogeneous systems outperformed the homogeneous systems on the average in both categories of comparison. In addition, the best system for non-decomposable systems was heterogeneous in nature. This stands in contrast to dimensionality and modality, where the best performer was homogeneous normative.

7.4. *The impact of basin in a function on the knowledge used to control search*

Figure 7.4 compares the performance of CAEP systems using the top20% acceptance function between functions with basins and functions without basins. While a homogeneous system, Nsd, was the best performer on functions without basins, the top 4 performers on basin functions all used situational knowledge. It is interesting to note that every CAEP system employing situational knowledge, such as CAEP(Ssd), CAEP(Sd), CAEP(Ns+Sd), CAEP(Ns+Sd'), and CAEP(Ns+Sd2) performed better on basin functions than on non-basin functions. Cultural systems that did not use situational knowledge, such as CAEP(Nsd), performed less well on functions with basins. It is suggested that for basins, situational knowledge can provide more useful information than normative knowledge. The tendency to imitate the best performers so far appears to be a good choice when the search space is smooth.

7.5. *The impact of valleys on the knowledge required to control search*

Figure 7.5 compares the performance of CAEP systems using the top20% acceptance function between functions with and without valleys. Again Nsd was the top performer on functions without valleys. But as with basins, its success ratio dropped markedly for valley functions (about 20%). The 3 heterogeneous functions were the top performers for valleys as they were for basins with Ns+Sd performing the best (85%), out of all the tested functions. More generally, every CAEP system employing situational knowledge,

such as CAEP(Sd), CAEP(Ns+Sd), CAEP(Ns+Sd'), and CAEP(Ns+Sd2) performed better on valley functions than systems using only normative knowledge, such as CAEP(Ns), CAEP(Nsd). This is consistent with our observations about basin functions as well.

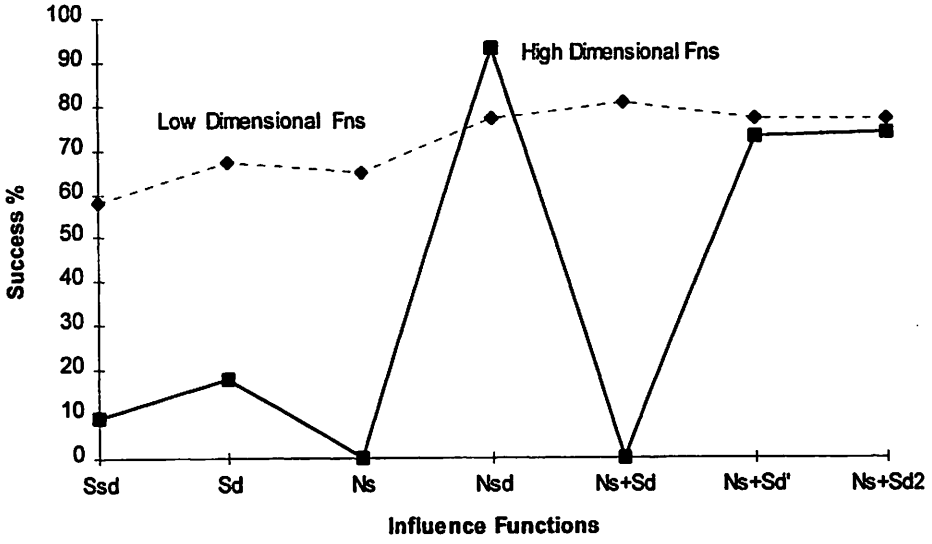

Figure 7.1 Mean performances between low-dimensional (n is less than 7) and high-dimensional (n is greater than or equal to 30) problems. Systems are based on top20% acceptance function.

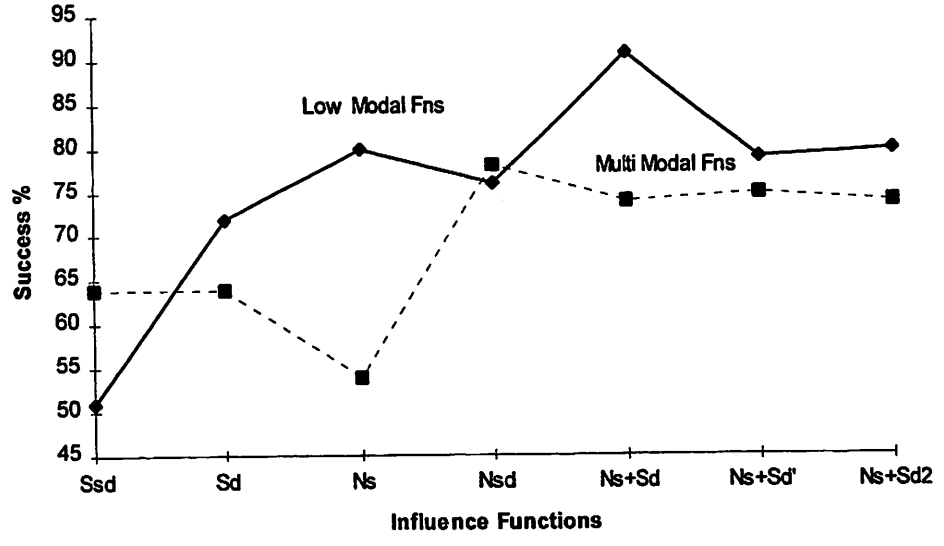

Figure 7.2 Mean performances between low-modal and high-modal (number of local minima is greater than 2) problems (based on top20% acceptance function)

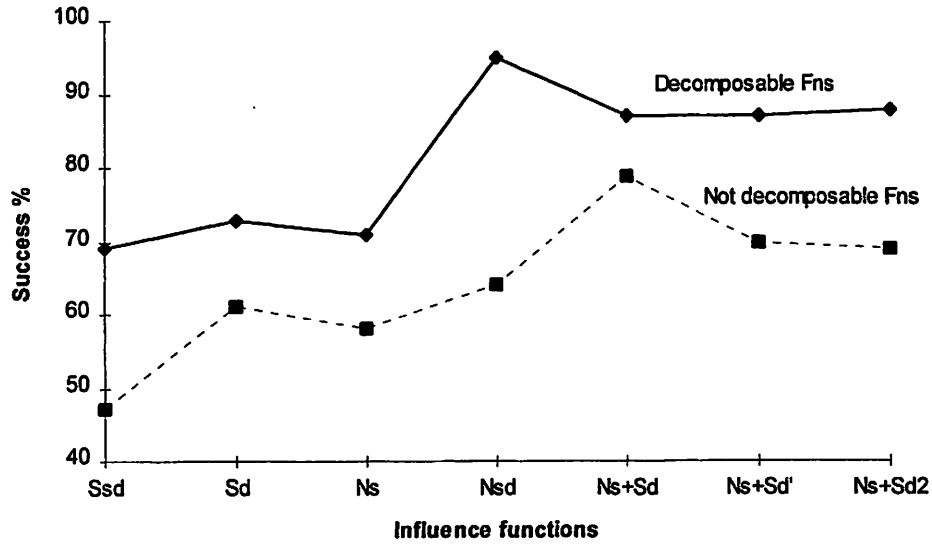

Figure 7.3 Mean performances between (partially) decomposable and not-decomposable problems (based on top20% acceptance function)

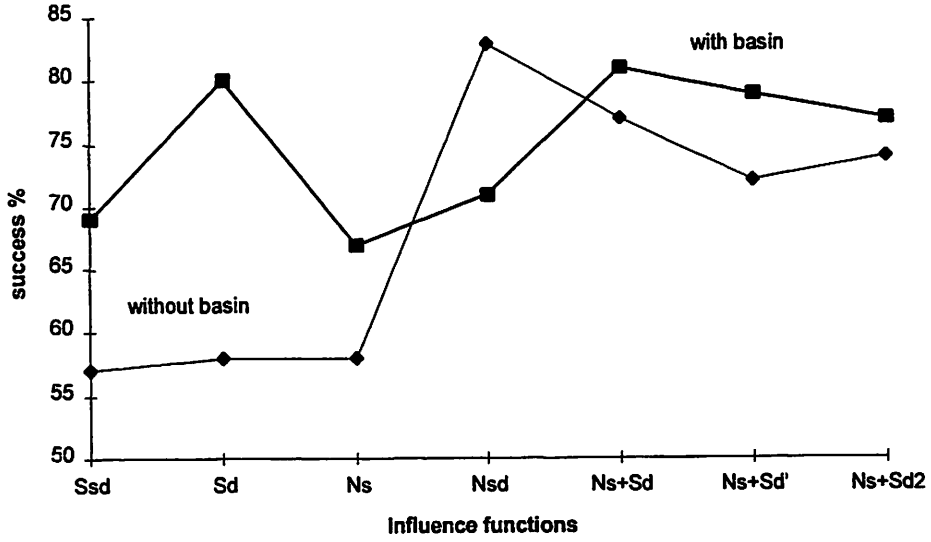

Figure 7.4 Mean performances between functions with and without basin (based on top20% acceptance function)

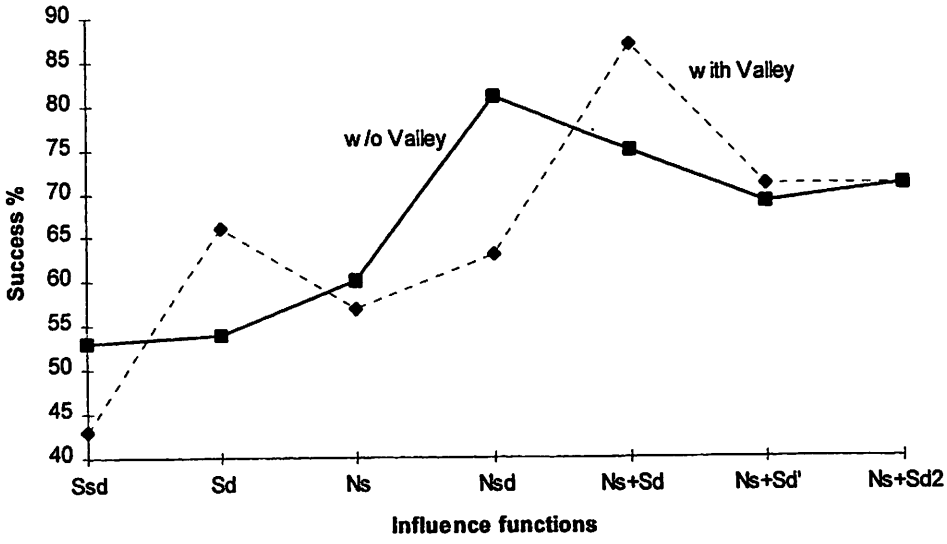

Figure 7.5 Mean performances between functions with valley and functions without valley (based on top20% acceptance function)

7.6. *Summary of the influence of function structure on system performance*

In section 5 we observed that different levels of self-adaptation at the population level favored different types of landscape features. The same is true for self-adaptation in the belief space. Table 7.1 summarizes the results of this section. There we see that pure normative knowledge, Nsd, does well with high dimensional problems and problems of high modality. These are problems where the system benefits from the generalizations produced by the normalization process. It also does well with problems that are decomposable. However, problems that contain basins and/or valleys are a different story. In these situations the problem is not with having too much information, as it is with having too little. In these circumstances, situational knowledge is useful in guiding the direction of mutation towards those individuals that are the best performers. In addition, situational knowledge is useful when the problem is not decomposable, as well. The lack of decomposability appears to make situational knowledge more important in directing the mutation process in CAEP.

In the literature it has been shown that for population only models there is a need to keep the elite or best individual around in the population. The experiments performed here suggest why; that for certain types of problem the elite individual will provide useful and necessary information with which to guide the population. Our results also suggest that it may not always be necessary to keep the elite individual in population only model for all problems.

	Dimension		Modality		Decompos.		Basin		Valley	
	Low	High	Low	Multi	Yes	No	Yes	No	Yes	No
Ssd	-	—	-	+	+	—	+	-	—	-
Sd	+	—	++	+	++	+	++	-	+	-
Ns	+	—	++	-	+	-	+	-	-	-
Nsd	++	+++	++	++	+++	+	+	+++	+	++
Ns+Sd	++	—	+++	++	+++	++	++	++	+++	++
Ns+Sd'	++	+	++	++	+++	+	++	+	+	+
Ns+Sd2	++	+	++	++	+++	+	++	++	+	++

+++: over 80% success ratio
++: 70-80% success ratio
+: 60-70% success ratio
-: 50-60% success ratio
—: 40-50% success ratio
—: below 40% success ratio

Table 7.1 Relationships between influence functions and function structural parameters

8. Conclusion and Future Directions

8.1. *Summary of experimental results*

Various population-only EP systems and CAEP systems using different influence functions and acceptance functions were tested on a variety of unconstrained optimization problems. Some of the conclusions that can be drawn from these experiments are summarized below.

(1) Self-adaptive population-only EP outperformed the basic EP system without self-adaptation across the board.

(2) Higher levels of self-adaptation in population-only EP systems produce better overall performance than lower-level self-adaptation. For example, the population-only population level adaptive EP system using 1/5 success rule was the best of the population-only systems over all the problems tested.

(3) However, our experiments suggest that self-adaptation at all levels (component, individual, and population) can be useful depending on the properties of the fitness landscape for the problem to be solved.

(4) By adding belief space knowledge to an EP system, CAEP systems produced marked improvements over population-only EP for all test functions in terms of system success ratio, mean best solution, and execution time in milliseconds (except for time in F16).

(5) CAEP systems that used knowledge to control only step size or direction did less well, in general, when compared with CAEP systems that coordinated both step size and direction.

(6) The most successful general purpose systems, such as CAEP(Nsd), used only normative knowledge to control both step size and direction. This system did best with decomposable problems with high dimensionality and/or high modality.

(7) For any given function tested for this paper, a relatively simple CAEP configuration can be produced that generates a 100% success ratio, using a simple best first search heuristic. This suggest the potential for the development of a reconfigurable knowledge-based function optimizer that is able to reconfigure itself to efficiently solve a given problem.

(8) Heterogeneous systems that use both normative and situational knowledge to control step size and direction respectively are best at providing efficient solutions for problems that contain basins or valleys, or that are non-decomposable. However, they have problems with high dimensional functions.

(9) Of all the landscape features examined here, problem dimensionality appeared to be the most critical.

(10) The effort estimation metric appeared to be a useful tool in the estimation of the number of generations required to solve a given problem for the set of problems used here. Only 7 problems were not able to be solved 100% of the time within the predicted constraints. This allows a more effective allocation of computational resources to a problem than otherwise might be the case. We expect that examinations of the structure of the seven problems will allow us to modify the weights of the cost drivers to produce an even better predictor.

8.2. *Implications for real world systems*

The results of the experiments provided some insight into why cultural systems are structured as they are. For example, the performance of Nsd begins to deteriorate when more than 20% of the population contributes to updating the belief space. However, it is not until around 50% when the performance begins to markedly degrade. Thus, 50% can be viewed as a compromise between performance and participation. That is, it may be the

point at which the best performance is produced using the most individuals. Since norms can be viewed as laws or standards for behavior it is perhaps no wonder that "the majority rules" is a way of reflecting the importance of this 50% guidelines.

Another issue pertains to the fact that larger-scale cultural systems need to be able to (1) allocate resources to solve specific problems (2) and to support an efficient solution to a given problem within a more general framework. We have attempted to provide some simple mechanisms for carrying out these activities in terms of our cultural algorithms model.

Metrics such as the effort metric proposed here can be developed by a system to predict the resources it should expect to use in order to solve a given problem. This is particularly important when a system must deal with a limited supply of resources. This allocation of resources should be predicted on problem structure as ours is. Likewise, the system needs to use problem structure information to select a problem solver that is computable with the available resources. The results of the experiments performed here suggest that this can be done within the context of our Cultural Algorithm model. Future research will investigate how such features can emerge within a cultural system.

BIBLIOGRAPHY

[Angeline 1995] Peter A. Angeline, "Adaptive and Self-Adaptive Evolutionary Computation," in *Computation Intelligence*, Eds. Marimuthu Palaniswami, et. al., IEEE Press, New York, 1995, pp. 152-163.

[Bäck 1993] Thomas Bäck and Hans-Paul Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, 1(1), pp. 1-24.

[Bäck 1994] Thomas Bäck, "Selective pressure in evolutionary algorithms: A characterization of selection mechanisms," *in Proceedings of the First IEEE conference* on *Evolutionary Computation*, pp. 57-62, IEEE Press, Piscataway, NJ, 1994

[Bäck 1996] Thomas Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.

[Bäck 1997] Thomas Bäck, U. Hammel, and Hans-Paul Schwefel, "Evolutionary Computation: History and Current State," *IEEE transactions* on *Evolutionary Computation*, 1(1), pp. 3-17, 1997

[Bohachevsky 1986] I. O. Bohachevsky, M.E. Johnson and M. L. Stein, "Generalized Simulated Annealing for Function Optimization," *Technometrics* 28(3), pp. 209-217

[Boyd 1985] R. Boyd and P. J. Richerson, *Culture and the Evolutionary Process*, University of Chicago Press, 1985

[Cavali-Sforza 1981] L. L. Cavali-Sforza and M. W. Feldman, *Cultural Transmission and Evolution: a Quantitative Approach.* Princeton University Press, 1981

[Chung 1996] Chan-Jin Chung and Robert G. Reynolds, "A Testbed for Solving Optimization Problems Using Cultural Algorithms," in *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, 1996

[Chung 1996b] ChanJin Chung and Robert G. Reynolds, "The Use of Cultural Algorithms to Support Self-Adaptation in Evolutionary Programming," *Proc. of 1996 Adaptive Distributed Parallel Computing Symposium*, Dayton, Ohio, Aug. 8-9, 1996, pp. 260-271.

[Chung 1996c] ChanJin Chung and Robert G. Reynolds, "Function Optimization Using Evolutionary Programming with Self-Adaptive Cultural Algorithms," *Proc. of the first Asian-Pacific Conference on Simulated Evolution and Learning*, Taejon, Korea, Nov. 8-12, 1996, pp. 21-28. (to appear in Lecture Notes in Artificial Intelligence, Springer-Verlag, 1997)

[Conte 1986] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, Software Engineering Metrics and Models, Benjamin Cummings Publishing Co., Menlo Park, CA., 1986, pp. 281-282.

[Darwin 1859] C. Darwin, *On the Origin of Species*, John Murray, London, 1859; A facsimile of the first edition with an Introduction by Mayr, Harvard University Press, 1964

[De Jong 1975] K. A. De Jong, *The analysis of the behavior of a class of genetic adaptive systems*, Doctoral dissertation, Univ. of Michigan, Ann Arbor

[Durham 1994] W. Durham, *Co-Evolution: Genes, Culture, and Human Diversity*, Stanford University Press, Stanford, CA

[Fogel 1962] L. J. Fogel, "Autonomous Automata," Ind. Res., Vol. 4, pp 14-19, 1962

[Fogel 1991] David B. Fogel, L. J. Fogel, and J. W. Atmar, "Meta-Evolutionary Programming," In *Proc. of the 25th Asilomar Conf. on Signals, Systems, and Computers*, edited by R.R. Chen. Pacific Grove, CA: IEEE Comp. Soc. Press, pp. 540-545

[Fogel 1994] L. J. Fogel, D. B. Fogel, and P. J. Angeline, "A Preliminary Investigation on Extending EP to Include Self-adaptation on Finite State Machines," *Informatica*, 18(4), pp. 387-398, 1994

[Fogel 1995a] David B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, NJ, 1995

[Fogel 1995b] David B. Fogel, "Phenotypes, Genotypes, and Operators in Evolutionary Computation," in *Proceedings of 1995 IEEE International Conference on Evolutionary Computation (ICEC'95)*, Volume 1, pp. 193-198.

[Fogel 1996] David B. Fogel, "Using Fitness Distributions to Design More Efficient Evolutionary Computations," in *Proceedings of 1996 IEEE International Conference on Evolutionary*

Computation (ICEC'96), pp. 11-19.

[Goldberg 1989] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, 1989

[Gehlhaar 1996] Daniel K. Gehlhaar and David B. Fogel, "Tuning EP for Conformationally Flexible Molecular Docking," in *Proceedings of the 5th Annual Conference on Evolutionary Programming*

[Grefenstette 1987] J. J. Grefenstette, "Incorporating problem specific knowledge into genetic algorithms," in *Genetic Algorithms and Simulated Annealing*, L. Davis ed., Los Altos, CA: Morgan Kaufmann, pp. 42-60, 1987.

[Hadley 1962] G. Hadley, *Linear Programming*, Addison-Wesley, Reading MA.

[Hinterding 1997] R. Hinterding, Z. Michalewicz, and A. E. Eiben, "Adaptation in Evolutionary Computation: A Survey," in Proceedings of the Fourth IEEE Conference on Evolutionary Computation, Indianapolis, IN, pages 65-69, IEEE Press Piscataway, NJ, 1997

[Hoffmeister 1990] F. Hoffmeister and T. Bäck, "Genetic Algorithms and Evolution Strategies: Similarities and Differences," in *Parallel Problem Solving from Nature 1*, H.P. Schwefel and R. Männer (eds.) (Springer-Verlag), pp. 455-469.

[Hoffmeister 1992] F. Hoffmeister, and T. Bäck, "Genetic Algorithms and Evolution Strategies: Similarities and Differences," Technical Report No. SYS-1/92, Systems Analysis Research Group, University of Dortmund, Dept. of Computer Science, Dortmund, Germany.

[Holland 1975;1992] J. H. Holland, *Adaptation in Natural and Artificial Systems*, First edition, Ann Arbor: The University of Michigan Press, 1975; First MIT Press edition, 1992

[Kauffman 1993] Stuart A. Kauffman, The Origins of Order, Oxford University Press, New York, 1993

[Kim 1995] J.-H Kim, J.Y. Jeon, H.K. Chae, and K.I. Koh, "A Novel Evolutionary Algorithm with Fast Convergence," in *Proceedings of 1995 IEEE International Conference on Evolutionary Computation (ICEC'95)* , 1995, pp. 819-824

[Koon 1995] G. K. Koon, and A. V. Sebald, "Some Interesting Test Functions for Evaluating Evolutionary Programming Strategies," in *Proceedings of the 4th Annual Conference on Evolutionary Programming*, 1995, pp.269-280

[Michalewicz 1994] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, (Second, Extended Edition,) Springer-Verlag, Berlin, 1994 (1st edition, 1992)

[Michalewicz 1994b] Zbigniew Michalewicz, Evolutionary Computation: An Overview, available at ftp://ftp.uncc.edu/coe/evol/, 1994

[Minkoff 1984] Eli C. Minkoff, *Evolutionary Biology*, Addison-Wesley Pub. Co.

[Pressman 1997] R. A. Pressman, Software Engineering: A Practitioner's Approach, McGraw-Hill Press, New York, 1997

[Reynolds 1978] Robert G. Reynolds, "On Modeling the Evolution of Hunter-Gatherer Decision-Making Systems," *Geographical Analysis*, Vol. 10, no. 1 (January 1978), pp. 31-46.

[Reynolds 1979] Robert G. Reynolds, *An adaptive computer model of the evolution of agriculture for hunter-gatherers in the valley of Oaxaca, Mexico*, Doctoral dissertation, Univ. of Michigan, Ann Arbor, 1979

[Reynolds 1986] Robert G. Reynolds, "An adaptive computer model of plan collection and early agriculture in the eastern valley of Oaxaca," in *Guila Naquitz: Archaic Foraging and Early Agriculture in Oaxaca*, Mexico, K. V. Flannery (Editor), Academic Press, 1986, pp. 439-500.

[Reynolds 1992] Robert G. Reynolds, and Elena Zannoni, "Why Does Cultural Evolution Proceed at a Faster Rate than Biological Evolution," *Proceedings of Simulating Societies Symposium*, United Kingdom, 1992

[Reynolds 1993] Robert G. Reynolds, and Jonathan I. Maletic, "The Use of Version Space Controlled Genetic Algorithms to Solve the Boole Problem," *International Journal on Artificial Intelligence Tools*, Vol. 2, No. 2, 1993, pp. 219-234.

[Reynolds 1994] Robert G. Reynolds, "An Introduction to Cultural Algorithms," in *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, Sebald, A.V.; Fogel, L.J.

(Editors), River Edge, NJ, World Scientific Publishing, 1994, pp. 131-139.

[Reynolds 1995] Robert G. Reynolds, Z. Michalewicz, M. J. Cavaretta, "Using Cultural Algorithms for Constraint Handling in GENOCOP," in *Proceedings of the 4th Annual Conference on Evolutionary Programming*, McDonnell, J.R.; Reynolds, R. G.; Fogel, D.B.; (Editors),.MIT press, 1995, pp 289 - 305

[Reynolds 1996] Robert G. Reynolds and ChanJin Chung, "A Self-adaptive Approach to Representation Shifts in Cultural Algorithms," in *Proceedings of IEEE International Conference on Evolutionary Computation(ICEC'96)*, Nagoya, Japan, 1996, pp. 94-99.

[Russel 1995] Stuart Russel and Peter Norvig, Artificial Intelligence - A Modern Approach, Prentice Hall, Englewood Cliffs, NJ, 1995

[Salomon 1996] Ralf Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions: A survey of some theoretical and practical aspects of genetic algorithms," in BioSystems, 39, 1996, pp. 263-278

[Saravanan 1994] N. Saravanan and D. B. Fogel, "Learning Strategy Parameters in Evolutionary Programming: An Empirical Study," in *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, editors: A. V. Sebald and L. J. Fogel, pp.269-280, 1994

[Schwefel 1981] H.-P. Schwefel, *Numerical Optimization of Computer Models*, John Wiley, Chichester, UK, 1981

[Schwefel 1995] Hans-Paul Schwefel, *Evolution and Optimum Seeking*, John Wiley & Sons, Inc., 1995

[Valente 1994] A. Valente and J. Breuker, "A Commonsense Formalization of Normative Systems," *Proceedings of the ECAI'94 Workshop on Artificial Normative Reasoning*, 1994

[Weinberger 1996] Edward D. Weinberger, np Completeness of Kauffman's n-k Model, A Tuneably Rugged Fitness Landscape, Santa Fe Institute Working Paper, #96-02-003, 1996

[Whitley 1995] D. Whitley, K. Mathias, S. Rana, and J. Dzubera, "Building Better Test Functions," In Eshelman, (ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 239-246. San Francisco, CA: Morgan Kaufmann, 1995

[Whitley 1996] D. Whitley, K. Mathias, S. Rana, and J. Dzubera, "Evaluating Evolutionary Algorithms," *Artificial Intelligence* Vol.85, pp. 245-276, 1996

[Winston 1992] P. H. Winston, *Artificial Intelligence*, 3rd edition, Addison-Wesley, Reading MA.

[Wolpert 1995] Wolpert, D. H. and Macready W. G., "No free lunch for search," Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM, 87501, USA, July 1995

[Yao 1996] Xin Yao and Yong Liu, "Fast Evolutionary Programming," in *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, 1996