Knowledge-Based Control of Self-Adaptive Evolutionary Search

Chan-Jin Chung

Dept. of Mathematics and Computer Science
Lawrence Technological University
Southfield, Michigan 48075-1058

Robert G. Reynolds

Dept. of Computer Science
Wayne State University
Detroit, Michigan 48202

Abstract

Self-adaptation has been frequently employed in evolutionary computation. Angeline [1995] defines three distinct adaptive levels which are: population, individual, and component level. Cultural Algorithms have been shown to provide a framework in which to model self-adaptation at each of these levels. Here, we examine the role that different forms of knowledge can play in the self-adaptation process at the population level for evolution-based function optimizers. In particular, we compare the relative performance of normative and situational knowledge in guiding the search process. An acceptance function using a fuzzy inference engine is employed to select acceptable individuals for forming the generalized knowledge in the belief space. Evolutionary programming is used to implement the population space. The results suggest that the use of a cultural framework can produce substantial performance improvements in execution time and accuracy for a given set of function minimization problems over population–only evolutionary systems.

1 Introduction

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 acquired problem solving knowledge during evolution is used to bias the problem solving process in order to identify patterns in an evolving population's performance environment [Reynolds, 1993, 1996; Chung 1996]. These patterns are used to influence the generation of candidate solutions, promote more instances of desirable candidates, or to reduce the number of less desirable candidates in the population.

Adaptive evolutionary computation takes place when an EC system is able to incorporate such knowledge into its representation and associated operators in order to facilitate the pruning and promoting activities mentioned above. These adaptations can take place at three different scales: the population level, the individual level, and the component level [Angeline, 1995]. At the population level, aspects of the system parameters that control all elements of the population can be modified. At the individual level, aspects of the system that control the action of specific individuals can be modified. At the component level, adaptive ECs dynamically alter how the individual components of each individual will be manipulated independently.

However, traditional ECs have limited or implicit mechanisms for representing and reasoning about the collective experience of individuals in a population. So we need an explicit mechanism for performing these activities that is compatible with an evolutionary learning perspective. In human societies, culture can be viewed as a vehicle for the storage of information that is 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 knowledge with which to bias the generation of individuals on at least a phenotypic level. This can be achieved by using collective knowledge to facilitate 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 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 support self-adaptation within an EC system.

Cultural Algorithms are computational models that consist of a social population and a belief space. 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 such a Cultural Algorithm is a dual inheritance system that supports the development of hybrid systems consisting of an underlying evolutionary search engine at the population level and a symbolic learning component at the belief level. In this study, the population component of the Cultural Algorithm will be Evolutionary Programming (EP) and the global knowledge will be represented using interval schema.

The purpose of this study is to find the relationships between cultural algorithm mechanisms and optimization performances. Our conjecture is that the knowledge needed to control evolutionary search by cultural algorithms will be dependent on the problem's characteristics and structure.

2 EP and Self-Adaptation

EC algorithms for optimization can be generally described by the following equation:

$$x[p,n]^{t+1} = s(v(x[p,n]^t))$$

where $x[p,n]^t$ is a population which consists of p candidate solutions with n parameters under a particular representation at time t. v is a variation operator to be applied to generate new solutions, and s is the selection operator that determines which candidate solutions will be survived in the next population $x[p,n]^{t+1}$. In EP, s is implemented by tournament selection mechanism. The following is the basic EP algorithm.

2.1 A basic EP Algorithm

This is a basic EP algorithm framework [Fogel, 1995] that will serve as the population component for the Cultural Algorithms described here.

(1) Select an initial population of p candidate solutions, $x[p,n]$, from a uniform distribution within the given domain for each of the n parameters.

(2) Assess the performance score of each parent solutions by the given objective function f.

(3) Generate p new offspring solutions from $x[p,n]$ by applying the variance operator, v. Now there are $2p$ solutions in the population.

$$x[2p,n] = v(x[p,n])$$

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

(5) For each individual, select c competitors at random from the population of $2p$ size. Conduct pairwise competitions between the individual and the competitors. This procedure can be described as:

For each x_i, $i = 1,\Lambda ,2p$, a value, number of wins, W_i is assigned according to:

$$w_i = \sum_{t=1}^{c} w_t^* \qquad w_t^* = \begin{cases} 1, & if\ f(x_i) \le f(x_o) \\ 0, & otherwise \end{cases}$$

where $o \neq i$, the index for a opponent $o = truncate(2pu + 1)$. $truncate(x)$ produces the greatest integer less than or equal to x, c is the number of competitions, and $u \sim U(0,1)$, uniform random variable.

(6) Select the p solutions that have the greatest number of wins (w_i) to be parents for the next generation.

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

Step (5) and (6) are procedures for the selection operator s usually used in EP.

2.2 Component level SA mechanism

Schwefel [1995] devised a method for Evolution Strategies to self-adapt σ, a parameter used to calculate mutation step sizes. This method can be represented as:

$$X[p,n]^{t+1} = s(v(\ X[p,n]^t))$$

where X is a composite structure of both real number x and σ. In this paper we use the following notation to represent this structure: $X = < x,\sigma >$. Here, we denote jth parameter value for ith individual as simply $x_{i,j}$, and the jth σ value for ith individual as simply $\sigma_{i,j}$. The variation operator in this method was defined as:

For all components $i = 1,\Lambda ,p$ and $j = 1,\Lambda ,n$

$$x_{p+i,j} = x_{i,j} + \sigma_{i,j} \cdot N_{i,j}(0,1)$$
$$\sigma_{p+i,j} = \sigma_{i,j} \cdot \exp(\tau' \cdot N_{i,j}(0,1) + \tau \cdot N_{i,j}(0,1))$$

where $N_{i,j}(0,1)$ is a realization of a Gaussian normal deviate for the jth parameter of ith individual. $x_{p+i,j}$ is an offspring of $x_{i,j}$ before selection. The offspring's step size is log normal perturbation of the parent's step size. The global factor $\tau' \cdot N_{i,j}(0,1)$ allows for an overall change of the mutability, whereas $\tau \cdot N_{i,j}(0,1)$ allows for individual changes in the step size of mutation. His recommended settings for the parameters τ and τ' are $\tau' = (\sqrt{2\sqrt{n}})^{-1}, \tau = (\sqrt{2n})^{-1}$ respectively.

2.3 Individual level SA mechanism

Here we developed an individual level self-adaptive EP that can be represented as:

$$< X[p,n], age[p] >^{t+1} = s(v(< X[p,n], age[p] >^t)),$$

where age_i means the age for the individual i. This represents a survival index for an individual. When an individual is initially generated by mutation, the age is zero. After the tournament selection, if an individual survives, then the count is increased by one. If the survival count reaches the number of individuals in the population, the counter is re-initialized to one. This index is used in the following way. The older an individual, the more likely it is to be associated with be a local optima. In order to escape from this locality, a larger perturbation may be required than normal. If we use this idea into Schwefel's self-adaptive method, the variation operator will be:

For all components $i = 1,\Lambda ,p$ and $j = 1,\Lambda ,n$

$$x_{p+i,j} = x_{i,j} + \sigma_{i,j} \cdot age_i \cdot N_{i,j}(0,1)$$
$$\sigma_{p+i,j} = \sigma_{i,j} \cdot \exp(\tau' \cdot N_{i,j}(0,1) + \tau \cdot N_{i,j}(0,1))$$

3 How to Culture EP

In the cultural model, the results of selection and performance evaluation at the population level are used as a basis for adjusting and reasoning about beliefs in the belief space. These new beliefs, in turn, effect the selection and adaptation of individuals at the population level. Therefore we view Cultural Algorithms as supporting population level self-adaptation [Reynolds, 1996].

Cultural Algorithm for optimization problems can be represented in general as:

$$x[p,n]^{t+1} = s_{culture}(v_{culture}(x[p,n]^t, beliefs^t), beliefs^t)$$

beliefs can be used for influencing variance operator and/or selection operator. *beliefs* itself can be adjusted as the following:

$$beliefs^{t+1} = adjust(beliefs^t, accept(p,t))$$

There are two basic categories of knowledge that we will consider: *normative* and *situational*. Normative knowledge is expressed here in interval form and provides standards for individual behavior and as well as 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. While there are other types of knowledge that can be contained within a cultural system, these two were selected because they are viewed to be fundamental to the operation of cultural systems.

3.1 Beliefspace structure

The formal syntax of the *beliefs* defined in this study is a pair structure $< E, N[n] >$, where E is a set of exemplar best individuals that constitute the situational knowledge. The normative component, N, is a collection of interval information describing each of the n parameters. Each one of the intervals in the belief space is represented as a triple $< I, L, U >$. I corresponds to a closed *interval*, a continuous set of real numbers x, represented as a an upper and lower boundary pair:

$$I = [l,u] = \{x \mid l \le x \le u\}.$$

Both of the bounds , l(lower bound) and u (upper bound), are initialized by the given domain values. L_j represents the performance score of the lower bound l for parameter j. U_j represents the performance score of the upper bound u for parameter j. They are initialized as $+\infty$.

3.2 Acceptance function

The following fuzzy *accept* function as shown in figure 1 determines the number of individuals which can be used to adjust the beliefspace.

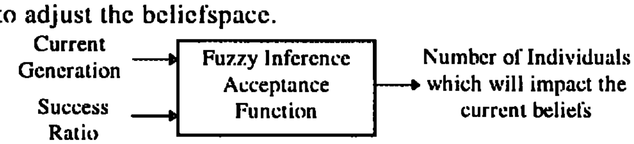

Figure 1. The Fuzzy Inference Acceptance Function

The antecedent membership functions used here are given in figures 2 and 3. The consequent membership function is shown in figure 4. The set of rules used here are shown in table 1.

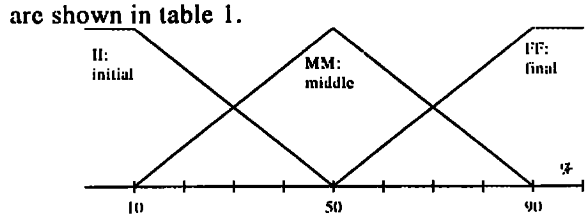

Figure 2 Membership functions for Current Generation

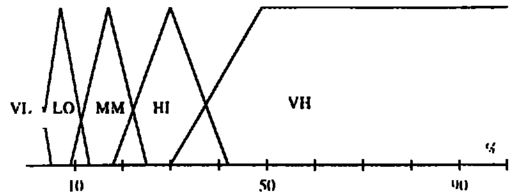

Figure 3 Membership functions for Success Ratio

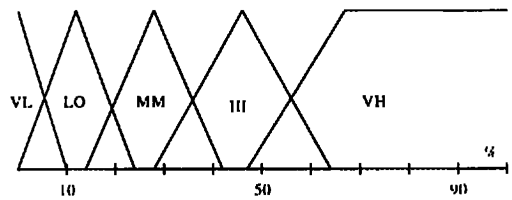

Figure 4 Membership functions for num. individuals

	VL	LO	MM	HI	VH
II	LO	MM		HI	VH
MM			MM		
FF	VL	LO		MM	MM

Table 1 Fuzzy Inference Rules

3.3 Adjusting the belief space

Here, the situational knowledge consists of the current best and the old best parameter vector found so far, that is, $E = \{\vec{E}^t, \vec{E}^{t-1}\}$ and updated by the following rule:

$$\vec{E}^{t+1} = \begin{cases} \vec{x}^t_{best} & if \quad f(\vec{x}^t_{best}) < f(\vec{E}^t) \\ \vec{E}^t & otherwise \end{cases}$$

where $\vec{x}^t_{best}$ denotes the best individual (solution parameter vector) in the population at time t.

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 as the following (assume *accepted* individuals from the operator $accept(p,t)$ are ith and kth individual):

For the left boundary and its score for parameter j:

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

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

where the ith individual affects the lower bound for parameter j. l_j^t is the lower limit for parameter j at generation t and L_j^t denotes the performance score for that lower limit.

For the right boundary and its score for parameter j:

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

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

where the kth individual affects the upper bound for parameter j. u_j^t represents upper limit for variable j at generation t and U_j^t denotes the performance score for it.

3.4 Cultured EP algorithm

The first step below should be inserted between (2) and (3) and the second step between (6) and (7) in the basic EP algorithm shown in subsection 2.1.

- Initialize the belief space with the given problem domain and candidate solutions. The structure of the belief space was given in subsection 3.1.
- Update the belief space using the acceptance function in 3.2. The beliefspace update rule is described in sections 3.3 and 3.4.

4 Cultured EP versions

The belief space knowledge can influence the evolutionary operator v, variation, in two ways: (1) determining the size of the changes, step size (2) and determining the direction of change, e.g. increase or decrease the current value. Four basic categories of Cultural Algorithms were produced by to support each of these possible permutations.

4.1 CAEP(Ns)

This version uses only normative knowledge for deciding the step size of mutation.

$$x_{p+i,j} = x_{i,j} + size(I_j) \cdot N_{i,j}(0,1)$$

where $size(I_j)$ represents the size of belief interval for the parameter j.

4.2 CAEP(Sd)

This version uses only situational knowledge for deciding the direction of mutation.

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

where $\sigma_{i,j}$ represents the individual level step size for ith variable of jth individual. E_j is the best exemplar parameter value as the situational knowledge for variable j in the belief space.

4.3 CAEP(Ns+Sd)

This version integrates both normative knowledge for the step size and situational knowledge for mutation direction together as shown in the following influence rule.

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

4.4 CAEP(Ns+Nd)

This version uses only normative knowledge for both step sizes and directions. The basic idea behind this version is to perturb small when the parameter value of a parent is in good acceptable range at random direction; otherwise perturb according to the current belief range found toward the current range in the belief space.

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

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.

5 Implementation and Test Results

In this study we implement systems with different level of self-adaptation as shown in table 2. We investigate how the different self-adaptation mechanisms and knowledge types affects problem solving performance of 28 different optimization problems, shown in Figure 5.

Self-adaptive level	System Name
Component Level	Schwefel's SA
Individual Level	Indv. Level SA
Population Level	CAEP(Ns), CAEP(Sd), CAEP(Ns+Sd), CAEP(Ns+Nd)

Table 2 Implemented Sysytems

For fair comparisons, we used the same population size (p=40), the same number of function evaluations, the same number of maximum generations, and the same same number of tournament competitions. Evolution starts from the same initial population for each problem. Used initial sigma for Schwefel's SA was 5.0.

It is possible that a perturbed offspring can violate the given domain constraints. When that happened a stochastic correction method is employed to prodcue a legal individual. For example, if a mutation violates the boundary for the parameter, the new value for the parameter will be forced to be stochastically near the boundary but within the required range. This is achieved by the following rule.

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

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

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

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

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

DeJong F4 without noise: $f_4(x) = \sum_{i=1}^{n} i \cdot x_i^4$

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

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

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

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

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

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

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's:
$$f_{14}(x) = \sum_{i=1}^{30} \left\{ \sum_{j=1}^{5} (j a_i^{j-1} x_{j+1}) - \left[\sum_{j=1}^{6} (a_i^{j-1} x_j)\right]^2 - 1 \right\}^2 + x_1^2$$

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

Griewank's: $f_{16}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(x_i \big/ \sqrt{i}\right) + 1$

Kowalik's: $f_{17}(x) = \sum_{i=1}^{11} \left(a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right)^2$

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

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

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

Beale's:
$$f_2(x) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$$

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

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's: $f_{27}(x) = \sum_{i=1}^{10} \left(e^{-0.1 i x_1} - e^{-0.1 i x_2} - x_3\left[e^{-0.1 i} - e^{-i}\right]\right)^2$

Figure 5 Fuction Test Cases

Since no single search algorithm is best for all optimization problems, we are trying to test as many different functions possible. A broad spectrum of function optimization problems [Fogel, 1995; Salomon, 1996; Schwefel, 1995], shown in figure 5, reflect different degree of complexity. Also we developed a metric to assess the difficulty of evolutionary learning for a function. This metric can also be used to predict the number of generations needed to solve a problem based upon features of its functional landscape. The features are based upon parameters that have long been associated with hard to solve functions in AI as suggested by Winston. The parameters in this metric are dimension, modality, and decomposibility. As shown in table 3, modality is assigned 0 if a function is unimodal, a 1 if a function has a few local minimums, a 3 if a function has many local minimas. If a parameter is independent of other parameters in a function, this function is regarded as easy to optimize, since optimization can be performed in a sequence of n independent optimization processes. From this observation, the following condition is developed to find whether a function is easy to optimize or not [Salomon 1996].

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

If this condition is satisfied, the function is as easy to optimize as decomposable functions, because it allows us to obtain solutions for each x_i independently of all other parameters. Table 3 shows all the characteristics of functions tested. Column 6 of the table shows the results of applying this condition; if the function is decomposable, the symbol D is assigned; if not, then ND is assigned. From the above information, we measure the function difficulty metrics, f_m, by the following function.

$$f_m(N, M, D) = 200 + 15 * N * 2^M * D$$

where N: Number of Parameters; M: Modality(0 ~3). D: (1 - Decomposable, 2 - Not decomposable). The 7th column of the table shows the calculated function metrics for each function. This value predicts the number of resources (in this case temporal in terms of generations) needed to solve the problem based upon its structure.

Fn	n	S	$f(x^*)$	mod-ality	decom-posable	Fn metrics
1	30	$[-5.12, 5.12]^n$	0	0	D	650
2	2	$[-6.0, 6.0]^n$	0	0	ND	440
3	5	$[-5.12, 5.12]^n$	-3.0000e+01	0	D	275
4	30	$[-1.28, 1.28]^n$	0	0	D	650
4'	30	$[-1.28, 1.28]^n$	0	0	D	650
5	2	$[-65.536, 65.536]^n$	9.9800e-01	2	ND	440
6	2	$[-6.0, 6.0]^n$	0	1	ND	320
7	5	$[-5.12, 5.12]^n$	0	2	ND	800
8	4	$[-10.0, 10.0]^n$	0	3	ND	1160
9	30	$[-32.0, 32.0]^n$	0	2	ND	3800
10	2	$[-2.0, 2.0]^n$	3.0000e+00	1	ND	320
11	2	$[-3.0, 3.0]^n$	-1.0316e+00	1	ND	320
12	2	$[-100.0, 100.0]^n$	-1.0000e+00	1	ND	320
13	5	$[0.0, \pi]^n$	-6.0000e+00	1	ND	500
14	6	$[-2.0, 2.0]^n$	2.2800e-03	3	ND	1640
15	3	$[-5.0, 5.0]^n$	-1.174985e+02	1	D	290
16	30	$[-600.0, 600.0]^n$	0	2	ND	3800
17	4	$[-1.0, 1.0]^n$	3.0750e-04	2	ND	680
18	2	$[-10.0, 10.0]^n$	0	1	ND	320
19	2	$[-2.0, 2.0]^n$	-3.7910e-03	1	ND	320
20	5	$[0.0, \pi]^n$	-4.6870e+00	2	ND	800
21	2	$[-5.0, 5.0]^n$	0	1	ND	320
22	5	$[0.0, 10.0]^n$	-1.4000e+00	2	ND	800

23	30	[-5.12, 5.12]^n		0	0	D	650
24	30	[-5.12, 5.12]^n		0	0	ND	1100
25	30	[-10.0, 10.0]^n		0	0	ND	1100
26	30	[-1000.0, 1000.0]^n		0	0	ND	1100
27	1	[-10.0, 10.0]^n		0	1	ND	380

Table 3 Characteristics of the test functions

Table 4 shows the test results from 20 runs of each on a Pentium processor for each system and test function. The first value in a cell shows the average CPU time in milliseconds taken to complete each run. Completion occurs if either the solution is found or the given number of generations has been reached. The time is measured in such a way that one the solution is found to 6 significant figures, then the process is stopped; otherwise the process is continued unless the maximum number of generations has been reached. The second value in parenthesis gives the average percentage of finding the global minimum for the 20 runs. The mean best value for the function at completion is given as the third value. The best system for each function is given in bold characters based upon accuracy with generations, and mean best value used as tie-breakers in that order. The bottom row shows the average % of runs that found the global minimum for each system.

Fn	Schwefel's sa	Indv. level sa	CAEP(Ns)	CAEP(Sd)	CAEP(Ns+Sd)	CAEP(Ns+Nd)
1	23159 (0%) 2.11499e+00	21571 (0%) 1.98736e+00	18951 (0%) 1.80092e+02	26092 (0%) 1.74288e-01	17090 (0%) 1.07348e+02	**4058 (100%) 4.90424e-33**
2	4707 (20%) 6.21798e-03	3227 (80%) 5.45158e-06	2971 (100%) 1.75892e-08	3542 (60%) 5.82244e-06	**1018 (100%) 9.33794e-25**	2247 (60%) 3.41105e-03
3	805 (100%) -3.00000e+01	771 (100%) -3.00000e+01	462 (100%) -3.00000e+01	1720 (70%) -2.97000e+01	**201 (100%) -3.00000e+01**	475 (95%) -2.99500e+01
4	23535 (0%) 8.66742e-02	22313 (0%) 1.33430e+00	19144 (0%) 9.45961e+01	19566 (85%) 5.96772e-05	17046 (0%) 1.90181e+01	**2214 (100%) 1.54030e-55**
4*	4443 (100%) -1.36954e+01	13465 (100%) -8.66874e+00	24516 (0%) 9.64203e+01	799 (100%) -1.83233e+01	22426 (0%) 1.98119e+01	**476 (100%) -2.07861e+01**
5	7330 (0%) 2.98211e+00	6234 (15%) 2.58548e+00	**2202 (100%) 9.98004e-01**	7393 (5%) 2.78389e+00	1901 (95%) 1.097407e+00	3249 (60%) 1.49462e+00
6	568 (100%) 0.00000e+00	1098 (100%) 6.80743e-18	442 (100%) 0.00000e+00	516 (95%) 1.09157e-02	217 (100%) 0.00000e+00	**208 (100%) 0.00000e+00**
7	14036 (0%) 6.81547e+00	11746 (0%) 4.50824e+00	11234 (0%) 1.17001e+01	12511 (5%) 4.77580e+00	10955 (5%) 5.78013e+00	**4316 (70%) 2.98488e-01**
8	15885 (0%) 2.20669e+00	14953 (0%) 1.92955e+00	13621 (0%) 7.83354e-01	16352 (0%) 1.04678e+00	**11551 (95%) 3.08568e-07**	13800 (0%) 3.64809e-01
9	162033 (0%) 1.69401e+01	15125 (9 0%) 1.64583e+01	141266 (0%) 2.05851e+01	180074 (0%) 1.52589e+01	12475 (8 0%) 1.91717e+01	**9593 (100%) 3.58637e-15**
10	580 (100%) 3.00000e+00	1365 (100%) 3.00000e+00	480 (100%) 3.00000e+00	337 (100%) 3.00000e+00	224 (100%) 3.00000e+00	**209 (100%) 3.00000e+00**
11	298 (100%) -1.03163e+00	532 (100%) -1.03163e+00	2954 (30%) -1.03152e+00	**194 (100%) -1.03163e+00**	354 (100%) -1.03163e+00	535 (100%) -1.03163e+00
12	1474 (100%) -1.00000e+00	2181 (95%) -9.50000e-01	3693 (0%) -1.26993e-01	1236 (100%) -1.00000e+00	884 (100%) -1.00000e+00	**390 (100%) -1.00000e+00**
13	2110 (100%) -6.00000e+00	4088 (100%) -6.00000e+00	4136 (100%) -6.00000e+00	973 (100%) -6.00000e+00	829 (100%) -6.00000e+00	**490 (100%) -6.00000e+00**
14	29687 (5%) 9.00007e-03	29741 (0%) 1.27008e-02	30052 (0%) 9.99888e-02	34704 (0%) 5.10732e-03	**16343 (5%) 9.48498e-03**	29141 (0%) 5.91309e-03
15	791 (100%) -1.17499e+02	1429 (100%) -1.17499e+02	1079 (100%) -1.17499e+02	435 (100%) -1.17499e+02	360 (100%) -1.17499e+02	**286 (100%) -1.17499e+02**
16	168086 (0%) 2.44829e+01	157982 (0%) 2.90656e+01	143918 (0%) 6.19296e+02	182339 (0%) 1.23537e+01	132710 (0%) 2.05707e+02	**48984 (65%) 4.18658e-03**
17	9258 (10%) 3.21697e-04	8870 (20%) 3.18204e-04	8667 (0%) 3.09451e-04	7961 (75%) 3.08578e-04	**2096 (100%) 3.07486e-04**	8763 (0%) 3.25554e-04
18	539 (100%) 1.21387e-36	803 (100%) 1.79738e-16	537 (100%) 2.00556e-37	307 (100%) 3.69772e-66	235 (100%) 7.72028e-92	**208 (100%) 1.97098e-24**
19	432 (100%) -3.79124e-03	723 (100%) -3.79124e-03	327 (100%) -3.79124e-03	255 (100%) -3.79124e-03	**153 (100%) -3.79124e-03**	159 (100%) -3.79124e-03
20	12995 (5%) -4.34164e+00	12836 (5%) -4.50881e+00	12364 (0%) -4.08997e+00	9958 (30%) -4.49094e+00	**9058 (40%) -4.65178e+00**	8568 (30%) -4.63587e+00
21	585 (100%) 5.68535e-32	1085 (100%) 5.26111e-14	621 (100%) 1.57926e-33	347 (100%) 0.00000e+00	**250 (100%) 1.57926e-33**	325 (100%) 1.55027e-15
22	14179 (10%) -8.13117e-01	8516 (55%) -1.20979e+00	11760 (50%) -1.18669e+00	14978 (5%) -7.82520e-01	5465 (65%) -1.30361e+00	**3192 (80%) -1.38730e+00**
23	25155 (0%) 7.25412e+19	23473 (0%) 1.11182e+21	20706 (0%) 3.16132e+22	27441 (0%) 3.07982e+17	18946 (0%) 2.30345e+16	**12547 (100%) 1.90480e-17**
24	42358 (0%) 3.13262e+02	39417 (0%) 6.31084e+01	36877 (0%) 4.02355e+04	48322 (0%) 1.01200e+00	33230 (0%) 5.86054e+03	**5971 (100%) 1.47495e-54**
25	39931 (0%) 8.78498e+00	37138 (0%) 1.02078e+01	34177 (0%) 6.26233e+09	43949 (0%) 5.28592e+00	30340 (0%) 5.56204e+06	**8098 (100%) 4.02695e-28**
26	39425 (0%) 3.58138e+01	36717 (0%) 6.32735e+01	32801 (0%) 8.34362e+01	43633 (0%) 5.34834e+01	30011 (0%) 8.34362e+01	**20817 (100%) 9.01613e-09**
27	635 (100%) 7.39999e-34	1495 (100%) 1.12858e-12	765 (100%) 7.92714e-10	623 (100%) 3.14916e-27	1277 (100%) 1.29620e-08	**456 (100%) 2.07820e-11**
%	45%	49%	42%	51%	58%	81%

Table 4 Performance Results

6 Conclusion

The test results indicate that the population only systems were always outperfromed by at least one of the cultural Algorithm configurations. However, since a "cultured" system must have additional overhead to process the knowledge contained in the belief space is it necessarily true that Cultural versions need more time to compute the solution as opposed to the population only systems. Our results demonstrate that if the accumulated knowledge is effective at guiding the search process, then fewer trials and fewer generations may be needed. Therefore, the best cultural system actually used less CPU time to reach the required global optimum for each problem.

Also, higher level self-adaptation gives better results in general. The results demonstrate that the use of a cultural framework for self-adaptation in EP can produce substantial performance improvements as expressed in terms of CPU time, the percentage of finding global minimum, and the mean best. The nature of these improvements depends on the type of knowledge used and the structure of the problem. For example, situational knowledge may not be useful for high dimensional problems, since systems which use situational knowledge like CAEP(Ns+Sd), CAEP(Sd), were not the best performers for such problems. Systems that used normative knowledge exclusively CAEP(Ns+Nd) consistently outperformed those using situational knowledge. Also the best performance is produced by systems that use knowledge to decide both step size and direction.

References

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

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

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

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

[Salomon, 1996] Salomon, Ralf, 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, pp.263-278

[Schwefel, 1995] Schwefel, H. P., *Evolution and Optimum Seeking*, John Wiley and Sons, Inc., New York.