

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

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: $f_3(x) = \sum_{i=1}^n \lfloor x_i \rfloor$
- 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: $f_5(x) = 1/0.002 + \sum_{i=1}^{25} \frac{1}{j + \sum_{k=1}^4 (x_k - a_{ik})^2}$
- 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: $f_7(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$
- Colville's: $f_8(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 + 90(x_4 - x_1^2)^2 + (1 - x_1)^2 + 0.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$
- Ackley's: $f_9(x) = -20 \exp(-0.2 \sqrt{1/n \sum_{i=1}^n x_i^2}) - \exp(\frac{1}{n} \sum_{i=1}^n \cos 2\pi x_i) + 20 + e$
- Goldstein-Price's: $f_{10}(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \cdot [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$
- Six hump camel back: $f_{11}(x) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$
- Easom's: $f_{12}(x) = -\cos(x_1) \cos(x_2) e^{-(0.1 + \pi) |x_1 - x_2|}$
- 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: $f_{14}(x) = \sum_{i=1}^n \left(\sum_{j=1}^i (j a_j^{i-j+1} x_{j+1}) - \left[\sum_{j=1}^i (a_j^{i-j+1} x_j) \right]^2 - 1 \right)^2 + x_i^2$
- Koon's F3: $f_{15}(x) = \frac{1}{2} \sum_{i=1}^n (x_i^4 - 16x_i^2 + 5x_i)$
- Griewank's: $f_{16}(x) = \frac{1}{4000} \sum_{i=1}^n x_i^4 - \prod_{i=1}^n \cos(x_i / \sqrt{i}) + 1$
- Kowalik's: $f_{17}(x) = \sum_{i=1}^{11} \left(a_i - \frac{x_i (b_i^2 + b_i x_2)}{b_i^2 + b_i x_1 + x_4} \right)^2$
- 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: $f_{20}(x) = -\sum_{i=1}^n \sin(x_i) \cdot \sin^{2m}(i \cdot x_i / \pi), m = 20$
- Beale's: $f_2(x) = [15 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$
- Langerman's: $f_{22}(x) = -\sum_{i=1}^m c_i \left(e^{-4i} \cos(\pi \cdot \|x - A(i)\|) \right), m = 5$
- 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: $f_{26}(x) = \max_i \{|x_i|\}, 1 \leq i \leq n$
- Box's: $f_{27}(x) = \sum_{i=1}^{10} \left(e^{(0.1 + i/x_i)} - e^{-(0.1 + i/x_i)} - x_i [e^{-(0.1 + i)} - e^{-(i-1)}] \right)^2$

Figure 5 Function 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 minimums. 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].

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

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.

| Fu | n | S | f(x') | modality | decomposable | Fm metrics |
|----|----|-------------------|---------------|----------|--------------|------------|
| 1 | 30 | [-5.12, 5.12] | 0 | 0 | D | 650 |
| 2 | 2 | [-6.0, 6.0] | 0 | 0 | ND | 340 |
| 3 | 5 | [-5.12, 5.12] | -3.00000e+01 | 0 | D | 275 |
| 4 | 30 | [-1.28, 1.28] | 0 | 0 | D | 650 |
| 4' | 30 | [-1.28, 1.28] | 0 | 0 | D | 650 |
| 5 | 2 | [-65.536, 65.536] | 9.98000e-01 | 2 | ND | 440 |
| 6 | 2 | [-6.0, 6.0] | 0 | 1 | ND | 320 |
| 7 | 5 | [-5.12, 5.12] | 0 | 2 | ND | 800 |
| 8 | 4 | [-10.0, 10.0] | 0 | 3 | ND | 1160 |
| 9 | 30 | [-32.0, 32.0] | 0 | 2 | ND | 3800 |
| 10 | 2 | [-2.0, 2.0] | 3.00000e-00 | 1 | ND | 320 |
| 11 | 2 | [-3.0, 3.0] | -1.03160e-00 | 1 | ND | 320 |
| 12 | 2 | [-100.0, 100.0] | -1.00000e+00 | 1 | ND | 320 |
| 13 | 5 | [0.0, pi] | -6.00000e+00 | 1 | ND | 500 |
| 14 | 6 | [-2.0, 2.0] | 2.28800e-03 | 3 | ND | 1640 |
| 15 | 3 | [-5.0, 5.0] | -1.174985e+02 | 1 | D | 290 |
| 16 | 30 | [-600.0, 600.0] | 0 | 2 | ND | 3000 |
| 17 | 4 | [-1.0, 1.0] | 3.07500e-04 | 2 | ND | 680 |
| 18 | 2 | [-10.0, 10.0] | 0 | 1 | ND | 320 |
| 19 | 2 | [-2.0, 2.0] | -3.79100e-03 | 1 | ND | 320 |
| 20 | 5 | [0.0, pi] | -4.68700e+00 | 2 | ND | 800 |
| 21 | 2 | [-5.0, 5.0] | 0 | 1 | ND | 320 |
| 22 | 5 | [0.0, 10.0] | -1.40000e+00 | 2 | ND | 800 |

| | | | | | | |
|----|----|---------------|---|---|----|------|
| 23 | 30 | -5.12, 5.12 | 0 | 0 | D | 650 |
| 24 | 30 | -5.12, 5.12 | 0 | 0 | ND | 1100 |
| 25 | 30 | -10.0, 10.0 | 0 | 0 | ND | 1100 |
| 26 | 30 | -100.0, 100.0 | 0 | 0 | ND | 1100 |
| 27 | 3 | -1.00, 10.0 | 0 | 1 | ND | 300 |

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%) | 21571 (0%) | 18951 (0%) | 26492 (0%) | 17090 (0%) | 4058 (100%) | |
| | 2.11499e+00 | 1.98736e+00 | 1.80092e+02 | 1.74288e-01 | 1.07346e+02 | 4.90424e-32 | |
| 2 | 4707 (20%) | 3227 (80%) | 2971 (100%) | 3542 (60%) | 1018 (100%) | 2247 (60%) | |
| | 6.21799e+03 | 5.45158e+06 | 1.25892e+08 | 5.82244e+06 | 9.33794e-25 | 3.41105e+03 | |
| 3 | 805 (100%) | 771 (100%) | 462 (100%) | 1720 (70%) | 201 (100%) | 475 (95%) | |
| | -3.00000e+01 | -3.00000e+01 | -3.00000e+01 | -2.97000e+01 | -3.00000e+01 | -2.99500e+01 | |
| 4 | 23535 (0%) | 22313 (0%) | 19144 (0%) | 19566 (85%) | 17046 (0%) | 2214 (100%) | |
| | 8.66742e-02 | 1.33430e+00 | 9.45961e+00 | 5.96772e-05 | 1.90181e+01 | 1.54030e-55 | |
| 4' | 4443 (100%) | 13465 (100%) | 24516 (0%) | 799 (100%) | 22426 (0%) | 476 (100%) | |
| | -1.36954e+01 | -8.66674e+00 | 9.64201e+01 | -1.83233e+01 | 1.98119e+01 | -2.07861e+01 | |
| 5 | 7330 (0%) | 6234 (15%) | 2202 (100%) | 7393 (5%) | 1901 (95%) | 3249 (80%) | |
| | 2.98211e+00 | 2.58548e+00 | 9.98004e-01 | 2.78389e+00 | 1.097407e+00 | 1.49462e+00 | |
| 6 | 568 (100%) | 1098 (100%) | 442 (100%) | 516 (95%) | 217 (100%) | 208 (100%) | |
| | 0.00000e+00 | 6.80743e-18 | 0.00000e+00 | 1.09157e-02 | 0.00000e+00 | 0.00000e+00 | |
| 7 | 14030 (0%) | 11746 (0%) | 11234 (0%) | 12511 (5%) | 10955 (5%) | 4316 (70%) | |
| | 6.81544e+00 | 4.50823e+00 | 1.17001e+01 | 4.77580e+00 | 5.78013e+00 | 2.98488e-01 | |
| 8 | 15885 (0%) | 14953 (0%) | 13621 (0%) | 16352 (0%) | 11551 (95%) | 13800 (0%) | |
| | 2.24699e+00 | 1.92959e+00 | 7.83354e-01 | 1.04678e+00 | 3.08568e-07 | 3.64809e-01 | |
| 9 | 162033 (0%) | 15125 (0%) | 141266 (0%) | 180074 (0%) | 12475 (0%) | 9593 (100%) | |
| | 1.69401e+01 | 1.64583e+01 | 2.05851e+01 | 1.52589e+01 | 1.91717e+01 | 3.58637e-15 | |
| 10 | 580 (100%) | 1365 (100%) | 480 (100%) | 337 (100%) | 224 (100%) | 209 (100%) | |
| | 3.00000e+00 | 3.00000e+00 | 3.00000e+00 | 3.00000e+00 | 3.00000e+00 | 3.00000e+00 | |
| 11 | 298 (100%) | 532 (100%) | 2954 (30%) | 194 (100%) | 354 (100%) | 535 (100%) | |
| | -1.03163e-00 | -1.03163e-00 | -1.03163e+00 | -1.03163e+00 | -1.03163e+00 | -1.03163e+00 | |
| 12 | 1474 (100%) | 2181 (95%) | 3693 (0%) | 1236 (100%) | 864 (100%) | 390 (100%) | |
| | -1.00000e+00 | -9.50000e-01 | -1.26993e-01 | -1.00000e+00 | -1.00000e+00 | -1.00000e+00 | |
| 13 | 2110 (100%) | 4098 (100%) | 4136 (100%) | 973 (100%) | 829 (100%) | 490 (100%) | |
| | -6.00000e+00 | -6.00000e+00 | -6.00000e+00 | -6.00000e+00 | -6.00000e+00 | -6.00000e+00 | |
| 14 | 29687 (5%) | 29741 (0%) | 30052 (0%) | 34704 (0%) | 16343 (5%) | 29141 (0%) | |
| | 9.00607e-03 | 1.27000e-02 | 9.99000e-02 | 5.10732e-03 | 9.48499e-03 | 5.91300e-03 | |
| 15 | 791 (100%) | 1429 (100%) | 1079 (100%) | 435 (100%) | 360 (100%) | 286 (100%) | |
| | -1.17499e+02 | -1.17499e+02 | -1.17499e+02 | -1.17499e+02 | -1.17499e+02 | -1.17499e+02 | |
| 16 | 166086 (0%) | 157982 (0%) | 143918 (0%) | 182339 (0%) | 132710 (0%) | 48984 (65%) | |
| | 2.44829e+01 | 2.90656e+01 | 6.19296e+02 | 1.23537e+01 | 2.05707e+02 | 4.18658e-03 | |
| 17 | 9258 (10%) | 8870 (20%) | 8667 (0%) | 7961 (75%) | 2096 (100%) | 8763 (0%) | |
| | 3.21697e-04 | 3.18204e-04 | 3.09451e-04 | 3.08578e-04 | 3.07486e-04 | 3.25554e-04 | |
| 18 | 539 (100%) | 803 (100%) | 537 (100%) | 307 (100%) | 235 (100%) | 208 (100%) | |
| | 1.21387e-36 | 1.79738e-16 | 2.00556e-37 | 3.69772e-66 | 7.72028e-92 | 1.97098e-24 | |
| 19 | 432 (100%) | 723 (100%) | 327 (100%) | 255 (100%) | 153 (100%) | 159 (100%) | |
| | -3.79124e-03 | -3.79124e-03 | -3.79124e-03 | -3.79124e-03 | -3.79124e-03 | -3.79124e-03 | |
| 20 | 12995 (5%) | 12836 (5%) | 12364 (0%) | 9958 (30%) | 9058 (40%) | 8568 (30%) | |
| | -4.34164e+00 | -4.50881e+00 | -4.08997e+00 | -4.49094e+00 | -4.65178e+00 | -4.63587e+00 | |
| 21 | 585 (100%) | 1085 (100%) | 621 (100%) | 347 (100%) | 250 (100%) | 325 (100%) | |
| | 5.68535e-32 | 5.26111e-14 | 1.57926e-33 | 0.00000e+00 | 1.57926e-33 | 1.55027e-15 | |
| 22 | 14179 (10%) | 8516 (55%) | 11761 (50%) | 14978 (5%) | 5465 (65%) | 3192 (80%) | |
| | -8.13117e-01 | -1.20979e+00 | -1.18669e+00 | -7.82520e-01 | -1.30361e+00 | -1.38730e+00 | |
| 23 | 25155 (0%) | 23473 (0%) | 20706 (0%) | 27441 (0%) | 18946 (0%) | 12547 (100%) | |
| | 7.25412e+19 | 1.11182e+21 | 3.16132e+22 | 3.07982e+17 | 2.30345e+16 | 1.99048e+17 | |
| 24 | 42358 (0%) | 39417 (0%) | 36877 (0%) | 48322 (0%) | 33230 (0%) | 5971 (100%) | |
| | 3.13202e+02 | 6.30084e+01 | 4.02355e+04 | 1.01200e+00 | 5.86054e+03 | 1.47495e-54 | |
| 25 | 39931 (0%) | 37138 (0%) | 34177 (0%) | 43949 (0%) | 30340 (0%) | 8098 (100%) | |
| | 8.78499e+00 | 1.02078e+01 | 6.26233e+00 | 5.28592e+00 | 5.56204e+06 | 4.02695e-28 | |

| | | | | | | |
|----|-------------|-------------|-------------|-------------|-------------|--------------|
| 26 | 39425 (0%) | 36717 (0%) | 32801 (0%) | 43633 (0%) | 30011 (0%) | 20817 (100%) |
| | 3.58138e+01 | 6.32735e+01 | 8.34362e+01 | 5.34834e+01 | 8.34362e+01 | 9.01613e-09 |
| 27 | 635 (100%) | 1495 (100%) | 765 (100%) | 623 (100%) | 1277 (100%) | 456 (100%) |
| | 7.39999e-14 | 1.12858e-12 | 7.92714e-10 | 3.14916e-27 | 1.29620e-08 | 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 outperformed 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.