Adaptive Terrain-Based Memetic Algorithms

Carlos R. B. Azevedo*
Center for Informatics
Federal University of Pernambuco
Cidade Universitária, Recife, Brazil
crba@cin.ufpe.br

V. Scott Gordon
Computer Science Department
California State University Sacramento
Sacramento, CA 95819-6023
gordonvs@ecs.csus.edu

ABSTRACT

The Terrain-Based Memetic Algorithm (TBMA) is a diffusion MA in which the local search (LS) behavior depends on the topological distribution of memetic material over a grid (terrain). In TBMA, the spreading of meme values – such as LS step sizes – emulates cultural differences which often arise in sparse populations. In this paper, adaptive capabilities of TBMAs are investigated by meme diffusion: individuals are allowed to move in the terrain and/or to affect their environment, by either following more effective memes or by transmitting successful meme values to nearby cells. In this regard, four TBMA versions are proposed and evaluated on three image vector quantizer design instances. The TBMAs are compared with K-Means and a Cellular MA. The results strongly indicate that utilizing dynamically adaptive meme evolution produces the best solutions using fewer fitness evaluations for this application.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms, Experimentation

Keywords

Memetic algorithms, adaptation, terrain-based models

1. INTRODUCTION

Adaptation in natural systems often requires a complex trade-off between genetic inheritance and individual learning. Bird flight and human walking are two examples of essential activities which require both adapted body structures (e.g., wings, legs) and training. A newborn's learning is usually performed through imitation, whereas biological adaptation is handled by natural selection and gene inheritance [31]. Therefore, complex human behaviors (e.g., communications) are expected to be influenced by both genetic and cultural evolution. The universal applicability of evolution to non-genetic systems was suggested by Dawkins, who has also introduced the concept of *meme* as a unit of cultural evolution in [7]. According to Dawkins, memes compete with their alleles for resources (i.e., time and space) in the "survival machines" (i.e., human brains).

In Evolutionary Computation, the term Memetic Algorithm (MA) was coined by Moscato [26] for designating population-based optimization heuristics inspired by both Darwinian evolution and Dawkins' meme concepts. Such algorithms make use of hybridization techniques for mixing Evolutionary Algorithms (EAs) with local improvements (e.g., hill climbing) [26]. It has been shown that MAs generally perform better than when either EAs or local search (LS) are used standalone [25, 27, 11].

While most works in MAs consider static local operations, MAs can benefit from the adaptation of LS [29]. In this regard, diffusion may play a central role in MAs. Nguyen et al. state that "*there exists a plethora of non-genetic transfers, which may include migration, diffusion, direct teaching and many others.*" [28]. Besides, Gen and Cheng [12] point out that there is a key difference between genes and memes: memes are typically processed and adapted before being diffused from one individual to another, whereas genes are fully inherited from parents.

The present paper is concerned with adaptation in MAs by meme diffusion. We propose four spatially distributed MAs, three of which may be considered adaptive MAs. A correspondence between meme and physical location is proposed, modeling cultural differences which may arise from sparse territories. All models are derived from the Terrain-Based Genetic Algorithm (TBGA) proposed by Gordon et al. [14], which made simultaneous usage of different parameter combinations such as mutation rates and number of crossover points for evolving individuals in a structured population. The proposed MAs integrate an accelerated version of K-Means – whose behavior depends on a scale factor parameter (η) – and are evaluated on vector quantizer design application for image coding. The proposed methods are compared with K-Means and with a Cellular MA.

The paper is organized as follows: Section 2 introduces vector quantization, the K-Means algorithm and its accelerated version. Section 3 provides background on adaptive EAs and MAs, and spatially distributed population EAs.

*Supported by the Brazilian Council for Technological and Scientific Development (CNPq).

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

Section 4 presents the proposed MAs. Section 5 describes the experimental methodology and settings used. Finally, Sections 6 and 7 present results and conclusion, respectively.

2. VECTOR QUANTIZATION

Vector Quantization [13] (VQ) has been established as an effective source coding technique that plays an important role in many signal compression systems, e.g. [35, 32]. Shannon's rate-distortion theory [5] establishes the superiority of VQ over scalar quantization. VQ is also used in other applications, such as speaker identification [17], steganography and digital watermarking [23, 6].

An N-level vector quantizer of dimension K can be defined as a mapping Q from a vector $\vec{x}$ in K-dimensional Euclidean space, $\mathbb{R}^K$, into a a finite subset W of $\mathbb{R}^K$ containing N distinct reproduction vectors. Thus,

$$Q : \mathbb{R}^K \rightarrow W, \tag{1}$$

where the codebook $W = \{\vec{w}_i\}|_{i=1}^N$ is the set of K-dimensional codevectors (reconstruction vectors).

The code rate of the quantizer, which measures the number of bits per vector component, is $R = \frac{1}{K}\log_2 N$. In image coding, R is expressed in bits per pixel (bpp).

The mapping Q leads to a partition of $\mathbb{R}^K$ into N regions S_i, $i = 1,\ 2,\ \ldots,\ N$, for which

$$\bigcup_{i=1}^{N} S_i = \mathbb{R}^K \text{ and } S_i \cap S_j = \emptyset \text{ if } i \neq j, \tag{2}$$

where each cell S_i is defined as

$$S_i = \{\vec{x} :\ Q(\vec{x}) = \vec{w}_i\}. \tag{3}$$

Codebook design plays a crucial role in the scenario of signal compression systems based on vector quantization. The most widely used technique for designing VQ codebooks is the K-Means algorithm [24].

2.1 The *K*-Means Algorithm

Let the iteration of K-Means be denoted by n. Given K, N and a distortion threshold $\varepsilon > 0$, the K-Means algorithm [24] consists of the following steps:

- Step 1: Given an initial codebook W_0 and a training set $X = \{\vec{x}_m\}|_{m=1}^M$, set $n \leftarrow 0$ and $D_{-1} \leftarrow \infty$.
- Step 2: Let W_n be the codebook at the n-th iteration and $\vec{w}_i^n$ the i-th codevector in W_n. Assign each input vector to the corresponding partition according to the nearest neighbor rule; determine the distortion $D_n = \sum_{i=1}^N \sum_{\vec{x}_m \in S_i} d(\vec{x}_m, \vec{w}_i^n)$.
- Step 3: If $(D_{n-1} - D_n)/D_{n-1} \leq \varepsilon$ then stop, with W_n representing the final codebook (designed codebook); else, continue.
- Step 4: Calculate the new codevectors as $\vec{w}_i^{n+1} = \mathcal{C}(\mathcal{V}(\vec{w}_i^n))$, where $\mathcal{C}(\mathcal{V}(\vec{w}_i^n))$ is the centroid of the partition $\mathcal{V}(\vec{w}_i^n)$; set $W_{n+1} \leftarrow W_n$; set $n \leftarrow n+1$ and go to Step 2.

In K-Means, the distortion decreases monotonically, since the codebook is updated to satisfy the nearest neighbor rule (Step 2) and the centroid condition (Step 4). Thus, K-Means may be regarded as a hill-climbing technique.

2.2 The Accelerated *K*-Means

In the accelerated K-Means (AKM), proposed by Lee et al. [22], the new codevector is updated according to

$$\vec{w}_i^{n+1} = \vec{w}_i^n + \eta(\mathcal{C}(\mathcal{V}(\vec{w}_i^n)) - \vec{w}_i^n), \tag{4}$$

where η is the scale factor, $\vec{w}_i^n$ denotes the codevector $\vec{w}_i$ at the end of the n-th iteration and $\mathcal{C}(\mathcal{V}(\vec{w}_i^n))$ is the centroid of the partition $\mathcal{V}(\vec{w}_i^n)$.

This method may be seen as a lookahead approach aiming at improving convergence, while reaching a smaller value of average distortion. Lee et al. experimentally showed that for $1.0 < \eta < 2.0$ their method converges faster than standard K-Means and results in a better codebook in terms of mean square quantization error. In the experiments reported in [22], when the value of η is about 1.8, the algorithm generally achieves good performance. It is worth mentioning that this method demands essentially the same computational effort of K-Means for each iteration, since the additional computation of Eq. (4) presents low complexity when compared to the centroid computation. In fact, $\eta = 1.0$ implements the standard K-Means.

3. BACKGROUND AND RELATED WORK

This section provides some background and related work regarding adaptive EAs and MAs, as well as structured population models such as the CGA, TBGA and their variants.

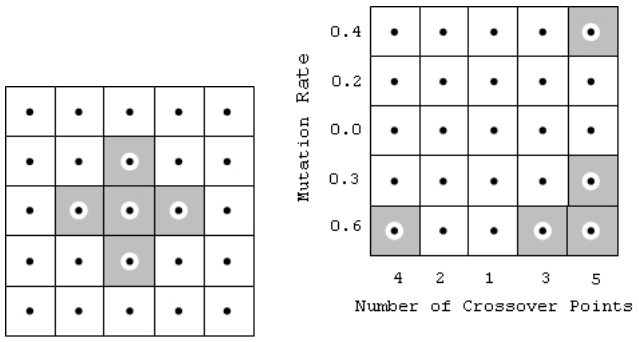
3.1 Parameter Adaptation in EAs

Adaptation of EAs parameters and operators, such as mutation rate and crossover, has been heavily studied over the past decades [10, 18]. For instance, in one of Evolution Strategies (ES) adapting schemes, the mutation operator is adjusted through the concatenation of the variances used in Gaussian perturbations within the solution's genotype [10].

In the specific case of MAs, the adaptation of memes (i.e. local search) is a subject of investigation. Krasnogor and Smith adopted in [20] the strategy of encoding memes' parameters directly into individual's genotype, thus evolving memes in an inheritance mechanism. In that work, the authors used self-adaptation in the sense of leaving the decision of which local search to apply to the evolutionary process itself. Other approaches were proposed for memes adaptation in MAs, but, to date, memes adaptation was not yet investigated using terrain-based models, though a related approach has been recently used to adapt memes in spatially distributed populations [28].

3.2 Cellular Genetic Algorithms

Genetic Algorithms (GAs) are believed to be more effective as function optimizers when some form of locality or spatial distribution is imposed on the mating and replacement strategies. This is consistent with some fundamental beliefs regarding natural evolution, such as Sewall Wright's claim that modeling biological evolution requires considering local interaction and spatial isolation [34]. Two methods have been commonly employed in EAs: distributing the individuals into islands or across a grid. Both incorporate isolation-by-distance, wherein periodic interaction between distant individuals is accomplished in the island model by migration, and in the grid-based structure by diffusion. Both also allow for individuals in different locations to be treated differently; one early example was the Injection Island GA



(a) CGA (b) TBGA

Figure 1: (a) Cellular Genetic Algorithm (CGA); (b) Terrain-Based Genetic Algorithm (TBGA)

developed by Eby et al., in which the authors subdivided a flywheel optimization problem across subpopulations [9]. Spatial isolation also has an added benefit of better utilizing parallel architectures, which was actually the initial motivation for these methods. When researchers discovered that parallel GAs performed better even when implemented serially, their fundamental advantages became evident [2, 15].

One particular method of incorporating spatial distribution into a GA is the Cellular GA (CGA). The term was coined by Whitley in his work with a cellular automata model [33]. In a CGA, individuals are assigned to locations on a grid and mating is limited to within demes (neighborhoods). Demes have been defined in a variety of ways, includig random walk, fixed topologies, 2 or 3-dimensional grids, and even by splitting the grid into islands. The latter was dubbed PEGA by Dorronsoro et al. [8], although it had been investigated earlier by Gordon and Whitley, who referred to it as I-CGA [16]. There is empirical evidence that higher locality (smaller deme) leads to faster and better problem solving in most cases [16]. Baluja's experiments in which 2-dimensional grids outperformed 3-dimensional ones [4], is consistent with the preference for smaller demes.

For these and other reasons, one of the most commonly-used CGAs has a fixed topology with a deme size of 4. Each individual is processed at every generation, and an individual's mate is selected from the best of the four strings located above, below, left, and to the right of the individual (see Fig. 1(a)). Crossover is always performed, yielding two offspring, and mutation is then applied to each offspring. If either resulting offspring has a better or equal fitness than the original node, then that node is replaced by the most fit offspring. Edge elements wrap around, forming a torus.

The CGA and its island version (I-CGA or PEGA) have been shown to be effective for numerical optimization, as well as for travelling salesmen, knapsack, vehicle routing, and other applications [8, 16].

3.3 Terrain-Based Genetic Algorithm

The Terrain-Based GA (TBGA) is a version of the CGA, in which various combinations of parameter values appear in different physical locations of the population, forming a sort of terrain in which individual solutions evolve [14]. The motivation for the TBGA is to provide an environment that contains a diverse combination of parameter settings in the hopes that individuals will exploit the best ones automatically. The TBGA has been shown to perform well as a function optimizer, and for automatically determining good parameter settings for the CGA by tracking the locations where high fitness strings occur.

Since the CGA as previously described has two dimensions, two parameters can be specified as terrain variables. For example, in Fig. 1(b), crossover points are linearly spread along the X-axis, and mutation rates along the Y-axis. Each cell then has a different combination of parameter settings, although neighboring cells have similar settings. Note that the lowest values are placed in the middle, and successively higher values are alternated from side-to-side, so that the logically adjacent positions at the extrema (due to the toroidal property of the grid) are also similar in value.

The TBGA has some drawbacks. One is that the method proposed for finding good CGA parameters removes its ability to exploit parameter setting strategies that change over time. Another drawback is that if a particular set of parameters is optimal, only a few strings are able to use them. Krink and Ursem devised a clever modification to the TBGA that corrects this latter deficiency, by creating the TB Patchwork Model (TBPM) [21], in which strings are allowed to migrate around the grid, and multiple strings are allowed to inhabit the same cell. The TBPM thus exploits the discovery of superior parameter settings immediately and dynamically and could possibly also discover dynamic parameter setting strategies that changes over time.

The TBPM is not the only form of CGA in which strings move around the grid – e.g. Janson et al. have proposed the Hierarchical CGA (H-cGA) in which strings with higher fitness are moved towards the center of the grid [19].

3.4 Cellular Memetic Algorithm

A MA consists of three phases in one of its most common forms: (i) selection of existing solutions for reproduction; (ii) application of variation operators on the selected solutions to derive new ones; and (iii) the application of individual learning (i.e., LS) to improve solutions. A Cellular MA (CMA) has been defined in [1] as CGAs "[...] *wherein some knowledge of the problem is included* [...]". In that work, the authors augmented the CGA to include specific crossover, mutation and LS operators for solving Boolean Satisfiability (SAT) instances and concluded that those incorporations were clearly beneficial for the original CGA.

An adaptive version of CMA has been proposed in [28] for numerical optimization and has been coined Diffusion MA (DMA). In DMA, memes (i.e., LS) are randomly assigned to individuals. Each individual may perform one of the different LS available from its neighboring meme pool. Meme selection is done through a reward mechanism which relies on the average fitness of the neighbors. Then, the desired meme is adopted by diffusion. This approach differs from the inheritence mechanism of Krasnogor and Smith [20] in that memes are not encoded in the genotype and thus cannot get passed down without an explicit decision.

4. PROPOSED MEMETIC ALGORITHMS

This section introduces a new class of MAs designated as Terrain-Based MAs (TBMAs). The fundamental difference between the TBMAs and TBGA concerns the distribution of memetic material across the terrain, instead of variation operators parameters. Thus, LS behavior varies according to the topological coordinates in the grid. In this paper, the

scale factor η of AKM (Section 2.2) is discretized along the terrain's axes, composing the parameter space of TBMAs for image VQ. In the following, four versions of TBMA are described. Three of them may be regarded as adaptive MAs.

It should be noted that in the TBMAs, two iterations of AKM are performed for improving the generated offspring.

4.1 Stationary TBMA

The Stationary TBMA (sTBMA) is the simplest of the proposed TBMAs. It can be regarded as a TBGA + AKM memetic approach. The pseudo-code for sTBMA is given in Algorithm 1. It is worth noting that each individual selects a mate from its Deme-4 neighborhood and that the scale factors values (η_1, η_2) for each of two AKM iterations depend on the individual's position in the terrain. Also, the population in sTBMA is updated in a batch procedure.

Algorithm 1 Stationary TBMA (sTBMA)

```
InitTerrainAndPopulation(T,P)
while not terminating condition do
  for all individuals W_i in P do
    // Let (x_i,y_i) denote the coordinates of W_i ≡ W_{x_i,y_i}
    W_j ← max{W_{x_i+1,y_i}, W_{x_i,y_i+1}, W_{x_i-1,y_i}, W_{x_i,y_i-1}}
    W_i' ← Crossover(W_i, W_j)
    η1 ← ScaleFactor1At(x_i,y_i)
    η2 ← ScaleFactor2At(x_i,y_i)
    LocalSearch(Mutate(W_i'), η1, η2)
    Evaluate(W_i')
  end for
  for all individuals W_i in P do
    if Fitness(W_i') ≥ Fitness(W_i) then
      W_i ← W_i'
    end if
  end for
end while
return P
```

Since sTBMA is analogous to TBGA, sTBMA has the same drawbacks already mentioned in Sect. 3.3.

4.2 Motioner TBMA

The Motioner TBMA (mTBMA) resembles a memetic TBPM + AKM approach. As stated in Section 3.3, by allowing individuals to migrate around the terrain, the solution space can be explored under better parameter settings. In the following, we present some useful definitions.

Definition 1. A *city* C is a collection of $h = 1, \cdots, h_{max}$ individuals (citizens) which share the same physical location in the terrain. The parameter h_{max} gives the maximum number of citizens a city can support.

Definition 2. A *citizen* is an intelligent agent which, at each generation, performs RULE 1.

RULE 1. *if there is a neighbor with better fitness value at city C_b and $|C_b| < h_{max}$, migrate to the city of the best adapted neighbor; else perform RULE 2 with probability p.*

RULE 2. *migrate to a random distance-1 cell located somewhere in any of the cardinal directions {N, E, S, W} and intermediate directions {NE, SE, SW, NW}.*

Definition 3. A *Mayor* is the most fit citizen in a city and cannot be replaced by any offspring.

Algorithm 2 Motioner TBMA (mTBMA)

```
InitTerrainAndPopulation(T,P)
while not terminating condition do
  N_c ← CountNumberOfCities(T)
  M ← ∅ // Initialize Mayors' subpopulation
  for all cities C_i in terrain do
    η1 ← ScaleFactor1At(x_i,y_i)
    η2 ← ScaleFactor2At(x_i,y_i)
    if |C_i| = 1 and W_i is isolated then
      LocalSearch(Mutate(W_i), η1, η2)
      RandomWalk(T,W_i) // Migrate isolated citizen
    else
      // Run |C_i| - 1 generations of GAKM
      GAKM(C_i,η1,η2, |C_i| - 1)
      M ← M ∪ max C_i
    end if
  end for
  P ← ⋃_i^{N_c} C_i − M
  P ← P ∪ sTBMA(T,M) // One generation of sTBMA
  MigrateCitizens(T,P) // Performs RULES 1 and 2
end while
return P
```

In Algorithm 2, cities are evolved by Genetic Accelerated K-Means (GAKM) [3], a steady-state MA that improves offspring with two iterations of AKM. In this step, tournament selection of size 1.5 is used, in which the two best out of three competitors mate. Replacement is done as follows: generate two offspring and count the number of genes inherited from each parent; replace a randomly chosen parent by its nearest offspring in terms of the number of common genes, without regard of fitness, if the chosen parent is not the Mayor. In this way, diversity is expected to be maintained even in such small cities, while elitism at the subpopulation level guarantees that the best solutions will not be lost.

To complete a mTBMA generation, isolated cities with size one are updated with mutation and LS. After this step, $|P| - N_c$ citzens have been generated and/or mutated, where N_c is the number of cities over the terrain. Mayors are determined for the other cities and the subpopulation of Mayors is evolved by a generation of sTBMA. Thus, mTBMA produces |P| new citzens per generation. The call `sTBMA`(T,M) permits a relatively small genetic flow between neighboring cities. Finally, all citizens have a chance of migrating to nearby cities or to start a new city (by moving to an empty cell), according to Rules 1 and 2. It is worth noting that isolated citizens perform a one-step random walk over the terrain after having been updated by mutation and LS.

The mTBMA also has two drawbacks: first, Rules 1 and 2 do not guarantee that citizens will use the best parameter setting at each stage of the optimization process – possessing high fitness does not necessarily mean that it has been achieved because of their current terrain-values. Second, the level of discretization of η_1 and η_2 excludes the possibility of exploiting intermediate values. Hence, neither sTBMA nor mTBMA can be said to utilize optimal LS behavior.

4.3 Local Adaptive-sTBMA

The Local Adaptive-sTBMA (LA-sTBMA) performs adaptation of LS by changing the terrain-values at the cell level. Its pseudo-code is the same as sTBMA, except for the execution of a local adjustment procedure at the end of every

k generations. This procedure updates the scale factors of each cell using Eq. (7) and works as follows: let (x_i, y_i) and (x_b, y_b) be the topological coordinates of individual W_i and of its best fitted neighbor W_b, respectively. Let $F(\cdot,\cdot)$ be the sum of the fitnesses of the individuals at (x,y) from generation 1 to current generation G, or

$$F(x,y) = \sum_{t=1}^{G} fitness(W_{x,y}^{t}). \quad (5)$$

Compute the adjustment factor α_i for cell (x_i,y_i) as

$$\alpha_i = \frac{F(x_b,y_b)}{F(x_b,y_b)+F(x_i,y_i)}. \quad (6)$$

Update the terrain-variable at cell (x_i,y_i) as

$$\eta(x_i,y_i) = (1-\alpha_i)\times\eta(x_i,y_i) + \alpha_i \times \eta(x_b,y_b). \quad (7)$$

After a number of generations, which will depend on the choice of k, the terrain is expected to converge to suitable values of both η_1 and η_2 at all cells, thus performing an online tuning of parameters. Note that, unlike other terrain-based GAs and MAs, the terrain parameters are not uniform along a row or column, because the adjustment factor given above is not done on the axes but is done independently for each cell. Although, like the TBGA and TBMA, neighboring cells are still likely to have similar terrain values.

One drawback of LA-sTBMA is that it cannot turn back to lost terrain-values. Hence, LA-sTBMA may be more useful in problems for which optimal parameters do not have a large variance in time.

4.4 Hierarchical Adaptive-TBMA

In the Hierarchical Adaptive-TBMA (HA-TBMA), individuals are ranked by fitness and compete for locations near the top ranked individual (leader) by swapping positions in the terrain. Again, the pseudo-code is the same as sTBMA, except for the following modifications.

First, at each generation, individuals execute the rule

RULE 3. *if there is a neighbor of lower rank on the shortest path to the leader, swap positions.*

Hence, near-ranked individuals are expected to mate, thus implying a hierarchy relation upon the terrain. Note that unlike mTBMA, each cell supports only one individual. The HA-TBMA resembles H-cGA [19], mentioned in Section 3.3. However, the cell that the leader seeks to follow is not fixed; it may change throughout the algorithm. The goal of the leader is to reach the cell which yields the highest success frequency (SF). The SF for cell (x,y) at generation G is

$$SF(G,x,y) = \sum_{t=2}^{G} I(t,x,y),\ G \geq 2, \quad (8)$$

with

$$I(t,x,y) = \begin{cases} 1 & \text{if } fitness(W_{x,y}^{t}) > fitness(W_{x,y}^{t-1}), \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

The leader thus aims to take advantage of the best parameter settings, leaving behind poorer cells to the less fit individuals. Furthermore, terrain-values are adjusted every k generations: let (x_L, y_L) be the coordinates of the leader. The terrain-values of cell (x_i, y_i) are updated as

$$\eta(x_i,y_i) = \frac{(\ |P| - R(W_i)\ )\times\eta(x_i,y_i) + \eta(x_L,y_L)}{(\ |P| - R(W_i)\ ) + 1}, \quad (10)$$

where R(W_i) is the rank of individual W_i (note that W_L has R(W_L) = 1). In this approach, memes are gradually changed towards the terrain-values in use by the leader. Eq. (10) weights the leader's influence according to individual rank: cells of lower ranked individuals are more susceptible to changes than higher ranked ones. Thus, the leader's influence propagates throughout the terrain like a "tsunami wave", in which the amplitude of parameter change becomes larger at further away cells occupied by worse individuals.

The HA-TBMA can also be said to have the same drawback mentioned for LA-sTBMA in Sect. 4.3.

5. EXPERIMENTAL METHODOLOGY

In all experiments, 5×5 2-D grids were used. Thus, $|P|$=25. Both η_1 and η_2 assumed the sequence [1.3 1.1 1.0 1.2 1.4] for each axis. Although earlier in Section 2.2 we stated that η=1.8 was known to be a good setting for AKM, previous results using AKM within an MA have shown that lower values for η tend to be more effective [3]. We include 1.0 in the range of possible settings [1.0...1.4], because at η=1.0, AKM is the same as standard K-Means, giving the TBMAs the option of utilizing that if it turns out to be desirable.

Individuals encode codebooks W, thus, genes are codevectors $\vec{w}_i$, $i = 1,\cdots,N$, in which N is the codebook size. The variation operators are defined as follows:

Crossover – An n-point crossover is used to generate offspring, where n is a random integer in $\{1,\cdots,N\}$. It should be noted that $n = N$ is equivalent to uniform crossover.

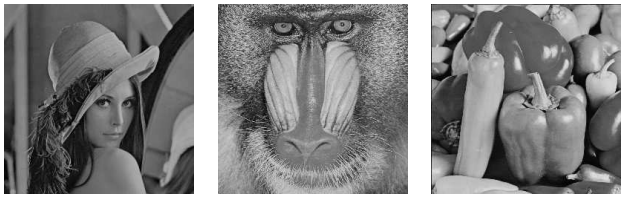
Mutation – For each gene $\vec{w}_i$, create a random noise vector $\vec{U} = [u_1 \cdots u_K]^T$, $u_j \sim N(0,1)\ \forall\ 1 \leq j \leq K$ and compute $\vec{w}_i \leftarrow \vec{w}_i + \vec{U}$, with probability $pmut$.

The fitness function is Peak Signal-to-Noise Ratio (PSNR), which assesses the objective quality of an image reconstructed by a given codebook. The PSNR is computed as

$$PSNR = 10 \times \log_{10}\left(\frac{255^2}{MSE}\right), \quad (11)$$

with

$$MSE = \frac{1}{K|X|} \times \sum_{i=1}^{N} \sum_{\vec{x}_m \in S_i} \|\vec{x}_m - \vec{w}_i\|^2. \quad (12)$$



(a) Lena. (b) Mandrill. (c) Peppers.

Figure 2: Images used as training sets.

The coding performance of the designed codebooks is evaluated on 256 × 256 monochrome images, originally encoded at 8 bpp: Lena, Mandrill and Peppers (Fig. 2). The distortion threshold $\epsilon = 10^{-3}$ is assumed for designing codebooks with K-Means. VQ with dimension K=16 is used, corresponding to the usage of 4×4 blocks of pixels. Codebook sizes of N=256 and 512 are considered, corresponding to coding rates of 0.5 and 0.5625 bpp. Also note

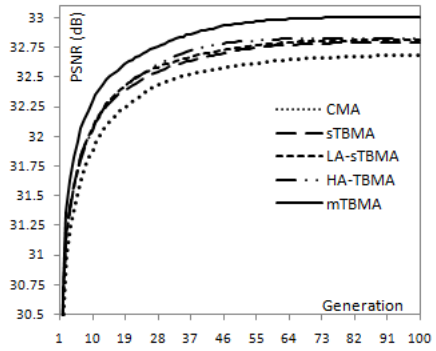


Figure 3: Average fitness of the best individual obtained at each generation, averaged over 20 runs on Peppers at 0.5625 bpp.

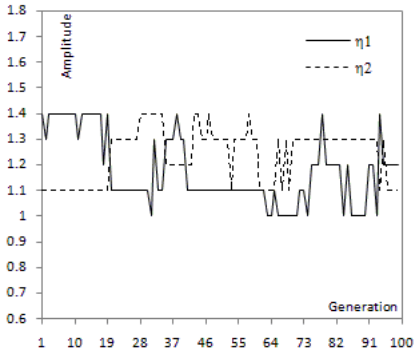


Figure 4: Terrain-values adopted by the best individual at each generation of best mTBMA run on Peppers at 0.5 bpp.

that, in CMA, two iterations of standard K-Means are performed for improving offspring. The mutation probability is $pmut = 0.00025$ for $N = 256$ and $pmut = 0.000125$ for $N = 512$, which correspond to $\approx 8\%$ of the offspring undergoing mutation each generation. In both LA-sTBMA and HA-TBMA, terrain-values are adjusted every $k = 2$ generations. In mTBMA, h_{max} is set to $\lfloor |P|/2 \rfloor = 12$ and p=0.2 (probability of migrating/starting a new city). Finally, 100 generations are performed for both CMA and the TBMAs.

6. RESULTS AND DISCUSSION

The mTBMA outperformed all other MAs in every case on the image VQ test suite used in this paper. An example of the performance of all investigated MAs on Peppers at 0.5625 bpp is plotted in Fig. 3, which shows the average fitness of the best individual obtained at each generation, averaged over 20 runs. It is observed that, by the end of 100 generations of CMA, the average fitness is around 32.7 dB. The mTBMA requires only 24 generations to achieve the same performance, a reduction of roughly 75% on the number of fitness evaluations computed before achieving the PSNR value obtained with CMA.

Table 1 summarizes the average final performance of all investigated methods. In every case, the MAs outperform K-Means in terms of average final PSNR values obtained for the images reconstructed with the resulting codebooks. Among the MAs, all of the various terrain-based models outperform the CMA, although some improvements are more pronounced than others.

Among the TBMAs, the mTBMA achieved the most significant improvements in all cases. There is not much difference in performance between the sTBMA, LA-sTBMA and HA-TBMA, although the HA-TBMA seems to possess a slightly better performance among those three TBMAs. The most dramatic gain was for Peppers at 0.5625 bpp (N=512), where mTBMA achieved an improvement of 2.09 dB, a reduction of $\approx 38\%$ in MSE over K-Means. In addition, as the codebook size (N) rises, the gains achieved with the investigated MAs over K-Means increase. This result supports the superiority of MAs over standalone hill-climbing LS techniques on the task of vector quantizer design.

The effects of the adjustment mechanisms implemented in HA-TBMA and mTBMA are contrasted in Fig. 5(a) and Fig. 5(b), respectively. Those figures show the terrain values adopted by the best individual at each generation, averaged over 20 independent simulations on Lena at 0.5625 bpp in Fig. 5(a) – and Peppers at 0.5 bpp in Fig. 5(b). Clearly the best individual in HA-TBMA settles on specific parameter values by about the 30th generation, with very little subsequent change. This was also observed for LA-sTBMA – in both TBMAs, η_1 and η_2 quickly converge to nearly the same pair of values at every cell. Interestingly, for 97% of the generations of HA-TBMA shown in Fig. 5(a), $\eta_1 > \eta_2$ holds, which is actually a reasonable heuristic for the problem of adjusting the scale factor in AKM if it were to be used standalone. For instance, Paliwal and Ramasubramanian proposed a linearly-decreasing variable scale factor as a function of the number of iterations of AKM [30].

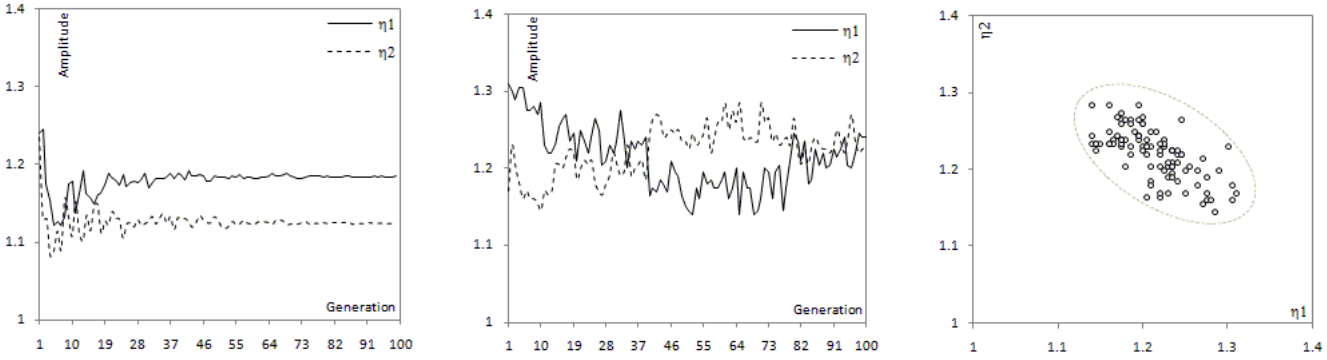
Despite the online tuning of parameters performed by HA-TBMA and LA-sTBMA, the adjustment mechanism did not lead to significant improvements over the results achieved with the simpler sTBMA (Table 1). The fact that mTBMA did exhibit significant improvement in every case suggests that the greater variability in terrain-values over time shown in Fig. 5(b) may play a valuable role in effective meme adaptation. In this regard, the mTBMA presumably utilizes migration for the tracking of suitable dynamically-changing terrain-values within the parameter space, and then utilizes cities for the building of distinct subpopulations that exploits those good parameter sequences.

Meme values utilized in very good individual mTBMA runs also changed over time. For example, the values of η_1 and η_2 for the single run that produced the best codebook for Peppers at 0.5 bpp (with a final fitness of 30.88 dB) are plotted in Fig. 4. The behavior is similar to the average behavior described earlier, and corroborates that exploiting dynamic adaptation of memes is a feature not only of the average performance, but also of the best performance for the algorithms that we tested.

Furthermore, the variability in terrain values utilized by mTBMA does not appear to simply be a chaotic sampling of the parameter space – there is an apparent pattern wherein as η_1 increases, η_2 decreases and vice-versa. This is evident both in the relative symmetry of the plots in Fig. 5(b), and the shape of the sample cluster of points (η_1, η_2) in Fig. 5(c). mTBMA has not only found an effective set of pa-

Table 1: Average PSNR (dB) values for codebooks designed with K-Means, CMA and the TBMAs.

Image (N)	K-Means	CMA	sTBMA	LA-sTBMA	HA-TBMA	mTBMA
Lena (256)	29.91 ± 0.08	30.85 ± 0.05	30.89 ± 0.05	30.90 ± 0.04	30.89 ± 0.05	31.03 ± 0.04
Lena (512)	31.14 ± 0.08	32.82 ± 0.08	32.90 ± 0.09	32.89 ± 0.09	32.94 ± 0.07	33.10 ± 0.05
Mandrill (256)	25.22 ± 0.02	25.50 ± 0.02	25.51 ± 0.02	25.52 ± 0.02	25.53 ± 0.02	25.58 ± 0.01
Mandrill (512)	26.19 ± 0.03	26.68 ± 0.04	26.70 ± 0.03	26.70 ± 0.03	26.71 ± 0.02	26.79 ± 0.02
Peppers (256)	29.70 ± 0.05	30.67 ± 0.05	30.70 ± 0.04	30.69 ± 0.06	30.71 ± 0.04	30.83 ± 0.03
Peppers (512)	30.92 ± 0.09	32.68 ± 0.11	32.79 ± 0.06	32.81 ± 0.07	32.83 ± 0.10	33.01 ± 0.08



(a) Meme values in HA-TBMA (b) Meme values in mTBMA (c) Cluster of memes in mTBMA

Figure 5: Terrain-values adopted by the best individual at each generation of TBMAs averaged over 20 runs. (a) HA-TBMA on Lena at 0.5625 bpp; (b) mTBMA on Peppers at 0.5 bpp; (c) Cluster of points (η_1, η_2) in mTBMA on Peppers at 0.5 bpp.

rameter values corresponding to the trajectory being taken through the solution space, it has incidentally also revealed that simultaneous extrema for both values (η_1 and η_2) are undesirable.

7. CONCLUSION

We have investigated the adaptive capabilities of a class of Memetic Algorithms called Terrain-Based Memetic Algorithms (TBMA). Specifically, four particular implementations dubbed Stationary TBMA (sTBMA), Local Adaptive Stationary TBMA (LA-sTBMA), Hierarchical Adaptive TBMA (HA-TBMA), and Motioner TBMA (mTBMA). The LA-sTBMA, HA-TBMA, and mTBMA can be considered adaptive MAs. We assessed the efficacy of the algorithms by testing them on a set of image Vector Quantization problems, and compared their performance against each other, as well as against a previously-described Cellular Memetic Algorithm and K-Means.

The MAs outperformed K-Means in all cases. Among the MAs, the TBMAs outperformed CMA in all cases. Among the TBMAs, the mTBMA significantly outperformed the other TBMAs and was always the best performing algorithm. Visualization of the progress of meme adaptation over time support the following conclusions:

- The sTBMA, LA-sTBMA, and HA-TBMA are able to find excellent meme values.
- The mTBMA is able to find even better sequences of dynamically changing meme values.

That is, although the adaptive sTBMA, LA-sTBMA, and HA-TBMA are all able to settle on very good parameters, the reason the mTBMA exhibits superior performance is that it is able to find and exploit dynamically-changing parameter values that correspond effectively with the trajectory that the VQ codebooks happen to be taking through the solution space during evolution. Interestingly, the algorithm that performed the best was the one that most closely resembled real-world human cultural dynamics, with individuals allowed to move semi-freely and form cities in a dynamic terrain. We conclude that memetic evolution is most effective when it can adapt dynamically, for the problems we tested. Additional research applying the mTBMA across a larger and more diverse test suite is a logical next step.

8. REFERENCES

[1] E. Alba, B. Dorronsoro, and H. Alfonso. Cellular memetic algorithms. *Journal of Computer Science and Technology*, 5(4):257–263, December 2005.

[2] E. Alba, A. J. Nebro, and J. M. Troya. Heterogeneous computing and parallel genetic algorithms. *J. Parallel Distrib. Comput.*, 62(9):1362–1385, 2002.

[3] C. R. Azevedo, T. A. Ferreira, W. T. Lopes, and F. Madeiro. Improving image vector quantization with a genetic accelerated k-means algorithm. In *ACIVS '08: Proceedings of the 10th International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 67–76, Berlin, Heidelberg, 2008. Springer-Verlag.

[4] S. Baluja. Structure and performance of fine-grain parallelism in genetic search. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 155–162, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[5] T. Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[6] Y. K. Chiang and P. Tsai. Steganography using overlapping codebook partition. *Signal Processing*, 88(5):1203–1215, May 2008.

[7] R. Dawkins. *The Selfish Gene*. Oxford University Press, New York, 1976.

[8] B. Dorronsoro, D. Arias, F. Luna, A. J. Nebro, and E. Alba. A grid-based hybrid cellular genetic algorithm for very large scale instances of the CVRP. In W. W. Smari, editor, *2007 High Performance Computing & Simulation Conference (HPCS 2007)*, pages 759–765, 2007.

[9] D. Eby, R. C. Averill, W. F. Punch, and E. D. Goodman. Optimal design of flywheels using an injection island genetic algorithm. *AI EDAM*, 13:327–340, 1999.

[10] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Trans. Evolutionary Comp.*, 3(2):124–141, 1999.

[11] P. Fränti, J. Kivijärvi, T. Kaukoranta, and O. Nevalainen. Genetic Algorithm for Codebook Generation in Vector Quantization. In *Proceedings of 3rd Nordic Workshop on Genetic Algorithms*, pages 207–222, Helsinki, 1997.

[12] M. Gen and R. Cheng. *Genetic Algorithms and Engineering Optimization*. Wiley-Interscience, Hoboken, 1999.

[13] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1992.

[14] V. S. Gordon, R. Pirie, A. Wachter, and S. Sharp. Terrain-based genetic algorithm (TBGA): Modeling parameter space as terrain. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99)*, pages 229–235, Orlando, FL, USA, 1999. Morgan Kaufmann Publishers Inc.

[15] V. S. Gordon and L. D. Whitley. Serial and parallel genetic algorithms as function optimizers. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 177–183, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[16] V. S. Gordon and L. D. Whitley. A machine independent analysis of parallel genetic algorithms. *Journal of Complex Systems*, 8(3):181–214, 1994.

[17] J. He, L. Liu, and G. Palm. A discriminative training algorithm for VQ-based speaker identification. *IEEE Trans. Speech Audio Process.*, 7(3):353–356, May 1999.

[18] R. Hinterding, Z. Michalewicz, and A. E. Eiben. Adaptation in evolutionary computation: a survey. In *Proceedings of the Fourth IEEE Conference on Evolutionary Computation*, pages 65–69, Indianapolis, IN, 1997.

[19] S. Janson, E. Alba, B. Dorronsoro, and M. Middendorf. Hierarchical cellular genetic algorithm. In *Evolutionary Computation in Combinatorial Optimization EvoCOP06*, pages 111–122, 2006.

[20] N. Krasnogor and J. Smith. Emergence of profitable search strategies based on a simple inheritance mechanism. In *International Genetic and Evolutionary Computation Conference (GECCO'01)*, pages 432–439, San Francisco, CA, 2001. Morgan Kaufmann Publishers Inc.

[21] T. Krink and R. Ursem. Parameter control using the agent based patchwork model. In *Proceedings of the Second Congress on Evolutionary Computation*, pages 77–83, 2000.

[22] D. Lee, S. Baek, and K. Sung. Modified K-means algorithm for vector quantizer design. *IEEE Signal Processing Letters*, 4(1):2–4, January 1997.

[23] C.-C. Lin, S.-C. Chen, and N.-L. Hsueh. Adaptive embedding techniques for VQ-compressed images. *Information Sciences*, 179(1-2):140–149, January 2009.

[24] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. Commun.*, 28(1):84–95, January 1980.

[25] P. Merz and B. Freisleben. A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. In *Proc. Congress on Evolutionary Computation*, pages 2063–2070. IEEE Press, 1999.

[26] P. Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, Pasadena, CA, USA, 1989.

[27] H. Muhlenbein and T. Mahnig. A comparison of stochastic local search and population based search. *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, 1:255–260, May 2002.

[28] Q. H. Nguyen, Y. S. Ong, and M. H. Lim. Non-genetic transmission of memes by diffusion. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1017–1024, New York, NY, USA, 2008. ACM.

[29] Y. S. Ong, M. H. Lim, N. Zhu, , and K.-W. Wong. Classification of adaptive memetic algorithms: A comparative study. *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, 36(1):141–152, 2006.

[30] K. K. Paliwal and V. Ramasubramanian. Comments on "modified k-means algorithm for vector quantizer design". *IEEE IEEE Trans. Image Process.*, 9(11):1964–1967, 2000.

[31] M. Ridley. *Evolution*. Blackwell Publishing, Oxford, 3 edition, 2004.

[32] K. Sasazaki, S. Saga, J. Maeda, and Y. Suzuki. Vector quantization of images with variable block size. *Applied Soft Computing*, 8(1):634–645, 2008.

[33] L. D. Whitley. Cellular genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, page 658, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[34] S. Wright. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In *Proceedings of the 6th International Congress on Genetics*, pages 356–366, 1932.

[35] P. Yahampath and P. Rondeau. Multiple-description predictive-vector quantization with applications to low bit-rate speech coding over networks. *IEEE Trans. Audio, Speech, Language Process.*, 15(3):749–755, March 2007.