A New QEA Computing Near-Optimal Low-Discrepancy Colorings in the Hypergraph of Arithmetic Progressions
(Extended Abstract)

Lasse Kliemann[1], Ole Kliemann[1], C. Patvardhan[2], Volkmar Sauerland[1], and Anand Srivastav[1]

[1] Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Christian-Albrechts-Platz 4
24118 Kiel, Germany
lki@informatik.uni-kiel.de
[2] Dayalbagh Educational Institute
Deemed University
Agra, India
cpatvardhan@googlemail.com

Abstract. We present a new quantum-inspired evolutionary algorithm, the *attractor population QEA* (apQEA). Our benchmark problem is a classical and difficult problem from Combinatorics, namely finding low-discrepancy colorings in the hypergraph of arithmetic progressions on the first n integers, which is a massive hypergraph (e. g., with approx. 3.88×10^{11} hyperedges for $n = 250\,000$). Its optimal low-discrepancy coloring bound $\Theta(\sqrt[4]{n})$ is known and it has been a long-standing open problem to give practically and/or theoretically efficient algorithms. We show that apQEA outperforms known QEA approaches and the classical combinatorial algorithm (Sárközy 1974) by a large margin. Regarding practicability, it is also far superior to the SDP-based polynomial-time algorithm of Bansal (2010), the latter being a breakthrough work from a theoretical point of view. Thus we give the first *practical* algorithm to construct optimal colorings in this hypergraph, up to a constant factor. We hope that our work will spur further applications of Algorithm Engineering to Combinatorics.

Keywords: estimation of distribution algorithm, quantum-inspired evolutionary algorithm, hypergraph coloring, arithmetic progressions, algorithm engineering, combinatorics.

1 Introduction

Experimentation is emerging as a tool in Combinatorics. For example, experimentation is used in a Polymath project on one of the most challenging open

V. Bonifaci et al. (Eds.): SEA 2013, LNCS 7933, pp. 67–78, 2013.

problems of Paul Erdős on homogeneous arithmetic progressions. In this paper we contribute to both, experimental algorithms for difficult discrepancy problems and highly-parallel evolutionary computation within the class of *estimation of distribution algorithms* (EDA).

Quantum-inspired evolutionary algorithms (QEA) belong to the class of EDAs, more precisely to the class of *univariate* EDAs. An EDA maintains a probability distribution, also called *model*, μ on the set of possible solutions, say $\{0,1\}^k$. Sampling μ yields concrete solutions, which can be used to tune μ with the intent to sample better solutions next time. In a univariate EDA, models of a simple kind are considered, namely which treat all of the k coordinates as independent random variables. Thus μ can be represented as a vector $Q = (Q_1, \ldots, Q_k) \in [0,1]^k$ with Q_i stating the probability of sampling 1 in coordinate i. Univariate EDAs have been studied since the 90ies; in 2002 [5], the term "quantum-inspired" was coined, based on the observation that the $Q_1, \ldots, Q_k$ behave similar to k qubits in a quantum computer: each is in a state between 0 and 1, and only upon observation takes on states 0 or 1 with certain probabilities. Hence what we call "sampling" is also called "observing" in the literature. We call the QEA from [5] the *standard QEA* (sQEA). It uses an *attractor*, which is the best solution found so far. The model is tuned towards the attractor in each generation. We stick to the term "quantum-inspired" since our version of univariate EDA also uses the idea of an attractor. A burden that comes with QEAs is the possibility of *premature convergence*, meaning: each Q_i moves close to one of the extremes (0 or 1), so the model Q essentially locks onto one particular solution, before a sufficiently good solution is found – *and* the algorithm does not provide a way to escape this dead end. We will show how our new QEA successfully deals with this problem.

We briefly introduce the hypergraph of arithmetic progressions and the discrepancy problem. Given $a, d, \ell \in \mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$, the set $A_{a,d,\ell} := \{a + id;\ 0 \leqslant i < \ell\}$ is the *arithmetic progression* (AP) with starting point a, difference d, and length ℓ. It contains exactly ℓ numbers, namely $a, a+d, a+2d, \ldots, a+(\ell-1)\,d$. For $n \in \mathbb{N}$ we call $\mathcal{A}_n := \{A_{a,d,\ell} \cap \{0, \ldots, n-1\};\ a, d, \ell \in \mathbb{N}_0\}$ the *set system* or *hypergraph of arithmetic progressions* in the first n integers. Elements of $\mathcal{A}_n$ are called *hyperedges* and elements of the ground set $V := \{0, \ldots, n-1\}$ are called *vertices*. The cardinality of $\mathcal{A}_n$ is approximately $n^2 \log(n)/2$; we will give a proof in the full version. Often, the ground set is $\{1, \ldots, n\}$ in the literature, but for our purposes starting at 0 is notationally more convenient. A *coloring* is a mapping $\chi : V \longrightarrow \{-1, +1\}$. Given a coloring χ and an AP $E \subseteq V$ we have its *discrepancy* $\operatorname{disc}_\chi(E) := \left|\sum_{v \in E} \chi(v)\right|$. The discrepancy of $\mathcal{A}_n$ with respect to χ is $\operatorname{disc}_\chi(\mathcal{A}_n) := \max_{E \in \mathcal{A}_n} \chi(E)$.

Previous and Related Work. Univariate EDAs have been studied since the 90ies, see, e. g.,, [2,13,7,5,15]. For a recent survey on general EDAs see [8] and the references therein. Particularly influential for our work have been [5] and [15], where sQEA and vQEA are presented, respectively. vQEA extends the attractor concept in a way to allow for better exploration. But for the discrepancy problem, vQEA is not well suited for reasons explained later. In recent years, variants of

QEAs have been successfully used on benchmark as well as on difficult practice problems, see, e. g.,, [1,10,11,14,6].

For $\mathcal{A}_n$, in 1964, it was shown by Roth [16] that there is no coloring with discrepancy below $\Omega(\sqrt[4]{n})$. More than 20 years later, it was shown by Matoušek and Spencer [12] that there exists a coloring with discrepancy $O(\sqrt[4]{n})$, so together with the earlier result we have $\Theta(\sqrt[4]{n})$. The proof is non-constructive, and the problem of efficiently computing such colorings remained open. For many years, Sárközy's approximation algorithm (see [4]) was the best known, provably attaining discrepancy $O(\sqrt[3]{n \log(n)})$; experiments suggest that (asymptotically) it does not perform better than this guarantee. Recently, in a pioneering work, the problem was solved by Bansal [3], using semi-definite programs (SDP). However, Bansal's algorithm requires solving a series of SDPs that grow in the number of hyperedges, making it practically problematic for $\mathcal{A}_n$. In our experiments, even for $n < 100$, it requires several hours to complete, whereas our apQEA (in a parallel implementation) only requires a couple of minutes up to $n = 100\,000$. Computing optimal low-discrepancy colorings for *general* hypergraphs is NP-hard [9].

Our Contribution. We use the problem of computing low-discrepancy colorings in $\mathcal{A}_n$ in order to show limitations of sQEA and how a new form of QEA can successfully overcome these limitations. Our new QEA uses an *attractor population* where the actual attractor is repeatedly selected from. We call it *attractor population QEA* (apQEA). The drawback of sQEA appears to be premature convergence, or in other words, a lack of exploration of the search space. We show that even by reducing the learning rate drastically in sQEA and by using local and global migration, it does not attain the speed and solution quality of apQEA. In addition to the exploration capabilities of apQEA, we show that – with an appropriate tuning of parameters – it scales well in the number of parallel processors: when doubling the number of processor cores from 96 to 192, running times reduces to roughly between 40% and 60%.

We also look at the combinatorial structure of $\mathcal{A}_n$. Based on an idea by Sárközy (see [4]) we devise a modulo coloring scheme, resulting in a search space reduction and faster fitness function evaluation. This, together with apQEA, allows us to compute low-discrepancy colorings that are optimal up to a constant factor, in the range up to $n = 250\,000$ vertices. For this n, the cardinality of $\mathcal{A}_n$ is approx. 3.88×10^{11}, which means a massive hypergraph. Precisely, we compute colorings with discrepancy not more than $3\sqrt[4]{n}$; we call $\lfloor 3\sqrt[4]{n} \rfloor$ the *target discrepancy*. We have chosen factor 3, because this appeared as an attainable goal in reasonable time in preliminary experiments. Better approximations may be possible with other parameters and more processors and/or more time. Colorings found by our algorithm can be downloaded[1] and easily verified.

Our problem sizes are a magnitude beyond that of the Polymath project. Of course, our problem is different, but related and in future work we plan to access the Erdős problem with our approach.

[1] `http://www.informatik.uni-kiel.de/~lki/discap-results.tar.xz`

Algorithm 1. sQEA

in parallel for each *model* $Q = (Q_1, \ldots, Q_k) \in \mathcal{M}$ **do**
  initialize model $Q := (1/2, \ldots, 1/2)$;
  initialize attractor $a :=$ random solution;
  repeat
    $a :=$ best attractor over all models;
    sample Q yielding $x \in \{0,1\}^k$;
    if $f(x) \leqslant f(a)$ **then**
      for $i = 1, \ldots, k$ **do if** $x_i \neq a_i$ **then**

$$Q_i := \begin{cases} \max\{0,\, Q_i - \Delta(Q_i)\} & \text{if } a_i = 0 \\ \min\{1,\, Q_i + \Delta(Q_i)\} & \text{if } a_i = 1 \end{cases}$$

    else $a := x$;
  until *satisfied or hopeless or out of time*;

2 Description of Algorithms

Fitness Function and Shortcutting. As fitness function (FF), we use the negative of the discrepancy, so higher fitness is better. The sample space is of the form $\{-1,+1\}^k$, but we will often write $\{0,1\}^k$, where 0 means -1. The concrete choice of k will be explained in Sec. 3. In a QEA, given two solutions x and x^* with known fitness $f(x^*)$, it is often enough to decide whether $f(x) > f(x^*)$ and only in this case it will be required to compute $f(x)$ exactly. If we can determine that $f(x) \leqslant f(x^*)$, then we do not need the exact value of $f(x)$. Since discrepancy involves a maximum, it provides an opportunity for *shortcutting*: as soon as a hyperedge is found in which discrepancy w.r.t. x is at least as high as $\mathrm{disc}(x^*)$, evaluation can be aborted and $f(x) \leqslant f(x^*)$ can be reported. This is a big time-saver, e. g., for $n = 100\,000$ vertices, on average we require about 2 milliseconds for a shortcut FF evaluation and about 790 milliseconds for a full one – and for apQEA there are usually many more shortcut ones than full ones.

Standard QEA (sQEA). A basic version of sQEA [5] is given as Alg. 1. The set $\mathcal{M}$ typically comprises 1 to 100 models; they are distributed among the available processors. For each model, an *attractor* a is maintained. Each iteration of the repeat loop is called a *generation*. In each generation, each of the models is sampled, and if the sample x cannot beat the attractor,[2] *learning* takes place: the model is shifted slightly towards a, where x and a differ. *Linear learning* means using a fixed amount, e. g.,, $\Delta(Q_i) = \Delta = \frac{1}{100}$. In [5,15] *rotation learning* is used: the point $(\sqrt{1-Q_i}, \sqrt{Q_i})$ in the plane is rotated either clockwise (if $a_i = 0$) or counter-clockwise (if $a_i = 1$), and the new value of Q_i becomes the square root of

[2] The original description suggests using $f(x) < f(a)$ as the test, so the sample is not required to beat the attractor but to be at least as good as the attractor. We will comment on this later.

the new ordinate. This is inspired by quantum computing; an actual benefit could be that towards the extremes (0 and 1) shifts become smaller. We will test sQEA and vQEA with linear as well as rotation learning and stick to linear learning for apQEA. The *learning resolution* gives the number of possible values that each Q_i can assume inside $[0, 1]$. For linear learning, this is $\frac{1}{\Delta}$. For rotation learning, this is determined by the angle by which we rotate; it is typically between 0.01π and 0.001π. Since the interval is $[0, \pi/2]$, this means a learning resolution between 50 and 500. As an extension, multiple samples can be taken in line 6 and the model is only shifted if none of them beats the attractor (an arbitrary one of them is chosen for the test $x_i \neq a_i$). If one of the samples beats the attractor, the best one is used to update the attractor in line 9. We always use 10 samples.

What happens in line 5 is called *synchronization* or *migration*. Another extension, intended to prevent premature convergence, is the use of *local* and *global migration*. Models are bundled into *groups*, and the attractor of a model Q is set to the best attractor over all models in Q's group (local migration). Only every T_g generations, the best attractor over *all* models is used (global migration). We call T_g the *global migration period*.

The repeat loop stops when we are "satisfied or hopeless or out of time". We are satisfied in the discrepancy problem when the discrepancy is lesser or equal to $3\sqrt[4]{n}$. A possible criterion for hopelessness is when all the models have only very little entropy left. *Entropy* is a measure of randomness, defined as $\sum_{i=1}^{k} -\log(Q_i)$, which is at its maximum k when $Q_i = \frac{1}{2}$ for all i, and at its minimum 0 if $Q_i \in \{0, 1\}$ for all i. In all our experiments with sQEA, we will impose a simple *time limit* guided by the times needed by apQEA.

Versatile QEA (vQEA). vQEA [15] works similar to sQEA with the exception that the attractor update in line 9 is carried out *unconditionally*. The description given in [15] states that the attractor of each model in generation $t+1$ is the best sample from generation t, over all models. This means that parallel processes have to synchronize after each generation. and also that at least one sample per generation must be fully evaluated, so only limited use of shortcutting is possible.

Attractor Population QEA (apQEA). Our new QEA, the apQEA, is given as Alg. 2. It strikes a balance between the approaches of sQEA and vQEA. In sQEA, the attractor essentially follows the best solution and only changes when better solutions are found, while in vQEA the attractor changes frequently and is also allowed to assume inferior solutions. In apQEA, the *attractor population* $\mathcal{P}$ is a set of solutions. From it, attractors are selected, e. g., using tournament selection. When a sample cannot improve the population ($f(x) \leqslant f_0$) the model is adjusted. Otherwise the solution is injected into the population. The number of generations that a particular attractor stays in function is called the *attractor persistence*; we fix it to 10 in all our experiments. Note that apQEA will benefit from shortcutting since $f(x)$ has only to be computed when $f(x) > f_0$. Note also that it is appropriate to treat the models *asynchronously* in apQEA, hence preventing idle time: the attractor population is there, any process may inject into it or select from it at any time (given an appropriate implementation).

Algorithm 2. apQEA

randomly initialize attractor population $\mathcal{P} \subseteq \{0,1\}^k$ of cardinality S;
in parallel for each *model* $Q = (Q_1, \ldots, Q_k) \in \mathcal{M}$ **do**
  initialize model $Q := (1/2, \ldots, 1/2)$;
  repeat
    $a :=$ select from $\mathcal{P}$;
    do *10* **times**
      $f_0 :=$ worst fitness in $\mathcal{P}$;
      sample Q yielding $x \in \{0,1\}^k$;
      if $f(x) \leqslant f_0$ **then**
        for $i = 1, \ldots, k$ **do if** $x_i \neq a_i$ **then**

$$Q_i := \begin{cases} \max\{0,\, Q_i - \Delta(Q_i)\} & \text{if } a_i = 0 \\ \min\{1,\, Q_i + \Delta(Q_i)\} & \text{if } a_i = 1 \end{cases}$$

      else
        inject x into $\mathcal{P}$;
        trim $\mathcal{P}$ to the size of S, removing worst solutions;
  until *satisfied or hopeless or out of time*;

A very important parameter is the size S of the population. We will see in experiments that larger S means better exploration abilities. For the discrepancy problem, we will have to increase S (moderately) when n increases.

Since the attractor changes often in apQEA, entropy oftentimes never reaches near zero but instead oscillates around values like 20 or 30. A more stable measure is the mean Hamming distance in the attractor population, i. e., $\frac{1}{\binom{S}{2}} \cdot \sum_{\{x,x'\} \in \binom{\mathcal{P}}{2}} |\{i;\, x_i \neq x'_i\}|$. However, it also can get stuck well above zero. To determine a hopeless situation, we instead developed the concept of a *flatline*. A flatline is a period of time in which neither the mean Hamming distance reaches a new minimum nor a better solution is found. When we encounter a flatline stretching over 25% of the total running time so far, we declare the situation hopeless. To avoid erroneously aborting in early stages, we additionally demand that the relative mean Hamming distance, which is the mean Hamming distance divided by k, falls below 1/10. Those thresholds were found to be appropriate (for the discrepancy problem) in preliminary experiments.

3 Modulo Coloring

Let $p \leqslant n$ be an integer. Given a partial coloring $\chi' : \{0, \ldots, p-1\} \longrightarrow \{-1, +1\}$, i. e., a coloring of the first p vertices, we can construct a coloring χ by repeating χ', i. e., $\chi : V \longrightarrow \{-1, +1\}$, $v \mapsto \chi'(v \bmod p)$. We call χ' a *generating coloring*. This way of coloring, with an appropriate p, brings many benefits. Denote $E_p := A_{0,1,p} = \{0, \ldots, p-1\}$, this is an AP and also the whole set on which χ' lives.

Assume that χ' is *balanced*, i. e., $\mathrm{disc}_{\chi'}(E_p) \leqslant 1$. Let $A_{a,d,\ell}$ be any AP and $\ell = qp + r$ with integers q, r and $r < p$. Then we have the decomposition:

$$A_{a,d,\ell} = \dot{\bigcup_{i=0}^{q-1}} \underbrace{A_{a+ipd,d,p}}_{B_i:=} \cup \underbrace{A_{a+qpd,d,r}}_{B_q:=} \ . \tag{1}$$

Assuming p is prime, we have $B_i \bmod p := \{v \bmod p;\ v \in B_i\} = E_p$ for each $i = 0, \ldots, q-1$, so $\mathrm{disc}_\chi(B_i) = \mathrm{disc}_{\chi'}(E_p) \leqslant 1$. It follows $\mathrm{disc}_\chi(A_{a,d,\ell}) \leqslant q\,\mathrm{disc}_{\chi'}(E_p) + \mathrm{disc}_\chi(B_q) \leqslant q + \mathrm{disc}_\chi(B_q)$. This is one of the essential ideas how Sárközy's $O(\sqrt[3]{n\log(n)})$ bound is proved and it gives us a hint (which was confirmed in experiments) that modulo colorings, constructed from balanced ones, might tend to have low discrepancy. It is tempting to choose p very small, but the best discrepancy we can hope for when coloring modulo p is $\lceil n/p \rceil$. Since we aim for $3\sqrt[4]{n}$, we choose p as a prime number so that $\lceil n/p \rceil$ is some way below $3\sqrt[4]{n}$, precisely we choose p prime with $n/p \approx 5/2 \cdot \sqrt[4]{n}$, i. e., $p \approx 2/5 \cdot n^{3/4}$.

Constructing balanced colorings is straightforward. Define $h := \frac{p+1}{2}$ (so $h = \Theta(n^{3/4})$) and let $x \in \{-1, +1\}^h$. Then $(x_1, \ldots, x_{h-1}, -x_{h-1}, \ldots, -x_1, x_h)$ defines a balanced coloring of E_p. We could have chosen different ways of ordering the entries of x and their negatives, but this *mirroring* construction has shown to work best so far. We additionally alternate the last entry x_h, so we use the following generating coloring of length $2p$:

$$(x_1, \ldots, x_{h-1}, -x_{h-1}, \ldots, -x_1, x_h, x_1, \ldots, x_{h-1}, -x_{h-1}, \ldots, -x_1, -x_h) \ . \tag{2}$$

Modulo coloring has further benefits. It reduces the search space from $\{-1, +1\}^n$ to $\{-1, +1\}^h$, where $h = \Theta(n^{3/4})$. Moreover, it allows a much faster FF evaluation: we can restrict to those $A_{a,d,\ell}$ with $a \leqslant 2p - 1$. We also make use of a decomposition similar to (1), but which is more complicated since we exploit the structure of (2); details will be given in the full version. We omit APs which are too short to bring discrepancy above the target, giving additional speedup.

4 Experiments and Results

Implementation and Setup. To fully benefit from the features of apQEA, we needed an implementation which allows *asynchronous* communication between processes. Our MPI-based implementations (version 1.2.4) exhibited unacceptable idle times when used for asynchronous communication, so we wrote our own client-server-based parallel framework. It consists of a server process that manages the attractor population. Clients can connect to it at any time via TCP/IP and do selection and injection; the server takes care of trimming the population after injection. Great care was put into making the implementation free of race conditions. Most parts of the software is written in Bigloo[3], an implementation of the Scheme programming language. The FF and a few other parts are written in C, for performance reasons and to have OpenMP available. OpenMP is used to distribute FF evaluation across multiple processor cores. So we have a

[3] `http://www-sop.inria.fr/indes/fp/Bigloo/`, version 3.9b-alpha29Nov12

two-level parallelization: on the higher level, we have multiple processes treating multiple models and communicating via the attractor population server. On the lower level, we have thread parallelization for the FF. The framework provides also means to run sQEA and vQEA. We always use 8 threads (on 8 processor cores) for the FF. If not stated otherwise, a total of 96 processor cores is used. This allows us to have 12 models fully in parallel; if we use more models then the set of models is partitioned and the models from each partition are treated sequentially. Experiments are carried out on the NEC$^{\text{TM}}$ Linux Cluster of the Rechenzentrum at Kiel University, with SandyBridge-EP$^{\text{TM}}$ processors.

Results for sQEA. Recall the important parameters of sQEA: number of models M, learning resolution R, global migration period T_g, and number of groups. For R, we use 50, 100 and 500, which are common settings, and also try 1000, 2000, and 3000. In [6], it is proposed to choose T_g in linear dependence on R, which in our notation and neglecting the small additive constant reads $T_g = 2R\lambda$ with $1.15 \leqslant \lambda \leqslant 1.34$. We use $\lambda = 1.25$ and $\lambda = 1.5$, so $T_g = 2.5R$ and $T_g = 3R$. In [6], the number of groups is fixed to 5 and the number of models ranges up to 100. We use 6 groups and up to 96 models. We use rotation learning, but made similar observations with linear learning.

We fix $n = 100\,000$ and do 3 runs for each set of parameters. Computation is aborted after 15 minutes, which is roughly double the time apQEA needs to reach the target discrepancy of 53. For sQEA as given in Alg. 1 best discrepancy we reach is 57. We get better results when using $f(x) < f(a)$ as the test in line 7, i. e., we also accept a sample that is as good as the attractor and not require that it is strictly better.[4] The following table gives mean discrepancies for this variant. The left number is for smaller T_g and the right for higher T_g, e. g., for $M = 12$ and $R = 50$ we have 60 for $T_g = 2.5 \cdot 50 = 125$ and 61 for $T_g = 3 \cdot 50 = 150$.

R	$M = 12$		$M = 24$		$M = 48$		$M = 96$	
50	60	61	59	59	58	57	58	58
100	59	59	57	59	59	56	56	56
500	57	56	57	57	55	56	56	57
1000	57	57	56	55	57	56	58	59
2000	57	57	57	58	59	60	63	62
3000	56	57	59	58	61	61	64	65

Target discrepancy 53 is never reached. For two runs we reach 54, namely for $(M, R, T_g) = (24, 1000, 3000)$ and $(96, 500, 1250)$. But for each of the 2 settings, only 1 of the 3 runs reached 54. There is no clear indication whether smaller or larger T_g is better. Entropy left in the end generally increases with M and R.

We pick the setting $(M, R, T_g) = (24, 1000, 3000)$, which attained 54 in 1 run and also has lowest mean value of 55, for a 5 hour run. In the 15 minutes runs with this setting, entropy in the end was 48 on average. What happens if we let the algorithm use up more of its entropy? As it turns out while entropy is brought down to 16 during the 5 hours, only discrepancy 56 is attained.

The main problem with sQEA here is that there is no clear indication which parameter to tune in order to get higher quality solutions – at least not within

[4] Using $f(x) < f_0$ in apQEA however has shown to be not beneficial.

reasonable time (compared to what apQEA can do). In preliminary experiments, we reached target discrepancy 53 on some occasions, but with long running times. We found no way to *reliably* reach discrepancy 55 or better with sQEA.

Finally, for $(M, R, T_g) = (24, 1000, 3000)$, we do experiments for up to $n = 200\,000$, with 3 runs for each n. The time limit is twice what apQEA requires on average, rounded to the next multiple of 5. The following table for each n gives the best result obtained over the 3 runs and the target for comparison.

$n =$	100 000	125 000	150 000	175 000	200 000
time limit in minutes	15	25	40	80	150
best sQEA result	54	60	63	68	69
target	53	56	59	61	63

Although we do multiple runs and allocate twice the time apQEA would need to attain the target, the best result for sQEA stays clearly away from the target.

Results for vQEA. Recall that vQEA in generation $t+1$ unconditionally replaces the attractor for each model with the best sample found during generation t. vQEA does not use a groups and global migration period, instead all models form a single group "to ensure convergence" [15]. Indeed, our experiments confirm that vQEA has no problem with running out of entropy. We conducted 5 runs with $R = 50$ and rotation learning for $n = 100\,000$. In all of the runs, the target of 53 was hit with about 100 of entropy left. However, the time required was almost 2 hours.[5] We also conducted 5 runs with linear learning, yielding the same solution quality at a 12% higher running time. We also did a run for $n = 125\,000$; there vQEA attained the target discrepancy after 3.5 hours.

The high running times were to be expected since vQEA can only make limited use of shortcutting. Since we take multiple samples in each generation (10 for each model), FF evaluation from the second sample on can make use of shortcutting. However, necessarily each generation takes at least the time of one full FF evaluation. We conclude that while vQEA has impressive exploration capabilities and delivers high solution quality "out of the box", i. e., without any particular parameter tuning, it is not well suited for the discrepancy problem.

Results for apQEA. Recall that the most important parameter for apQEA is the attractor population size S. We fix $R = 100$ with linear learning and $M = 12$ and vary S in steps of 10. Computation is aborted when the target of $3\lfloor\sqrt[4]{n}\rfloor$ is hit (a *success*) or a long flatline is observed (a *failure*), as explained in Sec. 2. For selecting attractors, we use tournament selection: draw two solutions randomly from $\mathcal{P}$ and use the better one with 60% probability and the inferior one with 40% probability (higher selection pressures appear to not help, this will be discussed in the full version). For each n and appropriate choices of S, we do 30 runs and record the following: whether it is a success or a failure, final discrepancy (equals target discrepancy for successes), running time (in minutes), mean final entropy.

[5] Even more, these 2 hours is only the time spent in FF evaluation. Total time was about 4 hours, but we suspect this to be partly due to our implementation being not particularly suited for vQEA resulting in communication overhead.

For failures, we also record the time at which the last discrepancy improvement took place (for successes, this value is equal to the running time). The following table gives results grouped into successes and failures, all numbers are mean values over the 30 runs.

		successes				failures				
$\frac{n}{1000}$	S	#	disc	time	entropy	#	disc	time	entropy	last imp.
100	20	28	$53_{\sigma=00}$	$05_{\sigma=01}$	$38_{\sigma=09}$	02	$54_{\sigma=00}$	$08_{\sigma=01}$	$22_{\sigma=01}$	$06_{\sigma=01}$
100	30	30	$53_{\sigma=00}$	$07_{\sigma=01}$	$63_{\sigma=12}$	00	na	na	na	na
125	20	17	$56_{\sigma=00}$	$09_{\sigma=02}$	$31_{\sigma=08}$	13	$57_{\sigma=00}$	$12_{\sigma=01}$	$19_{\sigma=09}$	$08_{\sigma=01}$
125	30	30	$56_{\sigma=00}$	$11_{\sigma=01}$	$48_{\sigma=11}$	00	na	na	na	na
150	30	26	$59_{\sigma=00}$	$16_{\sigma=02}$	$46_{\sigma=11}$	04	$60_{\sigma=00}$	$23_{\sigma=02}$	$32_{\sigma=09}$	$16_{\sigma=02}$
150	40	30	$59_{\sigma=00}$	$20_{\sigma=02}$	$62_{\sigma=10}$	00	na	na	na	na
175	30	16	$61_{\sigma=00}$	$24_{\sigma=04}$	$31_{\sigma=08}$	14	$62_{\sigma=00}$	$38_{\sigma=05}$	$21_{\sigma=08}$	$24_{\sigma=03}$
175	40	27	$61_{\sigma=00}$	$32_{\sigma=05}$	$42_{\sigma=09}$	03	$62_{\sigma=00}$	$47_{\sigma=02}$	$26_{\sigma=05}$	$30_{\sigma=01}$
175	50	30	$61_{\sigma=00}$	$39_{\sigma=05}$	$57_{\sigma=11}$	00	na	na	na	na
200	30	02	$63_{\sigma=00}$	$38_{\sigma=02}$	$22_{\sigma=06}$	28	$65_{\sigma=01}$	$47_{\sigma=08}$	$24_{\sigma=17}$	$30_{\sigma=05}$
200	50	28	$63_{\sigma=00}$	$53_{\sigma=08}$	$44_{\sigma=08}$	02	$64_{\sigma=00}$	$76_{\sigma=06}$	$28_{\sigma=02}$	$53_{\sigma=03}$
200	60	30	$63_{\sigma=00}$	$74_{\sigma=15}$	$53_{\sigma=12}$	00	na	na	na	na

We observe that by increasing S, we can guarantee the target to be hit. Dependence of S on n for freeness of failure appears to be approx. linear or slightly super-linear; ratios of S to $n/1000$ for no failures are 0.30, 0.24, 0.27, 0.29, and 0.30. But even if S is one step below the required size, discrepancy is only 1 away from the target (with an exception for $n = 125\,000$ and $S = 20$, where we recorded discrepancy 58 in 1 of the 30 runs). Running times for failures tend to be longer than for successes, even if the failure is for a smaller S. This is because it takes some time to detect a failure by the flatline criterion. Larger S effects larger entropy; failures tend to have lowest entropy, indicating that the problem is the models having locked onto an inferior solution. The table also shows what happens if we do *lazy S tuning*, i. e., fixing S to the first successful value $S = 30$ and then increasing n: failure rate increases and solution quality for failures deteriorates moderately. The largest difference to the target is observed for $n = 200\,000$ and $S = 30$, namely we got discrepancy 67 in 1 of the 30 runs; target is 63. For comparison, the best discrepancy we found via sQEA in 3 runs for such n was 69 and the worst was 71. We conclude that a mistuned S does not necessarily have catastrophic implications, and apQEA can still beat sQEA.

Convergence. We plot (Fig. 1) discrepancy over time for 2 runs: the first hour of the 5-hour sQEA run with $(M, R, T_g) = (24, 1000, 3000)$; and 1 for apQEA with $S = 30$, which is kept running after the target was hit (until the flatline criterion leads to termination). sQEA is shown with a dashed line and apQEA with a solid line. In the first minutes, sQEA brings discrepancy down faster, but is soon overtaken by apQEA (which reaches 51 in 10 minutes, target is 53).

Effect of Parallelization. We double number of cores from 96 to 192 and increase number of models to $M = 24$, so that they can be treated in parallel with 8 cores each. The following table shows results for 5 runs for each set of parameters. First consider $n = 175\,000$ and $200\,000$. The best failure-free settings for S are 30 and 50.

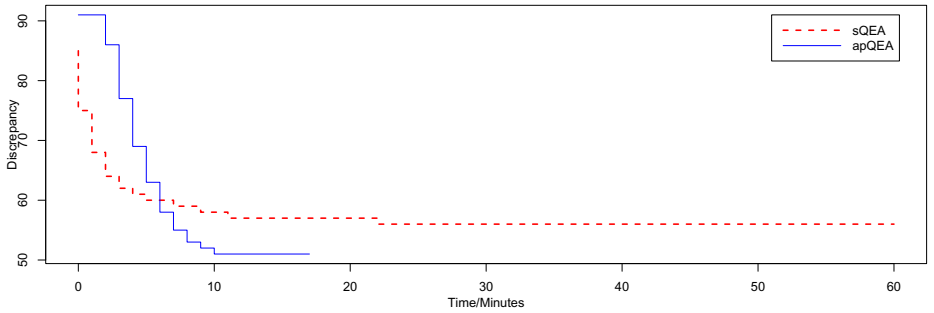

Fig. 1. Discrepancy plotted over time for sQEA and apQEA

In comparison with the best failure-free settings for 96 cores, time is reduced to $17/39 = 44\%$ and $42/74 = 57\%$, respectively. 50% or less would mean a perfect scaling. When we do not adjust S, i. e., we use 50 and 60, time is reduced to $26/39 = 67\%$ and $53/74 = 72\%$, respectively. We also compute for $n = 250\,000$.

		successes				failures				
$\frac{n}{1000}$	S	#	disc	time	entropy	#	disc	time	entropy	last imp.
175	20	02	$61_{\sigma=00}$	$14_{\sigma=02}$	$22_{\sigma=02}$	03	$62_{\sigma=00}$	$20_{\sigma=02}$	$31_{\sigma=12}$	$13_{\sigma=00}$
175	30	05	$61_{\sigma=00}$	$17_{\sigma=04}$	$43_{\sigma=07}$	00	na	na	na	na
175	50	05	$61_{\sigma=00}$	$26_{\sigma=03}$	$72_{\sigma=12}$	00	na	na	na	na
200	40	04	$63_{\sigma=00}$	$30_{\sigma=03}$	$42_{\sigma=12}$	01	$64_{\sigma=00}$	$42_{\sigma=00}$	$32_{\sigma=02}$	$28_{\sigma=00}$
200	50	05	$63_{\sigma=00}$	$42_{\sigma=10}$	$52_{\sigma=05}$	00	na	na	na	na
200	60	05	$63_{\sigma=00}$	$53_{\sigma=08}$	$60_{\sigma=11}$	00	na	na	na	na
250	50	04	$67_{\sigma=00}$	$58_{\sigma=07}$	$42_{\sigma=08}$	01	$68_{\sigma=00}$	$110_{\sigma=00}$	$29_{\sigma=02}$	$70_{\sigma=00}$
250	60	05	$67_{\sigma=00}$	$83_{\sigma=16}$	$61_{\sigma=14}$	00	na	na	na	na

5 Conclusion and Current Work

We have seen apQEA outperforming sQEA in terms of speed and solution quality by a large margin on the discrepancy problem. For this problem, apQEA is easy to tune, since a single parameter, the size S of the attractor population, has a clear and foreseeable effect: it improves solution quality (if possible) at the price of an acceptable increase in running time. vQEA has shown that it is possible to achieve the same solution quality as apQEA without parameter tuning, at the price of an enormous running time and inter-process communication overhead. It may be possible to have the best of both worlds in one algorithm, i. e., to get rid of the S parameter in apQEA. Moreover we believe that it should be attempted to mathematically analyze why vQEA and apQEA succeed where sQEA fails. We also plan new applications of apQEA in the vast area of coloring of (hyper)graphs and other combinatorial problems. Concerning arithmetic progressions, we will investigate further ways to speed up the FF by exploiting combinatorial structures.

Acknowledgements. We thank the Deutsche Forschungsgemeinschaft (DFG) for Grants Sr7/12-3 (SPP "Algorithm Engineering") and Sr7/14-1 (binational workshop). We thank Nikhil Bansal for interesting discussions.

References

1. Babu, G.S., Das, D.B., Patvardhan, C.: Solution of real-parameter optimization problems using novel quantum evolutionary algorithm with applications in power dispatch. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2009, Trondheim, Norway, pp. 1927–1920 (May 2009)
2. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie Mellon University, Pittsburgh, PA (1994)
3. Bansal, N.: Constructive algorithms for discrepancy minimization. In: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, Las Vegas, Nevada, USA, pp. 3–10 (October 2010)
4. Erdős, P., Spencer, J.: Probabilistic Methods in Combinatorics. Akadémia Kiadó, Budapest (1974)
5. Han, K.H., Kim, J.H.: Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Transactions on Evolutionary Computation 6(6), 580–593 (2002)
6. Han, K.H., Kim, J.H.: On setting the parameters of quantum-inspired evolutionary algorithm for practical applications. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2003, Canberra, Australia, pp. 178–184 (December 2003)
7. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. Technical report, Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory (1997)
8. Hauschild, M., Pelikan, M.: An introduction and survey of estimation of distribution algorithms (2011)
9. Knieper, P.: The Discrepancy of Arithmetic Progressions. PhD thesis, Institut für Informatik, Humboldt-Universität zu Berlin (1997)
10. Mani, A., Patvardhan, C.: An adaptive quantum inspired evolutionary algorithm with two populations for engineering optimization problems. In: Proceedings of the International Conference on Applied Systems Research, NSC 2009, Dayalbagh Educational Instute, Agra, India (2009)
11. Mani, A., Patvardhan, C.: A hybrid quantum evolutionary algorithm for solving engineering optimization problems. International Journal of Hybrid Intelligent Systems 7, 225–235 (2010)
12. Matoušek, J., Spencer, J.: Discrepancy in arithmetic progressions. Journal of the American Mathematical Society 9, 195–204 (1996)
13. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. binary parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
14. Patvardhan, C., Prakash, P., Srivastav, A.: A novel quantum-inspired evolutionary algorithm for the quadratic knapsack problem. In: Proceedings of the International Conference on Operations Research Applications in Engineering and Management, ICOREM 2009, Tiruchirappalli, India, pp. 2061–2064 (May 2009)
15. Platel, M.D., Schliebs, S., Kasabov, N.: A versatile quantum-inspired evolutionary algorithm. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, Singapore, pp. 423–430 (September 2007)
16. Roth, K.F.: Remark concerning integer sequences. Acta Arithmetica 9, 257–260 (1964)