International Journal of Automation and Computing 04(3), July 2007, 273-280
DOI: 10.1007/s11633-007-0273-3

Combinations of Estimation of Distribution Algorithms and Other Techniques

Qingfu Zhang[1,*] Jianyong Sun[2] Edward Tsang[1]
[1] Department of Computer Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK
[2] School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK

Abstract: This paper summaries our recent work on combining estimation of distribution algorithms (EDA) and other techniques for solving hard search and optimization problems: a) guided mutation, an offspring generator in which the ideas from EDAs and genetic algorithms are combined together, we have shown that an evolutionary algorithm with guided mutation outperforms the best GA for the maximum clique problem, b) evolutionary algorithms refining a heuristic, we advocate a strategy for solving a hard optimization problem with complicated data structure, and c) combination of two different local search techniques and EDA for numerical global optimization problems, its basic idea is that not all the new generated points are needed to be improved by an expensive local search.

Keywords: Estimation distribution algorithm, guided mutation, memetic algorithms, global optimization.

1 Introduction

Estimation of distribution algorithms (EDAs) have been recognized as a major paradigm in evolutionary computation. There is no traditional crossover or mutation in EDAs. Instead, they explicitly extract global statistical information from the selected solutions (often called parents) and build a posterior probability distribution model of promising solutions, based on the extracted information. New solutions are sampled from the model thus built and fully or in part replace the old population. Since the dependence relationships in the distribution of the promising solutions are highly relevant to the variable interactions in the problem, EDAs are promising methods for capturing the structure of variable interactions, identifying and manipulating crucial building blocks, and hence efficiently solving hard optimization and search problems with interactions among the variables. Many EDA-like algorithms have been developed for various optimization and search problems in recent years. Instances of EDAs include population-based incremental learning (PBIL)[1], univariate marginal distribution algorithm (UMDA)[2], mutual information maximization for input clustering (MIMIC)[3], combining optimizers with mutual information trees (COMIT)[4], factorized distribution algorithm (FDA)[5], Bayesian optimization algorithm (BOA)[6], Bayesian evolutionary algorithm (BEA)[7], and global search based on reinforcement learning agents (GSBRL)[8], to name a few.

We have been working on the theory and application of EDAs since 2000. One of our main focuses is to hybridize EDAs with other techniques and design efficient optimization algorithms. The purpose of this paper is to summarize some of our work on the design of practical EDAs. Section 2 describes a new offspring generator which can be regarded as a combination of the conventional mutation operator and the EDA offspring generating scheme. Section 3 describes a hybrid algorithm with guided mutation for the maximum clique problem. Section 4 introduces our recent work on refining a heuristic by using EDAs. Section 5 presents a method for combining EDAs with local searches for global continuous optimization. Section 6 concludes this paper.

2 Guided mutation

One of the key issues in the design of evolutionary algorithms (EAs) is how to generate offspring. The proximate optimality principle[9], an underlying assumption in most (if not all) heuristics, assumes that good solutions have similar structure. This assumption is reasonable for most real-world problems, e.g., the percentage of common edges in any two locally optimal solutions of a traveling salesman problem obtained by the Lin-Kernighan method is about 85% on average[10]. Based on this assumption, an ideal offspring generator should be able to produce a solution which is close to the best solutions found so far. Suppose the current population in an evolutionary algorithm with local search consists of the best locally optimal solutions found so far, a new solution generated by the conventional mutation is close (similar) to its parent, but may be far away from other better solutions since the mutation does not utilize any global information extracted from the current population. EDAs extract global statistical information from the previous search and then represent it as a probability model, which characterizes the distribution of promising solutions in the search space. New solutions are generated by sampling from this model. However, the location information of the locally optimal solutions found so far has not been directly used in EDAs, there is no mechanism to directly control the similarity between new solutions and a given solution. The idea behind the proposed operator which we call guided mutation is to combine the global statistical information and location information of the solutions found so far to overcome the shortcoming of GAs and EDAs. In the following, we present a version of guided mutation for the binary search space.

Let the search space be $\Omega = \{0,1\}^n$, a probability

Manuscript received date March 5, 2007; revised date May 23, 2007
*Corresponding author. E-mail address: qzhang@essex.ac.uk

vector $p = (p_1, \cdots, p_n) \in [0,1]^n$ is used to characterize the distribution of promising solutions in the search space, where p_i is the probability that the value of the i-th position of a promising solution is one. The guided mutation operator uses a probability vector $p \in [0,1]^n$ to guide to mutate an $x \in \{0,1\}^n$ in the following way:

```
Guided mutation Operator $y = GM(p, x, \beta)$
Input: $p = (p_1, \cdots, p_n) \in [0,1]^n, x = (x_1, \cdots x_n) \in \{0,1\}^n$ and $\beta \in [0,1]$.
Output: $y = (y_1, \cdots, y_n) \in \{0,1\}^n$.
For $i := 1$ to $n$ do
    Flip a coin with head probability $\beta$;
    If the head turns up, with probability $p_i$ set $y_i = 1$, otherwise set $y_i = 0$;
    Otherwise, $y_i := x_i$.
End For
```

Remark 1. In the above guided mutation operator, y_i is directly copied from the parent x or randomly sampled from the probability vector p. The larger β is, the more elements of y are sampled from the probability vector p. In other words, β, similar to the mutation rate in conventional mutation, controls the similarity between offspring and the parent, while the parent can be chosen from the best solutions found so far.

Remark 2. In the correlated mutation[11] for real vectors, the probability of generating an offspring in the steepest ascent direction is larger than in other directions. In the conventional mutation for binary strings, the probability of a vector y being generated from the parent vector x is entirely determined by the Hamming distance between x and y. The guided mutation operator can be regarded as a discrete counterpart of the correlated mutation. The probability vector p in the guided mutation can be learned and updated at each generation for modeling the distribution of promising solutions. Since some elements of the offspring y are sampled from the probability vector p, it can be expected that y should fall in or close to a promising area. Meanwhile, this sampling also provides diversity for the search afterwards.

Remark 3. Variable linkages are not taken into account in the probability model in the above version of guided mutation. It is trivial to use other complicated probability models in guided mutation. However, we do not advocate to do so. Our working experiences show that combinations of several simple techniques often work better than a single complicated one.

Remark 4. The above version is for the binary search space. It is easy to generalize it to other search spaces such as permutation spaces.

3 A hybrid algorithm with guided mutation for the maximum clique problem

In this section, we briefly introduce our work on combination of guided mutation and local search for dealing with the maximum clique problem.

3.1 Maximum clique problem and its solution representation

3.1.1 Problem

A clique of a graph is a set of pairwise adjacent nodes. A maximal clique is a clique which is not a proper subset of any other clique. A maximum clique is a clique with the maximum cardinality (which is called the maximum clique number). A maximum clique is maximal but not vice versa. Given a graph, the maximum clique problem (MCP) is to find a maximum clique. The MCP is NP-complete. Moreover, there is no polynomial-time algorithm for approximating the maximum clique within a factor of $n^{1-\varepsilon}$ unless $P = NP$[12], where n is the number of the nodes of the graph. These facts indicate that the MCP is very difficult to solve.

3.1.2 Representation and fitness

Given a graph $G = (V, E)$ where $V = \{1, 2, \cdots, n\}$ is its node set and $E \subset V \times V$ is its edge set. A set of nodes $U \subset V$ is encoded as a string $x = (x_1, x_2, \cdots, x_n) \in \{0,1\}^n$ where $x_i = 1$ if and only if node i is in U. Therefore the search space is $\Omega = \{0,1\}^n$. The fitness of x is defined as its cardinality if x represents a clique. Since every new solution generated by the guided mutation will be repaired, there is no need to define fitness values for infeasible solutions.

3.2 Algorithm

3.2.1 Partitioning of the search space

The search space, $\Omega = \{0,1\}^n$ for the MCP can be divided into $n+1$ subspaces as follows:

$$\Omega = \Omega_0 \bigcup \Omega_1 \cdots \bigcup \Omega_n$$

where $\Omega_i = \{x = (x_1, x_2, \cdots, x_n) : \sum\limits_{j=1}^{n} x_j = i\}$. Obviously, for any Ω_i and Ω_j, we have $\Omega_i \bigcap \Omega_j = \emptyset$. Any string in Ω_i represents a set of nodes with cardinality i.

The proposed algorithm explores different subspaces in the search space during different search phases. At the beginning, a lower bound low of the maximum clique number is computed. The first search phase will be for the area $\bigcup\limits_{i=low+1}^{low+\Delta} \Omega_i$, where Δ is a predefined integer number. If a maximal clique with a larger cardinality s is found, then the current search phase will end and the search area for next phase will be $\bigcup\limits_{i=s+1}^{s+\Delta} \Omega_i$. In such a way, the search is focused on these promising Ω_i.

3.2.2 Repair heuristic

We can induce a set of nodes from any string in $\{0,1\}^n$. This set of nodes, however, may not be a clique. To produce a clique, we use Marchiori′s heuristic[13] to repair it.

3.2.3 Initialization and update of the probability vector

The probability vector p for guided mutation needs to be initialized and updated in the algorithm.

Suppose that the current population $Pop(t)$ has N binary strings $x^j = (x_1^j, \cdots, x_n^j)$, $j = 1, 2, \cdots, N$, $p = (p_1, \cdots, p_n)$

will be initialized as

$$p_i = \frac{\sum_{j=1}^{N} x_i^j}{N}. \tag{1}$$

At each generation t in the algorithm, some binary strings are selected from the current population $Pop(t)$ to form the parent set $Parent(t)$, which is then used for updating the probability vector p. Let $Parent(t)$ contain M strings $y^j = (y_1^j, \cdots, y_n^j)$, $j = 1, 2, \cdots, M$, the probability vector p is updated in the same way as in the PBIL algorithm[1]:

$$p_i := (1-\lambda)p_i + \lambda \frac{\sum_{j=1}^{M} y_i^j}{M} \tag{2}$$

for $i = 1, 2, \cdots, n$. $\lambda \in (0, 1]$ is the learning rate.

3.2.4 The framework of the algorithm

EA/G works as follows:

Step 1 Set $t := 0$, randomly pick an $x \in \Omega$ and apply the Repair Operator to x to obtain a maximal clique U. Set $low := |U|$.

Step 2 Randomly pick N strings from $\cup_{i=low+1}^{low+\Delta}\Omega_i$ and apply the Repair Operator to each of them, the N resultant strings form $Pop(t)$. Then initialize the probability vector p by (1).

Step 3 Select the $N/2$ best strings from $Pop(t)$ to form the parent set $Parent(t)$ and then update the probability vector p by (2).

Step 4 Apply the Guided Mutation Operator to the fittest string in $Parent(t)$ $N/2$ times and then apply the Repair Operator to the resultant strings to get $N/2$ cliques. Add these $N/2$ cliques to $Parent(t)$ to form $Pop(t+1)$. If the stopping condition is met, return the largest clique found so far.

Step 5 Set $t := t+1$. Let S be the largest clique in $Pop(t)$. If $|S| > low$, set $low := |S|$ and go to Step 2.

Step 6 If all the strings in $Pop(t)$ are identical, go to Step 2, else go to Step 3.

In Step 1, a lower bound low for the maximum clique number is obtained. The population $Pop(t)$ and the probability vector p are initialized in Step 2 for the search on $\cup_{i=low+1}^{low+\Delta}\Omega_i$. Step 3 selects the fittest strings to become parents and then update p. In Step 4, $N/2$ strings are generated by applying the guided mutation operator to the fittest string in the current parent set. The mutated strings are then repaired. These resultant strings join their parents to form the population of the next generation. If a larger clique is found (Step 5), the search will move to a new area in the search space. If all the members in the current population are identical (Step 6), the search will be restarted to diversify the population.

3.3 The experimental results

The experimental results in [14] show that our algorithm outperforms hybrid genetic algorithm (HGA), the best GA for the MCP, on the DIMACS benchmark problem. However, an issue naturally arises: Is guided mutation better than other advanced EDA offspring generators in our algorithm framework? To investigate this issue, we have compared our algorithm with an algorithm, which is the same as our algorithm, except it uses the MIMIC way to generate new solutions. Some of the experimental results on 5 test problems are given in Table 1.

Table 1 The comparison results between EA/MIMIC and EA/G. *Best*: the size of the largest clique found, *Avg*: the average size of the cliques found. *std*: The standard deviation of the sizes of the cliques found

Graph	EA/MIMIC		EA/GuidedMutation	
	Avg(*std*)	*Best*	*Avg*(*std*)	*Best*
brock200_4	16.3(0.9)	17	16.5(0.5)	**17**
brock400_2	22.1(0.3)	23	24.7(0.4)	**25**
brock400_4	22.7(0.9)	23	25.1(2.6)	**33**
brock800_2	18.5(0.5)	19	20.1(0.4)	**21**
brock800_4	18.4(0.5)	19	19.9(0.5)	**21**

Clearly, the performance of MIMIC is poorer than that of guided mutation on these five test problems. It may be due to the facts that a) unlike the guided mutation, MIMIC cannot directly control the similarity between offspring and the best solutions found so far. and b) only a very limited number of dependence relationships can be considered in MIMIC, which is far away from enough for capturing the variable linkage structure in a complicated optimization problem. Other advanced EDAs may have the similar shortcomings.

4 EA with guided mutation for refining a heuristic

Suppose that we have a problem-specific construction heuristic for an optimization problem. The heuristic has a set of control parameters with relatively simple data structures. For any given parameter setting, the heuristic can construct a solution to the problem, the quality of which entirely depends on the parameter setting. An EA can be used to tune these control parameters to find a nearly-optimal parameter setting and thus generate a good solution to the original problem. This approach can be regarded as an instance of indirect encoding EAs since the actual search space of the EA is the parameter space of the heuristic and each parameter setting can be decoded (i.e., transformed) to a solution to the original problem *via* the heuristic. If solutions to the problem have a complex data structure and a parameterized construction heuristic is relatively easy to design, this approach could be a reasonable choice.

We have recently adopted this approach for solving several complicated telecommunication network design problems. In the following, we present our work for designing a network protection problem in wavelength-division multiplexing (WDM) networks.

4.1 Problem

As a case study, we consider a shared-path protection (SPP) problem with shared risk link group (SRLG) constraints. This problem can be modeled as an optimization problem in a simple directed graph. Given:

- V : the set of nodes in the graph under consideration.
- E : the set of directed links (edges) in the graph.
- W : the number of wavelengths available on each link. The wavelengths are numbered from 1 to W.
- R : the set of connection requests. $M = |R|$. The requests are numbered from 1 to M. Each connection request has a source node and a destination node[1]. It requires a working lightpath and a backup lightpath from the stated source to the stated destination.
- G : the set of SRLGs. Each SRLG contains a set of links in E. Links in the same SRLG share the same risk, i.e., these links may break at the same time due to a destructive event. If two paths in the network have links in the same SRLG, we say that they are SRLG-joint. Otherwise, we call them SRLG-disjoint.

The goal is to determine a working lightpath and a backup lightpath for each connection request in R. The constraints are:

C1 The number of wavelengths used on each link cannot not exceed W.

C2 The working lightpath and the backup lightpath, for each connection request, must be SRLG-disjoint.

C3 Two working lightpaths cannot use the same wavelength on the same link.

C4 A backup lightpath cannot share the same wavelength on the same link with any working lightpath.

C5 If two working lightpaths are SRLG-joint, their backup lightpaths cannot use the same wavelength on the same link.

The objective is to minimize the cost

$$\sum_{e\in E}(F_e + S_e) \tag{3}$$

where F_e is the number of wavelengths on link e used in working lightpaths and S_e the number of wavelengths on link e used in backup lightpaths.

This problem has been studied and modeled as an integer linear programming (ILP) problem in [15]. It is a NP-complete problem[16]. An ILP approach involves too many constraints and variables, even for a small-sized network[17]. Therefore, dealing with any practical-sized networks needs to resort to heuristics.

A solution to this problem consists of a set of working paths, a set of protection paths and a wavelength assignment scheme on these working and protection paths. Designing effective and efficient GA or EDA operators on these solutions could be a very challenging task.

[1]Two different requests in C may have the same source and destination.

4.2 Basic heuristics

The proposed basic heuristic consists of three phases. In the first phase, a set of working working paths is routed. The second phase computes computes protection paths for all the working paths established in the first stage. In the third phase, a wavelength is assigned to each path generated in the first two phases. permutation $\pi = (\pi_1, \pi_2, \cdots, \pi_M)$, permutation $\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_M)$ and a scalar parameter c. The output is a working lightpath set WP, a backup lightpath set BP and a wavelength assignment $A : BP \cup WP \rightarrow \{1, 2, \cdots .\}$. It can be written as:

$$(BP, WP, A) = BH(\pi, \sigma, c).$$

The performance of this heuristic is determined by its control parameters. The details of this heuristic can be found in [18].

4.3 Tuning control parameters

We used EA/G, an evolutionary algorithm in which a guided mutation is used for generating new trial solutions to tune π and σ. EA/G works with the permutation space. c is tested on several selected values. The tuning procedure work as follows:

Step 1. Tuning c

Step 1.1. Randomly generate 10 pairs of permutations (π^1, σ^1), (π^2, σ^2), $\cdots$, (π^{10}, σ^{10}). Let $c_i = \frac{i}{10}$ $(i = 1, 2, \cdots, 10)$.

Step 1.2. Compute

$$c^\star = \arg \min_{c\in\{c_1,c_2,\cdots,c_{10}\}} \frac{1}{10}\sum_{j=1}^{10} cost(\pi^j, \sigma^j, c)$$

Step 2. Tuning π

Step 2.1. Randomly generate a permutation $\tilde{\sigma}$.

Step 2.2. Use EA/G to tune π where the cost of π is set as $cost(\pi, \tilde{\sigma}, c^\star)$. Set $\pi^\star$ to be the best setting of π found in EA/G (GA).

Step 3 Tuning σ

Use EA/G to tune σ where the cost of σ is set as $cost(\pi^\star, \sigma, c^\star)$. Set $\sigma^\star$ to be the best setting of σ found in EA/G (GA).

Then the best solution found to the RWAP is the solution generated by BH with parameter setting $(\pi^\star, \sigma^\star, c^\star)$.

We can iterate the above procedure many times in order to lower the cost of the solution obtained as in the alternating variables optimization method[19]. Taking the computational overhead into consideration, we chose not to perform iteration in our experimental study.

4.4 Experimental results

We have compared the our algorithm with HA, a heuristic developed by H. Zang[17]. To investigate the performance of guided mutation, we have also substituted EA/G by a GA in our algorithm and compared it with our approach. Table 2 gives some of experimental results:

Table 2 The solution quality of HA, BH/GA and BH/EA/G on test network instances

instances	HA	BH/GA		BH/EA/G	
	cost	*best*	*avg*	*best*	*avg*
T_1-1	478.0	369.0	381.5	365.0	372.50
T_1-2	458.0	378.0	382.7	368.0	374.55
T_1-3	487.0	370.0	381.4	364.0	368.70
T_1-4	445.0	394.0	401.5	377.0	384.60
T_1-5	492.0	393.0	410.3	384.0	390.75
T_1-6	462.0	390.0	399.0	377.0	384.90
T_1-7	451.0	383.0	386.4	384.0	403.10
T_1-8	785.0	660.0	671.4	636.0	649.20
T_1-9	1389.0	1257.0	1268.4	1221.0	1239.1

BH/EA/G is our proposed approach and BH/GA is the same as BH/EA/G except that a GA is used for tuning π and σ.

It can be observed from this table that both BH/EA/G and BH/GA perform much better than HA, which implies that our approach for combining problem-specifical heuristic with evolutionary algorithm is effective. It is also clear that BH/EA/G outperforms BH/GA. It indicates that guided mutation does improve the performance of evolutionary algorithms.

5 EDA+Two local search techniques for continuous optimization

Combinations of evolutionary algorithms and local search (Memetic algorithms[20]) have been proved to be very successful in dealing with hard optimization problems. Most, if not all, of memetic algorithms apply the same local search to all the new solutions generated by EA operators. Since the local search on most new solutions will produce local optima, which are not desirable solutions. Therefore, most computational costs are wasted. We have proposed EDA/L, a combination of EDA and two different local search for continuous optimization problems, as an attempt to overcome this shortcoming.

5.1 Problem

We are considering the following global optimization problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in D \end{array} \tag{4}$$

where $x = (x_1, x_2, \cdots, x_n) \in \mathbf{R}^n$, $f : \mathbf{R}^n \to \mathbf{R}$ is the objective function and $D = \{x \mid a_i \le x_i \le b_i \text{ for } i = 1, 2, \cdots, n\}$ is the feasible region. This problem arises in almost every field of science, engineering and business. Often, the objective function $f(x)$ is non-convex and has many local minima in the feasible region that are not the global minimum.

5.2 Algorithm

The framework of the proposed algorithm is as follows:

Step 1. **Parameter Setting** Population size: N, the best solution value found so far: $Fbest = \infty$, the number of new solutions sampled from the probability model at each iteration: K, the maximal number of function evaluations in the simplex method: S, and the number of solutions undergoing UOBDQA at each iteration: J. $J < K$.

Step 2. **Initialization** Generate N solutions $\tilde{x}^1, \tilde{x}^2, \cdots, \tilde{x}^N$ from D, using the uniform design technique. Apply the simplex method with at most S function evaluations to each solution $\tilde{x}^i$ and obtain $x^i (1 \le i \le N)$. Then let $x^1, x^2, \cdots, x^N$ constitute the initial population.

Step 3. **Reproduction** Build a probability model based on the statistical information extracted from some selected solutions in the current population. Sample K new solutions $\tilde{x}^{N+1}, \cdots \tilde{x}^{N+K}$ from this model and then apply the simplex method with at most S function evaluations to each solution $\tilde{x}^i$ and obtain x^i $(N + 1 \le i \le N + K)$.

Step 4. **Comparison** Compare the function values of all x^i $(1 \le i \le N + K)$, order and relabel them such that

$$f(x^1) \le f(x^2) \le \cdots \le f(x^{N+K}).$$

Step 5. **Update of *Fbest*** Apply UOBDQA to x^i $(1 \le i \le J)$ and obtain J local minima y^i $(1 \le i \le J)$. If $Fbest > \min_{1 \le i \le J} f(y^i)$, set $Fbest = \min_{1 \le i \le J} f(y^i)$.

Step 6. **Stopping Condition** If the stopping condition is met, stop.

Step 7. **Update of the Population** Let $x^{J+1}, \cdots, x^{J+N}$ constitute new population. Go to Step 3.

Experimental design techniques[21] such as orthogonal design and uniform design have been proposed to improve the performance of genetic algorithms[22,23]. The total number of orthogonal design points is often much more than 2^n while the number of uniform design points is relatively smaller. This is the reason why we use the uniform design in Initialization (Step 2). The probability model built in Reproduction (Step 3) step models the distribution of the best solutions in the current population. Therefore, sampling solutions from this model should fall in promising areas with high probability. In Step 4, since all the solutions have undergone the incomplete simplex method with at most S function value evaluations, the best ones, i.e., $x^1, x^2, \cdots, x^J$ in Step 5, should be more likely to be close to the global optimum than other solutions. We apply UOBQDA only to these best solutions in Step 5.

5.2.1 Reproduction and Sampling

Let $x^1, x^2, \cdots, x^N$ be the current population, we select the M best solutions from the current population. Without loss of generality, we assume that the M selected solutions are $x^1, x^2, \cdots, x^M$. To model the distribution of these M best solutions by a fixed-width histogram distribution[24], we divide the search space $[a_i, b_i]$ of each variable x_i into H subintervals with the same length. The j-th subinterval is[2]

$$[a_i + \frac{j-1}{H}(b_i - a_i), a_i + \frac{j}{H}(b_i - a_i)), \quad (1 \le j \le H).$$

[2]Precisely, the H-th subinterval is $[a_i + \frac{H-1}{H}(b_i - a_i), b_i]$.

Then we count $W(i,j)$, the number of the selected solutions whose values of x_i fall in the j-th subinterval. The marginal probability density of x_i is modeled by

$$p_i(x_i)=\begin{cases}\frac{W(i,1)}{M}\frac{H}{b_i-a_i} & a_i\le x_i< a_i+\frac{1}{H}(b_i-a_i)\\ \frac{W(i,2)}{M}\frac{H}{b_i-a_i} & a_i+\frac{1}{H}(b_i-a_i)\le x_i<a_i+\\ & \frac{2}{H}(b_i-a_i)\\ \vdots & \vdots\\ \frac{W(i,H)}{M}\frac{H}{b_i-a_i} & a_i+\frac{H-1}{H}(b_i-a_i)\le x_i\le b_i.\end{cases}\tag{5}$$

A new solution $\widetilde{x}=(\widetilde{x}_1,\widetilde{x}_2,\cdots,\widetilde{x}_n)$ is generated by sampling the value of each x_i from (5) independently. To generate $\widetilde{x}_i$, we first randomly select a subinterval $[a_i+\frac{j-1}{H}(b_i-a_i),a_i+\frac{j}{H}(b_i-a_i))$ with probability $\frac{W(i,j)}{M}$, and then pick up a number from this subinterval with uniform distribution. In Reproduction, we generate K new solutions in this way.

5.2.2 UOBDQA

Trust region methods using quadratic interpolation model, developed by Powell[25], is a new class of derivative free local optimization algorithms for finding a local minimum of the objective function $f(x)$. Such algorithms start with an initial point v, ρ_{beg} and ρ_{end}, the initial and final values of a trust region radius ρ. The algorithms generate a set of interpolation points (including v) in a neighborhood of v. Then an initial quadratic model is formed by interpolating these points. The algorithm generate a new point, either by minimizing the current quadratic model within a trust region, or by a procedure that improves the accuracy of the model. One of the interpolation points is replaced by the resultant point. The typical distance between successive points at which f is calculated are of magnitude of the trust region radius ρ. The initial value of ρ is set to be ρ_{beg}. ρ will be reduced when the objective function stops decreasing for such changes to the points. The algorithms stop when ρ is smaller than ρ_{end}.

Several instances of the trust region methods have been implemented by Powell[25]. Unconstrained optimization by quadratic approximation (UOBYQA), as its name suggests, builds a general quadratic model in a trust region. It needs to maintain $(n+1)(n+2)/2$ interpolation points at each iteration. The amount of routine work of an iteration takes $O(n^4)$. Consequently, UOBYQA will be prohibitive for the large value of n. To overcome this shortcoming, unconstrained optimization by diagonal quadratic approximation (UOBDQA) was proposed, which chooses a quadratic of the form:

$$Q(v^*+d)=f(v^*)+g^{\mathrm{T}}d+d^{\mathrm{T}}Dd$$

where v^* is the point with the least objective function among the current interpolation points. g is a vector and D is a diagonal matrix of dimension n. Therefore, $2n+1$ interpolation points are needed to determine this quadratic model. The amount of routine work of each iteration is $O(n^2)$ instead of $O(n^4)$ in UOBYQA, which makes UOBDQA more useful than UOBYQA for solving large scale optimization problems. For the detailed description of UOBYQA and UOBDQA, please refer to [25].

5.3 Experimental results

5.3.1 Test suite

We have tested EDA/L on 10 widely-used test functions. The following are 4 of these functions.

$f_1=\sum\limits_{i=1}^{n}(-x_1\sin(\sqrt{|x_i|}))$

where $n=30$ and $-500\le x_i\le 500$ for all $i=1,\ldots,30$.

$f_2=\sum\limits_{i=1}^{n}(-x_i^2-10\cos(2\pi x_i)+10)$

where $n=30$ and $-5.12\le x_i\le 5.12$ for all $i=1,\ldots,30$.

$f_3=-20\exp(-0.2\sqrt{\frac{1}{n}\sum\limits_{i=1}^{n}x_i^2})-\exp(\sum\limits_{i=1}^{n}\cos(2\pi x_i))$

where $n=30$ and $-32\le x_i\le 32$ for all $i=1,\ldots,30$.

$f_4=\frac{1}{4000}\sum\limits_{i=1}^{n}(-x_i^2-\prod\limits_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1)$

where $n=30$ and $-600\le x_i\le 600$ for all $i=1,\ldots,30$.

All these functions have many local minima.

5.3.2 Comparison with other algorithms

We have compared EDA/L with several algorithms, including:

- Orthogonal genetic algorithm with quantization (OGA/Q)[22]. This algorithm applies the orthogonal design to improve the performance of the genetic algorithm for global optimization.
- Fast evolution strategy (FSA)[26]. This algorithm uses Cauchy mutation instead of Gaussian mutation in generating new test points.
- Particle swarm optimization (PSO)[27]. It is a algorithm inspired by bird flocking.

Tables 3 compares the quality of the solutions found by EDA/L with the above three algorithms, while Table 4 compares the computational costs. The experimental results of these three algorithms are from [11, 26, 27], respectively.

Table 3 Comparison of the mean of the smallest function value found

functions	EDA/L	OGA/Q	FSA	PSO
f_1	-12569.4811	-12569.4537	-12554.5	N/A
f_2	0	0	4.6×10^{-2}	46.4689
f_3	4.141×10^{-15}	4.440×10^{-16}	1.8×10^{-2}	N/A
f_4	0	0	1.6×10^{-2}	0.4498

Table 4 Comparison of the mean no of the function evaluations

functions	EDA/L	OGA/Q	FSA	PSO
f_1	52,216	302,166	900,030	N/A
f_2	75,014	224,710	500,030	250,000
f_3	106,061	114,421	150,030	N/A
f_4	79,096	134,000	200,030	250,000

We can see that EDA/L can give better solutions than other algorithms on all the test functions except f_3 in which the solution obtained by OGA/Q is slightly better than EDA/L. We also see that the computational cost of our algorithm is the smallest on all the test problems. These results show that combinations of EA with two different local search techniques is worthwhile investigating[28].

6 Conclusions

EDAs are a novel optimization tool. However, a single technique is hard for solving complicated optimization problems. Therefore, how to combine EDAs with other techniques represents an important research direction. This paper has summarized some of our efforts in hybridizing EDAs with other techniques. We have proposed guided mutation in which GA and EDA ideas are combined. We have showed that an EA with guided mutation can efficiently used for tackling with the maximum clique problem. We have also advocated using an evolutionary algorithm with guided mutation for tuning control parameters of a problem-specific heuristic for solving a hard optimization problem. We have showed that a combination of an EDA with two local search techniques works very well for numerical global optimization.

Our work has shown that combinations of simple techniques could be effective and efficient for solving hard problems. Encouraged by our success in scalar optimization, we have recently made effort in combining traditional mathematical programming methods, machine learning and evolutionary algorithms for solving multi-objective optimization problems[29,30].

References

[1] S. Baluja. Population-based Icremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, 1994.

[2] H. Mühlenbein, G. Paaß. From Recombination of Genes to the Estimation of Distributions: I. Binary parameters. *Lecture Notes in Computer Science: Parallel Problem Solving from Nature*, vol. 1141, pp. 178–187, 1996.

[3] J. S. De Bonet, C. L. Isbell, P. Viola. MIMIC: Finding Optima by Estimating Probability Densities. *Advances in Neural Information Processing Systems*, vol. 9, no. 1, pp. 424–431, 1996.

[4] S. Baluja, S. Davies. Combining Multiple Optimization Runs with Optimal Dependency Trees. Technical Report CMU-CS-97-157, Carnegie Mellon University, 1997.

[5] H. Mühlenbein, T. Mahnig. The Factorized Distribution Algorithm for Additively Decomposed Functions. In *Proceedings of Congress on Evolutionary Computation*, Vol. 1, pp. 752–759, 1999.

[6] M. Pelikan, D. E. Goldberg, E. Cantú-Paz. Linkage Problem, Distribution Estimation and Bayesian Networks. *Evolutionary Computation*, vol. 8, no. 3, pp. 311–340, 2000.

[7] B. T. Zhang. Bayesian Evolutionary Algorithms for Learning and Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Workshop Program*, pp. 220–222, 2000.

[8] V. V. Miagkikh, W. F. Punch III. Global Search in Combinatorial Optimization Using Reinforcement Learning Algorithms. In *Proceedings of the Congress on Evolutionary Computation*, Washington D.C., USA, July 6-9, vol. 1, pp. 189–196, 1999.

[9] F. Glover, M. Laguna. *Tabu Search*, Kluwer Academic Publishers, New York, 1997.

[10] S. Lin, B. W. Kernighan. An Effective Heuristic Algorithm for the Travelling Salesman Problem. *Operations Research*, vol. 21, pp. 498–516, 1973.

[11] A. E. Eiben, J. E. Smith. *Introduction to Evolutionary Computing*, Springer-Verlag, London, 2003.

[12] J. Hastad. Clique is Hard to Approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, vol. 182, no. 1, pp. 105–142, 1999.

[13] E. Marchiori. Genetic, Iterated and Multistart Local Search for the Maximum Clique Problem. *Applications of Evolutionary Computation*, vol. 2279, pp. 112–121, 2002.

[14] Q. Zhang, J. Sun, E. Tsang. Evolutionary Algorithm with the Guided Mutation for the Maximum Clique Problem. *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 2, pp. 192–200, 2005.

[15] H. Zang, C. Ou, B. Mukherjee. Path-protection Routing and Wavelength-assignment in WDM Mesh Networks under Shared-Risk-Group Constraints. In *Proceedings of Asia-Pacific Optical and Wireless Communications Conference*, Beijing, China, vol. 4585, pp. 49–60, 2001.

[16] J. Q. Hu. Diverse Routing in Optical Mesh Networks. *IEEE Transactions on Communcations*, vol.51, no. 3, pp. 489-494, 2003.

[17] H. Zang, C. Ou, B. Mukherjee. Path-protection Routing and Wavelength-Assignment (RWA) in WDM Mesh Networks under Duct-layer Constraints. *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 248–258, 2003.

[18] Q. Zhang, J. Sun, G. Xiao, E. Tsang. Evolutionary Algorithms Refining a Heuristic: A Hybrid Method for Shared-path Protections in WDM Networks under SRLG Constraints. *IEEE Transactions on System, Man, and Cybernetics – Part B: Cybernetics*, vol. 37, no. 1, pp. 51–61, 2007.

[19] R. Fletcher. *Practical Methods of Optimization*, John Wiley and Sons, New York, vol. 1, 1980.

[20] E. K. Burke, G. Kendall. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Springer-Verlag, New York, LLC, 2005.

[21] S. E. Maxwell, H. D. Delaney. *Designing Experiments and Analyzing Data: A Model Comparison Perspective*, Lawrence Erlbaum Associates, London, 2003.

[22] K. S. Leung, J. Sun, Z. B. Xu. Efficency Speed-up Strategies for Evolutionary Computation: An Adaptive Implementation. *Engineering Computation*, vol. 19, no. 3, pp. 272–304, 2003.

[23] Q. Zhang, Y. W. Leung. An Orthogonal Genetic Algorithm for Multimedia Multicast Routing. *IEEE Transactions on Evolutionary Computation*, vol. 3, no.1, pp. 53–62, 1999.

[24] S. Tsutsui, M. Pelikan, D. Goldberg. Evolutionary Algorithm Using Marginal Histogram Models in Continuous Domain. In *Optimization by Building and Using Probabilistic Models*, San Francisco, California, USA, pp. 230–233, 2001.

[25] M. J. D. Powell. On Trust Region Methods for Unconstrained Minimization without Derivatives. Technical Report DAMTP 2002/NA02, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, 2002.

[26] X. Yao, Y. Liu. Evolutionary Programming Made Faster. *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[27] J. Kennedy, R. Eberhart. Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks,* Perth, Australia, vol. 4, pp. 1942–1948, 1995.

[28] Q. Zhang, J. Sun, E. Tsang, J. Ford. Hybrid Estimation of Distribution Algorithm for Global Optimisation. *Engineering Computations*, vol. 21, no. 1, pp. 91–107, 2003.

[29] Q. Zhang, A. Zhou, Y. Jin. Emodelling the Regularity in an Estimation of Distribution Algorithm for Continuous Multiobjective Optimisation with Variable Linkages. *IEEE Transactions on Evolutionary Computation*, to be published.

[30] Q. Zhang, H. Li. A Multi-objective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, to be published.

Qingfu Zhang received the B.Sc. in mathematics from Shanxi University, Shanxi, China in 1984, the M.Sc. in applied mathematics and the Ph.D. in information engineering from Xidian University, Xi′an, China, in 1991 and 1994, respectively.

He is currently a reader in the Department of Computer Science, University of Essex, UK. Before joining Essex in 2000, he was with the National Laboratory of Parallel Processing at Computing in National University of Defence Science and Technology (China), Hong Kong Polytechnic University (Hong Kong), the German National Research Center for Information Technology (now Fraunhofer-Gesellschaft, Germany) and University of Manchester Institute of Science and Technology (UK). His main research interests include evolutionary computation, optimization, neural networks, data analysis and their applications.

Dr. Zhang is an associate editor of the *IEEE Transactions on Systems, Man, and Cybernetics-Part B* and a Guest Editor of a forthcoming special issue on evolutionary algorithms based on probabilistic models in the *IEEE Transactions on Evolutionary Computation*.

Jianyong Sun received the B.Sc. in mathematics from Xi′an Jiaotong University, China in 1997, and the Ph.D. degree in computer science from University of Essex, UK., in 2005.

He is now a research associate at School of Computer Science, University of Birmingham, UK. In 2000, he worked in Department of Computer Science and Engineering, the Chinese University of Hong Kong (Hong Kong) as a research assistant. From 2002 to 2003, he worked in Department of Computer Science, University of Essex (U. K.) as a senior research officer. His main research interests include evolutionary computation, optimization, theory and algorithm of metaheuristic, telecommunication networks and machine learning.

Edward Tsang has a first degree in business administration (major in finance) and a Ph.D. in computer science. He has broad interest in applied artificial intelligence, in particularly computational finance, scheduling, heuristic search, constraint satisfaction and optimization. He is currently a professor in computer science at the University of Essex where he leads the Computational Finance Group and Constraint Satisfaction and Optimization Group. He is also the deputy director of the Center for Computational Finance and Economic Agents (CCFEA), an interdisciplinary center. He chaired the Technical Committee for Computational Finance under the IEEE Computational Intelligence Society from 2004 to 2005.