A Modified Screening Estimation of Distribution Algorithm for Large-Scale Continuous Optimization

Krishna Manjari Mishra and Marcus Gallagher

School of Information Technology and Electrical Engineering,
The University of Queensland, Brisbane, 4072. Australia.
{krishna,marcusg}@itee.uq.edu.au

Abstract. Continuous Estimation of Distribution Algorithms (EDAs) commonly use a Gaussian distribution to control the search process. For high-dimensional optimization problems, several practical issues arise when estimating a large covariance matrix from the selected population. Recent work in continuous EDAs has aimed to address these issues. The Screening Estimation of Distribution Algorithm (sEDA) is one such algorithm which, uniquely, utilizes the objective function values obtained during the search. A sensitivity analysis technique is then used to reduce the rank of the covariance matrix, according to the estimated sensitivity of the fitness function to individual variables in the search space.

In this paper we analyze sEDA and find that it does not scale well to very high-dimensional problems because it uses a large number of additional fitness function evaluations per generation. A modified version of the algorithm, named sEDA-lite is proposed which requires no additional fitness evaluations for sensitivity analysis. Experiments on a variety of artificial and real-world representative problems evaluate the performance of the algorithm compared with sEDA and EDA-MCC, a related, recently proposed algorithm.

Keywords: Estimation of Distribution algorithms, High-Dimensional Optimization problems, Screening Technique.

1 Introduction

Estimation of Distribution Algorithms (EDAs) are a class of evolutionary optimization algorithms, where probabilistic models play the key role in controlling the search process. In EDAs, the selected population is used to learn a probability distribution and subsequent solutions obtained by sampling from this distribution. The general procedure for an EDA is summarized in Algorithm 1. A number of different types of density estimation models have been used in EDAs for both discrete and continuous search spaces. Probabilistic models can be used to identify and represent interactions among the variables and can represent a priori information about the problem structure, which may assist the search process. EDAs have been developed in both the discrete and continuous setting, and have been successfully applied to solving a variety of problems.

G. Dick et al. (Eds.): SEAL 2014, LNCS 8886, pp. 119–130, 2014.

Algorithm 1. General pseudocode framework for an EDA

1: Initialization: set $t = 0$, Generate initial population uniformly in search space
2: Evaluate $f(\mathbf{x}')$ for each individual $\mathbf{x}'$ in the current population
3: Select promising individuals
4: Build probabilistic model $p(\mathbf{x})$ based on selected individuals
5: Generate new population by sampling from $p(\mathbf{x})$
6: $t = t + 1$
7: Goto step 2 until a stopping criterion is met

Reviews of EDAs can be found in [13,16,19]. This paper focuses on continuous EDAs ($\mathbf{x} \in I\!R^n$).

Truncation selection is typically applied in EDAs. The parameters of $p(\mathbf{x})$ are typically fitted using maximum likelihood estimation. While several different models have been considered for continuous optimization, a Gaussian distribution is most commonly used. The continuous Univariate Marginal Distribution Algorithm (UMDA_c)[13] uses a factorized Gaussian model (i.e. a diagonal covariance matrix) which assumes all the variables are independent. UMDA_c is easy to apply and computationally robust and efficient, but the model may have difficulty in solving problems with strong dependencies between variables. Multivariate Gaussian EDAs, such as the Estimation of Multivariate Normal Algorithm ($\text{EMNA}_\text{global}$) address this issue by modelling dependencies between all pairs of variables using a full covariance matrix [13].

The behaviour of basic Gaussian EDAs has been shown to be sometimes undesirable. On some fitness landscapes, performance is poor due to premature shrinking of the model variance at an exponential rate (Eg., in slope-like regions of the search space, described in [1] or in an elliptical region [10]). To address such issues, a number of enhancements have been proposed. Adaptive variance scaling (AVS) provides a way to control the rate of contraction and expansion of the model and scale the variance to improve the progress of the EDA model. Anticipated Mean Shift (AMS) additionally modifies sampled solutions in the direction of mean shift of the previous generation [1,2]. Nevertheless, the task of covariance matrix estimation remains a fundamental step in state of the art Gaussian EDAs.

In practice, numerical issues can arise with estimating the full covariance matrix. The covariance matrix, Σ, is by definition positive semi-definite, but this is not guaranteed in implementation because of finite precision representation. Computational errors or numerical issues arise when the sample used to estimate the model does not adequately span all dimensions of the search space, which becomes likely when the sample size is relatively small compared to the problem dimensionality. As a result of these issues, several techniques have been proposed to avoid the covariance matrix becoming ill-posed in EDAs. The Eigenspace EDA (EEDA) [20] was one of the first modifications in this direction (see [7] for a discussion and comparison of variants). Dong et al. [6] developed the Covariance Matrix Repairing (CMR) method, where a positive value (equal to the absolute value of the smallest eigenvalue) is added to the diagonal of Σ.

The Eigen-decomposition EDA (ED-EDA) builds on this previous work with eigenvalue-based repairing strategies [7] . Experimental results show that these different covariance repairing methods can avoid numerical difficulties. The algorithms also show good performance results with respect to the best solution found and because of the improved numerical properties, a smaller population can be used.

More recently, Dong et al.[5] proposed the EDA Model Complexity Control (EDA-MCC) to scale up continuous EDAs to high-dimensional problems using a sparse covariance matrix, with reduced computational cost and a smaller population. EDA-MCC uses $\text{EMNA}_{\text{global}}$ for each subset of the variables. A set of artificial test problems were used for comparing EDA-MCC with UMDA_c, $\text{EMNA}_{\text{global}}$ and EEDA [20]. While EDA-MCC does not outperform traditional EDAs on low dimensional problems, EDA-MCC shows significantly better results on high dimensional problems. Other existing statistical methods have also been applied to control the amount of covariance/dependency modelling in EDAs. In [11], regularization techniques were adopted into EDAs. The resulting algorithm shows the ability to solve high dimensional problems with a comparable quality of solutions using much smaller populations.

The screening Estimation of Distribution Algorithm (sEDA), was proposed in [14] as an EDA to control the degree of covariance modelling. However unlike other approaches, this algorithm explicitly uses the objective function values obtained during the search. Using this information, a notion of variable importance is derived by adapting a screening technique from experimental design. The algorithm also improves on numerical stability in EDAs by allowing the level of dependency modelling to be controlled. It performs better than traditional EDAs on low dimensional problems. In this paper, a modified version of sEDA, called sEDA-lite is proposed which improves on the previous algorithm, specifically by allowing the algorithm to scale to higher dimensional problems using less function evaluations than sEDA.

This paper is structured as follows. In Section 2, the existing sEDA algorithm is described. We analyse the issues arising in high dimensions when using sEDA. In Section 3, we propose sEDA-lite to address these issues. In Section 4, we compare the solutions of sEDA-lite with solutions of UMDA_c, EEDA and EDA-MCC on a set of artificial test problems. In addition to this, the solution of sEDA-lite is also compared with UMDA_c, $\text{EMNA}_{\text{global}}$ and sEDA on a couple of real world problems. Conclusions of this paper are drawn in Section 6.

2 Screening Estimation of Distribution Algorithms (sEDA)

The continuous global optimization problem is to find $\mathbf{x}^*$ such that

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in S,$$

where $S \subseteq \mathbb{R}^n$ is the set of feasible solutions, $f(\mathbf{x})$ is the fitness or objective function and $\mathbf{x} = (x_1, \ldots, x_n)$ is an individual or candidate solution vector.

In the Screening EDA [14], variables are modeled based on their estimated influence on the fitness function. From this, the most important variables are then modeled using the $\text{EMNA}_{\text{global}}$ model (full covariance matrix), while the variables which are least important are modeled using the UMDA_c model (no covariance), to try and capture the advantages and limit the potential problems of both approaches. Hence, the sEDA uses a multivariate Gaussian model where the covariance matrix contains some degree of sparseness. To select which variables to model using covariance, a technique from sensitivity analysis known as the Morris method is used.

The Morris method [15], is based on measuring the mean and standard deviation of changes in the fitness function value given perturbations of individual variables, calculated via so-called *elementary effects* terms. The elementary effect for the ith variable, $E_i(\mathbf{x})$, is defined as follows. Let Δ be a pre-determined amount to perturb each variable. For a given $\mathbf{x}$,

$$E_i(\mathbf{x}) = \frac{f(x_1, x_2, ..., x_{i-1}, x_i + \Delta, x_{i+1}, ..., x_n) - f(\mathbf{x})}{\Delta} \tag{1}$$

where $\mathbf{x} = (x_1, x_2, ..., x_n)$ is a given starting or "baseline" vector in the solution space. The perturbations, Δ are by default determined according to a full factorial sampling grid of some fixed resolution or increment size. In other words, for each variable x_i, over some fixed range and increment size, the value of x_i is changed and f is recalculated, producing a sample or set of values of $E_i(\mathbf{x})$.

Given a set of elementary effect values, the mean, $\overline{E}_i(\mathbf{x})$, and standard deviation, $std(E_i(\mathbf{x}))$ over the sample can be calculated. If this calculation is done over an arbitrary set of points in an arbitrary order, the absolute values, $|E_i|$ are used [4,12] and we take this approach in sEDA. A high value of $\overline{E}_i(\mathbf{x}^*)$ indicates that x_i has a large average influence on the value of f. A high value of $std(E_i(\mathbf{x}))$ indicates that variable x_i has a fluctuating influence on the value of f, which may indicate that it is involved in interactions with other variables [15].

In sEDA, the Morris method is adapted by calculating elementary effects values using the selected population on each generation of the algorithm rather than being based on a predetermined grid of points. Specifically, the mean of the selected population is calculated for each dimension x_i. A new set of candidate solutions is then generated, by creating new solution vectors where the mean value is substituted in turn for each problem variable (e.g. $\mathbf{x^i} = x_i, \ldots, x_{i-1}, m_i, x_{i+1}, \ldots, x_n$) in each individual in the selected population. This produces $n \times \tau \times M_{sel}$ new individuals, where M_{sel} is the size of the selected population[1], n is the dimension of the problem and τ ($0 < \tau < 1$) is the selection ratio, which are evaluated using f to produce a set of elementary effect values.

Given these values and their sample mean and standard deviation, sEDA uses the concepts of dominance and Pareto optimality from multi-objective optimization (see, e.g. chapter 9 of [8]) to determine which variables are the most "important". We consider the mean and standard deviation of elementary effects as two different (aka decision-making) criteria. One solution is said to dominate

[1] Rounding if $M_{sel} \cdot \tau$ is not an integer.

the other if its score is at least as high for all objectives, and is strictly better for at least one. The set of all non-dominated solutions is called the Pareto set or the Pareto front.

A fixed fraction η $(0 < \eta < 1)$ of the variables need to be selected for covariance modeling in the sEDA. Variables that belong to the Pareto set are selected first. If more variables are required, then those which have the minimum (Euclidean) distance to the Pareto front are selected. On the other hand, if the number of variables on the Pareto front is greater than required, then a random subset of these variables is selected. Hence at each generation, a sparse matrix is formed that has $n \times \eta$ variables[2] which are modeled using covariance parameters while the remaining variables are modelled using only variance terms. The mean vector of the selected population along with the sparse covariance matrix is then used to generate the new population for the next generation.

2.1 Scaling of sEDA

Due to the nature of the algorithm, sEDA as described above will require a relatively large number of function evaluations when applied to high-dimensional problems. This is due to the fact that for each generation, the population size is directly proportional to the dimension of the problem. Hence, the number of function evaluations per generation is $O(nM_{sel})$.

3 Scaling sEDA to High-Dimensional Problems: sEDA-lite

In this section we describe a modified version of sEDA, called sEDA-lite. The algorithm uses the same principles as sEDA but differs in the calculation of the elementary effects values. As discussed in Section 2.1, using the mean of the selected population to calculate elementary effects values in sEDA results in significant increase in the number of fitness function evaluations required per generation. In sEDA-lite, the main innovation is to instead use the *median* of each dimension in the selected population to calculate elementary effects values. Like the mean, the median is representative of the center of the selected population. However, the median is by definition located at one of the given individual points. Hence, all calculations in Equation(1) are carried out between individuals in the selected population (i.e. their fitness values have already been evaluated). Hence for each generation of sEDA-lite the number of function evaluations is reduced from $M_{sel} + (M_{sel} \times n \times \tau)$ to M_{sel}.

The median of the selected population is taken as the central/reference point for the elementary effect calculations. Other points in the selected population represent perturbations around this point and hence the elementary effect values measure the sensitivity of the objective function to changes in the solution values in the region of the search space represented by the current selected population.

[2] Rounding if $n \times \eta$ is not an integer.

The mean is also still calculated, since this is used as a parameter of the EDA model itself.

sEDA-lite uses the same Pareto optimal concept as in sEDA, to select the important variables in the problem as the ones that have the largest mean/standard deviation of elementary effect values. After selection, the covariance matrix for the EDA model is formed as a sparse matrix, with non-zero covariance terms for selected variables. This is used in combination with the EDA mean vector (estimated from the selected expanded population) and the model is then used to generate the new population as in a standard EDA. The process is repeated until some stopping criterion is met.

Algorithm 2. Pseudo code for sEDA-lite

Given: Population size M, dimensionality n, selection parameter $0 < \tau < 1$, model selection parameter $0 < \eta < 1$.
Begin (set $t = 0$)
Initialize population P by generating M individuals uniformly in S.
while stopping criteria not met **do**
  Evaluate f for population P.
  Truncation selection: $P_{sel} = M_{sel}$ best individuals from P; $M_{sel} = \text{Rnd}(M \cdot \tau)$.
  Calculate mean, $\boldsymbol{\mu}$ and median, $\tilde{\boldsymbol{m}}$ of P_{sel}
  Calculate $\tilde{m} = \text{median}(P_{sel})$, where $\tilde{m} = \tilde{m}_1, \cdots, \tilde{m}_n$.
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** M_{sel} **do**
      Calculate $E_{ij}(\boldsymbol{x})$ using Eqn.1, where $\tilde{m}$ is the baseline point and the perturbation value is given by j^{th} individual
    end for
  end for
  Calculate $mean(E)$ and $std(E)$ over M_{sel} perturbations in E_{ij}.
  Determine the Pareto optimal solutions/variables for objectives $abs(mean(E))$ and $std(E)$. Let this number of variables be p_o .
  Let $B = \text{Round}(n \cdot \eta)$.
  If $p_o > B$, randomly choose B variables from p_o.
  If $p_o < B$, select/add the next $B - p_o$ variables nearest to the Pareto front.
  Build Σ_t using covariance terms for the B selected variables and variance terms only for the remaining $n - B$ variables
  $p(\mathbf{x}) \leftarrow (\boldsymbol{\mu}, \Sigma_t)$.
  Generate P new population by sampling from $p(\mathbf{x})$.
end while

4 Experimental Design

To evaluate the performance of sEDA-lite, we have carried out experiments on 3 different sets of problems. The first is a set of commonly used artificial test functions. The second set of problems are Circle in a square (CiaS) packing problems and the third set are the 50-customer Location Allocation problems with different numbers of facilities. While the artificial functions are useful for

comparison with other algorithms, we also consider it important to evaluate the technique on real-world representative problems. The problems used here are all scalable in terms of dimensionality. They are also known to have features that make them difficult to solve for many algorithms, e.g. they are not everywhere differentiable and contain a large number of local optima.

4.1 Artificial Test Problems

The artificial test problems considered in this paper are taken from Dong et al. [5]. The problems are categorized into separable unimodal (F_1 and F_2), non separable problems ($F_3, \cdots, F_{10}$) and multimodal problems (F_{11}, F_{12}, F_{13}). The offset values used in the test functions are same as described in [5], with the exception of F_4 and F_6. For these 2 functions, the offset values were generated randomly. While this means the results are not precisely comparable, we generated offset values for these functions using the same formula as given in [5].

The problem sizes were 50D and 100D for each artificial test function. The maximum number of function evaluation was set at $10000 \times n$. The population sizes used in [5], were tested for sEDA-lite (i.e., 200, 500, 1000 and 2000). From this, a population size of 2000 was used for all functions except F_1 and F_2, where 200 was used since these very simple functions do not require a large population. Initially, rough experiments were conducted to determine reasonable algorithm parameter values: $\tau = \{0.1, 0.2, 0.3, 0.5\}$ and $\eta = 0.1$ were trialled, though not explored exhaustively. The parameter values that seemed to work best for each set of problems were then used. The algorithm stopped when the difference between the global optimum and the optimal values obtained from the algorithm is 1E-12 or it attained maximum number of function evaluations. The results reported are based on 25 repeated trials.

4.2 Circle in a Square Problem

Given the 2D unit square and a pre-specified number of circles, n_c, constrained to be of equal size, the circles in a square (CiaS) problem is to find an optimal packing; i.e. to position the circles and compute the radius length of the circles such that the circles occupy the maximum possible area within the square. All circles must remain fully enclosed within the square, and cannot overlap. The problem can be formulated as finding the positions of n_c points inside the unit square such that their minimum pairwise distance is maximized:

$$d_{n_c} = \max \min_{i \neq j} \parallel \mathbf{w}^i - \mathbf{w}^j \parallel_2 \tag{2}$$

$$\mathbf{w}^i \in [0,1]^2, i = 1, \ldots, n_c \tag{3}$$

The feasible search space of a CiaS problem is defined by the unit hypercube $[0,1]^{2n_c} \subset I\!R^{2n_c}$, and solutions outside this are infeasible. To ensure the generation of such candidate solutions by the EDAs, any value in a solution vector generated that lies outside the feasible region is reset to the (nearest) boundary.

For the (CiaS) problems, experiments were conducted on the number of circles, ranging from 2,...,50. The problem dimensionality n is equal to $2 \times n_c$. UMDA$_c$, EMNA$_{\text{global}}$, sEDA and sEDA-lite were implemented on this problem. The population size of all the algorithms was set to 2000 except sEDA, where the population was 50 times n. The value of τ for all the algorithms is set to 0.2. Since EMNA$_{\text{global}}$ is performing better than UMDA$_c$ in CiaS problem discussed in [14], the value of η as 0.5 was set for sEDA and sEDA-lite. The algorithm is stopped after 2E+06 function evaluations or if the difference between the best fitness value and the global optimum is 1E-04. The results were computed based on 25 repeated trials.

4.3 Location Allocation

In the continuous location allocation problem [18], the aim is to determine the location of n_f facilities in a 2D Euclidean space in order to serve customers at c fixed points so that the distances between each customer and the nearest facility are minimized [3]. There is no restriction on the capacity of the facilities to serve customers.
The (uncapacitated, continuous) location-allocation problem is formulated as follows:

$$\min \left(\sum_{j=1}^{c} \min_{i} d(X_i, A_j) \right)$$

where X is the vector consisting of the coordinates of the facilities. For n_f facilities problem, there are $2n_f$ variables for optimization. The $X_i th$ facility has coordinate values (x_i, x_{n_f+i}). A is the vector consisting of the given coordinates of the customers in the problem. For the $A_j th$ customer the coordinate values are represented as (a_{1j}, a_{2j}). $d\,(X_i, A_j)$ is the Euclidean distance from the location of facility X_i, to the location of a customer at fixed point A_j. For $n_f > 1$, this problem is known to be non convex and generally contains a large number of local minima [3].

In this paper we consider the widely used (e.g.[3,17]) 50-customer problem with a unit weight value for all the customers. The data A_i for the problem is given in [9]. For the 50-Customer problems, experiments were conducted using, $n_f = 5, 10, 15, 20, 25, 35$. The dimensionality of the problem $n = 2 \times n_f$. UMDA$_c$, EMNA$_{\text{global}}$, sEDA and sEDA-lite were compared on these problems. The population size of all the algorithms was set to 2000 except sEDA, where the population was $50 \times n$ to allow the algorithm a sufficient number of generations to converge. The value of τ for all the algorithms was 0.3, while the value of η was 0.1 for sEDA and sEDA-lite. The maximum number of function evaluations was $10000 \times n$. The results are over 25 repeated trials.

5 Results

5.1 Artificial Test Problems

The results of sEDA-lite are compared here with the values of UMDA$_c$, EEDA and EDA-MCC which are taken from [5]. The results of EMNA$_{\text{global}}$ are not repeated since it was previously found to perform worst on the test problems [5]. Comparative results for sEDA are also not reported here since it requires a prohibitive number of function evaluations for these larger-scale problems. The comparative results between UMDA$_c$, EEDA, EDA-MCC and sEDA-lite are listed in Table 5.1.

The results of the experiments for separable problems (F_1 and F_2), show that all the algorithms for 50D and 100D can solve these problems without difficulty. Functions from $F_3 \ldots F_{10}$ are non-separable problems with only a few local optima. On these functions, UMDA$_c$ and EEDA do not show the best performance. The performance of EDA-MCC is significantly better than rest of the algorithms in problems F_3 and F_5. Since the offset values are generated randomly, the previous solutions of F_4 and F_6 for EEDA and EDA-MCC are not reported here. We recomputed the results of UMDA$_c$ on these functions and compared with sEDA-lite for the same offset values. sEDA-lite clearly outperforms UMDA$_c$. The results also show EDA-MCC and sEDA-lite outperformUMDA$_c$ in 50D F_7 and F_8 functions. Solution comparison Table 5.1 shows that EEDA performs well on the 50D F_{10} function. Overall, from functions F_3 to F_{10}, the performance of sEDA-lite is similar to EDA-MCC.

Functions from F_{11} to F_{13} are multimodal functions. In these functions EDA-MCC and EEDA do not perform well. The performance of UMDA$_c$ and sEDA-lite are similar for function F_{11}, however, on the functions F_{12} and F_{13}, the performance of sEDA-lite exceeds UMDA$_c$. It is to be expected that there would be some variability in the relative performance of the algorithms. Overall, sEDA-lite is generally competitive and in some cases provides the best performance for these problems.

5.2 Results for the Circle in a Square (CiaS) Problems

The performance of the algorithms on a large set of CiaS problems (4D - 100D) is presented in Figure 1. The x-axis denotes the problem size (n_c) while the y-axis is a performance ratio given by $d_n/f(x_n)$, where d_n is the known global optimum and $f(x_n)$ is the solution found by the algorithm.

The results show that up to $n_c = 16$, UMDA$_c$, sEDA and sEDA-lite perform similarly. EMNA$_{\text{global}}$ does not perform as well, likely because it requires a larger population size and/or number of function evaluations. When $16 < n_c < 24$, sEDA actually performs slightly better than the other algorithms, but its performance then quickly degrades when $n_c > 24$. This is when the total budget of function evaluations for this experiment means that sEDA cannot perform sufficient generations, due to the requirement for calculating elementary effects values during execution. However sEDA-lite does not suffer from this, maintaining performance that is a little better than UMDA$_c$ up to $n_c = 50$.

Table 1. Solution quality comparison. Each cell contains the mean and standard deviation of the difference between the best fitness value and the global optimum. Bold font represents the best result. A "+" indicates a statistically significant difference (t-test, unequal variances, 0.05 level) when compared with sEDA-lite. A "-" sign indicates no significant difference. ζ indicates previous results for the algorithms which are incomparable due to the random values of the offset.

Prob.	Dim	UMDA$_c$	EEDA	EDA-MCC	sEDA-lite
F_1	50	**0±0**	**0±0**	**0±0**	**0±0**
	100	**0±0**	**0±0**	**0±0**	**0±0**
F_2	50	**0±0**	**0±0**	**0±0**	**0±0**
	100	**0±0**	5.3e-10±1.4e-09(-)	**0±0**	**0±0**
F_3	50	2.6e-04±1.5e-05(+)	1.8e-08±2.4e-09(+)	**0±0(+)**	4.2e-07±2.7e-08
	100	2.6e-02±8.3e-02(-)	1.5e-03±8.5e-04(+)	**0±0(+)**	2.5e-06±4.2e-06
F_4	50	4.1e+01±2.3e+00(+)	ζ	ζ	**3.6e+01±2.1e+00**
	100	5.3e+01±2.5e+00(+)	ζ	ζ	**4.8e+01±2.8e+00**
F_5	50	1.5e+01±4.1e+00(+)	2.4e-02±3.7e-03(+)	**0±0(+)**	1.0e+01±4.0e+00
	100	1.3e+02±2.7e+01(-)	3.8e-01±4.7e-02(+)	**0±0(+)**	1.2e+02±2.1e+01
F_6	50	6.5e+01±1.4e+01(+)	ζ	ζ	**3.8e+01±1.1e+01**
	100	1.1e+03±1.6e+02(+)	ζ	ζ	**8.6e+02±1.6e+02**
F_7	50	4.8e+01±3.4e-02(+)	5.0e+01±9.2e+00(+)	**4.7e+01±2.1e-01(-)**	**4.7e+01±2.4e-02**
	100	9.7e+01±6.4e-02(-)	9.7e+01±3.7e-01(-)	**9.6e+01±7.5e-02(+)**	9.7e+01±3.1e-02
F_8	50	4.1e+02±9.1e+02(+)	5.2e+02±1.0e+03(+)	**4.8e+01±1.5e-01(-)**	**4.8e+01±1.4e+00**
	100	9.3e+02±3.1e+03(-)	4.4e+04±4.4e+04(+)	**9.6e+01±1.3e-01(+)**	1.1e+02±2.8e+01
F_9	50	4.3e+07±4.1e+06(+)	4.1e+06±1.4e+06(+)	**3.6e+06±1.5e+06(+)**	4.0e+08±4.6e+07
	100	4.3e+07±3.1e+06(+)	2.2e+07±3.7e+06(+)	**9.6e+06±2.5e+06(+)**	2.5e+09±3.7e+08
F_{10}	50	4.9e+03±1.8e+02(+)	**2.0e+03±2.0e+03(+)**	3.1e+03±3.4e+02(+)	4.5e+03±1.6e+02
	100	5.9e+03±4.3e+02(+)	4.4e+03±6.0e+02(+)	**1.9e+03±3.6e+02(+)**	5.3e+03±3.4e+02
F_{11}	50	**0±0(-)**	3.1e+02±1.3e+01(+)	2.9e+02±1.4e+01(+)	**0±0**
	100	**0±0(-)**	7.3e+02±1.5e+01(+)	7.5e+02±1.6e+01(+)	**0±0**
F_{12}	50	2.1e+00±9.5e-01(+)	3.1e+02±1.7e+01(+)	3.0e+02±1.4e+01(+)	**1.5e+00±9.9e-01**
	100	8.6e+00±2.1e+00(+)	7.3e+02±2.5e+01(+)	7.4e+02±2.3e+01(+)	**0±0**
F_{13}	50	7.8e+00±8.3e-01(+)	2.7e+01±1.1e+00(+)	2.6e+01±9.2e-01(+)	**5.9e+00±5.4e-01**
	100	1.5e+01±2.0e+00(+)	3.8e+01±2.6e+01(+)	6.5e+01±1.6e+00(+)	**1.1e+01±7.3e-01**

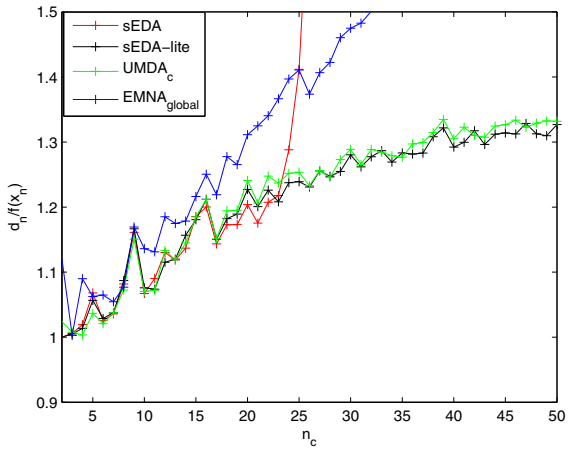

Fig. 1. Median Performance of UMDA$_c$, EMNA$_{\text{global}}$, sEDA and sEDA-lite on CiaS problems

5.3 Results for the 50-Customer Location Allocation Problem

Table 5.3 shows the results for UMDA$_c$, EMNA$_{\text{global}}$, sEDA and sEDA-lite. The performance of sEDA-lite is relatively good, particularly for high-dimensional problems. While it is possible that sEDA and EMNA$_{\text{global}}$ could give similar (or even better) performance, the amount of function evaluations required is prohibitive. Note however that there is some difference between these average performance results and the known optimal values. Also, these results are not as good as some of the previously reported results [3,17].

Table 2. Solution quality comparison (mean and standard deviation) for 50 Customer problem. Bold font represents the best result. A "+" indicates a statistically significant difference (t-test, unequal variances, 0.05 level) when compared with best result. A "-" sign indicates no significant difference.

(n_f)	Optimum	UMDA$_c$	EMNA$_{\text{global}}$	sEDA	sEDA-lite
5	72.2369	72.688±0.551(+)	77.454±3.650(+)	**72.560 ±0.591**	72.542±0.543(-)
10	41.6851	**43.401±0.823**	52.539±2.753(+)	47.930±3.809(+)	43.614±0.764(+)
15	27.6282	29.295±0.948(+)	42.991±3.171(+)	49.376±3.224(+)	**28.952±1.080**
20	19.3560	21.135±0.494(+)	36.292±2.559(+)	48.554±9.155(+)	**20.889±0.321**
25	13.3016	14.653±0.705(-)	34.402±2.772(+)	50.093 ±16.515(+)	**14.218±0.492**
30	8.7963	10.104±0.970(+)	31.237±2.173(+)	49.796±20.015(+)	**9.592±0.538**
35	5.0483	7.364±0.605(-)	29.917±2.293(+)	47.771±20.717(+)	**7.246±0.757**

6 Summary

This paper has proposed a modified version of the sEDA algorithm called sEDA-lite. Like the original algorithm, sEDA-lite is a Gaussian-EDA with a sparse covariance matrix model, that uses a screening technique to predict the important variables which is subsequently used to control covariance modelling. However, sEDA-lite achieve this without using any additional objective function values per generation to carry out the modelling. As a result, it can be effectively applied to high-dimensional problems without a prohibitive number of function evaluations. Experimentally, sEDA-lite has been shown to be competitive with UMDA$_c$, EEDA, EDA-MCC, EMNA$_{\text{global}}$ and sEDA in various problems.

References

1. Bosman, P.A.N., Grahl, J., Thierens, D.: Enhancing the performance of maximum–likelihood gaussian eDAs using anticipated mean shift. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN X. LNCS, vol. 5199, pp. 133–143. Springer, Heidelberg (2008)

2. Bosman, P.A.N.: On empirical memory design, faster selection of Bayesian factorizations and parameter-free Gaussian EDAs. In: Raidl, G., et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference — GECCO–2009, pp. 389–396. ACM Press, New York (2009)
3. Brimberg, J., Hansen, P., Mladenovic, N., Taillard, E.D.: Improvements and comparison of heuristics for solving the uncapacitated multisource Weber problem. Operations Research 48(3), 444–460 (2000)
4. Campolongo, F., Cariboni, J., Saltelli, A.: An effective screening design for sensitivity analysis of large models. Environmental Modelling & Software 22(10), 1509–1518 (2007)
5. Dong, W., Chen, T., Tino, P., Yao, X.: Scaling up Estimation of Distribution Algorithms for continuous optimization. IEEE Transactions 17(6), 797–822 (2013)
6. Dong, W., Yao, X.: Covariance matrix repairing in Gaussian based EDAs. In: IEEE Congress on Evolutionary Computation (CEC), pp. 415–422. IEEE (2007)
7. Dong, W., Yao, X.: Unified eigen analysis on multivariate Gaussian based Estimation of Distribution Algorithms. Information Sciences 178(15), 3000–3023 (2008)
8. Eiben, A.E., Smith, J.E.: Multimodal problems and spatial distribution. In: Introduction to Evolutionary Computing, pp. 153–172. Springer (2003)
9. Eilon, S., Watson-Gandy, C.D.T., Christofides, N.: Distribution management: mathematical modelling and practical analysis. Griffin, London (1971)
10. Hansen, N.: The CMA evolution strategy: a comparing review. In: Towards a New Evolutionary Computation, pp. 75–102. Springer (2006)
11. Karshenas, H., Santana, R., Bielza, C., Larrañaga, P.: Regularized continuous Estimation of Distribution Algorithms. Applied Soft Computing (2012)
12. King, D.M., Perera, B.J.C.: Morris method of sensitivity analysis applied to assess the importance of input variables on urban water supply yield–a case study. Journal of Hydrology 477, 17–32 (2013)
13. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer (2001)
14. Mishra, K.M., Gallagher, M.: Variable screening for reduced dependency modelling in gaussian-based continuous estimation of distribution algorithms. In: 2012 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2012)
15. Morris, M.D.: Factorial sampling plans for preliminary computational experiments. Technometrics 33(2), 161–174 (1991)
16. Pelikan, M., Goldberg, D.E., Lobo, F.G.: A survey of optimization by building and using probabilistic models. Computational Optimization and Applications 21(1), 5–20 (2002)
17. Salhi, S., Gamal, M.D.H.: A genetic algorithm based approach for the uncapacitated continuous location–allocation problem. Annals of Operations Research 123(1), 203–222 (2003)
18. Scaparra, M.P., Scutellà, M.G.: Facilities, locations, customers: Building blocks of location models. a survey. Technical Report TR-01-18, Universits' degli Studi di Pisa (2001)
19. Hansen, N., Büche, D., Ocenasek, J., Kern, S., Müller, S.D., Koumoutsakos, P.: Learning probability distributions in continuous evolutionary algorithms–a comparative review. Natural Computing 3(1), 77–112 (2004)
20. Wagner, M., Auger, A., Schoenauer, M.: EEDA: A new robust estimation of distribution algorithms. Technical Report INRIA RR-5190 (2004)