Parallel BMDA with Probability Model Migration

Josef Schwarz and Jiri Jaros

Department of Computer Systems, Faculty of Information Technology,
Brno University of Technology, CZ, {schwarz, jarosjir}@fit.vutbr.cz

Summary. The chapter presents a new concept of parallel Bivariate Marginal Distribution Algorithm (BMDA) using the stepping stone communication model with the unidirectional ring topology. The traditional migration of individuals is compared with a newly proposed technique of probability model migration. The idea of the new adaptive BMDA (aBMDA) algorithms is to modify the classic learning of the probability model (applied in the sequential BMDA [24]). In the proposed strategy, the adaptive learning of the resident probability model is used. The evaluation of pair dependency, using Pearson's chi-square statistics is influenced by the relevant immigrant pair dependency according to the quality of resident and immigrant subpopulation. Experimental results show that the proposed adaptive aBMDA significantly outperforms the traditional concept of migration of individuals.

1 Introduction

The concept of traditional parallel genetic algorithm (PGA) is well known. It stems from the idea that the large problem can be successfully solved using decomposition of the original problem into smaller tasks. Consequently, the tasks can be solved simultaneously using multiple processors.

This divide-and-conquer technique can be applied to GA in many distinct ways. Mostly, the population is divided into a few subpopulations or demes, and each of these demes evolves separately on different processors. Exchange of information among subpopulations is possible via a migration operator. In this context, the term island model is commonly used. Island populations are free to converge toward different optima. The migration

operator is supposed to mix good features that emerge locally in the different demes.

Many topologies can be defined for connecting the demes like mesh, torus, hypercube or ring. The most common models are the island model and the stepping stones model. In the basic island model, migration can occur between any subpopulations, whereas in the stepping stone model, migration is restricted to neighboring demes. In [7], the theory is published providing rational decisions for the proper setting of control parameters. An interesting survey of PGA is published in [2]. An effective technique for the massive parallelization of compact GA was published in [15]. An extremely prestigious PGA which is capable to solve billion-variable optimization problems was recently published in [10].

This chapter concerns the application of the stepping stone model (for simplicity we will use the term island-based model) for bivariate marginal distribution algorithm BMDA. This new approach using probability model migration is conceptually different from the traditional parallel genetic algorithms with migration of individuals/solutions and also from the EDAs using parallel building of pseudo-sequential probabilistic models.

The sections are organized as follows: Section 2 introduces the basic concept of EDA algorithm and current techniques used in the parallelization of the EDA algorithms. In Section 3 the sequential BMDA is described including the factorization and graphical representation of the probability model. Section 4 presents the motivation and a new idea of learning the probability model using a concept of probability model migration. Experimental results are shown in Section 5, Section 6 concludes the chapter.

2 Traditional EDAs

EDAs belong to the advanced evolutionary algorithms based on the estimation and sampling of graphical probabilistic models [4, 5, 6, 11, 13, 22, 23, 26]. They do not suffer from the disruption of building blocks known from the theory of standard genetic algorithms. The canonical sequential EDA is described in Fig. 1.

EDAs often surpass classical EAs in the number of required fitness function evaluations. However, the absolute execution time is still limiting factor which determines the size of practically tractable problems. Referring to Fig. 1 the most time consuming task is the estimation of probability model for many problems. Most papers on EDAs concentrate on parallel construction and sampling of probabilistic models. The well-known algo-

rithm employing parallel construction of Bayesian network is EBNA algorithm targeted for MIMD architecture and designed both for MPI and POSIX threads, published in [17, 20, 21]. In [25], the theory of population sizing and timing to convergence is published.

```
Set t ← 0;
Generate initial population D(0);
  While termination criteria is false do
  begin
    Select a set of promising solution D^s(t);
    Construct a new probability model M from D^s(t) using chosen metric;
    Sample offspring O(t) from M;
    Evaluate O(t);
    Create D(t+1) as a subset of O(t) ∪ D(t) with cardinality N;
    t ← t + 1;
  end
```

Fig. 1. The pseudo code of canonical EDA

A new idea of the multideme parallel estimation of distribution algorithm (PEDAs) based on PBIL algorithm was published in [1]. In [16], mixtures of distribution with Bayesian inference are discussed. Parallel learning of belief networks in large domains is investigated in [27]. Using the concept of PBIL algorithm [3, 12, 19], the classical phenomenon of migration in island-based EAs was carried over into probability distribution of EDAs. A new approach of probability vector crossover was implemented with very good performance.

2.1 Linkage learning in EDA algorithms

In competent genetic algorithms, various sophisticated linkage learning techniques must be implemented to discover Building Blocks (BBs). In EDA algorithms, the linkage learning is automatically incorporated into a graphical probabilistic model. EDAs support an effective detection, mixing and reproduction of BBs, so that they are capable to solve complex optimization problems including deceptive problems. The choice of the model complexity is very significant and it is determined by the fitness function complexity. We can recognize three categories of model complexity: without dependency (UMDA), pairwise dependency (MIMIC, BMDA) and multivariate dependency (BOA, EBNA).

2.2 Migration of probabilistic parameters for UMDA

The concept of migration of probabilistic parameters instead of individuals was firstly published in [8] where on UMDA platform the convex combination of univariate probability models is investigated for various network topologies (ring, star etc.).

Further enhancement of this concept is described in [9] where the local search methods are used to identify which parts of the immigrant model can improve the resident model.

In following sections we describe the proposal of a new concept of island-based BMDA algorithm with unidirectional ring topology based on the combination of two adjacent bivariate probability models.

3 Sequential BMDA

The well known representative of bivariate EDAs is the Bivariate Marginal Distribution Algorithm (BMDA) proposed by Pelikan and Mühlenbein [19, 24]. This algorithm uses a factorization of the joint probability distribution that exhibits second-order dependencies.

EDAs are also population based algorithm but unlike GAs the new population is generated by sampling the recognized probability model.

Let us denote:

$D = (X^0, X^1,..., X^{N-1})$ with $X \in D$, is the population of strings /solutions/individuals,

$\boldsymbol{X} = (X_0, X_1,..., X_{n-1})$ is a string/solution of length n with X_i as a variable,

$x = (x_0, x_1,..., x_{n-1})$ is a string/solution with x_i as a possible instantiation of variable X_i, $x_i \in \{0,1\}$,

$p(\boldsymbol{X}) = p(X_0, X_1,..., X_{n-1})$ denotes the n dimensional probability distribution,

$p(x_0, x_1,..., x_{n-1}) = p(X_0 = x_0, X_1 = x_1,..., X_{n-1} = x_{n-1})$ denotes a probability of a concrete n dimensional vector.

The probabilistic model used in BMDA can be formalized by $\mathcal{M} = (G, \Theta)$, where G is dependency graph and $\Theta = (\theta_0, \theta_2,\ldots, \theta_{n-1})$ is a set of parameters which are estimated by local conditional or marginal probability for each node/variable of the dependency graph.

A greedy algorithm for building dependency graphs is used. At the beginning, the root node is selected and subsequently the nodes with maximum dependency value are searched among the remaining nodes and joined. These pairwise dependencies in BMDA are discovered by Pearson's chi-square statistics:

$$\chi^2_{i,j} = N\left(\sum_{\forall x_i \in Dom(X_i)} \sum_{\forall x_j \in Dom(X_j)} \frac{m^2(x_i, x_j)}{m(x_i)m(x_j)} - 1\right) \quad (1)$$

where N is the size of parent population and $m(x_i, x_j)$, $m(x_i)$ resp. $m(x_j)$ denote the number of individuals in the parent population with concrete values of x_i and/or x_j. These values are stored in the contingency tables. From the theoretical point of view this metric can be seen as statistical testing of hypothesis – for example binary genes X_i and X_j are considered to be independent at 95 percent confidence level if $\chi^2_{i,j} < 3.84$. Like COMIT, BMDA also uses a variant of minimum spanning tree technique to learn a model.

However, during the tree construction, if none of the remaining variables can be "rooted" to existing tree, BMDA starts to form additional tree from remaining variables. The final probability distribution is thus a forest distribution (a set of mutually independent dependency trees):

$$p(\boldsymbol{X}) = \prod_{X_r \in R} p(X_r) \prod_{X_i \in V \setminus R} p(X_i \mid X_{j(i)}) \quad (2)$$

where V is the set of nodes of dependency tree, R is the set of root nodes and $X_{j(i)}$ denotes the parent node of X_i. Given the tree dependence structure, the univariate marginal probability distributions are estimated from the promising/parent population:

$$p(X_i = 1) = \frac{m(X_i = 1)}{N} \quad (3)$$

and the bivariate conditional probability distributions $p(X_i \mid X_{j(i)})$ are estimated as

$$p(x_i \mid x_{j(i)}) = \frac{m(x_i, x_{j(i)})}{m(x_{j(i)})} \quad (4)$$

For example, the joint probability distribution for the dependency graph in Fig. 2 can be expressed by the factorization:

1. $p(X) = p(X_4)\, p(X_3 \mid X_4)\, p(X_2 \mid X_3)\, p(X_1 \mid X_2)\, p(X_0 \mid X_1)$
2. $p(X) = p(X_2)\, p(X_3 \mid X_2)\, p(X_0 \mid X_4)\, p(X_4)\, p(X_1 \mid X_4)$

The time complexity of the complete BMDA algorithm can be expressed by the formula: $O(n^3)+O(4Nn^2)+O(Nn)$, where the first component is a cubical time complexity of the dependency graph construction, the second component is a quadratic time complexity of contingency tables

collection and the third component of the formula reflects a linear complexity of new solution sampling.

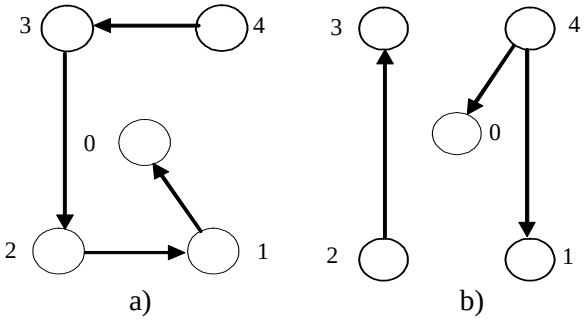

Fig. 2. Example of dependency graph for: a) COMIT, b) BMDA

4 Island-based BMDA

4.1 Migration of individuals

In traditional island-based PGA algorithms, an infrequent migration of individuals among subpopulation is incorporated. The migration process is controlled by several parameters. It is necessary to determine number and size of the subpopulations, the frequency and the intensity of migration and the method used for selection of candidate migrants. By analogy, it is possible to build island-based parallel BMDA, whereas the GA demes are replaced by BMDA ones. In BMDA and generally in EDAs, as it is known, new individuals are generated by the sampling of the probabilistic model. Consequently, a question pops up, whether it is possible to replace the migration of individual just by the probability model transfer. This topic is investigated in the next subsection.

4.2 Migration of the probabilistic model

The principal motivation for the proposal of a new concept of BMDA parallelization is to discover the efficiency of the transfer of probabilistic parameters in comparison with the traditional transfer of individuals. The main goal is to find a robust computational tool for hard optimization

problems. The present approaches recently published in [1, 8, 9] use a simpler probability model only (PBIL, UMDA).

In concordance with the theoretical conclusion shown in [24] and on the basis of experimental works done in [17], we used the island-based communication model with unidirectional ring topology with synchronization, see Fig. 3.

We have simulated the island-based system partly on a single processor computer and partly on a real parallel system composed of a cluster of eight Linux-based workstations. It is evident that we can simply decomposed the migration process in the ring loop into pairwise interactions of two adjacent islands - one of them is considered to be a resident island specified by resident probabilistic model and the second one is considered to be an immigrant island which probabilistic model is transferred to participate on the building up a new resident model after a predefined migration rate.

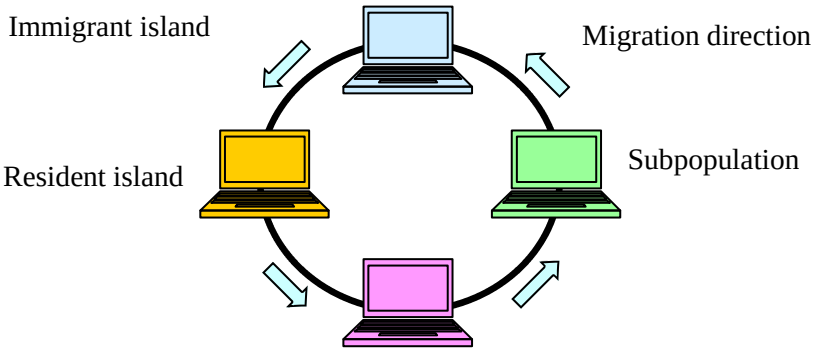

Fig. 3. Ring topology of island-based BMDA

We focused on the problem of how to compose the resident model with the incoming model belonging to the immigrant island. In general, the modification of the resident model by the immigrant model can be formalized by the adaptation rule [3, 19]:

$$\mathcal{M}'_R = \beta \ \mathcal{M}_R \circ (1-\beta)\, \mathcal{M}_I , \qquad (5)$$

where operator ° can be e. g. sum operator and the coefficient β in the range <0, 1> specifies the influence of the immigrant model.

4.3 Adaptive Learning of Probabilistic Model

We applied the adaptive learning for the both parts of the probabilistic model $\mathcal{M}_R = (G_R , \Theta_R)$ – the dependence graph G_R and the parameter set Θ_R. The new dependency graph G'_R is not built by the aggregation of the original graph G_R and the incoming graph G_I but by means of Pearson's chi-square statistics:

$$\chi^2_{i,j} = \beta\chi^2_{i_R,j_R} + (1-\beta)\chi^2_{i_I,j_I} \tag{6}$$

The new parameters Θ'_R are calculated by the simple adaptation rule:

$$\Theta'_R = \beta\,\Theta_R + (1-\beta)\Theta_I \tag{7}$$

The adaptation coefficient β is defined by the formula:

$$\beta = \begin{cases} \dfrac{F_R}{F_I + F_R} & \text{if } F_I \geq F_R \\ 0.9 & \text{otherwise} \end{cases}, \tag{8}$$

where F_R represents the mean fitness value of the resident subpopulation and F_I represents the mean fitness value of the immigrant subpopulation.

Procedure *(Output: $\mathcal{M}'_R$, Input:$SubPop_I$, $SubPop_R$)*
Calculate F_R for the resident subpopulation;
Calculate F_I for the immigrant subpopulation;
Calculate β:

$$\beta = \begin{cases} \dfrac{F_R}{F_I + F_R} & \text{if } F_I \geq F_R \\ 0.9 & \text{otherwise} \end{cases}$$

For *i=0 to n-1* ***do begin***
For *j=0 to n-1* ***do begin***
Calculate $\chi^2_{i_R,j_R}, \chi^2_{i_I,j_I}$;
Store in Chisqr_Table[i,j]: $\chi^2_{i,j} = \beta\chi^2_{i_R,j_R} + (1-\beta)\chi^2_{i_I,j_I}$
end
end
Build the new dependency graphs G'_R according ***Chisqr_Table****;*
Calculate set of the parameters: $\Theta_R(G'_R)$, $\Theta_I(G'_R)$;
Learning of the parameters: $\Theta'_R = \beta\,\Theta_R + (1-\beta)\Theta_I$
Store the new resident model: $\mathcal{M}'_R = (G'_R , \Theta'_R)$,
Sample the adapted model $\mathcal{M}'_R$;
Replacement of $SubPop_R$;
end

Fig. 4. Adaptive learning of the resident model

The major part of all experiments was implemented using this pseudo parallel version of the algorithm aBMDA, see Fig. 4

4.4 Parallel implementation on a cluster of workstations

In the parallel version of aBMDA, it is necessary to transfer some components of the probability model from the immigrant node to the resident one. In the proposed version, the contingency tables are transferred. The spatial complexity of all transported tables is $O(4n^2)$, where n is the cardinality of the solved problem. Seeing that chi-square is symmetric, and dependencies between the same variables have no sense, the spatial complexity can be reduced to $O(2(n^2 - n))$.

In contrast to the probabilistic model migration, the migration of individuals used in iBMDA works with the spatial complexity $O(nkN)$, where kN is the number of migrating individuals. Because the communication overhead in modern interconnection networks depends more strongly on the start-up latency of communication than on a transported message size, we can consider that the communication overhead will be nearly the same for both approaches. Moreover, using an asynchronous or non-blocking type of migration [14], the communication overhead could be simply overlapped.

Our parallel implementation of aBMDA derives benefits from overlapping of communication and computation, based on non-blocking MPI [18] communication subroutines MPI_ISend, MPI_IRecv and MPI_Wait. The basic idea is shown in Fig. 5.

The information exchange between the resident and the immigrant node begins with the initiation of receiving request. During the receiving procedure, the resident node can compute its contingency tables and the mean fitness value F_R of the resident population. Next, all computed data are packed into a simple send buffer using standard C routine *memcpy* and sent using non-blocking communication to the neighbor node. The resident chisqr-table is computed from the resident contingency tables in the next step. Now, the resident node has to wait until the immigrant data are completely received. After finalization, the data from the immigrant node are unpacked from a receive buffer and the immigrant chisqr-table is computed. Now, the probabilistic model composition can be started. First, the resident and the immigrant chisqr-tables are combined together using beta parameter to produce a new chsqri-table. A new dependency graph is created according to the information stored in the learned chisqr-table. Second, a set of parameters Θ'_R are calculated using new dependency graph,

and the original resident and the immigrant contingency tables. As a result the new probabilistic model $\mathcal{M}'_R = (G'_R, \Theta'_R)$, is determined.

Procedure MakeExhangeIslandInformation();
MPI_IRecv*(Receive buffer);*
Calculate the mean fitness value F_R of the resident island;
Calculate resident contingency tables;
Pack resident contingency tables and F_R into a send buffer;
MPI_ISend*(Send buffer);*
Calculate Chisqr_Table_Resident[i,j] = $\chi^2_{i_R, j_R}$
MPI_Wait*(Waiting for receiving finish);*
Unpack immigrant contingency tables and F_I from a receive buffer;
Calculate β;
Calculate the Chisqr_Table_Imigrant[i,j]= $\chi^2_{i_I, j_I}$ *;*
Calculate items of the composed ***Chisqr_Table[i,j]:*** $\chi^2_{i,j} = \beta\chi^2_{i_R, j_R} + (1-\beta)\chi^2_{i_I, j_I}$
Build the new dependency graphs G'_R according to new ***Chisqr_Table****;*
Calculate set of the parameters: $\Theta_R(G'_R)$, $\Theta_I(G'_R)$ using contingency tables ;
Learning of the parameters: $\Theta'_R = \beta\,\Theta_R + (1-\beta)\Theta_I$
Compose new resident model: $\mathcal{M}'_R = (G'_R, \Theta'_R)$
MPI_Wait*(Waiting for sending finish);*
end

Fig. 5. MPI communication between the resident and the immigrant node

The migration of individuals used in iBMDA can be realized in the similar way. The computation of contingency tables and mean fitness value is simply replaced by the selection of individuals intended for the migration. In this case, only selected individuals are packed into a send buffer and transported to the neighbor node. Received solutions are then unpacked in the resident node and incorporated into resident population. Finally, a new population is created in the standard way.

Besides the described type of communication, the MPI_Gather [18] operation was employed after each generation. During this operation, all necessary information from all processing nodes are collected to compute global statistics including the global mean fitness value, the best global solution, etc.

5 Experimental Results

In our experiments, we compared four different variants of the BMDA algorithm. The first group consists of two versions of parallel BMDA algorithm:

1. aBMDA, with adaptive learning of dependency graph.
2. iBMDA, with the migration of individuals.

These two parallel BMDA algorithms work with 8 island subpopulations, each consisting of 256 individuals as a portion of the full population with 2048 individuals.

The second group used for the comparison includes two classical variants of BMDA:

3. sBMDA, sequential BMDA, with full population of 2048 individuals (as the whole eight-island model).
4. oBMDA - sequential BMDA with reduced population consisting of 256 individuals (as in case of one island).

The fixed subpopulation size has been used for the whole range of problem size. We have not wittingly used the possibility of the adaptation of the subpopulation size according to problem size as discussed in [25]. Our goal was to compare namely the parallel adaptive aBMDA version with traditional iBMDA version under limited resources (subpopulation size). The value of the population size for sBMDA is set to 2048 derived partially from our experience and from the experimental results published in [25] for the 3-Deceptive problem.

In all BMDA variants, truncation-based selection strategy was used, i.e. all individuals were ordered by their fitness value and the better half was used for model building. The truncation-based replacement strategy was also used for the replacement operator, i.e. the new generated solutions (offspring) replace the worse half of the subpopulation. The probabilistic model is built in each generation. Frequency of the model migration or individual migration was even - once per five generations. In case of the algorithm with migrating individuals, the elitism is used, that is, 13 best individuals of the immigrant subpopulation (i.e. about k=5 percent of the subpopulation) replace the worse individuals of the resident subpopulation. First stop condition was met after 500 generations; the second condition was activated if there was no improvement in the interval of 50 generations.

5.1 Specification of Benchmarks

For our experimental study, four well known benchmarks with various complexity and known global optimum were used. The OneMax and TwoMax problems served as the basic benchmarks for the testing of the basic performance. The Quadratic problem represents the adequate benchmark that should be solvable just by any BMDA algorithm. The 3-Deceptive task belongs to the hard deceptive benchmark for BMDA and it is often used for the testing of BOA algorithms.

OneMax: $$f_{OneMax}(x) = \sum_{i=0}^{n-1} x_i \tag{9}$$

TwoMax: $$f_{TwoMax}(x) = \left| \sum_{i=0}^{n-1} x_i - \frac{n}{2} \right| + \frac{n}{2} \tag{10}$$

Quadratic: $$f_{Quadratic}(x) = \sum_{i=0}^{\frac{n}{2}-1} f_2(x_{\pi(2i)}, x_{\pi(2i+1)}) \tag{11}$$

$$\text{where } f_2(u,v) = 0.9 - 0.9(u+v) + 1.9uv$$

3-Deceptive: $$f_{3-Deceptive}(x) = \sum_{i=0}^{\frac{n}{3}-1} f_3(x_{\pi(3i)} + x_{\pi(3i+1)} + x_{\pi(3i+2)}) \tag{12}$$

$$\text{where } f_3(u) = \begin{cases} 0.9 & if \quad u = 0 \\ 0.8 & if \quad u = 1 \\ 0 & if \quad u = 2 \\ 1 & otherwise \end{cases}$$

The four mentioned objective functions were used to form fitness functions (FF) without additional modification. We have tested four variants of BMDA using 30 independent runs. To have baseline to island based versions, we first tested the classic sequential BMDA (sBMDA) with ordinary population of 2048 individuals and the classical sequential BMDA with reduced population (oBMDA).

The first metric is represented by the often used success rate of the global optimum discovery. The second metric is calculated as the average value of the best fitness function (FF) over 30 runs. The third metric is computed as the mean value of the number of correctly discovered buildings blocks (BBs) over 30 runs. These metrics/statistics are discussed in the next sections.

5.2 OneMax Problem

The sBMDA and aBMDA algorithms succeeded in the whole range of the problem size, see Fig. 6. Classical iBMDA version produces comparative result only up to 260 variables. The rapid drop follows after this threshold. oBMDA was very significantly outperformed by all algorithms.

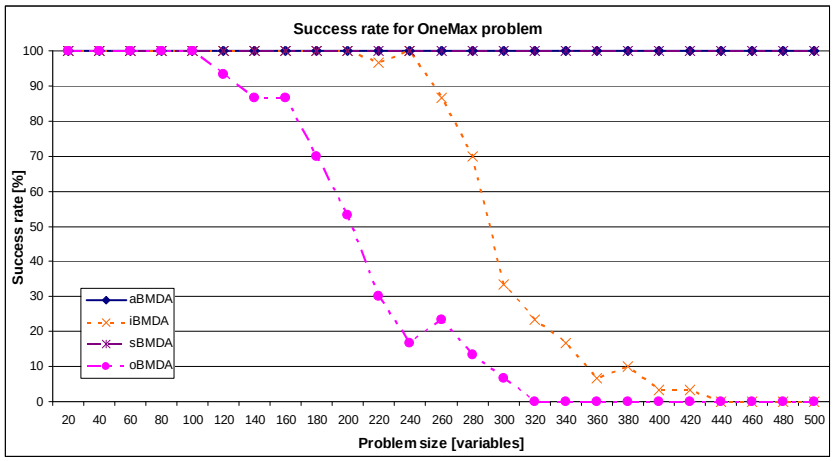

Fig. 6. Success rate for OneMax problem

5.3 TwoMax Problem

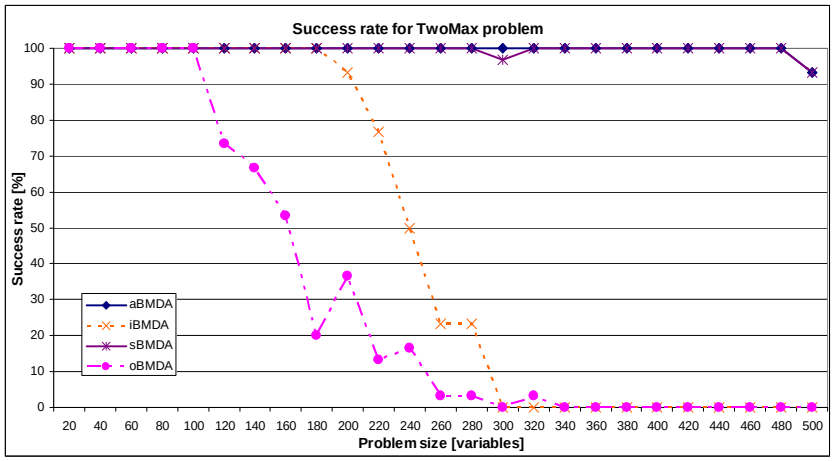

Fig. 7. Success rate for TwoMax problem

In case of TwoMax problem, see Fig. 7, the results of the tested algorithms are similar to the results achieved for OneMax problem. The aBMDA version outperformed all other versions and achieved the same results as sBMDA. The drop of success rate for the iBMDA version with migration of individuals is stronger than in the case of OneMax problem.

5.4 Quadratic Problem

To achieve global solution for this problem, the probability model with the bivariate dependency is required. This benchmark is thus perfectly suitable for testing and comparing all BMDA variants.

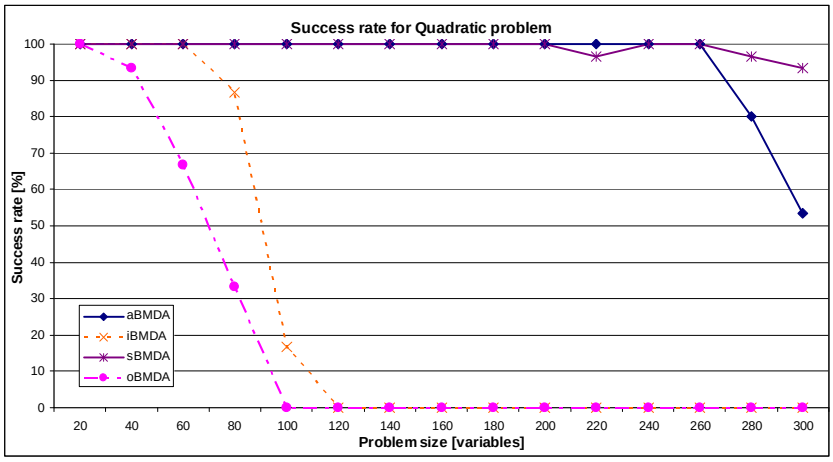

Fig. 8. Success rate for Quadratic problem

In Fig. 8, the success rate for all compared algorithms can be seen. The best results were reached by sBMDA that succeeded in the nearly whole range of the problem sizes. The similar behavior can be observed even for aBMDA version that achieved 100 percent success rate up to 260 variables.

Besides the success rate metric the second metric represented by mean±std statistics of the fitness function are presented in Table 1. It is evident that aBMDA version and sBMDA version provide the same results up to 260 variables. For higher number of variables sBMDA achieves better results. The best value achieved for each problem size is written in bold.

Table 1. Statistics results (mean±std of FF) for Quadratic problem

Problem	Algorithm				
size	aBMDA	iBMDA	sBMDA	oBMDA	Optimum
60	**30.0±0.00**	**30.0±0.00**	**30.0±0.00**	29.9±0.06	30
80	**40.0±0.00**	39.9±0.30	**40.0±0.00**	39.8±0.10	40
100	**50.0±0.00**	49.8±0.80	**50.0±0.00**	49.6±0.14	50
120	**60.0±0.00**	59.7±0.13	**60.0±0.00**	59.3±0.24	60
140	**70.0±0.00**	67.7±3.14	**70.0±0.00**	69.1±0.25	70
260	**130.0±0.00**	127.1±0.37	**130.0±0.00**	126.3±0.57	130
280	139.9±0.24	136.5±0.30	**139.9±0.02**	135.8±0.54	140
300	149.7±0.46	146.1±0.27	**149.9±0.02**	145.3±0.64	150

In Table 2, the third metric represented by mean±std statistics for the number of correctly recognized buildings blocks (BBs) are shown.

Table 2. Statistics results (mean±std of BBs) for Quadratic problem

Problem	Algorithm				
size	aBMDA	iBMDA	sBMDA	oBMDA	Optimum
60	**30.0±0.00**	**30.0±0.00**	**30.0±0.00**	29.6±0.61	30
80	**40.0±0.00**	39.8±0.37	**40.0±0.00**	38.8±1.02	40
100	**50.0±0.00**	48.7±0.78	**50.0±0.00**	46.5±1.41	50
120	**60.0±0.00**	56.7±1.26	**60.0±0.00**	53.8±2.48	60
140	**70.0±0.00**	64.5±1.36	**70.0±0.00**	60.8±2.38	70
260	**130±0.00**	101±3.71	**130±0.00**	93.4±5.86	130
280	129±2.45	105±3.27	**140±0.00**	98.0±5.15	140
300	147±4.64	111±2.77	**149.9±0.25**	104±4.93	150

5.5 3-Deceptive Problem

The problem was investigated for the variable range from 21 to 120, see Fig. 9. For higher number of variables, the drop of success rate is significant for all proposed algorithms. It is caused by rather high complexity of the 3-Deceptive problem that requires a more complex model and also larger population size for efficient performance. The best success rate was achieved by aBMDA version. Very similar values were also achieved by sBMDA. On the other hand, the worst results were obtained by oBMDA and iBMDA.

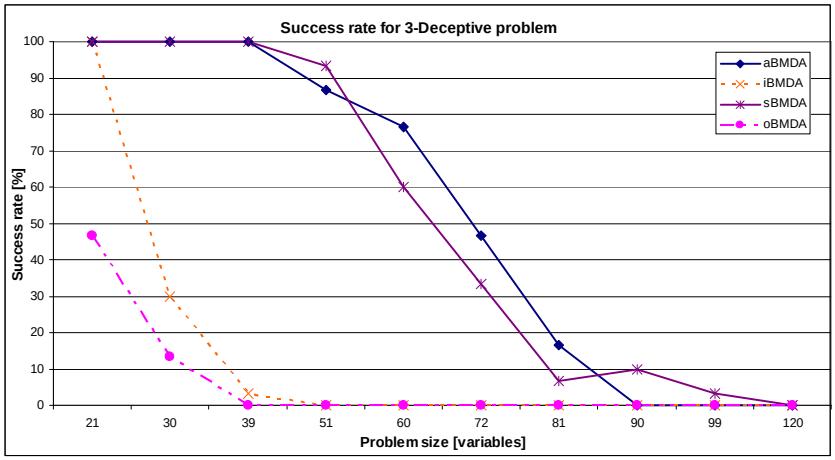

Fig. 9. Success rate for 3-Deceptive problem

In Table 3, the mean±std statistics of the fitness function are presented. The best results were obtained by the aBMDA version and by sBMDA. The worst mean fitness values were achieved by oBMDA algorithm followed by iBMDA. The mean values and standard deviation of the discovered BBs are presented in Table 4. The adaptive aBMDA version proves significant correlation between the mean value of fitness and the mean value of BBs. For the case of the 99–variable problem the mean number of BBs is 26.3 which is 80 percent of total 33 blocks. Note that iBMDA discovered only 11.9 BBs (36 percent). It is interesting to compare these values with the experimental results published for BOA algorithm in [25], where the achieved number of building blocks (BBs) for 99 variables and for the population size estimated to 250 equals to 25 percent.

Table 3. Statistics results (mean±std of FF) for 3-Deceptive problem

Problem	Algorithm				
size	aBMDA	iBMDA	sBMDA	oBMDA	Optimum
21	**7.00±0.00**	**7.00±0.00**	**7.00±0.00**	6.90±0.05	7
30	**10.0±0.00**	9.92±0.06	**10.0±0.00**	9.75±0.12	10
39	**13.0±0.00**	12.7±0.11	**13.0±0.00**	12.1±1.99	13
51	16.9±0.04	16.3±0.09	**16.9±0.03**	16.1±0.16	17
60	**19.9±0.06**	19.1±0.13	**19.9±0.06**	18.8±0.22	20
72	23.9±0.15	22.7±0.14	**23.9±0.09**	22.4±0.24	24
81	26.8±0.15	25.4±0.16	**26.8±0.10**	25.1±0.23	27
90	29.5±0.20	28.1±0.15	**29.7±0.16**	28.6±0.57	30
99	32.4±0.35	30.8±0.15	**32.6±0.18**	30.5±0.22	33
120	38.6±0.29	37.1±0.14	**39.2±0.2**	38.6±0.59	40

Table 4. Statistics results (mean±std of BBs) for 3-Deceptive problem

Problem size	Algorithm				
	aBMDA	iBMDA	sBMDA	oBMDA	Optimum
21	**7.00±0.00**	**7.00±0.00**	**7.00±0.00**	6.47±0.50	7
30	**10.0±0.00**	9.23±0.56	**10.0±0.00**	7.60±1.11	10
39	**13.0±0.00**	10.6±1.11	**13.0±0.00**	7.33±1.72	13
51	16.8±0.45	10.6±0.92	**16.9±0.39**	8.27±1.67	17
60	**19.7±0.59**	11.3±1.49	19.6±0.56	7.23±2.39	20
72	**23.1±1.02**	11.5±1.56	23.0±0.93	8.23±2.29	24
81	25.0±1.51	11.5±1.62	**25.3±0.98**	7.92±2.39	27
90	25.8±2.08	11.3±1.41	**27.8±1.61**	8.50±2.26	30
99	26.3±2.99	11.9±1.83	**29.4±1.85**	8.67±2.95	33
120	26.2±3.72	11.6±1.80	**32.1±2.09**	9.03±1.74	40

5.6 Discussion to pseudo-parallel version of algorithms

In our experiments two groups of algorithms are compared:

1. The new proposed island-based aBMDA with probabilistic model learning and the traditional island-based iBMDA with individual migration.
2. Sequential sBMDA version with the full population size and reduced sequential oBMDA version.

In the first experiment, the success rate metric was applied. Both aBMDA and sBMDA versions are capable to find global optima with 100 percent success rate up to 500 variables in case of OneMax and TwoMax problems and up to 260 variables in case of Quadratic problems. For difficult problems, like 3-Deceptive, the algorithms lack the ability to find repeatedly the optimal solution for problem size larger than 39.

It is evident that aBMDA is effective optimization tool outperforming iBMDA version based on the traditional migration of individuals. From this point of view the range of solvable problem size is at least two times larger in case of aBMDA version.

In the second experiment the statistics including mean±std values of fitness function (FF) was processed for two harder problems – the Quadratic problem in Table 1 and 3-Deceptive problem in Table 3. The best values are written in bold. From Table 1, it is evident that for Quadratic problem, aBMDA and sBMDA have reached the global optima up to 260 variables and outperformed very significantly iBMDA version. In the case of 3-Deceptive problem, aBMDA outperformed all other algorithms besides the sBMDA that is better for the problem size exceeding 90 variables. Note that for the 120-variable problem the mean value of FF in case of aBMDA equals to 38.6 which is close to global optimum represented by value 40.

In Table 2 and Table 4, the statistics results for BBs are shown for Quadratic and 3-Deceptive problems. From Table 2 it is evident, that in case of Quadratic problem, aBMDA discovered all BBs up to 260 variables while iBMDA was successful up to 30 variables only. In case of 3-Deceptive problems, see Table 4, aBMDA outperformed iBMDA in the whole range of problem size. Note that aBMDA achieved approximately two time higher mean value of BBs for the problem size exceeding 60 variables.

The computational complexity of all algorithms measured by the number of generation is comparable. For example, in case of Quadratic problem with 60 variables the average computational time is about 20 generations, see Fig. 10. Note that oBMDA was able to find the global optima for this instance of Quadratic problem only in 66 percent of the 30 runs, see Fig. 8.

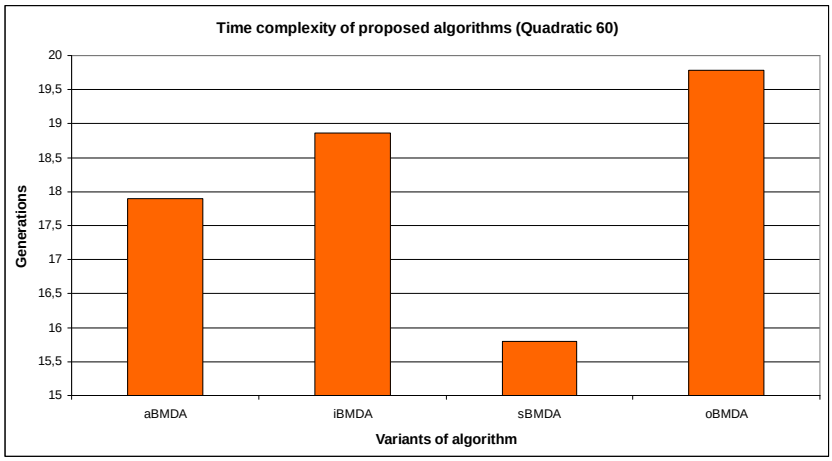

Fig. 10. Time complexity of the proposed algorithms for Quadratic problem

5.7 Performance of parallel implementation

The parallel implementation of aBMDA and iBMDA was tested on a cluster of 8 Linux-based workstations equipped with Intel E6550 processor and 2GB RAM connected together by 1Gb LAN network. In case of sequential sBMDA only one station was used.

First, the comparison of the mean execution time T_G related to one generation was performed. For T_G calculation, five independent runs, each

composed of 20 generation (including 4 migration cycles), were carried out.

The algorithms were compared using OneMax benchmark, see Fig. 11. Let us note that the convergence toward global optima was not checked in this case. From Fig. 11, a marked difference between sequential and parallel approaches is evident, as it was expected. The values of T_G for the both parallel algorithms are comparable. Finally, OneMax benchmark is relatively simple, as the fitness function evaluation does not influence the execution time very much. For more complex problems, a gap between sequential and parallel approaches will be deeper, because computational complexity will dominate communication complexity.

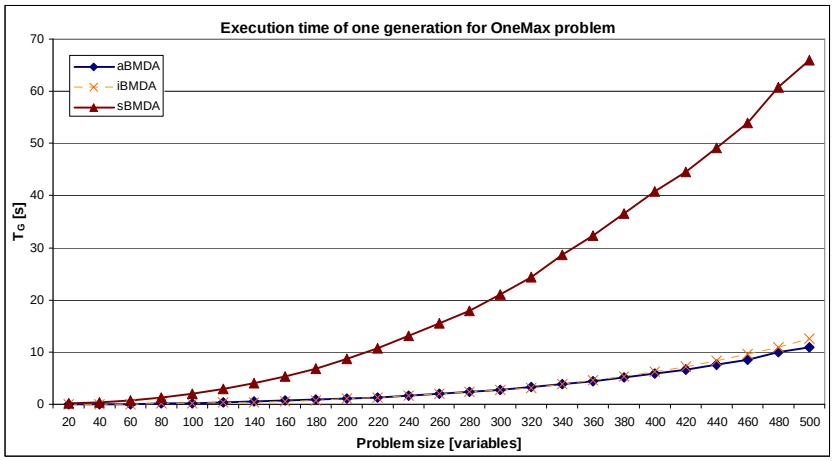

Fig. 11. The mean execution time T_G for OneMax

The speed-up of parallel implementations for OneMax problem is displayed in Fig. 12. It varies between 5 and 9 for both parallel algorithms. The decrease of the speed-up for larger instances of OneMax problem is caused by the increased parallelization overhead. The overhead consists of the quadratic time complexity of contingency tables transport and the subsequent model composition in the resident node.

The performance of aBMDA algorithm is slightly better. For simpler instances (say up to 200 variables) the achieved speed-up was larger than a number of processing nodes (8 in our experiment) – it is known as the phenomenon of superlinear speed-up [3].

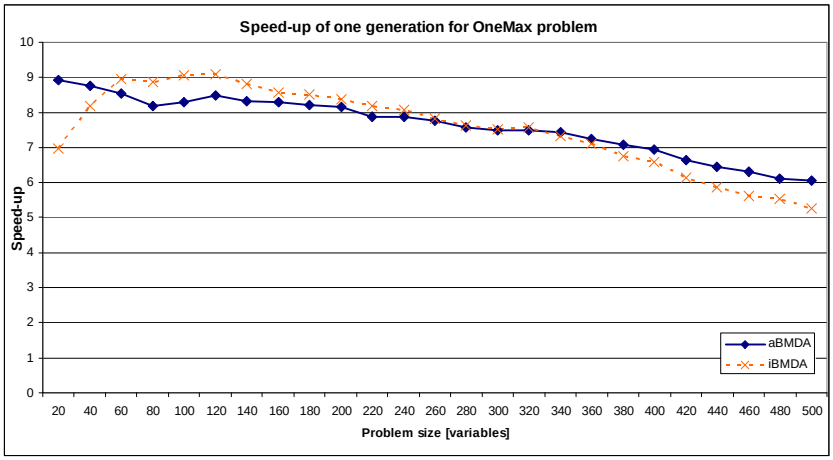

Fig. 12. Speed-up of iBMDA and aBMDA with regard to sBMDA for OneMax problem

The value of speed-up was also investigated for Quadratic problem, see Fig. 13. Both parallel algorithms achieved superlinear speed-up in the whole range of problem instances as the consequence of the more complex benchmark.

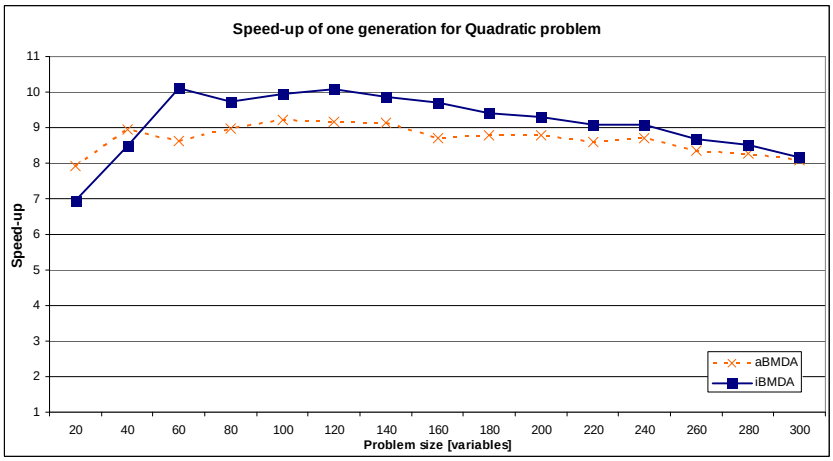

Fig. 13. Speed-up of iBMDA and aBMDA with regard to sBMDA for Quadratic problem

Let us note that the knowledge of the concrete speed-up can be utilize for prediction of the execution time of the optimization process and also

for the setting of proper population size and the number of processing nodes.

Finally, we investigated the speed-up of the optimization tasks which resulted in 100% success rate. The value of the speed-up was calculated using the following schema: a) first the number of generation required for achievement the global optima for each version of BMDA and for each instance of the problem was measured and averaged during 30 independent runs, b) this value was multiplied by the mean execution time of one generation T_G for relevant version of BMDA, c) finally, this value was normalized according the values obtained by sequential sBMDA and plotted in the Fig. 14 and Fig. 15.

From Fig. 14, it is evident that iBMDA algorithm was not able to find optimal solutions for the whole range of OneMax instances. It succeeded only in the range from 20 to 220 variables. aBMDA solved the tasks reliably with the speed-up higher than 6 and even with superlinear speed-up up to 280 variables.

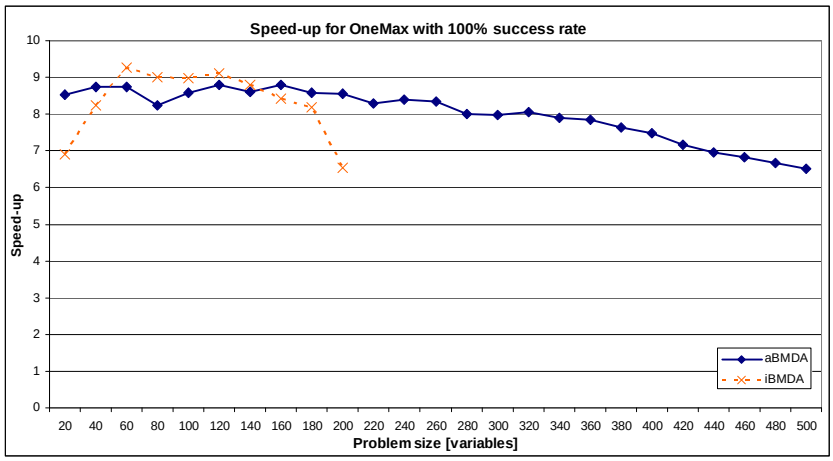

Fig. 14. Speed-up of iBMDA and aBMDA with regard to sBMDA for OneMax problem in case of 100% success rate

The capability of the speed-up for Quadratic problem (Fig. 15) was investigated up to 260 variables, whereas aBMDA was still able to produce optimal solutions with 100% success rate. The speed-up varied between 4.4 and 8.7. iBMDA algorithm achieved comparable speed-up as aBMDA but only up to 60 variables.

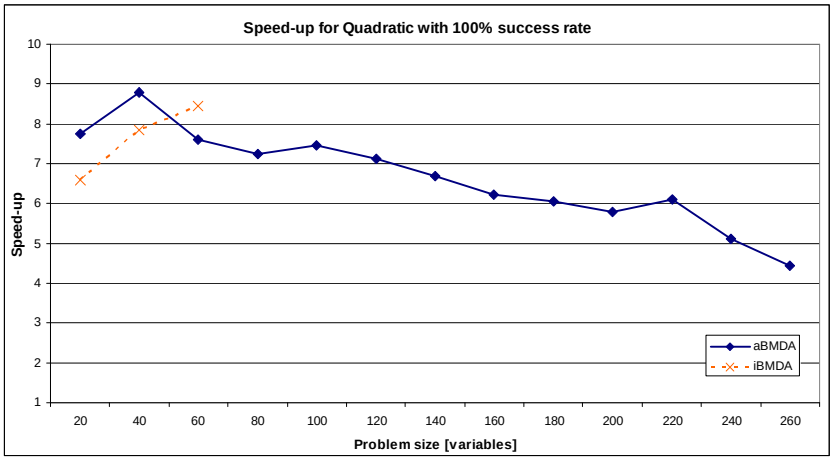

Fig. 15. Speed-up of iBMDA and aBMDA with regard to sBMDA for Quadratic problem in case of 100% success rate

5.7.1 Discussion to parallel implementation

The message passing interface MPI used for parallel implementation of BMDA algorithms appears as an efficient software tool. Two variants of parallel BMDA was investigated – aBMDA and iBMDA. They are mutually compared and also compared to the sequential sBMDA.

From Fig. 12 and Fig. 13 it is possible to conclude, that the communication overhead is negligible. The values of achieved speed-up of processing were very close to the number of utilized processing nodes, and even higher in some cases. The ability of finding global optima and its speed-up was also examined for both tested parallel algorithms. From Fig. 14 and Fig. 15, it is evident that aBMDA algorithm is much more robust than iBMDA. The speed-up of optimization process with respect to sBMDA (with 100 percent success rate) varies from 4.4 to 8.7, but for the most of problem instances it is higher than 6.

6 Conclusions

This chapter has presented a new idea of parallel BMDA algorithms using island-based model with the unidirectional ring topology. The cooperation of demes was realized via migration of probabilistic models instead of the traditional migration of individuals. Each two neighbor demes are organized as a pair of resident and immigrant demes. We have introduced an

adaptive learning technique, based on the quality of resident and immigrant subpopulation, which consists of the adaptation of the resident probabilistic model by the incoming neighbor immigrant model.

We have introduced an efficient tool how to learn graphical probabilistic model and associated probabilistic parameters. The comparative experimental studies demonstrated that the proposed parallel algorithm aBMDA outperforms the traditional iBMDA using migration of individuals, mainly according to the problem size and the level of the success rate. This is true for all the applied benchmarks beginning with OneMax and ending with the Quadratic problem. The 3-Deceptive problem is a hard problem for all compared algorithms but aBMDA and sBMDA provide relatively best solutions. Note that the sequential sBMDA with full population provides competitive results compared with the aBMDA, indeed but the time complexity of aBMDA version can be significantly reduced by parallel processing.

The future work will be focused on more sophisticated modifications of learning techniques with limited size of parameter transfer. We also aim to parallelize the Bayesian Optimization Algorithm (BOA) using a modified concept of probabilistic model migration.

Acknowledgement

This work was partially supported by the Grant Agency of the Czech Republic under No.~102/07/0850 Design and hardware implementation of a patent-invention machine and the Research Plan No. MSM 0021630528 - Security-Oriented Research in Information Technology.

References

1. Ahn CW, Goldberg DE, Ramakrishna RS (2003) Multiple-Deme Parallel Estimation of Distribution Algorithms: Basic Framework and Application. (Illigal Report No. 2003016)
2. Alba E, Troya JM (1999) A Survay of Parallel Distributed Genetic Algorithms. Complexity, vol. 4, pp 303-346
3. Baluja S, (1994) Population-Based Incremental Learning: A method for Integrating Genetic Search Based Function Optimization and Competitive Learning. (Technical report CMU-CS-94-163, Carnegie Mellon University)
4. Bosman PAN, Thierens D (1999) An Algorithmic Framework For Density Estimation Based Evolutionary Algorithms. (Technical report UU-CS-1999-46, Utrecht University)

5. Bosman PAN, Thierens D (1999) Linkage information processing in distribution estimation algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99. Orlando, Florida, Morgan Kaufmann Publishers, San Fransisco, CA, vol I, pp 60-67
6. Bosman PAN, Thierens D (2000) Continuous iterated density estimation evolutionary algorithms within the IDEA framework. In Proceedings of the Optimization by Building and Using Probabilistic Models OBUPM workshop. Genetic and Evolutionary Computation Conference GECCO-2000, pp 197-200
7. Cantu-Paz E (2000), Efficient and Accurate Parallel genetic algorithm. Kluwer Academic Publishers
8. delaOssa L, Gámez JA, Puerta JM (2004) Migration of probability models instead of individuals: an alternative when applying the island model to edas In: Parallel Problem Solving from Nature - PPSN VIII, vol 3242 of LNCS, Springer, pp 242–252
9. delaOssa L, Gámez JA, Puerta JM (2005) Improving model combination through local search in parallel univariate EDAs, Evolutionary Computation, vol. 2, pp 1426-1433
10. Goldberg DE, Sastry K, Llorà X (2007) Toward routine billion-variable optimization using genetic algorithms. Complexity vol. 12 pp 27–29
11. Heckerman D, Geiger D, Chickering M (1994) Learning Bayesian networks: The combination of knowledge and statistical data. (Technical Report MSR-TR-94-09, Microsoft Research, Redmond, WA)
12. Höhfeld M, Rudolph G (1997) Towards a theory of population-based incremental learning. In: Proceedings of 4th International Conference on Evolutionary Computation, IEEE Press, pp 1-5
13. Larrañaga P, Lozano J. A. (2002) Estimation of Distribution Algorithms. Kluwer Academic Publishers, London ISBN 0-7923-7466-5.
14. Lin SC, Punch WF, Goodman ED (1994) Coarse-grain parallel genetic algorithms: categorization and a new approach In Sixth IEEE Conference on Parallel and Distributed Processing, IEEE Press, Piscataway, NJ, pp 28-37
15. Lobo FG, Lima CF, Martires H (2005) Massive parallelization of the compact genetic algorithm. In: Ribeiro R. (ed) Proceedings of the International Conference on Adaptive and Natural Computing Algorithms (ICANNGA-2005), Springer, pp 530-533
16. Marin JM, Mengersen K, Robert CP (2005) Bayesian Modelling and Inference on Mixtures of Distribution. In Dey D, Rao CR (eds) Handbook of Statistics 25, pp 459-507
17. Mendiburu-Alberro A (2006) Parallel implementation of estimation of Distribution Algorithms based on probabilistic graphical models. Application to chemical calibration models (PhD thesis, the University of Bascue Country, Donostia-San Sebastian)
18. MPI-2 Extension to message-Passing Interface, Message Passing Interface Forum, 2003, Document available at url: http://www.mpi-forum.org/docs/mpi2-report.pdf
19. Mühlenbein H (1997) The equation for response to selection and its use for prediction. In: Evolutionary Computation 5(3). pp 303-346

20. Ocenasek J (2002) Parallel Estimation of Distribution Algorithms. (PhD. Thesis, Faculty of Information Technology, Brno University of Technology, Brno, Czech Rep.)
21. Ocenasek J, Schwarz J, Pelikan M (2003) Design of Multithreaded Estimation of Distribution Algorithms. In: Genetic and Evolutionary Computation Conference - GECCO 2003. Springer Verlag: Berlin, pp 1247-1258
22. Pelikan M, Goldberg DE, Cantú-Paz E (1998) Linkage problem, distribution estimation, and Bayesian networks. (IlliGAL Report No. 98013, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, Illinois)
23. Pelikan M, Goldberg DE, Lobo F (1999) A survey of optimization by building and using probabilistic models. (IlliGAL Report No. 99018, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, Illinois)
24. Pelikan M, Mühlenbein H (199) The bivariate marginal distribution algorithm. In: Advances in Soft Computing - Engineering Design and Manufacturing. Springer-Verlag, London, pp 521-535
25. Pelikan M, Sastry K, Cantú-Paz E (2000) Bayesian Optimization Algorithm, Population Sizing and Time to Convergence. (IlliGAL Report No. 2000001, Illinois Genetic Algorithm Laboratory, University of Illinois at Urbana-Champaing, p 13)
26. Pelikan M, Sastry K, Goldberg DE (2001) Evolutionary Algorithms + Graphical Models = Scalable Black-Box Optimization. (IlliGAL Report No. 2001029, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, Illinois)
27. Xiang Y, Chu T (1999) Parallel Learning of Belief Networks in Large and Difficult Domains. In: Data Mining and Knowledge Discovery, vol 3, pp 315-339