Structure vs. Efficiency of the Cross-Entropy Based Population Learning Algorithm for Discrete-Continuous Scheduling with Continuous Resource Discretisation

Piotr Jędrzejowicz and Aleksander Skakovski*

Abstract. In the chapter, we consider the population learning algorithm (PLA2), earlier designed by the authors, and study how the interconnection topology and heterogeneity of the constituent modules influence its efficiency. PLA2 is a population-based approach which takes advantage of the features common to the social education system rather than to the evolutionary processes. The problem of scheduling nonpreemtable tasks on parallel identical machines under constraint on discrete resource and requiring, additionally, renewable continuous resource to minimize the schedule length is chosen as the problem to cope with. A continuous resource is divisible continuously and is allocated to tasks from given intervals in amounts unknown in advance. Task processing rate depends on the allocated amount of the continuous resource. To eliminate time consuming optimal continuous resource allocation, an NP-hard problem Θ_Z with continuous resource discretisation is introduced and sub-optimally solved by PLA2. The PLA2's island design can be easily transferred to an agent system with cooperating agents.

1 Introduction

A problem of scheduling jobs on multiple machines under constraint on discrete resource and requiring, additionally, renewable continuous resource to minimize the schedule length is considered in the chapter. In the problem two types of resources are considered: discrete and continuous. A discrete resource is divisible

Piotr Jędrzejowicz · Aleksander Skakovski
Department of Information Systems, Gdynia Maritime University,
Morska 83, 81-225 Gdynia, Poland
e-mail: {pj,askakow}@am.gdynia.pl

* Corresponding author.

I. Czarnowski et al. (Eds.): Agent-Based Optimization, SCI 456, pp. 77–102.
DOI: 10.1007/978-3-642-34097-0_4

discretely, for example a set of machines or a set of mechanical or pumping machines. A continuous resource is divisible continuously and is allocated to the jobs from given intervals in amounts unknown in advance. In practice a continuous resource may be limited in amount - for example power (electric, pneumatic, hydraulic) supplying a set of machines, limited gas flow intensity supplying forge furnaces in a steel plant, or limited fuel flow intensity in refueling terminals.

The problem of scheduling jobs on multiple machines under constraint on discrete resource and requiring, additionally, renewable continuous resource was intensively explored in [9], [10], [11], [12], and we define the problem in the same way. Namely, we consider n independent, nonpreemptable jobs, each of them simultaneously requiring for its processing at time t a machine from a set of m parallel, identical machines (the discrete resource) and an amount (unknown in advance) $u_i(t) \in [0, 1]$, $i = 1, 2, \ldots, n$, of a continuous renewable resource. The job model is given in the form:

$$\dot{x}_i(t) = \frac{dx_i(t)}{d(t)} = f_i[u_i(t)],\ x_i(0) = 0,\ x_i(C_i) = \tilde{x}_i\ , \tag{1}$$

where $x_i(t)$ is the state of job i at time t, f_i is an increasing continuous function, $f_i(0) = 0$, C_i is (unknown in advance) completion time of job i, and $\tilde{x}_i$ is its processing demand (final state). We assume, without loss of generality, that $\sum_{i=1}^{n} u_i(t) = 1$ for every t. The problem is to find a sequence of jobs on machines and, simultaneously, a continuous resource allocation that minimizes the given scheduling criterion. The problem is computationally complex and is at least as hard as the classical RCPSP (Resource Constrained Project Scheduling Problem), since the existence of an additional continuous resource cannot make the problem any simpler [11], [12]. The defined problem can be decomposed into two interrelated sub problems: (i) to find a feasible sequence of jobs on machines, and (ii) to allocate the continuous resource among jobs already sequenced. The notion of a feasible sequence is of crucial importance. According to [10] a feasible schedule can be divided into $p \le n$ intervals defined by completion times of consecutive jobs. Let Z_k denote the combination of jobs processed in parallel in the k-th interval. Thus, in general, a feasible sequence FS of combinations Z_k, $k = 1, 2,..., p$, can be associated with each feasible schedule. Feasibility of such a sequence requires that the number of elements in each combination does not exceed m and that each job appears exactly in one or in consecutive combinations in FS (nonpreemptability). It has been shown in [9] that for concave job models and the schedule length minimization problem, it is sufficient to consider feasible sequences of combinations Z_k, $k = 1, 2,..., n - m + 1$, composed of exactly m jobs each. For a given feasible sequence FS of jobs on machines, we can find an optimal continuous resource allocation, i.e. an allocation that leads to a schedule minimizing the given criterion from among all feasible schedules generated by FS. At this point, a convex mathematical programming problem has to be solved, in the general case (see [9]). An optimal schedule for a given feasible sequence (i.e. a schedule resulting

from an optimal continuous resource allocation for this sequence) is called a semi-optimal schedule. In consequence, a globally optimal schedule can be found by solving the continuous resource allocation problem optimally for all feasible sequences. Unfortunately, in general, the number of feasible sequences grows exponentially with the number of jobs. Therefore it is justified to apply some approximation algorithm or metaheuristic.

Because finding an optimal allocation of a continuous resource to a feasible schedule requires using specialized and time-consuming solver, an idea of continuous resource discretisation was proposed in [12]. We use the same approach in the chapter. Namely, we assume that the number of possible continuous resource allocations to a task J_i is D_i, i.e. is fixed, and the amount of the continuous resource for each $l_i = 1, 2, \ldots, D_i$ is known in advance (in the original problem there was infinite number of the continuous resource allocations to a task and the amount of the continuous resource to be allocated was not known in advance). Because a different amount of the continuous resource is allocated to task J_i for each l_i, l_i is called a processing mode of task J_i. Such discretisation of the continuous resource allows treating it as a discrete resource.

The problem of scheduling jobs on multiple machines under additional continuous resource with continuous resource discretisation is NP-hard [12]. A population-learning algorithm (SLA) first proposed in [6] was used to tackle the problem, since it was effective in solving other scheduling problems considered in [5], [3], [4]. Promising results obtained by the proposed in [8] version of PLA - PLA1 proved the approach for solving Θ_Z to be effective and caused the design of PLA2 proposed in [8]. PLA2 uses four main procedures: a cross-entropy (CE), a Tabu Search (TS) procedure, an island-based evolutionary algorithm (IBEA), and a population-based evolutionary algorithm (PBEA). All mentioned procedures could be viewed as independent and cooperating agents and used to design an agent system. Because all the procedures used in PLA2 were thoroughly described in [7] and [8] we only briefly remind the procedures in this work in Sections 3.1-3.4 respectively.

The main goal of our research was to find out whether the interconnection topology of a learning stages (or islands), might have some effect on the algorithm's efficiency. For this reason we proposed six versions of PLA2 that differ from each other by their structure and migration scheme. We assume that the efficiency of the algorithm is its ability to yield "good" quality solutions of a problem within a given number of fitness function evaluations. On this basis we have compared all proposed versions of PLA2 making them to carry out the same or approximately the same number of fitness function evaluations. Assuming such approach it is easier to judge on the efficiency of the proposed versions of PLA2 by only comparing the quality of the solutions they yielded. A computational experiment, described in Section 4, was carried out to test the influence of the interconnection topology of the available islands and the possible influence of migration size between CE-island and IBEA-islands on the quality of the PLA2 found solutions.

2 Problem Formulation

We define a problem Θ_Z in the same way as in [12]. Namely, let $\boldsymbol{J} = \{J_1, J_2, \ldots, J_n\}$ be a set of nonpreemtable tasks, with no precedence relations and ready times $r_i = 0$, $i = 1, 2, \ldots, n$, and $\boldsymbol{P} = \{P_1, P_2, \ldots, P_m\}$ be a set of parallel and identical machines, and there is one additional renewable discrete resource in amount $U = 1$ available. A task J_i can be processed in one of the modes $l_i = 1, 2, \ldots, D_i$ (D_i – the number of processing modes of task J_i), for which J_i requires a machine from $\boldsymbol{P}$ and amount of the additional resource known in advance. The processing mode of J_i cannot change during the processing. For each task two vectors are defined: a processing times vector $\tau_i = [\tau_i^1, \tau_i^2, ..., \tau_i^{D_i}]$, where $\tau_i^{l_i}$ is the processing time of task J_i in mode $l_i = 1, 2, \ldots, D_i$ and a vector of additional resource quantities allocated in each processing mode $u_i = [u_i^1, u_i^2, ..., u_i^{D_i}]$. The problem is to find processing modes for tasks from $\boldsymbol{J}$ and their sequence on machines from $\boldsymbol{P}$ such that schedule length $Q = \max\{C_i\}$, $i = 1, ..., n$ is minimized.

3 Population Learning Algorithm

Population learning algorithm proposed in [6] has been inspired by analogies to a social phenomenon rather than to evolutionary processes. The population learning algorithm takes advantage of features that are common to social education systems:

- A generation of individuals enters the system.
- Individuals learn through organized tuition, interaction, self-study and self-improvement.
- Learning process is inherently parallel with different schools, curricula, teachers, etc.
- Learning process is divided into stages.
- More advanced and more demanding stages are entered by a diminishing number of individuals from the initial population (generation).
- At higher stages more advanced education techniques are used.
- The final stage can be reached by only a fraction of the initial population.

All individuals (solutions) used in the PLA2 procedure can be characterized in the following manner:

- an individual (a solution) is represented by an n-element vector $S = [c_i |\ 1 \le i \le n]$,
- all processing modes of all tasks are numbered consecutively. Thus processing mode l_b of task J_b has the number $c_b = \sum_{i=1}^{b-1} D_i + l_b$,
- all S representing feasible solutions are potential individuals,

– each individual can be transformed into a schedule by applying LSG, which is a specially designed list-scheduling algorithm for discrete-continuous scheduling,
– each schedule produced by the LSG can be directly evaluated in terms of its fitness.

The PLA2 model can be also viewed as an island model, were islands are connected to each other according to some topology and exchange individuals in order to collectively find best possible solution to the problem. Such island-based design can be easily transferred to an agent system with cooperating agents. We used three kinds of learning procedures to design PLA2: cross-entropy (CE), Tabu search (TS), and an island-based evolutionary algorithm (IBEA) all combined into some structures. As a learning procedure IBEA uses population-based evolutionary algorithm (PBEA) to evolve a population on an island, which is described in Section 3.3. PBEA is also used to evolve solutions on an island independently on IBEA, for example in case with random solution migration among islands. We distinguish two categories of island groups – heterogeneous and homogeneous, dependently on the type of the learning procedures carried out on the islands. We refer to the group of islands as heterogeneous, if the learning procedures carried out on at least one island is different from the learning procedures carried out on the rest of the islands in the group. We refer to the group of islands as homogeneous, if the same learning procedure is carried out on each island in the group. In our work, we will refer to a particular island as heterogeneous (Ht), if CE or TS procedure is carried out on it, and homogeneous (Hm), if PBEA is carried out on it. We will use the terms a learning procedure and an island interchangeably. The main goal of our research was to find out whether a topology of a learning stages (or islands), might have some effect on the algorithm's efficiency. For this reason we proposed several versions of PLA2 that differ from each other by their structure and migration scheme. We will refer to these versions of PLA2 as algorithms, and some letter code will be assigned to each of them. In order to distinguish the algorithms, we considered two their basic types, each of them having two topology schemes. In the first basic type, all islands participate in the solution evolution and migration at least once, but only *selected* islands take part in the cyclic solution migration among islands (the letter code for this version will contain letter "S"). In the second basic type – all islands take part in the cyclic solution migration among islands (the letter code for this version will contain letter "A"). As it was mentioned above, each basic type appears in two topology schemes. In the first topology scheme, islands are located on a directed ring and the individuals migrate among the islands along the ring (the letter code for this scheme will contain letter "O" and we will refer in the following text to this topology as a *ring topology*). In the second topology scheme – individuals migrate between randomly chosen pairs of the islands (the letter code for this scheme will contain letter "X" and we will refer in the following text to this topology as a *random topology*). Moreover, the letter code for the algorithms in which CE procedure sends multiple solutions to the island-in-pair during the migration phase will contain letter "m". For the version where CE procedure sends a single solution to the island-in-pair

during the migration phase the letter code will contain letter "s". Therefore, the letter code "AO-m" stands for the algorithm in which all islands comprise a directed ring of heterogeneous islands and procedure CE sends multiple solutions to the island-in-pair during the migration phase. In our present research, we consider six versions of PLA2, namely: SO, SX, AO-m, AO-s, AX-m, and AX-s. Because all proposed algorithms are versions of PLA2, they have common phases, which are shown in a generalized versions as S- or A-algorithms. The pseudo codes, as well as figures illustrating all the proposed algorithms are given below. In a simplified graphic illustration of the algorithms in Figures 1 - 4, solid lines show islands participating in the cyclic solution migration and dash-dot lines show islands where learning procedures are carried out only once.

```
S-algorithm
Begin
  Create an initial population P0 of the size x0 - 1
  using procedure cross-entropy (CE).
  Create an individual TSI in which all tasks Ji are to
  be executed in mode li = 1 (a mode characterized by
  minimal quantity of additional resource u_i^1 and max-
  imal task processing time τ_i^1, 1 ≤ i ≤ n).
  Improve the individual TSI with the tabu search (TS)
  procedure.
  Create population P1 = P0 + TSI.
  Distribute equally individuals from P1 among all Hm-
  islands.
  Carry out the appropriate Learning stage SO or SX
  designed for SO and SX algorithms respectively.
  Output the best solution to the problem.
End.

Learning stage SO
Begin
  Improve individuals on Hm-islands with procedure
  IBEA.
End.

Learning stage SX
Begin
  Improve individuals on each Hm-island with procedure
  PBEA, cyclically exchanging best solutions between
  randomly chosen pairs of Hm-islands.
End.
```

In all proposed algorithms, $x_0 = K \cdot PS$, where K – the number of homogeneous islands and PS – the population size on an island defined in procedure IBEA.

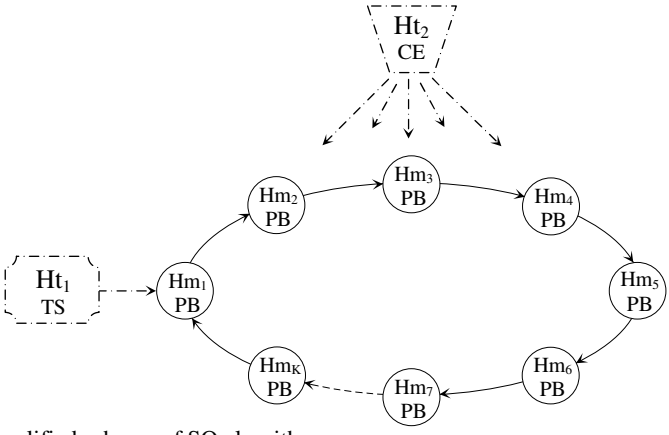

Fig. 1 A simplified scheme of SO algorithm

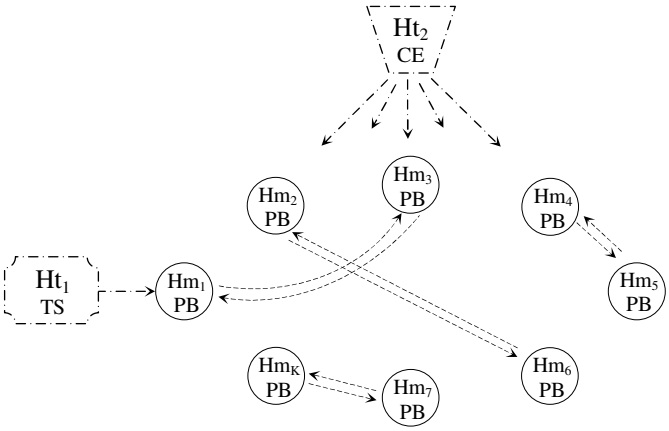

Fig. 2 A simplified scheme of SX algorithm

```
A-algorithm
Begin
  Create an initial population P0 of the size x0 using
  cross-entropy procedure (CE).
  Distribute equally individuals from P0 among all Hm-
  islands.
  Create an individual TSI in which all tasks Ji are to
  be executed in mode li = 1 (a mode characterized by
  minimal quantity of additional resource u_i^1 and max-
  imal task processing time τ_i^1, 1 ≤ i ≤ n).
  Send TSI to the Tabu Search (TS) procedure.
```

```
  Carry out the appropriate Learning stage AO or AX
  designed for AO and AX algorithms respectively.
  Output the best solution to the problem.
End.
```

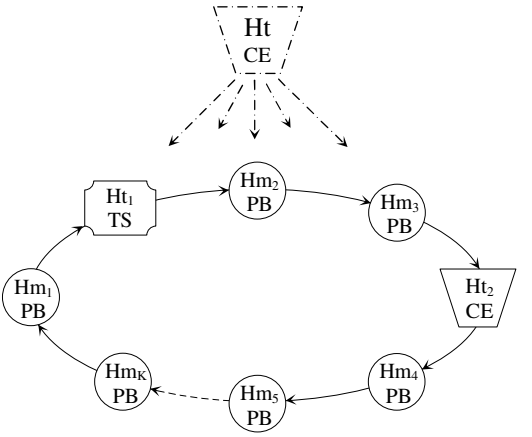

Fig. 3 A simplified scheme of AO algorithm

```
Learning stage AO
Begin
  Create a directed ring of all available islands as
  follows:
  Hm1, Ht1(TS), Hm2, Hm3, Ht2(CE), … , HmK, where K • 3.
  Improve individuals on the islands with the assigned
  to the islands procedures cyclically sending best
  solution from each island along the ring.
End.

Learning stage AX
Begin
  Improve individuals on all available islands with
  the assigned to the islands procedures cyclically
  exchanging best solution between randomly chosen
  pairs of islands.
End.
```

In the AO algorithm, CE procedure receives multiple solutions from the homogeneous islands Hm_1, Hm_2 and Hm_3, and sends a single solution to Hm_4 (or Hm_1, when $K = 3$) in AO-s algorithm, or multiple solutions in AO-m algorithm. In AX algorithm CE procedure also receives multiple solutions from Hm_1, Hm_2 and Hm_3, and sends to the randomly chosen island a single solution in AX-s algorithm, or multiple solutions in AX-m algorithm. On all homogeneous islands Hm_i, $i = 1, 2, \ldots, K$ we used PBEA procedure.

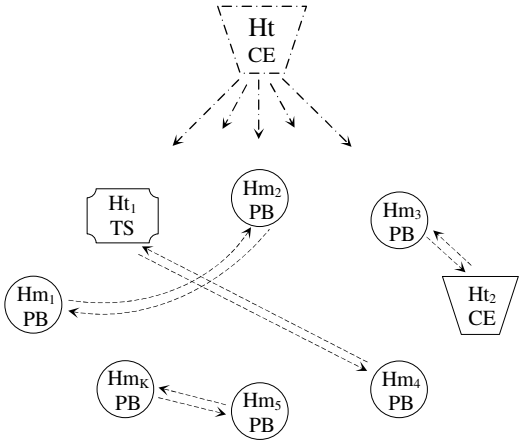

Fig. 4 A simplified scheme of AX algorithm

3.1 A Cross-Entropy Algorithm

In PLA2 the proposed CE procedure is perceived as the procedure preparing some solution basis for further improvement by procedure IBEA. In CE procedure a cross-entropy (CE) method first proposed in [13] is used since it was effective in solving various difficult combinatorial optimization problems [1]. It follows from the definition of the solution vector S that a number c_i in S unequivocally identifies a task and the task processing mode. In order to use CE method, we would like to know the probability of locating a task J_i on a particular place j in the vector. For this reason we introduce two success probability vectors $\hat{p}_j$ and $\hat{p}'_{ji}$ related to each task J_i and its place j in solution S. Vector $\hat{p}_j = [p_{ji} \mid 1 \le i \le n]$, $1 \le j \le n$ contains p_{ji} values, which is the probability that on the place j there will be located a task i. Vector $\hat{p}'_{ji} = [p_{jil} \mid 1 \le l \le D_i]$, $1 \le j \le n$, $1 \le i \le n$ contains p_{jil} values, which is the probability that on place j task i will be executed in mode l. A procedure CE using cross-entropy method for combinatorial optimization described in [1] and modified for solving Θ_Z problem is shown in the following pseudo code:

```
Procedure CE
Begin
 Set ic = 1 (ic - iteration counter), ic^stop - maximal
 number of iterations, a:= 1.
 Set p̂_j = [p_ji = 1/n | 1 ≤ i ≤ n], 1 ≤ j ≤ n.
 Set p̂'_ji = [p_jil = 1/D_i | 1 ≤ l ≤ D_i], 1 ≤ j ≤ n, 1 ≤ i ≤ n.
```

While $ic \le ic^{stop}$ do

Generate a sample S_1, S_2, … , S_s, … , S_N of solutions with success probability vectors $\hat{p}_j$ and $\hat{p}'_{ji}$.

Order S_1, S_2, … , S_s, … , S_N by values of their fitness function non-decreasingly.

Set $\gamma = \lceil \rho \cdot N \rceil, \rho \in (0,1)$.

Set

$$\hat{p}_j = \left[p_{ji} = \frac{\sum_{s=1}^{\gamma} I(S_s(j) = i)}{\gamma} \middle| 1 \le i \le n \right], \tag{2}$$

$1 \le j \le n$, $I(S_s(j) = i) = 1$, $I(S_s(j) \bullet i) = 0$, where $S_s(j)$ – number of the task located on j-th place in s-th solution S.

Set

$$\hat{p}'_{ji} = \left[p_{jil} = \frac{\sum_{s=1}^{\gamma} I(S_s(ji) = l)}{\gamma} \middle| 1 \le l \le D_i \right], \tag{3}$$

$1 \le j \le n$, $1 \le i \le n$, $I(S_s(ji) = l) = 1$, $I(S_s(ji) \ne l) = 0$, where $S_s(ji)$ – an execution mode of task i located on j-th place in s-th solution S.

Save the first $h = \lceil K \bullet PS \,/\, ic^{stop} \rceil$ best solutions from the ordered sample into P_0 under address a. Set $a := a + h$.

Set $ic := ic + 1$.

EndWhile.

EndProcedure.

In the presented pseudo code, a parameter N is the number of solutions in a sample generated in each iteration. A parameter ρ determines the percentage of the best solutions in the current sample that are used to calculate new values for the vectors $\hat{p}_j$ and $\hat{p}'_{ji}$. The both parameters were determined empirically and set $N = 1000$ and $\rho = 0{,}2$. Parameters K – the number of islands and PS – the population size are defined in procedure IBEA and PBEA respectively.

3.2 Tabu Search

Tabu search is another metaheuristic used in the considered versions of PLA (see [4]). To present general idea of the present implementation of the tabu search procedure we introduce the neighborhoods N_t and N_{md} of a solution S. N_t is a set of

solutions generated from S by moving a task $J_i \in S$ from place i to the rest $n - 1$ places. Thus we yield $|N_t| = n \cdot (n - 1)$ neighbors. N_{md} is a set of solutions generated from S by assigning to task $J_i \in S$ one by one in a row all of its D modes, assuming that all tasks can be executed in D modes. Thus we yield another $|N_{md}| = n \cdot (D - 1)$ neighbors. The considered tabu search procedure is shown in the following pseudo code:

```
Procedure TS
Begin
 Set S0 = initial solution TSI (li = 1, 1 • i • n).
 Set the best solution Sbest = S0.
 Set Tabu List TL = ∅.
 Set Nt = {S0} and Nmd = ∅.
 Set nit = 7 (determined empirically).
 Repeat the following max_number_of_iterations times:
   Find the best legal neighbour Sbln of S0, i.e. the
   best across Nt and Nmd neighbour which is not on TL.
   Set S0 = Sbln.
   If Sbln is more fit than Sbest then Sbest = Sbln.
   Put Sbln on the Tabu list.
   If the fitness of S0 has not improved after nit num-
   ber of iterations construct a new solution by mov-
   ing a task Ji in S0 to one of the chosen randomly
   less frequently visited places on the task list and
   assigning to it one of the chosen randomly less
   frequently assigned execution modes.
 EndRepeat.
End.
```

The size of the Tabu List (TL) was determined empirically and set to 500 solutions.

3.3 An Island-Based Evolutionary Algorithm

The following pseudo-code shows main stages of the IBEA algorithm:

```
Procedure IBEA
Begin
 Set the number of islands K, the number of popula-
 tions PN to be evolved on each island.
 While no stopping criteria is met do
   For each island Ik do
      Evolve PN generations using procedure PBEA.
```

```
      Send the best solution to I_(k mod K) + 1.
      Incorporate the best solution from
      I_((K+k -2) mod K) + 1 instead of the best one.
    EndFor
  EndWhile
  Find the best solution across all islands and save it
  as the final one.
End.
```

3.4 A Population-Based Evolutionary Algorithm

Population-based evolutionary algorithm (PBEA) proposed in [7] as a part of IBEA for solving discrete-continuous scheduling problem is used as a learning procedure to evolve solutions on homogeneous islands in all considered versions of PLA2. PBEA algorithm is shown in the following pseudo-code:

```
Procedure PBEA
Begin
 Set population size PS.
 Set ic:= 0; (ic - iteration counter).
 While no stopping criteria is met do
   Set ic:= ic + 1,
   Calculate fitness factor for each individual in
   PP_ic-1 using LSG,
   Form new population PP_ic:
     Select randomly a quarter of PS of individuals
     from PP_ic-1 (probability of selection depends on
     fitness of an individual).
     Produce a quarter of PS of individuals by apply-
     ing crossover operator to previously selected in-
     dividuals from PP_ic-1.
     Produce a quarter of PS of individuals by apply-
     ing mutation operators to previously selected in-
     dividuals from PP_ic-1.
     Generate half of a quarter of PS of individuals
     from set of potential individuals (random task
     processing mode and task order).
     Generate half of a quarter of PS of individuals
     from set of potential individuals (random task
     processing mode and ascending order of the task
     numbers).
   EndWhile
End.
```

LSG algorithm used within PBEA is carried out in three steps as follows:

```
Procedure LSG
Begin
 Construct a list of tasks from the code representing
 individuals. Set loop over tasks on the list.
 Within the loop, allocate current task to a machine
 considering the amount of a continuous resource al-
 lotted to the task, and minimizing the beginning time
 of its processing. Continue with tasks until all have
 been allocated.
 Calculate the fitness of the individual S as
 Q_u = max{C_i}, i = 1, ... , n.
End.
```

4 Computational Experiments

The proposed six versions of the cross-entropy based population learning algorithm for solving discrete-continuous scheduling problems with continuous resource discretisation were implemented and tested. The efficiencies of all six algorithms were compared to each other, as well as to the tabu search (TS) procedure, used within each of the algorithms which was run in addition as an independent algorithm. In the procedure CE, as it was mentioned earlier, parameters ρ and N were determined empirically and set $N = 1000$ and $\rho = 0{,}2$. The size of the Tabu List (TL) was determined empirically as well, and set to 500 solutions. For testing purposes three combinations of n x m were considered (n – the number of tasks and m – the number of machines): 10x2, 10x3, and 20x2. For each combination n x m 100 instances of a problem Θ_Z were generated and three discretisation levels D were considered: 10, 20, and 50. This way we considered nine sizes of the problem: 10x2x10, 10x2x20, 10x2x50, 10x3x10, … , 20x2x50, which makes 900 instances of the problem in total. In all problem instances, we have used the same as in [12] the task processing rate function calculated according to the formula:

$$f_i^{l_i} = u_i^{l_i 1/\alpha_i}, \alpha_i \in \{1, 2\}, \tag{4}$$

where α_i could take the values 1 and 2 with the same probability. The values of the task processing times were calculated according to the formula:

$$\tau_i^{l_i} = \frac{\tilde{x}_i}{f_i^{l_i}},\ i = 1, 2, \ldots, n. \tag{5}$$

The continuous resource was discretised uniformly according to the formula:

$$u_i^{l_i} = \frac{l_i}{D_i},\ D_i = D,\ D \in \{10, 20, 50\},\ i = 1, 2, \ldots, n. \tag{6}$$

Each of the considered algorithms carried out about 720000 fitness function evaluations to yield one solution for the instance of the problem. Each instance was tested 43 times by all the proposed algorithms. Mean time required by the considered algorithms to find a solution for the problem sizes 10x2 and 10x3 for all discretisation levels on Pentium (R) 4 CPU 3.00GHz compiled with aid of Borland Delphi Personal v.7.0 was approximately 4 - 7s, and for the problem size 20x2 for all discretisation levels approximately 8 – 13s.

In order to evaluate the efficiency of the proposed algorithms we used such parameters as relative errors (minimum, average, maximum) of the solutions yielded by the algorithms, as well as percentage of the best found solutions of the same quality as the best-known solutions. Relative errors (RE) of the solutions compared to the best-known solutions were calculated according to the formulae $RE = (Q_{PLA2} - Q_{best-known})/Q_{best-known}$, where Q – the quality of a considered solution. The set of the best-known solutions was determined by the authors while using all designed by them procedures and algorithms, namely PBEA, IBEA, TS, PLA1, PLA2, AX-m, AX-s, SX, SO, AO-m, AO-s, for solving problem Θ_Z. We have determined RE_{min} and RE_{max} for every size of the considered problem as a minimum or respectively maximum RE across 4300 REs calculated while solving each of the 100 instances 43 times. We have also determined RE_{avg} as a mean value of 4300 REs obtained within 43 runs of 100 instances of the considered problem. The values of RE_{min}, RE_{avg} and RE_{max} of the solutions found by all proposed algorithms for all problem sizes are presented in Tables 1 - 9. The values of REs in Tables 1 - 9 show how much schedules yielded by the proposed algorithms were longer than the best known schedule for the same case. For example, in Table 1 for the case 10x2x10 for SO algorithm, $RE_{avg} = 3.28\%$ means that the schedule length of all schedules yielded by SO algorithm was on average 3.28% longer than the best-known. For the same case, $RE_{max} = 9.76\%$ means that the longest schedule among all schedules yielded by SO algorithm was 9.76% longer than the best-known. To make it easier to evaluate their efficiency, in Tables 1 - 9 below, we have ordered the algorithms according to their REs non-decreasingly.

Table 1 The algorithms ordered non-decreasingly according to their RE_{min}, RE_{avg} and RE_{max} for the size 10x2x10 of the problem Θ_z

Nr	Algm	RE_{min}	Algm	RE_{avg}	Algm	RE_{max}
1	SX	0,01%	SO	3,28%	SO	9,76%
2	AO-m	0,01%	SX	3,31%	TS	9,91%
3	AO-s	0,01%	TS	3,49%	SX	10,00%
4	AX-m	0,01%	AX-m	3,54%	AX-s	11,39%
5	AX-s	0,01%	AX-s	3,58%	AX-m	12,33%
6	TS	0,01%	AO-m	6,30%	AO-m	14,68%
7	SO	0,19%	AO-s	6,30%	AO-s	16,53%

For the problem size 10x2x10, according to the values of the RE_{min} in Table 1, SO algorithm has the largest RE_{min}, however quite close to the REs of the other algorithms. On the other hand, considering RE_{avg}, SO algorithm has the lowest RE_{avg}, and this way, is the leader in a group of the algorithms: SO, SX, TS, AX-m, AX-s, with similar RE_{avg} values. The algorithms AO-m, AO-s make another group of the same RE_{avg} values, where RE_{avg} is about twice higher than in the first group. Considering RE_{max}, it is also possible to classify the algorithms into two groups: with low RE_{max}: SO, TS, SX, and with high RE_{max}: AX-s, AX-m, AO-m, AO-s. In the first group the algorithms differ from 9,76% to 10,00%, while in the second – from 11,39% to 16,53%. For the problem size 10x2x10, $RE_{min} \in [0{,}01\%, 0{,}19\%]$, $RE_{avg} \in [3{,}28\%, 6{,}30\%]$, $RE_{max} \in [9{,}76\%, 16{,}53\%]$. Generally, according to Table 1, the algorithms exploiting the directed ring migration scheme (the ring topology) or random migration scheme (the random topology), built exclusively on homogeneous islands, as well as scheme built on all islands with the random topology perform better, than the algorithms exploiting the ring topology built on all islands - both homogeneous and heterogeneous. This way, for the size 10x2x10 – SO, SX and AX-s perform better than the rest of the algorithms.

Table 2 The algorithms ordered non-decreasingly according to their RE_{min}, RE_{avg} and RE_{max} for the size 10x3x10 of the problem Θ_z

Nr	Algm	RE_{min}	Algm	RE_{avg}	Algm	RE_{max}
1	AX-s	0,00%	SO	4,67%	SO	15,92%
2	SO	0,00%	AX-m	4,68%	AX-m	16,07%
3	AX-m	0,00%	SX	4,69%	TS	17,72%
4	SX	0,01%	AX-s	4,74%	SX	18,39%
5	TS	0,07%	TS	5,36%	AX-s	19,33%
6	AO-m	0,10%	AO-m	8,66%	AO-s	25,67%
7	AO-s	0,28%	AO-s	8,71%	AO-m	26,06%

For the problem size 10x3x10, according to the values of the RE_{min} in Table 2, algorithms AX-s, SO, AX-m were able to find the best-known solutions, and algorithms: SX, TS, AO-m, AO-s could not. However, RE_{min} values of the latter group do not differ significantly from the best-known solutions, namely, from 0,01% to 0,28%. Considering RE_{avg}, SO algorithm has the lowest RE_{avg}, and this way, is the leader in a group of the algorithms: SO, AX-m, SX, AX-s with RE_{avg} values differing from 4,67% to 4,74%. TS algorithm is in-between the first group and the third, made of AO-m and AO-s algorithms, whose RE_{avg} values are considerably higher than in the first group, i.e. 8,66% and 8,71% respectively. Speaking about RE_{max}, it is also possible to classify the algorithms into two groups of similar values of RE_{max}: SO, AX-m, TS, SX, from 15,92% to 18,39%, and a group of high RE_{max}: AX-s, AO-s, AO-m, from 19,33% to 26,06%. For the problem size 10x3x10, we also give the intervals to which belong the values of the considered parameters, i.e. $RE_{min} \in [0{,}00\%, 0{,}28\%]$, $RE_{avg} \in [4{,}67\%, 8{,}71\%]$, and finally,

$RE_{max} \in [15{,}92\%, 26{,}06\%]$. As it could be seen, the RE_{avg} and RE_{max} of all algorithms have increased while scheduling 10 tasks on 3 machines compared to scheduling 10 tasks on 2 machines. In this case again, homogeneous ring topology, as well as random topology for both homogeneous and heterogeneous structures preformed better than other algorithms. In addition, it can be noticed, that the algorithms where CE procedure sends multiple solutions during the migration phase perform better, than when it sends a single solution. To finalize, for the size 10x3x10 – SO, AX-m and SX perform better, than the other algorithms.

For the problem size 20x2x10, according to the values of the RE_{min} in Table 3, only algorithm AX-m was able to find the best-known solutions, and the rest of the algorithms - could not. According to RE_{min}, the algorithms: AX-s, SX, SO, TS make a middle group with the values from 0,26% to 0,68%. The algorithms AO-m and AO-s make the third group with RE_{min} values from 1,12% to 1,16%, and are nearly twice as high as in the middle group. Considering RE_{avg}, AX-m algorithm has the lowest RE_{avg}, and is the leader in a group of the algorithms: AX-m, AX-s, SO, SX, TS with RE_{avg} values differing from 4,76% to 6,26%. The algorithms AO-s, AO-m make the third group of the similar RE_{avg} values, where RE_{avg} is from 9,19% to 9,31% which are considerably higher than in the middle group. Considering RE_{max}, it is also possible to classify the algorithms into two groups: with low RE_{max}: SO, SX, AX-m, TS, AX-s, with the values from 11,47% to 12,54%, and another group: AO-m, AO-s, with high RE_{max} values from 16,71% to 18,19%. For the problem size 20x2x10, the intervals of the REs are as follow: $RE_{min} \in [0{,}00\%, 1{,}16\%]$, $RE_{avg} \in [4{,}76\%, 9{,}31\%]$, $RE_{max} \in [11{,}47\%, 18{,}19\%]$. For this problem size, our observations on the topology point at the random topology of both heterogeneous and homogeneous structures, as well as homogeneous ring topology as most efficient ones. Here, CE procedure sending multiple solutions during the migration phase, perform better than when it sends only a single solution. For the size 20x2x10, AX-m, SO and AX-s are the most efficient algorithms.

Table 3 The algorithms ordered non-decreasingly according to their RE_{min}, RE_{avg} and RE_{max} the size 20x2x10 of the problem Θ_z

Nr	Algm	RE_{min}	Algm	RE_{avg}	Algm	RE_{max}
1	AX-m	0,00%	AX-m	4,76%	SO	11,47%
2	AX-s	0,26%	AX-s	4,86%	SX	11,74%
3	SX	0,40%	SO	5,46%	AX-m	11,81%
4	SO	0,51%	SX	5,56%	TS	12,09%
5	TS	0,68%	TS	6,26%	AX-s	12,54%
6	AO-m	1,12%	AO-s	9,19%	AO-m	16,71%
7	AO-s	1,16%	AO-m	9,31%	AO-s	18,19%

Table 4 The algorithms ordered non-decreasingly according to their RE_{min}, RE_{avg} and RE_{max} for the size 10x2x20 of the problem Θ_z

Nr	Algm	RE_{min}	Algm	RE_{avg}	Algm	RE_{max}
1	SX	0,00%	SO	2,05%	SO	6,67%
2	AO-m	0,00%	SX	2,07%	SX	7,04%
3	AO-s	0,00%	AX-m	2,27%	TS	7,66%
4	AX-m	0,00%	AX-s	2,33%	AX-m	9,14%
5	AX-s	0,00%	TS	2,36%	AX-s	12,40%
6	TS	0,00%	AO-m	4,99%	AO-m	17,18%
7	SO	0,00%	AO-s	5,01%	AO-s	17,60%

For the problem size 10x2x20, according to the values of the RE_{min} in Table 4, all algorithms were able to find the best-known solutions. Here, $RE_{min} = 0{,}00\%$, $RE_{avg} \in [2{,}05\%, 5{,}01\%]$, $RE_{max} \in [6{,}67\%, 17{,}60\%]$. The algorithms implementing ring or random topologies realized on homogeneous islands, as well as random topology realized on all islands have considerably lower RE_{avg} and RE_{max} in comparison with the algorithms exploiting the ring topology built on all islands. Thus, SO, SX and AX-m algorithms outperform the other algorithms while solving the problem of the size 10x2x20.

Table 5 The algorithms ordered non-decreasingly according to their RE_{min}, RE_{avg} and RE_{max} for the size 10x3x20 of the problem Θ_z

Nr	Algm	RE_{min}	Algm	RE_{avg}	Algm	RE_{max}
1	AX-m	0,00%	AX-m	3,61%	AX-m	14,71%
2	SX	0,00%	AX-s	3,73%	SO	15,18%
3	AX-s	0,00%	SO	3,87%	AX-s	15,89%
4	TS	0,00%	SX	4,06%	SX	16,66%
5	SO	0,00%	TS	4,95%	TS	18,28%
6	AO-m	0,00%	AO-s	8,13%	AO-m	24,24%
7	AO-s	0,08%	AO-m	8,13%	AO-s	26,03%

For the problem size 10x3x20, according to the values of the RE_{min} in Table 5, all algorithms, except for AO-s, were able to find the best-known solutions. Here, $RE_{min} \in [0{,}00\%, 0{,}08\%]$, $RE_{avg} \in [3{,}61\%, 8{,}13\%]$, $RE_{max} \in [14{,}71\%, 26{,}03\%]$. The overall results for the size 10x3x20 are nearly the same as for 10x2x20, i.e. the algorithms implementing random or ring topologies realized on homogeneous islands, as well as random topology realized on all islands have considerably lower RE_{avg} and RE_{max} in comparison with the algorithms exploiting the ring topology built on all islands. However, for this size of the problem AX-m algorithm has lower RE_{avg} and RE_{max} than SO. Thus, AX-m, SO and AX-s algorithms outperform the other algorithms while solving the problem of the size 10x3x20.

Table 6 The algorithms ordered non-decreasingly according to their RE_{min}, RE_{avg} and RE_{max} for the size 20x2x20 of the problem Θ_z

Nr	Algm	RE_{min}	Algm	RE_{avg}	Algm	RE_{max}
1	SX	0,00%	AX-m	2,05%	AX-m	9,08%
2	AX-m	0,00%	SO	2,43%	SO	9,76%
3	AX-s	0,00%	AX-s	2,53%	AX-s	9,90%
4	SO	0,00%	AO-m	2,60%	TS	10,81%
5	AO-m	0,00%	AO-s	2,72%	SX	11,00%
6	AO-s	0,00%	SX	2,84%	AO-m	15,19%
7	TS	0,53%	TS	5,09%	AO-s	15,61%

For the problem size 20x2x20, according to the values of the RE_{min} in Table 6, all island-based algorithms were able to find the best-known solutions. Here, $RE_{min} \in [0{,}00\%, 0{,}53\%]$, $RE_{avg} \in [2{,}05\%, 5{,}09\%]$, $RE_{max} \in [9{,}08\%, 15{,}61\%]$. The overall results for the size 20x2x20 are much alike as for 10x2x20, i.e. the algorithms implementing random or ring topologies realized on homogeneous islands, as well as random topology realized on all islands have considerably lower RE_{avg} and RE_{max} in comparison with the algorithms exploiting the ring topology built on all islands. Again, AX-m algorithm has lower RE_{avg} and RE_{max} than SO for this size of the problem. Thus, AX-m, SO and AX-s algorithms outperform the other algorithms while solving the problem of the size 20x2x20.

Table 7 The algorithms ordered non-decreasingly according to their RE_{min}, RE_{avg} and RE_{max} for the size 10x2x50 of the problem Θ_z

Nr	Algm	RE_{min}	Algm	RE_{avg}	Algm	RE_{max}
1	TS	0,00%	AX-m	2,47%	SO	8,69%
2	SX	0,00%	AX-s	2,53%	TS	8,79%
3	AX-m	0,00%	SO	2,77%	SX	9,23%
4	AX-s	0,00%	SX	2,79%	AX-m	11,02%
5	SO	0,00%	TS	3,09%	AX-s	11,19%
6	AO-m	0,00%	AO-m	5,77%	AO-s	15,41%
7	AO-s	0,03%	AO-s	5,78%	AO-m	16,61%

For the problem size 10x2x50, according to the values of the RE_{min} in Table 7, all island-based algorithms, except for AO-s, were able to find the best-known solutions. Here, $RE_{min} \in [0{,}00\%, 0{,}03\%]$, $RE_{avg} \in [2{,}47\%, 5{,}78\%]$, and finally $RE_{max} \in [8{,}69\%, 16{,}61\%]$. The overall results for the size 10x2x50 show that the algorithms implementing random or ring topologies realized on homogeneous islands, as well as random topology realized on all islands have considerably lower RE_{avg} and RE_{max} in comparison with the algorithms exploiting the ring topology built on all islands. For the size 10x2x50 the result do not allow to unequivocally determine the most efficient algorithm, thus we distinguish AX-m, SO, SX and AX-s algorithms as more efficient than AO-m and AO-s algorithms.

For the problem size 10x3x50, according to the values of the RE_{min} in Table 8, all algorithms were able to find the best-known solutions. Here, $RE_{min} \in [0{,}00\%, 0{,}06\%]$, $RE_{avg} \in [3{,}31\%, 5{,}91\%]$, $RE_{max} \in [14{,}66\%, 36{,}35\%]$. The overall results for the size 10x3x50 show that the algorithms implementing random or ring topologies realized on homogeneous islands, as well as random topology realized on all islands have considerably lower RE_{avg} and RE_{max} in comparison with the algorithms exploiting the ring topology built on all islands. For this size of the problem, AX-m, AX-s and SX algorithms outperform the other algorithms while solving the problem.

Table 8 The algorithms ordered non-decreasingly according to their RE_{min}, RE_{avg} and RE_{max} for the size 10x3x50 of the problem Θ_z

Nr	Algm	RE_{min}	Algm	RE_{avg}	Algm	RE_{max}
1	AX-s	0,00%	AX-m	3,31%	AX-m	14,66%
2	SX	0,00%	AX-s	3,46%	AX-s	16,59%
3	AO-s	0,00%	SO	3,86%	SX	17,34%
4	AX-m	0,00%	SX	4,18%	TS	18,16%
5	SO	0,00%	TS	5,72%	AO-s	25,24%
6	AO-m	0,00%	AO-s	5,86%	AO-m	27,02%
7	TS	0,06%	AO-m	5,91%	SO	36,35%

Table 9 The algorithms ordered non-decreasingly according to their RE_{min}, RE_{avg} and RE_{max} for the size 20x2x50 of the problem Θ_z

Nr	Algm	RE_{min}	Algm	RE_{avg}	Algm	RE_{max}
1	SO	0,00%	AX-m	3,84%	SX	11,31%
2	SX	0,00%	AX-s	3,95%	SO	11,77%
3	AX-m	0,00%	SO	5,18%	AX-m	12,44%
4	AX-s	0,00%	SX	5,33%	AX-s	12,49%
5	AO-m	0,87%	TS	6,19%	TS	12,65%
6	AO-s	0,96%	AO-s	7,73%	AO-s	17,45%
7	TS	1,05%	AO-m	7,80%	AO-m	18,76%

For the problem size 20x2x50, according to the values of the RE_{min} in Table 9, all algorithms, except for AO-m and AO-s, were able to find the best-known solutions. Here, we give the intervals of the REs' values: $RE_{min} \in [0{,}00\%, 0{,}96\%]$, $RE_{avg} \in [3{,}84\%, 7{,}80\%]$, $RE_{max} \in [11{,}31\%, 18{,}76\%]$. The overall results for the size 20x2x50 show that the algorithms implementing random or ring topologies realized on homogeneous islands, as well as random topology realized on all islands have considerably lower RE_{avg} and RE_{max} in comparison with the algorithms exploiting the ring topology built on all islands. For the size 20x2x50 the result do not allow to

unequivocally determine the most efficient algorithm, thus we distinguish AX-m, SO, SX and AX-s algorithms as more efficient than AO-m and AO-s algorithms.

In order to determine the percentages of the best found solutions for a particular problem size that are of the same quality as the best-known solutions, we had determined the best solutions found within 43 runs of the algorithm for each of 100 problem instances. Next, we counted how many solutions out of obtained 100 had the same quality as the best known for the same problem size and gave this number in percents. The percentages of the best solutions found by the proposed algorithms of the same quality as the best-known solutions are given in Tables 10 - 12.The results in the Tables 10 – 12 confirm unequivocally the previous conclusion, that the algorithms implementing random or ring topologies realized on homogeneous islands, as well as random topology realized on all islands are more efficient that the algorithms exploiting the ring topology built on all islands. Similarly as for REs, AX-m, AX-s and SX prevail other algorithms with clear dominance of AX-m and AX-s, i.e. the algorithms that implement the random topology realized on all islands.

Table 10 The percentage of the best solutions (PBFS), ordered non-increasingly, found by the proposed algorithms that have the same quality as the best-known solutions for the discretisation level $D = 10$

10x2x10	PBFS	10x3x10	PBFS	20x2x10	PBFS
AX-m	69%	AX-s	52%	AX-s	47%
SX	68%	AX-m	49%	AX-m	28%
AX-s	66%	SX	46%	SX	11%
SO	50%	SO	33%	SO	10%
TS	44%	TS	22%	AO-s	2%
AO-s	16%	AO-m	8%	TS	2%
AO-m	8%	AO-s	3%	AO-m	0%

Table 11 The percentage of the best solutions (PBFS), ordered non-increasingly, found by the proposed algorithms that have the same quality as the best-known solutions for the discretisation level $D = 20$

10x2x20	PBFS	10x3x20	PBFS	20x2x20	PBFS
AX-m	36%	AX-m	53%	SX	32%
AX-s	36%	AX-s	37%	AX-m	32%
SX	35%	SX	27%	AX-s	25%
SO	31%	SO	17%	SO	12%
TS	16%	AO-m	9%	AO-m	4%
AO-m	10%	AO-s	9%	AO-s	2%
AO-s	8%	TS	6%	TS	0%

Table 12 The percentage of the best solutions (PBFS), ordered non-increasingly, found by the proposed algorithms that have the same quality as the best-known solutions for the discretisation level $D = 50$

10x2x50	PBFS	10x3x50	PBFS	20x2x50	PBFS
AX-m	42%	AX-m	37%	AX-m	47%
AX-s	30%	AX-s	27%	AX-s	40%
SO	18%	SX	17%	SX	9%
SX	18%	AO-s	12%	SO	3%
TS	4%	SO	10%	TS	1%
AO-m	3%	AO-m	6%	AO-m	0%
AO-s	1%	TS	0%	AO-s	0%

Although, it was possible to determine several most efficient algorithms for each conducted test, we still can't distinguish the most efficient one. In order to do so, we need some universal measure, that could be applied for evaluation of the proposed algorithms. For this reason, we need to transform, or more precisely - normalize RE_{min}, RE_{avg}, RE_{max} and PBFS in a such way, that it would be possible to obtain some estimates that could be aggregated into one estimate, this way enabling the choice of the most efficient algorithm. Because RE and PBFS have opposite evaluation meaning, i.e. the lower RE – the better performance of the algorithm, the lower PBFS – the worse performance of the algorithm, we introduce a new parameter NB = 1 – PBFS instead of PBFS. Thus, let

$$ne_p = \frac{x - x_{\min}}{x_{\max}} \tag{7}$$

be the formula which we apply to RE_{min}, RE_{avg}, RE_{max} and NB within a paricular problem size in order to obtain a normalized estimate *ne*. In the Equation (7), p – one of the considered parameters, i.e. RE_{min}, RE_{avg}, RE_{max} or NB, x – the value of the considered parameter of the particular algorithm, x_{min}, x_{max} – the minimum or respectively maximum value of the considered parameter within the same problem size among all considered algorithms. After calculating *ne* values for all parameters of all algorithms for all problem sizes, the values obtained for each algorithm were summed into an aggregated estimate. The values of the aggregated estimates were used to make a ranking of the considered algorithms, which is shown in Table 13. As it could be seen in Table 13, the ranking implies the superiority of the algorithms implementing random topology realized on both heterogeneous and homogeneous islands over the algorithms implementing the ring topology. Thus, according to the ranking AX-m algorithm is the most efficient among all considered algorithms.

Table 13 A ranking of the considered algorithms according to the aggregated estimate values

Alg-m	Aggregated estimate	Ranking
AX-m	0,83	1
AX-s	2,05	2
SX	3,06	3
SO	4,94	4
TS	9,61	5
AO-m	13,68	6
AO-s	16,50	7

Table 14 The ranges and deltas of RE_{avg} and RE_{max} values for the considered problem sizes ordered by ΔRE_{avg} and ΔRE_{max} non-decreasingly

prbl. size	RE_{avg} range	ΔRE_{avg}	prbl. size	RE_{max} range	ΔRE_{max}
20x2x20	2,05% - 5,09%	2,59%	20x2x20	9,08% - 15,61%	6,53%
10x3x50	3,31% - 5,91%	2,60%	20x2x10	11,47% - 18,19%	6,72%
10x2x20	2,05% - 5,01%	2,96%	10x2x10	9,76% - 16,53%	6,77%
10x2x10	3,28% - 6,30%	3,02%	20x2x50	11,31% - 18,76%	7,45%
10x2x50	2,47% - 5,78%	3,31%	10x2x50	8,69% - 16,61%	7,92%
20x2x50	3,84% - 7,80%	3,96%	10x3x10	15,92% - 26,06%	10,14%
10x3x10	4,67% - 8,71%	4,04%	10x2x20	6,67% - 17,60%	10,93%
10x3x20	3,61% - 8,13%	4,52%	10x3x20	14,71% - 26,03%	11,32%
20x2x10	4,76% - 9,31%	4,64%	10x3x50	14,66% - 36,35%	21,69%

Table 15 The range and delta of PBFS values for the considered problem sizes ordered by ΔPBFS non-decreasingly

prbl. size	PBFS range	ΔPBFS
10x2x20	8% - 36%	28%
20x2x20	2% - 32%	30%
10x3x50	6% - 37%	31%
10x2x50	1% - 42%	41%
10x3x20	9% - 53%	44%
20x2x10	0% - 47%	47%
20x2x50	0% - 47%	47%
10x3x10	3% - 52%	49%
10x2x10	8% - 69%	61%

As it could be seen from the experimental results described above, it is possible to reduce the REs of the solutions found by the considered algorithms just by changing the interconnection topology of the constituent islands. The Table 14 shows that by changing the interconnection topology RE_{avg} can be reduced by 2,59% - 4,64% and RE_{max} by 6,53% - 21,69% dependently on the problem size. The RE_{avg} and RE_{max} ranges were taken from the Tables 1 – 9. Similarly, the Table 15 shows that the percentage of the best found solutions that have the same quality as the best-known solutions can be increased by 28% - 61% dependently on the problem size. The PBFS ranges were taken from the Tables 10 – 12.

Finally, in Tables 16 – 17, we observe the influence of the level of the continuous resource discretisation D on the REs of the found solutions. In Table 16, for the problem size 10x2xD, $D \in \{10, 20, 50\}$, for almost all algorithms except for AO-s, both RE_{min} and RE_{avg} have the lowest values when $D = 20$. Thus, the influence of D on the REs for the considered problem size could generalized by the following relations: $RE_{min/avg}(D = 20) < RE_{min/avg}(D = 50) < RE_{min/avg}(D = 10)$. For the RE_{max}, the results are mixed and it's impossible to derive one clear rule for all algorithms. The influence of the discretisation level D on the REs of the solutions found by the considered algorithms for the problem size 10x3xD, $D \in \{10, 20, 50\}$, according to the Table 17 could be described generally by the following relations: $RE_{min}(D = 20) \leq RE_{min}(D = 50) < RE_{min}(D = 10)$, except for AO-s, and $RE_{avg}(D = 50) < RE_{avg}(D = 20) < RE_{avg}(D = 10)$, except for SX and TS. For the RE_{max}, the results are mixed and it's impossible to derive one clear rule for all algorithms. The influence of the discretisation level D on the REs of the solutions found by the considered algorithms for the problem size 20x2xD, $D \in \{10, 20, 50\}$, according to the Table 18 could be described generally by the following relations: $RE_{min}(D = 20) \leq RE_{min}(D = 50) < RE_{min}(D = 10)$ except for TS, and $RE_{avg}(D = 20) < RE_{avg}(D = 50) < RE_{avg}(D = 10)$. For the RE_{max}, the results are mixed and it's impossible to derive one clear rule for all algorithms. Below we tabularise the obtained relations together in Table 19:

Table 16 The influence of the level of the continuous resource discretisation D on the REs of the found solutions for the problem size 10x2xD, $D \in \{10, 20, 50\}$

Algm	RE_{min}			RE_{avg}			RE_{max}		
	10	20	50	10	20	50	10	20	50
AO-m	0,01%	0,00%	0,00%	6,30%	4,99%	5,77%	14,68%	17,18%	16,61%
AO-s	0,01%	0,00%	0,03%	6,30%	5,01%	5,78%	16,53%	17,60%	15,41%
AX-m	0,01%	0,00%	0,00%	3,54%	2,27%	2,47%	12,33%	9,14%	11,02%
AX-s	0,01%	0,00%	0,00%	3,58%	2,33%	2,53%	11,39%	12,40%	11,19%
SO	0,19%	0,00%	0,00%	3,28%	2,05%	2,77%	9,76%	6,67%	8,69%
SX	0,01%	0,00%	0,00%	3,31%	2,07%	2,79%	10,00%	7,04%	9,23%
TS	0,01%	0,00%	0,00%	3,49%	2,36%	3,09%	9,91%	7,66%	8,79%

Table 17 The influence of the level of the continuous resource discretisation D on the REs of the found solutions for the problem size 10x3xD, $D \in \{10, 20, 50\}$

Algm	RE_{min}			RE_{avg}			RE_{max}		
	10	20	50	10	20	50	10	20	50
AO-m	0,10%	0,00%	0,00%	8,66%	8,13%	5,91%	26,06%	24,24%	27,02%
AO-s	0,28%	0,08%	0,00%	8,71%	8,13%	5,86%	25,67%	26,03%	25,24%
AX-m	0,00%	0,00%	0,00%	4,68%	3,61%	3,31%	16,07%	14,71%	14,66%
AX-s	0,00%	0,00%	0,00%	4,74%	3,73%	3,46%	19,33%	15,89%	16,59%
SO	0,00%	0,00%	0,00%	4,67%	3,87%	3,86%	15,92%	15,18%	36,35%
SX	0,01%	0,00%	0,00%	4,69%	4,06%	4,18%	18,39%	16,66%	17,34%
TS	0,07%	0,00%	0,06%	5,36%	4,95%	5,72%	17,72%	18,28%	18,16%

Table 18 The influence of the level of the continuous resource discretisation D on the REs of the found solutions for the problem size 20x2xD, $D \in \{10, 20, 50\}$

Algm	RE_{min}			RE_{avg}				RE_{max}		
	10	20	50	10	20	50		10	20	50
AO-m	1,12%	0,00%	0,87%	9,31%	2,60%	7,80%	215	16,71%	15,19%	18,76%
AO-s	1,16%	0,00%	0,96%	9,19%	2,72%	7,73%	251	18,19%	15,61%	17,45%
AX-m	0,00%	0,00%	0,00%	4,76%	2,05%	3,84%	215	11,81%	9,08%	12,44%
AX-s	0,26%	0,00%	0,00%	4,86%	2,53%	3,95%	251	12,54%	9,90%	12,49%
SO	0,51%	0,00%	0,00%	5,46%	2,43%	5,18%	215	11,47%	9,76%	11,77%
SX	0,40%	0,00%	0,00%	5,56%	2,84%	5,33%	251	11,74%	11,00%	11,31%
TS	0,68%	0,53%	1,05%	6,26%	5,09%	6,19%	215	12,09%	10,81%	12,65%

Table 19 The relations among the REs on different discretisation levels D, $D \in \{10, 20, 50\}$, for the considered problem sizes

Prbl.size	Relations among the REs on different discretisation levels D, $D \in \{10, 20, 50\}$
10x2xD	$RE_{min}(D = 20) < RE_{min}(D = 50) < RE_{min}(D = 10)$
	$RE_{avg}(D = 20) < RE_{avg}(D = 50) < RE_{avg}(D = 10)$
	RE_{max} – mixed
10x3xD	$RE_{min}(D = 20) \bullet RE_{min}(D = 50) < RE_{min}(D = 10)$, except for AO-s
	$RE_{avg}(D = 50) < RE_{avg}(D = 20) < RE_{avg}(D = 10)$, except for SX and TS
	RE_{max} – mixed
20x2xD	$RE_{min}(D = 20) \bullet RE_{min}(D = 50) < RE_{min}(D = 10)$ except for TS
	$RE_{avg}(D = 20) < RE_{avg}(D = 50) < RE_{avg}(D = 10)$
	RE_{max} – mixed

As it could be seen in Table 19, it's impossible to determine unequivocally the discretisation level on which REs of the found solutions are the lowest. However, it could be pointed at the relation $RE(D = 20) < RE(D = 50) < RE(D = 10)$ as the most frequent relation. This might impose the conclusion, that the high discretisation level does not ensure the lowest values of the REs and the additional research is needed to identify the most appropriate discretisation of the continuous resource.

5 Conclusion

In the chapter, we consider the population learning algorithm (PLA2), earlier designed by the authors for solving the problem of scheduling non-preemtable tasks on parallel identical machines under constraint on discrete resource and requiring, additionally, renewable continuous resource to minimize the schedule length. The PLA2 model can be also viewed as an island model, were homogeneous as well as heterogeneous islands are connected to each other according to some topology and exchange individuals in order to collectively find best possible solution to the problem. The PLA2's island-based design can be easily used to construct an agent system with cooperating agents. The main goal of our research was to find out whether a topology of a learning stages (or islands), might have some effect on the algorithm's efficiency. For this reason we proposed six versions of PLA2 that differs from each other by their structure and migration scheme. The most important conclusion that can be drawn from the experimental results is that the interconnection topology of the constituent islands might have a noticeable impact on the quality of the solutions yielded by PLA2. It is possible to reduce the relative errors of the solutions found by PLA2 by order of 2,59% - 4,64% for RE_{avg} and 6,53% - 21,69% for RE_{max} dependently on the problem size. Similarly, the percentage of the best found solutions that have the same quality as the best-known solutions can be increased dependently on the problem size by 28% - 61%. The ranking of the considered algorithms that was designed to reveal the most efficient interconnection topology implies the superiority of the algorithms implementing random topology realized on all available islands, i.e. heterogeneous and homogeneous, or exclusively on homogeneous islands, over the algorithms implementing the ring topology. However, the algorithm implementing the directed ring topology realized exclusively on homogeneous islands for some problem sizes yielded solutions that had the lowest RE_{avg} and RE_{max} values. It should be mentioned here, that all the conclusions are valid for particular implementations of the algorithms used in the experiments. The values of some parameters of the algorithms were determined during their tuning and should be determined on the way of an exhaustive experiment. Our further research should concern other parameters and interconnection topologies that might allow to improve the efficiency of the algorithms that implement the island-based model.

References

1. De Boer, P.-T., Kroese, D.P., Mannor, S., Rubinstein, R.Y.: A Tutorial on the Cross-Entropy Method. Annals of Operations Research 134(1), 19–67 (2005)
2. Czarnowski, I., Gutjahr, W.J., Jędrzejowicz, P., Ratajczak, E., Skakovski, A., Wierzbowska, I.: Scheduling Multiprocessor Tasks in Presence of Correlated Failures. Central European Journal of Operations Research 11(2), 163–182 (2003); Luptaćik, M., Wildburger, U.L. (eds.) Physika-Verlag, A Springer-Verlag Company, Heidelberg
3. Jędrzejowicz, J., Jędrzejowicz, P.: Population–Based Approach to Multiprocessor Task Scheduling in Multistage Hybrid Flowshops. In: Palade, V., Howlett, R.J., Jain, L. (eds.) KES 2003. LNCS (LNAI), vol. 2773, pp. 279–286. Springer, Heidelberg (2003)
4. Jędrzejowicz, J., Jędrzejowicz, P.: PLA–Based Permutation Scheduling. Foundations of Computing and Decision Sciences 28(3), 159–177 (2003)
5. Jędrzejowicz, J., Jędrzejowicz, P.: New Upper Bounds for the Permutation Flowshop Scheduling Problem. In: Ali, M., Esposito, F. (eds.) IEA/AIE 2005. LNCS (LNAI), vol. 3533, pp. 232–235. Springer, Heidelberg (2005)
6. Jędrzejowicz, P.: Social Learning Algorithm as a Tool for Solving Some Difficult Scheduling Problems. Foundation of Computing and Decision Sciences 24, 51–66 (1999)
7. Jędrzejowicz, P., Skakovski, A.: A Population Learning Algorithm for Discrete-Continuous Scheduling with Continuous Resource Discretisation. In: Chen, Y., Abraham, A. (eds.) 6th International Conference on Intelligent Systems Design and Applications, ISDA 2006 Special session: Nature Imitation Methods Theory and Practice (NIM 2006), October 16-18, vol. II, pp. 1153–1158. IEEE Computer Society, Jinan (2006)
8. Jędrzejowicz, P., Skakovski, A.: A Cross-Entropy Based Population Learning Algorithm for Discrete-Continuous Scheduling with Continuous Resource Discretisation. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part I. LNCS (LNAI), vol. 5177, pp. 82–89. Springer, Heidelberg (2008)
9. Józefowska, J., Węglarz, J.: On a methodology for discrete-continuous scheduling. European J. Oper. Res. 107(2), 338–353 (1998)
10. Józefowska, J., Mika, M., Różycki, R., Waligóra, G., Węglarz, J.: Solving discrete-continuous scheduling problems by Tabu Search. In: 4th Metaheuristics International Conference MIC 2001, Porto, Portugal, July 16-20, pp. 667–671 (2001)
11. Józefowska, J., Różycki, R., Waligóra, G., Węglarz, J.: Local search metaheuristics for some discrete-continuous scheduling problems. European J. Oper. Res. 107(2), 354–370 (1998)
12. Różycki, R.: Zastosowanie algorytmu genetycznego do rozwiązywania dyskretno-ciągłych problemów szeregowania. PhD dissertation, Istitute of Computing Science, Poznań University of Technology, Piotrowo 3A, 60-965, Poznań, Poland (2000)
13. Rubinstein, R.Y.: Optimization of computer simulation models with rare events. European Journal of Operations Research 99, 89–112 (1997)