Multi-Objective Optimization Using Bat Algorithm to Solve Multiprocessor Scheduling and Workload Allocation Problem

Pablo Eliseo Reynoso Aguirre, Pedro Flores Pérez and María de Guadalupe Cota Ortiz
Mathematics Department, University of Sonora, Hermosillo 83000, Mexico
Corresponding author: Pablo Eliseo Reynoso Aguirre (pablo.reynoso9@gmail.com)

Abstract: Bat Algorithm is an optimization Meta heuristic, classified as a bio inspired heuristic. In 2010, a Computer Scientist and Researcher Dr. Xin She Yang developed this algorithm. The principal motivation of this methodology is the bat swarm pursuit to their prey, using bat eco localization. Its performance has been scored as a good technique compared with genetic algorithms, particle swarm optimization and simulated annealing for a certain kind of optimization problems. This methodology has been applied to a wide range of optimization problems related with: energetic modeling of gas turbine generators, optimum allocation of capacitors, estimation of dynamic and biologic parameters, data clustering applying k-means methods, and neural network training using feed forward, etc. In this paper applies Bat Algorithm to scheduling and work allocation on a multiprocessor, in which the main objectives of the problem that intends to minimize are makespan and tardiness. In this kind of problems usually use a scheduler, which divides the amount of available time of all the processors between all the processes associated to the problem in order to be executed. Each processor can be able to execute only one task at the time and at the end, the scheduler decides which will be the next process to be performed in every processor. Naturally, the scheduler only selects tasks that can be able to be executed and that has not any dependence restriction incomplete which has to be required for its execution.

Keywords: Bat algorithm, multi-objective optimization, multiprocessor scheduling, makespan, tardiness.

1. Introduction

In recent decades, scheduling has become one of the most studied optimization problems, which is still an important problem in the development at research field. The scheduling appears in many areas of science, mainly in engineering and industry. Recently multiprocessor scheduling has become a topic of particular interest to many researchers in the area of optimization due to the high demand for software applications as well as embedded software development for new electronic devices. This problem is addressed in different ways depending on the constraints and optimization criteria of operational environments. Particularly, for Optimization and Computer Science, Multiprocessor Scheduling represents the distribution of n processes or tasks between m processors in order to optimize specific objectives of the type of system related to the problem to be solved. The multiprocessor scheduling problems are classified as impaired decision making of the NP-hard class. This kind of problems is defined by properties of jobs, type of system, and type of processor [1, 2].

The characteristics associated with the specific problem to be solved in this paper are defined.

1.1 Job Properties

The job properties related to the problem being solved in this work are [2]

- Multiple processes compose jobs, and processes have precedence relations between processes associated
with the same job;
- The processes are executed in complete form only on a single processor;
- The processing time of each job is deterministic, i.e., these times are known from the beginning.

1.2 Processor Properties

The processor properties related to the problem being solved in this work are [2]
- Processors are physically and logically homogeneous; therefore have the same capabilities computationally;
- Processors are organized in grid architecture; therefore the processes exchange information, and then consider the communication times.

1.3 System Properties

The system properties related to the problem being solved in this work are [2, 3]
- The system model is associated with due dates (deadlines), so the goal is to minimize system tardiness, which is the maximum amount of time delay of all jobs;
- The system model is static; therefore, the objective is to minimize system makespan, which is the total execution time of all jobs on the system. This type of system assumes a priori, the number of jobs to be executed;
- The system model considers scheduling architecture without multi processor voltage setting;
- The system model considers the overlap of communication time between processes with execution times.

For organizational purposes, the paper is arranged in six sections as follows: the first two sections, introduction and multiprocessor scheduling problem, described briefly the problematic that has been solved in this paper; the following two sections, the original bat algorithm and proposed bat algorithm discussed the methodology applied in the solution of the problematic; and at the end, the last two sections, experimental results and conclusions, show which are the results of the objectives of the problem by using the methodology selected in this paper, and summarize how good were the results establishing comparisons with other methodologies in order to identify achievements.

2. Multiprocessor Scheduling Problem

Particularly Computing Programs where processes are executed in parallel processors are typically represented by directed a cyclic graphs DAG. In such graphs, the associated arcs have a direction, and there are no cycles in the nodes. Notably in the implementation of the problem, the nodes represent processes and an arc between two nodes represents the data dependency between them, i.e., precedence constraint. The values associated with the arcs of a DAG represent the cost the cost of communication between the respective processes, and the execution time of each process is given in table [4].

In this paper is addressed a non-connected DAG graph, in which each sub graph is connected and represents a job. The following describes theoretically the problem statement: Let be $P$ a set of $m$ homogeneous processors connected in a grid bidirectional configuration [4], the expression of the set of processors is given by

$$P = \{p^{(1)}, p^{(2)}, \ldots, p^{(m)}\}$$

where
- $P$: Set of $m$ homogenous processors.
- $p^{(m)}$: $m$-th processor.

To illustrate the proposed model for the architecture of the processors in a grid, in this paper describes an example of grid model, which is composed by 9 processors (Fig. 1).

This set of processors models a program using a non-connected directed a cyclic graph composed by $n$ independent jobs. The expression of the non-connected DAG is given by

$$DAG = \{f^{(1)}, f^{(2)}, \ldots, f^{(n)}\}$$

where
Multi-Objective Optimization Using Bat Algorithm to Solve Multiprocessor Scheduling and Workload Allocation Problem

Fig. 1  Grid architecture of processors using bidirectional connections.

DAG: Non-connected directed a cyclic graph.

Each job is composed of s number of processes. The expression of the number of processes is given by

\[ s = \text{num}(j^{(i)}) \]

where

\( j^{(i)} \): i-th job.

\( s \): Number of processes in i-th job.

Consequently the total of processes that compose the DAG are q. The following expression that denote the sum of number of processes for every job is given by

\[ q = \sum_{i=1}^{n} \text{num}(j^{(i)}) \]

where

\( n \): Number of jobs associated to DAG.

\( j^{(i)} \): i-th job.

\( s \): Number of processes in i-th job.

The precedence relation of processes \( a, b \in \text{DAG} \), assuming that \( a < b \), denotes that the process \( b \) cannot be executed until the process \( a \) is completed, because process \( a \) is a predecessor of process \( b \) [2]. The weights associated to the arcs represent the cost of communication between processes. As an illustration, this paper implements the solution of scheduling of a non-connected directed a cyclic graph, which comprises 10 processes distributed in three jobs that are performed in two processors [2]. The numerical example problem solved in this work is shown in Fig. 2, Tables 1 and 2.

![Constraint graph associated to the problem in this paper.](image)

Fig. 2

**Table 1  Execution time of the processes.**

<table>
<thead>
<tr>
<th>Process</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (ms)</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>10</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table 2  Limit date time of Jobs to avoid delays.**

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limit time (ms)</td>
<td>22</td>
<td>14</td>
<td>32</td>
</tr>
</tbody>
</table>

2.1 Fitness Functions in Multiprocessor Scheduling Problem

The evaluations of fitness functions that measure efficiency assume by first a simulation scheduling of processes, performing an intelligent processor assignment. Then the evaluation functions for the problem are represented by Eqs. (1)-(3):

\[ M = \max(\text{ft}(j^{(i)})) \mid \forall j^{(i)} \in \text{DAG} \]  

(1)

where

\( M \): Makespan.

DAG: Non-connected directed a cyclic graph.

\( j^{(i)} \): i-th job.

\( \text{ft}(j^{(i)}) \): Finish time of the i-th job.

\( \max \): Maximum value.

\( \forall j^{(i)} \in \text{DAG} \): For all the jobs in the DAG.

\[ T = \max(0, \max(\text{ft}(j^{(i)}) - d_i)) \mid \forall j^{(i)} \in \text{DAG} \]  

(2)

where

\( T \): Tardiness.

DAG: Non-connected directed a cyclic graph.

\( j^{(i)} \): i-th job.

\( \text{ft}(j^{(i)}) \): Finish time of the i-th job.

\( \max \): Maximum value.

\( d_i \): Ith due time associated to the i-th job.
For all the jobs in the DAG.

In addition to determine a value of efficiency for the tardiness related to problems of any size, this condition requires to use the average value of tardiness, which is given by

\[ T = \frac{T}{n} \]  

(3)

where

\[ T: \text{Tardiness.} \]

\[ n: \text{Number of jobs associated to DAG.} \]

3. Original Bat Algorithm

The Bat algorithm is a balanced combination of swarm and path algorithms. The particle swarm optimization algorithm scheme provides algorithms population taking into account their positions and velocities in varying dimensions. The intensive local search algorithm is a path algorithm, which provides the random walk that is a local search around the fittest individuals found looking for the best overall. Bat algorithm also has characteristics of a path algorithm known as simulated annealing, Due to variables as volume and emission rate strongly tied to the temperature and the cooling factor from simulated annealing. Applying this metaheuristic to scientific computing, we can see the numerical problem as bats, pursuit by a computer simulation, looking forward an approximate solution to the global optimum [5] (Fig. 3).

Furthermore, this paper describes the 3 main aspects of idealization from Bat Algorithm are described below [6, 7]:

(1) Bats use echolocation technique to detect prey type or obstacle and distance in which this is.

(2) Bats fly randomly with a velocity \( v_i \) at a position \( x_i \) in certain time \( t = i \), through \( f_{\text{min}} \) fixed frequency, at pulse emission rate \( r \), and wave length \( L \), adjusted at defined volume \( A \) during the bat search. In addition bats can automatically adjust the parameters.

(3) The volume that bats emit, which is an ultrasound, varies the initial volume from a positive constant \( A_0 \) to a minimum value \( A_{\text{min}} \).

3.1 Population Size of Bat Swarm

The initial population, i.e., the number of \( n \) virtual bats in the swarm, is usually selected as a random value, this variable is in the range 10-20 bats, while the number of generations \( N \) is usually chosen between 50 and 100. In this paper \( n \) is selected as 30 and the value of \( N \) is taken as 40. Applying this configuration to find good results required approximately less than or equal to 1,200 different bats [8].

3.2 Movement of Bats

Eqs. (4)-(6) associated respectively with frequency, updating positions \( x_i \) and updating velocities \( v_i \) of bats, are expressed below [5]:

\[ f_i = f_{\text{min}} + (f_{\text{max}} - f_{\text{min}}) \beta \]  

(4)

where

\( f_i: \) Frequency of the \( i \)-th bat.

\( f_{\text{min}}: \) Minimum frequency of the bats.

\( f_{\text{max}}: \) Maximum frequency of the bats.

\( \beta: \) Random value that parameterized frequency, which value should be \( \in [0, 1] \).
\[ v_i^t = v_i^{t-1} + (x_i^{t-1} - x_i^*) f_i \]  
(5)

where
- \( v_i^t \): Actual velocity of the \( i \)-th bat.
- \( v_i^{t-1} \): Previous velocity of the \( i \)-th bat.
- \( x_i^{t-1} \): Previous position of the \( i \)-th bat.
- \( x_i^* \): Best actual position found of the \( i \)-th bat.
- \( f_i \): Frequency of the \( i \)-th bat.

\[ x_i^t = x_i^{t-1} + v_i^t \]  
(6)

where
- \( x_i^t \): Actual position found of the \( i \)-th bat.
- \( x_i^{t-1} \): Previous position of the \( i \)-th bat.
- \( v_i^t \): Actual velocity of the \( i \)-th bat.

### 3.3 Loudness of Bats

The loudness volume \( A_i \), Eq. (9), and the pulse emission rate \( r_i \), Eq. (8), have an important role in the selection of best solutions among the best founded, and contributes to the good explorations on the random walks Eq. (7), the associated equations are shown below [5]:

\[ x_{rw}^t = x_i^t + \varepsilon A_i^t \]  
(7)

where
- \( x_{rw}^t \): Bat selected for the random walk.
- \( x_i^t \): Actual position of the bat selected in random walk.
- \( A_i^t \): Actual volume of the bat selected in random walk.
- \( \varepsilon \): Random value that parameterized volume in random walk, which value should be \( \in [-1, 1] \).

\[ r_i^{t+1} = r_i^0 \left(1 - \varepsilon\right) \]  
(8)

where
- \( r_i^{t+1} \): New value for emission pulse rate of \( i \)-th bat.
- \( r_i^0 \): Initial value for emission pulse rate of \( i \)-th bat.
- \( \gamma \): Random value that adjusts the emission rate pulse of the bat, which value should be \( > 0 \).

\[ A_i^{t+1} = \alpha A_i^t \]  
(9)

where
- \( A_i^{t+1} \): New value for volume of \( i \)-th bat.
- \( A_i^t \): Actual value for volume of \( i \)-th bat.
- \( \alpha \): Random value that adjusts the volume of the bat, which value should be \( 0 < \alpha < 1 \).

In the aftermath it is expected that at the end of the iterations or when the stopping criteria has been reached at Bat Algorithm, after certain time tending to infinite, the conditions of the following values volume and the pulse rate emission of the bats, became their initial values [5], as the expression given below:

\[ t \to \infty : A_i^t \to \infty, r_i^t \to r_i^0 \]

### 4. Proposed Bat Algorithm

This section addresses the version proposed of the Bat Algorithm implemented to solve the multiprocessor scheduling problem, in which describes the initial values used in the variables related to bats parameters. The values related to general features of the bat population are shown below:
- Population size: 30;
- Number of generations: 40;
- Dimension: 10;
- Minimum frequency: 0.0;
- Maximum frequency: 1.0.

The initial values related to the single features of the bat are shown below:
- Initial frequency: 0.0;
- Initial velocity: A random value, should be \( 0 < \text{value} < 1 \);
- Initial position: A random schedule of the tasks
- Initial wavelength: 0.5;
- Initial volume: Random value, should be \( 0 < \text{value} < 1 \);
- Initial pulse emission rate: 0.5.

Moreover, this paper presents a proposal of Bat Algorithm, which introduce a multi objective optimization approach for the algorithm, in order to minimize the objectives Makespan and Tardiness related to the scheduling problem solved in the work. Multi objective approach is a technique implemented by the usage of the Normalized Weight Additive Utility Function NWAUF Eq. (10). This function supports the idea of working with weights for each one of the objectives of the problem, as a rule the sum
of all the weights associated to the objectives, must be one. This weight represents the relative importance for each one of the objectives of the problem [2]. The NWAUF mathematic expression implemented in Bat Algorithm for a schedule $j$ is given by

$$U_j = \sum_{i=1}^{k} w_i f_{i,j}$$

(10)

where

- $k$: The number of objectives associated with the scheduling problem.
- $U_j$: Utility composed value, considering all the objectives related to the problem in the $j$-th schedule.
- $f_{i,j}$: Fitness value of the $i$-th objective in the $j$-th schedule.
- $w_i$: Weight of the $i$-th objective related to the scheduling problem.

The following conditions of this expression are met:

$$W = \sum_{j=1}^{k} w_j = 1; \ w_j > 0 \ \forall w \in W$$

where

- $k$: The number of objectives associated with the scheduling problem.
- $w_i$: Weight of the $i$-th objective related to the scheduling problem.
- $W$: Set of weights for each one of the objectives associated with the scheduling problem.

Furthermore, in order to cover the aspect of the problem of addressing multiple objectives, this work presents some modifications to the original Bat Algorithm for this specific implementation, also the contribution of new methods associated with the resolution of the scheduling problem treated.

The following modifications to the original algorithm are described below:

- The use of a random integer speed in range of 1-10 instead of a speed vector which its dimension was according the number of processes, thinking of an uniformly move in searching of the scheduling processes permutations;
- The usage of a constant lower value in random walks validation than the bat initial emission rate. This leads to improved few individuals at first, making few random walks as he goes looking for a better overall bat, giving rise to random walks with the passage of time and at the end being always perform the random walk, ensuring a better and more complete exploration even when have a very rough global optimum, thus attempting to make fully searchable in the solution space.

In addition the proposed contributions in the implementation of Bat Algorithm to solve the Scheduling problem are described below:

- The programming of a library (.h) in language C/C++ that provides many functions, structures and lists for the implementation of the scheduling problem representing the nonconnected directed acyclic graph and allowing efficient search on the lists;
- The development of an intelligent assignment process, that distributes processors tasks and workloads in such a way that minimizes the objectives of the problem makespan and tardiness;
- The establishment of a procedure to limit the solution space with boundaries, this function limits internal processes to processes of the numerical problem, processes (positions $x$ of the bat) that have been moved;
- The development of a method that fixes constraint precedencies. After being limited solution space boundaries, this method changes precedence indexes of the processes associated to precedence constraints of the non-connected DAG respect to other processes, and the result leaves a partial order in the processes scheduling, leaving one new valid scheduling permutation, this functionality helps greatly to explore more and better solutions.

5. Experimental Results

The following experimental results of makespan and tardiness objectives related to the numerical problem composed by processes, processors, and jobs are denoted respectively by $n = 10$, $m = 2$, $j = 3$. These results are presented bellow in Tables 3 and 4.
Table 3 Comparison of results between algorithms: SPT, EDD, CR, LPT, GA, BIH, BA minimizing makespan objective.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Makespan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest Processing Time</td>
<td>38</td>
</tr>
<tr>
<td>Earliest Due Date</td>
<td>34</td>
</tr>
<tr>
<td>Critical Ratio</td>
<td>34</td>
</tr>
<tr>
<td>Longest Processing Time</td>
<td>34</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td>29.47</td>
</tr>
<tr>
<td>Bat Intelligent Hunting</td>
<td>28.01</td>
</tr>
<tr>
<td>Bat Algorithm</td>
<td>28</td>
</tr>
</tbody>
</table>

Table 4 Comparison of results between algorithms: SPT, EDD, CR, LPT, GA, BIH, BA minimizing tardiness objective.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Makespan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest Processing Time</td>
<td>5.33</td>
</tr>
<tr>
<td>Earliest Due Date</td>
<td>1</td>
</tr>
<tr>
<td>Critical Ratio</td>
<td>4.33</td>
</tr>
<tr>
<td>Longest Processing Time</td>
<td>4.33</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td>1.12</td>
</tr>
<tr>
<td>Bat Intelligent Hunting</td>
<td>1.07</td>
</tr>
<tr>
<td>Bat Algorithm</td>
<td>0.33</td>
</tr>
</tbody>
</table>

6. Simulations and Testing

The following simulations and testing described the distribution and processing of the best values obtained for the objectives makespan and tardiness respectively, related to the numerical problem discussed in this work. The notation of the table variables is as follows:

- \( P \) —Available processors to execute,
- \( T \) —Remaining tasks to be executed,
- \( T_s \) —Selected task in current processing,
- \( P_s \) —Selected processor to execute the selected task,
- \( t_i \) —Entry time,
- \( t_f \) —Exit time. These simulations are described below. First of all, before describing the simulation of distribution and processing, we assume the following ideas and actions:

- The system generates initial population of solutions (bats), covering all precedence constraints, applying and intelligent allocation of tasks in processors, restricting bounds, repairing mistaken precedencies;
- Objectives weights are considered at the time of searching for approximate solutions to global optima during bat searching.

6.1 Makespan Best Value Simulation

During the performing of the bat algorithm for this simulation, were consider the following weights \( w_1 = 0.35 \) and \( w_2 = 0.65 \) used for makespan and tardiness objectives respectively. In this instance at runtime the following result shown above is found, the goal of producing value of 28 for the first objective has reached to find the lowest value found in this scheduling system, while the goal of the second objective produced the value of 2 (Table 5).

- Initially the two processors are unemployed, and the heads of job tasks from the numerical problem are the tasks 1t, 5t and 7t. Randomly one of three tasks available, which will be chosen for execution, is selected. This time was obtained randomly the task 1t, which is assigned the 1p processor as the first available processor found in all processors. The 1t task enters the unit time 0 and ends at the time unit 2 as its execution time takes only two units measured in milliseconds.

- Subsequently, since the precedence of 1t task is completed, the tasks 2t and 3t are added to the list of available tasks to be executed. Is selected randomly among 2t, 3t, 5t, and 7t tasks, task 5t, which is executed in the processor 2p processor being available at that time is obtained. This task enters the time unit 0 and ends in time unit 10.

- Then, after being fulfilled restriction task 5t, task 6t is added as a new process in the list of pending cases. Randomly one of the task available from the list 2p, 3p, 6p and 7p is selected, the task 2t is selected, which based on the contribution of intelligent allocation, the best processor you get to run this task is processor 1p. This task enters the time unit 2 and ends in time unit 8.

- Later in the task list 3t, 6t and 7t, 7t task is obtained randomly, and based on the contribution of
Table 5  Simulation of multiprocessor task scheduling for the best makespan value using Bat Algorithm.

<table>
<thead>
<tr>
<th>P</th>
<th>T</th>
<th>T_i</th>
<th>P_s</th>
<th>t_i</th>
<th>t_f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2</td>
<td>1, 5, 7</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1, 2</td>
<td>2, 3, 5, 7</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1, 2</td>
<td>2, 3, 6, 7</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>1, 2</td>
<td>3, 6, 7</td>
<td>7</td>
<td>1</td>
<td>8</td>
<td>17</td>
</tr>
<tr>
<td>1, 2</td>
<td>3, 6, 8, 9</td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>1, 2</td>
<td>3, 8, 9</td>
<td>3</td>
<td>2</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>1, 2</td>
<td>4, 8, 9</td>
<td>9</td>
<td>1</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>1, 2</td>
<td>4, 8, 9</td>
<td>3</td>
<td>2</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>1, 2</td>
<td>4, 10</td>
<td>4</td>
<td>2</td>
<td>19</td>
<td>28</td>
</tr>
<tr>
<td>1, 2</td>
<td>10</td>
<td>10</td>
<td>1</td>
<td>26</td>
<td>28</td>
</tr>
</tbody>
</table>

intelligent mapping shows that the processor that minimizes the objective is the 1p processor, the task 7t begins at time unit 8 and ends at the time unit 17.

- Next to, once completed the constraint of precedence of 7t, tasks 8t and 9t are added to the list of pending tasks. Consequently, tasks in the task list are 3t, 6t, 8t and 9t, and randomly the task obtained was 6t. With this task, based on the contribution of intelligent allocation, the processor observed that minimizes the objectives is 2p, and the task goes into the time unit 10 and ends at the time unit 15.

- Immediately, the task selected is 3t, with which, based on the contribution of intelligent allocation, the processor is observed that minimizes processor objectives is 2p. This task enters the time unit 15 and ends at the time unit 19.

- At the end of 2t and 3t tasks, the precedence constraint of both tasks is released and the 4t task is added to the list of tasks to run, and randomly selects a task from the list of pending cases 4t, 8t and 9t, 9t task is obtained with which, based on the contribution of intelligent allocation, the processor is observed that minimizes processor objectives is 1p. This task enters the time unit 17 and ends at the time unit 18.

- After randomly selecting a task from the list of pending cases 4t and 8t, and 8t process is obtained, and based on the contribution of intelligent allocation, we observe that the processor that minimizes the objective is the 1p processor. This task enters the time unit 18 and ends at the time unit 26.

- Next to, the precedence constraint of tasks 8t and 9t is released and the task 10t is added to the list of pending cases. Randomly selected from the list of pending cases 4t and 10t and 4t task is selected, which based on the contribution of intelligent allocation, it appears that the processor that minimizes the objective is the 2p processor. This task enters the time unit 19 and ends at the time unit 28.

- Consequently, 10t process is selected, due to be the only one in the list of remaining cases, which based on the contribution of intelligent allocation, we observe that the processor that minimizes the objective is the 1p processor, so that the task 10t enter the time unit 26 and ends at the time unit 28.

6.2 Tardiness Best Value Simulation

During the performing of the bat algorithm for this simulation, were consider the following weights $w_1 = 0.10$ and $w_2 = 0.90$ used for makespan and tardiness objectives respectively. In this instance at runtime the following result shown above is found, the goal of producing value of 32 for the first objective, while the goal of the second objective has reached to find the lowest value found in this scheduling system, which was 0.33333 (Table 6).

- Initially the two processors are unemployed, and the heads of job tasks from the numerical problem are the tasks 1t, 5t and 7t. Randomly one of three tasks available, which will be chosen for execution, is selected. This time was obtained randomly the task 1t, which is assigned the 1p processor as the first available processor found in all processors. The 1t task enters the unit time 0 and ends at the time unit 2 as its execution time takes only two units measured in milliseconds.

- Subsequently, since the precedence of 1t task is completed, the tasks 2t and 3t are added to the list of available tasks to be executed. Is selected randomly among 2t, 3t, 5t, and 7t tasks, task 5t, which is executed
Table 6  Simulation of multiprocessor task scheduling for the best tardiness value using Bat Algorithm.

<table>
<thead>
<tr>
<th>P</th>
<th>T</th>
<th>T_i</th>
<th>P_s</th>
<th>t_i</th>
<th>t_f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2</td>
<td>1, 5, 7</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1, 2</td>
<td>2, 3, 5, 7</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1, 2</td>
<td>2, 3, 6, 7</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>1, 2</td>
<td>3, 6, 7</td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>1, 2</td>
<td>3, 7</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>1, 2</td>
<td>4, 7</td>
<td>4</td>
<td>1</td>
<td>12</td>
<td>21</td>
</tr>
<tr>
<td>1, 2</td>
<td>7</td>
<td>7</td>
<td>2</td>
<td>15</td>
<td>24</td>
</tr>
<tr>
<td>1, 2</td>
<td>8, 9</td>
<td>8</td>
<td>2</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>1, 2</td>
<td>9</td>
<td>9</td>
<td>1</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>1, 2</td>
<td>10</td>
<td>10</td>
<td>1</td>
<td>27</td>
<td>29</td>
</tr>
</tbody>
</table>

in the processor 2p processor being available at that time is obtained. This task enters the unit time 0 and ends at the time unit 10.

- Then, after being fulfilled restriction task 5t, task 6t is added as a new process in the list of pending cases. Randomly one of the task available from the list 2p, 3p, 6p and 7p is selected, the task 2t is selected, which based on the contribution of intelligent allocation, the best processor you get to run this task is processor 1p. This task enters the time unit 2 and ends in time unit 8.

- Later in the task list 3t, 6t and 7t, 6t task is obtained randomly, and based on the contribution of intelligent mapping shows that the processor that minimizes the objective is the 2p processor, since the task that has a precedence constraint that was the 5t task, which was processed on the same processor, this give as a result a communication time of 0. The task 6t begins at time unit 10 and ends at the time unit 15.

- Next to, a process of the tasks list and 7t 3t is again selected and the task obtained was 3t. With this task, based on the contribution of intelligent allocation, the processor observed that minimizes the objectives is 1p, and the task goes into the time unit 8 and ends at the time unit 12.

- Immediately at the end of 2t and 3t tasks, the precedence constraint of both tasks is released and the 4t task is added to the list of tasks to run, randomly selects a process from the list of pending cases 4t and 7t, and 4t task is selected, which, based on the contribution of smart assignment is observed that the processor that minimizes the objective is 1p processor. This task enters the time unit 12 and ends at the time unit 21.

- Immediately, being executed the 4t task, it is observed that have completed the first two jobs of the numerical problem. Then 7t task is selected due to be the only process in the list of pending, and based on the contribution of intelligent allocation, we observe that the processor that minimizes the objective is the 2p processor. This task enters the time unit 15 and ends at the time unit 24.

- Followed by the precedence constraint from 7t is released, and tasks 8t and 9t tasks are added to the list of pending cases. Randomly selected from the list of pending cases, 8t and 9t, task 8t is selected, and based on the contribution of intelligent allocation it appears that the processor that minimizes the objective is the 2p processor. This task enters the time unit 24 and ends at the time unit 32.

- Subsequently 9t task is selected, due to be the only process in the list, and based on the contribution of intelligent allocation, we observe that the processor that minimizes the objective is the 1p processor. This task enters the time unit 26 and ends at the time unit 27.

- Consequently, having been executed the tasks with precedence constraint 8t and 9t, 10t task is added to the list of pending tasks, as is it is the last pending process, it is selected directly using once again the contribution of intelligent allocation, the processor noting that minimizing the objectives is the 1p processor, so that the task enters the 10p time unit 27 and ends at the time unit 29.

7. Conclusions

Making a balance between the changes to the algorithm bats can be concluded that the first modification to the algorithm, where it refers to using
an integer random value of velocity, resulting in a good efficiency when seeking new solutions for algorithm and good performance by making more simple and rapid updating positions of the bat, since a value, instead of a vector of values in the variable velocity.

Moreover the second amendment made, which involves the use of a particular constant value, gave good results in terms of efficiency, when seeking new solutions in certain bats in the random walk procedure, since the objective was to enhance few individuals at first, making a few random walks, and thus give rise to that with the passage of time, always perform the random walk, and improve all bats performing local search around them even when already good local optimum is found that meet the constraints of the problem. Its performance may be slightly lower, because over time as many times as the random walk is performed.

Improvements to the bat algorithm implementation to solve the problem of multiprocessor scheduling presented in this work can conclude the following:

The programming library “scheduling utilities” to translate numerical problem to constraint graphs facilitated processing and searches performed when the simulation of the creation of the initial schedulings, distribution simulation processes, and evaluation of the target maximum delay of work, which was necessary to perform an existing schedulings simulations saved. This achievement streamlines the results and good performance when performing the simulations was obtained.

The development of intelligent allocation, aimed to distribute the processes between corresponding processors in the system, appropriately, intelligently looking which is the best processor to perform a certain operation, so that in terms of efficiency achievement minimizes the two objectives of the problem and parallel performance issues this method achieves significantly minimize the total production time.

Establishing the procedure for limiting the solution space is a method of limiting the processes of the initial problem, the same processes after being found a new permutation scheduling, having been moved positions of the bat through variable speed. This has a great impact on efficiency issues that causes the algorithm to keep track of what the problem is, resolving also does not affect the algorithm performance issues because it is a very simple function, that for practical purposes uses a function that calculates the numerical module of a process.

The development of the method for repairing precedencies, is a method implemented in this work which makes a very important role in finding new schedulings that minimize the goals outlined in this paper. This contribution has good efficiency and achieves that in the search for new solutions precedence constraints of the problem are respected, and also leaves a partial order on scheduling, managing to make more complete searches out of local optima thus achieving a good performance.

The results of the Tardines, which is the maximum delay working, and Makespan, which is the production interval improved all the results of the algorithms mentioned in Ref. [2], so it is concluded that the algorithm bats multi target approach has very good results for this problem and scheduling and work allocation problems of this type.

References


