A modified binary bat algorithm for machine loading in flexible manufacturing systems: a case study

A modified binary bat algorithm for machine loading in flexible manufacturing systems: a case study

Authors: Giorgia CASELLA[a], Teresa MURINO[b], Eleonora BOTTANI[c]

Affiliations: [a] Department of Engineering and Architecture, University of Parma, viale delle Scienze 181/A, 43124 Parma, Italy. Email: giorgia.casella@unipr.it; ORCID: 0000-0003-4137-3084

[b] Department of Chemical, Materials and Industrial Production Engineering, University of Naples "Federico II", Piazzale Tecchio 80, 80125 Napoli, Italy. Email: murino@unina.it; ORCID: 0000-0003-2007-5598

[c] Department of Engineering and Architecture, University of Parma, viale delle Scienze 181/A, 43124 Parma, Italy. Email: eleonora.bottani@unipr.it; ORCID: 0000-0002-6319-8080

Corresponding author: Eleonora Bottani, Full professor of Industrial logistics - Department of Engineering and Architecture, University of Parma, viale G.P.Usberti 181/A, 43124 Parma (Italy). Phone: +39 0521 905872; fax: +39 0521 905705; email: eleonora.bottani@unipr.it

Abstract

The machine loading (ML) problem of flexible manufacturing systems (FMS) has been recognized as one of the most important planning problems in industry. This study aims to minimize the system unbalance by developing and testing a modified binary bat algorithm (MBBA), which satisfies the technological constraints such as the availability of machine time and tool slots. The proposed algorithm, coded in Matlab®, is tested on a case study referring to a major Italian company, which manufactures equipment and plants for the food industry. Two scenarios are evaluated to this end: an AS IS scenario, reflecting the current configuration of the production system, and a TO BE one, in which the MBBA is implemented for improving the system's performance, by determining a new sequence of jobs, able to minimise the variance of the processing time across the various machines. The application of the MBBA reveals significant improvements in processing time compared to the approach currently used by the company. The results of the TO BE scenario allow deriving useful indications to operations managers, helping them to identify an alternative strategy to enhance the efficiency of the targeted production department.

Keywords: *machine loading problem; flexible manufacturing systems; modified binary bat algorithm; system unbalance.*

A modified binary bat algorithm for machine loading in flexible manufacturing systems: a case study

Abstract

The machine loading (ML) problem of flexible manufacturing systems (FMS) has been recognized as one of the most important planning problems in industry. This study aims to minimize the system unbalance by developing and testing a modified binary bat algorithm (MBBA), which satisfies the technological constraints such as the availability of machine time and tool slots. The proposed algorithm, coded in Matlab®, is tested on a case study referring to a major Italian company, which manufactures equipment and plants for the food industry. Two scenarios are evaluated to this end: an AS IS scenario, reflecting the current configuration of the production system, and a TO BE one, in which the MBBA is implemented for improving the system's performance, by determining a new sequence of jobs, able to minimise the variance of the processing time across the various machines. The application of the MBBA reveals significant improvements in processing time compared to the approach currently used by the company. The results of the TO BE scenario allow deriving useful indications to operations managers, helping them to identify an alternative strategy to enhance the efficiency of the targeted production department.

Keywords: *machine loading problem; flexible manufacturing systems; modified binary bat algorithm; system unbalance.*

1 Introduction and background

The advent of global economy and trade has resulted in competition that has led enterprises to face a dynamic environment. Such an environment is characterized by a large volume of uncertainty, such as rapid market changes, increased product variety, competitive prices, and short product life cycles (Biswas & Mahapatra, 2008). These conditions coerce manufacturing firms to enhance response times and flexibility in all processes. Flexible manufacturing systems (FMSs) have been proved to respond to this challenge positively because of their ability to produce a variety of parts using the same system in the shortest possible lead time (Kumar, Murthy, & Chandrashekara, 2012). An FMS is an integrated manufacturing system which consists of a network of multi-functional numerically controlled (NC) machine tools, each with automatic tool changing capabilities and automated material transfer systems. The FMS are widely applied in industrial environments as they provide the flexibility of a low volume – high variety manufacturing along with the efficiency of a high volume – low variety manufacturing

(Ginoria, Samuel, & Srinivasan, 2015). However, building such systems using high technology equipment requires a huge initial investment. Indeed, FMSs are particularly expensive, and it is therefore crucial to manage these systems effectively for achieving optimum results with the lowest risk; this largely depends on the decision made to manage FMSs (Yusof, Budiarto, & Deris, 2011) (Mahmudy, 2015). In general, FMS planning consists of pre-release and post-release decisions. The pre-release decision problems consider the prearrangement of parts and tools before starting the activities of an FMS. FMS scheduling or post-release decisions consider instead sequencing and routing of part types, when the system is in progress (Kumar, Prakash, Tiwari, Shankar, & Baveja, 2006). Various post-release decisions in an FMS include: 1) job selection; 2) machine grouping; 3) determination of production ratio; 4) batching of the jobs; 5) allocation of pallets and fixtures; 6) allocation operations and tools among machines (machine loading (ML) problem) (Prakash, Khilwani, Tiwari, & Cohen, 2008). To be more precise, the ML problem in an FMS consists in assigning the machine, operations of selected jobs, and the tools necessary to perform these operations by satisfying the technological constraints (available machine time and tool slot constraints) when the system is working (Swarnkar & Tiwari, 2004). Stecke (1983) studied ML problems in detail and described its six main objectives: 1) balancing the assigned machine processing times; 2) minimizing the number of movements from machine to machine; 3) balancing the workload per machine for a system of groups of pooled machines of equal size; 4) unbalancing the workload per machine for a system of pooled machines of equal size; 5) filling the tool magazines as densely as possible; 6) maximizing the sum of operation priorities. Due to the various objectives covered, which involve the simultaneous determination of many factors, ML lies under the broad category of NP-hard problems (Yusof, Budiarto, & Deris, 2014). Numerous methods based on mathematical, heuristics, meta-heuristics models, as well as simulation, have been suggested by researchers in the pursuit of obtaining quality solutions to loading problems and reduce computational burden (Biswas & Mahapatra, 2008). Although analytical and mathematical programming-based methods are robust in their applications, they often become impractical when the problem size increases. This consideration motivated the researchers to develop fast and effective heuristics for solving loading problems in large-sized FMSs. The major limitation of heuristics, however, lies in their inability to estimate the results with a new or completely changed environment, as they are generally rule-based and mostly rely on empirical data. Therefore, recent literature has instead proposed meta-heuristic approaches for solving the ML problem (Kumar, Murthy, & Chandrashekara, 2012).

A careful analysis of the literature has identified the main techniques that researchers have proposed to solve the ML problem. For the sake of simplicity, in the following the approaches found will be divided into 4 main categories:

- Mathematical approaches (green): this group encompass global optimization techniques, which aim to find the optimal solution to the problem. These techniques generally start from an integer,

mixed, or binary linear programming model, which is solved directly or through "exact" techniques (e.g., the branch and bound method (Berrada & Stecke, 1986)).

- Heuristic algorithms (yellow): these approaches aim at finding good solutions to the problem, reducing computational time compared to the previous category (Radharamanan, 1986).
- Meta-heuristic algorithms (blue): this category includes algorithms which make use of higher-level modern techniques for finding the solution of the problem (Souier, Sari, & Hassam, 2013).
- Other techniques (grey): all approaches that cannot be classified under the previous ones fall under this residual category.

Table 1 summarizes the main studies about the ML problem, along with the approach and methodology used and the problem objectives.

[Table 1 near here]

The category of heuristic techniques includes 17 studies, and as shown in Table 1, these studies focus mainly on two key objectives: maximising the throughput and minimising the system unbalance. Meta-heuristic algorithms are by far more numerous (37 studies), which confirms their suitability to be applied to the ML problem. A wide variety of meta-heuristic approaches have been proposed in literature, including Genetic Algorithms (GAs) (Kumar & Shanker, 2000; Tiwari & Vidyarthi, 2000; Turkcan, Akturk, & Storer, 2007; Tyagi & Jain, 2008; Koşucuoğlu & Bilge, 2012; Ginoria, Samuel, & Srinivasan, 2015; Gaur, Indu, & Chawla, 2019), Particle Swarm Optimization (PSO) (Biswas & Mahapatra, 2007; Ponnambalam & Kiat, 2008; Santuka, Mahapatra, Dhal, & Mishra, 2015; Chawla, Chanda, & Angra, 2018), Immune algorithm (Prakash, Khilwani, Tiwari, & Cohen, 2008; Dhal, Mahapatra, Datta, & Mishra, 2010; Pandey, 2011), Simulated Annealing (Mukhopadhyay, Singh, & Srivastava, 1998; Tiwari, Kumar, Kumar, Prakash, & Shankar, 2006), Ant Colony Optimization (Verma, Shukla, Tiwari, & Shankar, 2007; Prakash, Tiwari, & Shankar, 2008) and Firefly algorithm (Bottani, Centobelli, Cerchione, Del Gaudio, & Murino, 2017). Some hybrid approaches were also developed, in which more algorithms are combined together for maximizing the solution performance; examples include hybrid Genetic Algorithm with Simulated Annealing (Yogeswaran, Ponnambalam, & Tiwari, 2009; Mandal, Pandey, & Tiwari, 2010), or with Particle Swarm Optimization (Kumar, Murthy, & Chandrashekara, 2012), or with Variable Neighborhood Search (Mahmudy, Marian, & Luong, 2013), and hybrid Simulated Annealing with Tabu Search (Arikan & Erol, 2012).

To the best of the authors' knowledge, the BA has not been proposed to solve the ML problem. Indeed, the BA is relatively new and its application to the operations management contexts is still at the early stage. Only a couple of studies (i.e., Ramesh, Mohan, & Reddy, 2013; Musikapun & Pongcharoen, 2012) have proposed the usage of the BA for solving scheduling problems; however, the problems addressed are quite far from the machine loading problem under examination in this paper. At the same time, the BA has some possible advantages compared to other meta-heuristic algorithms, and therefore

applying the BA to ML problems is expected to provide additional insights on its effective solution. Indeed, when proposing the BA, Yang X.-S. (2010) observed clearly that any meta-heuristic algorithm has advantages and disadvantages, and that the reason for developing the BA was exactly to combine the major advantages of other algorithms already available in literature. In the light of this consideration, the BA was expressly designed to be more powerful than the existing meta-heuristic algorithms, such as PSO and GA as well as Harmony Search (HS), and Yang X.-S. (2010) also demonstrated the superior performance of the BA over these algorithms. In addition, some algorithms (such as PSO and HS) can actually be seen as special cases of the BA under appropriate simplifications, which implicitly means that the BA is always able to solve problems for which those algorithms are used (Yang X.-S. , 2010).

Since evaluating the ML problem in FMSs and minimizing the system unbalance are increasingly important topics, this paper contributes to the literature by developing a Modified Binary Bat Algorithm (MBBA) suitable for application in the context of ML problems in FMSs for reducing the system unbalance – that is variability in the amount of under-and over-utilized time among the set of machines of the system – taking into account the technological constraints of the problem (i.e., tool slots and total machining time available). For testing purposes, a case study is presented, referring to an Italian company that manufactures equipment and plants for the food industry, and targeting the production department of that company. To prove the effectiveness of the proposed approach, two scenarios are evaluated and compared: an AS IS scenario, which represents the current organization of the production department (with relating performance), and a TO BE one, in which the MBBA is implementation for reducing the system unbalance; accordingly, for this latter scenario, the jobs sequence that minimizes the variance of the total processing time across the various machines was determined.

The remainder of the paper is organized as follows. Section 2 presents the methodological part of the study, including the notation used throughout the paper, a background on the ML problem structure, the theoretical background of Bat Algorithm (BA) and Binary Bat Algorithm (BBA), coming finally to the formulation of the proposed MBBA, customized for solving the ML problem. Section 3 is dedicated to the presentation of the case study, the implementation of the proposed algorithm and the description of the corresponding results. Finally, the key contribution, implications and limitations of the study are summarised in Section 4, together with a set of suggestions for future research activities.

2 Machine loading problem: formulation and solution approach

2.1 Nomenclature

The nomenclature used in this paper, to detail the mathematical model for the ML problem (in terms of notation, parameters, and decision variables), as well as the BA and the new MBBA, is shown in Table 2.

[Table 2 near here]

2.2 Theoretical background

2.2.1 Machine loading problem formulation

The loading problem in manufacturing deals with selecting jobs from a set of part types to be manufactured and assigning their operations to the relevant machines in a given planning horizon, at the same time meeting the technological constraints and trying to optimize certain performance measures (e.g., the minimization of the system unbalance, the maximization of the throughput or the minimization of the processing time – Kumar et al., 2006). System unbalance can be defined as the sum of unutilized or over-utilized times on all the machines available in the system. Minimization of the system unbalance is equivalent to maximization of machine utilization (Kumar, Murthy, & Chandrashekara, 2012). Such minimization is achieved by identifying an appropriate job sequence that also meets the given technological constraints.

Following the notation proposed in Table 2, the (binary) decision variables of the ML problem can have the following forms:

$$x_j = \begin{cases} 1 \; if \; job \; j \; is \; selected \\ 0 \qquad\qquad otherwise \end{cases} \tag{1}$$

$$x_{j,m} = \begin{cases} 1 \; if \; job \; j \; is \; assigned \; to \; machine \; m \\ 0 \qquad\qquad\qquad\qquad otherwise \end{cases} \tag{2}$$

$$x_{o,j,m} = \begin{cases} 1 \; if \; operation \; o \; of \; job \; j \; is \; assigned \; to \; machine \; m \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad otherwise \end{cases} \tag{3}$$

The (unitary) processing time for processing operation *o* of job *j* is expressed as $t_{o,j}$. Each job is to be manufactured in a batch of size b_j, which implies that the total processing time for the job can be computed as follows:

$$T_j = \sum_{o=1}^{O} t_{o,j} * b_j \tag{4}$$

For assigning an operation of a job to a machine, various technical aspects have to be evaluated. As a matter of fact, each machine *m* of the system has an available time for manufacturing (t_m), which cannot be exceeded. When a job is assigned to machine *m*, the remaining processing time can be computed as follows:

$$t_{o,j,m}^{r} = t_m - x_{o,j,m} * t_{o,j} \tag{5}$$

Also, a machine has a set of tools U_m needed for production; an operation of a job can be assigned to machine *m* only if there is the availability of the tool slots on that machine.

Once all the operations have been assigned to the machines of the system, the processing time of each machine, T_m, can be evaluated using the equation below:

$$T_m = \sum_{j=1}^{J} T_j x_{j,m} \tag{6}$$

The "system unbalance" reflects the asymmetry in the idle time and over utilized time among all machines of the system. Its computation requires the determination of the variance (σ_T^2) of the processing time across the machines, i.e. the deviation of the processing times of each machine from the average value of the total processing time μ_T, as shown in the following set of formulae:

$$\mu_T = \frac{1}{M} * \sum_{m=1}^{M} T_m \quad (7)$$

$$\sigma_T^2 = \frac{1}{M-1} * \sum_{m=1}^{M} (T_m - \mu_T)^2 \quad (8)$$

The system unbalance (eq.7) is obviously one out of the possible parameters to be used as objective function in ML problems (Kumar et al., 2006); that index was privileged in this study in view of the application of the solution approach in a real context, with a large amount of data. In such a context, a dispersion index is quite reliable and easy to compute even if the data analysed increases. Also, by minimising the variance of the total processing times, the operations will be assigned to the machines in such a way as to distribute the workload evenly in the manufacturing system.

Gathering the set of considerations and equations above, the ML problem can be formulated in mathematical terms as follows:

$$\textit{objective function: } \min \textit{system unbalance} = \min \sigma_T^2 \quad (9)$$

subject to:

$$\sum_{j=1}^{J} \sum_{o=1}^{O} t_{o,j,m}^{r} \geq 0 \qquad \forall m \quad (10)$$

$$\sum_{o=1}^{O} U_{o,j,m}\, x_{o,j,m} \leq U_m \quad \forall j \quad (11)$$

Constraints (10) and (11) reflects the key technological aspects of the system, previously mentioned. To be more precise, constraint (10) guarantees that the time remaining on machine *m* after allocating operation *o* of job *j* is still admissible. Constraint (11), instead, ensures that the operation *o* of job *j* will be loaded only if there is the availability of the tool slots on each machine *m*. This is also expected to ensure that the operation will be completed on the same machine once a machine has been selected for that job. Obviously, if both constraints are met, the assignment can be confirmed; otherwise, it will be modified.

2.2.2 Bat Algorithm

This sub-section and the following one provide the theoretical background of the proposed algorithm.

The BA is a quite recent meta-heuristic, swarm intelligence optimization algorithm, proposed by Yang X.-S., (2010) and (Reda, Hamdy, & Rashed, 2022). The BA was used successfully for continuous constrained optimization problems, for which it demonstrates a significant increase in the convergence rate to the global optimal solution (for well-tuned parameters), compared to other algorithms (Umar & Rashid, 2021). In particular, the BA approaches the most promising solution areas, increasing the chance

of a balance between the exploration and exploitation phase, which could be particularly helpful when examining large size optimization problems. This is often the case for the ML problem (Kumar et al., 2006), in which the number of operation/machine-allocation combinations can be particularly relevant, making the exhaustive search in the solution space practically unfeasible.

Nature-inspired BA works with the echolocation strategy of the bats to locate the prey and differentiate it from the barriers. Bats, indeed, find their targets by varying the loudness and the pulse rate, and according to these inputs, positions themselves. Loudness A_0 and pulse rate r parameters of the natural bats are also used in the BA (Yang X.-S., 2013), as well as the position x_i, frequency F_{min} and velocity v_i vectors. To be more precise, using the position and velocity vectors, the bat moves in the solution space which is a continuous real-time domain (Akila & Christe, 2022).

Mathematically, from iteration $iter$ of the algorithm to $iter + 1$, the movement of the virtual bat is described using Eqs.12–14 (Tsai, Pan, Liao, Tsai, & Istanda, 2012):

$$x_i^{iter+1} = x_i^{iter} + v_i^{iter+1} \quad (12)$$

$$v_i^{iter+1} = v_i^{iter} + \left(x_i^{iter} - x_{best}\right) f_i \quad (13)$$

$$f_i = F_{min} + (F_{max} - F_{min})\beta \quad (14)$$

Eqs.12-14 ensure the exploitability and exploration phases of the algorithm. However, the exploitation phase is performed by randomly selecting a solution among the current best solutions to perform a random wall as follows:

$$x_{new} = x_{old} + \varepsilon A^{iter} \quad (15)$$

The basic steps of the BA can be summarized in the pseudo-code shown in Figure 1.

[Figure 1 near here]

In Figure 1, $rand$ is random number taken from a uniform distribution in the range [0,1]. A_i and r_i (if they are not set at a constant value) are updated from iteration $iter$ to $iter + 1$ as follows:

$$A_i^{iter+1} = \alpha x_i^{iter} \quad (16)$$

$$r_i^{iter+1} = r_i^{iter}[1 - \mathrm{e}\,(-\varphi^{Iter})] \quad (17)$$

For any $0 < \alpha < 1$ and $\varphi > 0$, at the end, A_i and r_i converge to zero and r_0, respectively (Jaafer, Al-Bazoon, & Dawood, 2020).

2.2.3 Binary Bat Algorithm

The BBA, which was first proposed by (Nakamura, Pereira, Costa, Rodrigues, & Papa, 2012), is one of the recent meta-heuristic optimization algorithms that simulate the echolocation behaviour of bats for solving different problems. The artificial bats move in the binary search space based on the position,

velocity, and frequency vectors, which are updated at each iteration (Mouwafi, El-Ela, El-Sehiemy, & Al-Zahar, 2021). In the BBA, artificial bats explore and hunt prey in the binary space, searching by changing the position from "0" to "1", which requires a change in the position and speed update of the standard BA. The change required can be applied by introducing a transfer function, which will "force" the bats to move in the binary space (Šipoš, Klaić, Nyarko, & Fekete, 2021). The BBA restricts the new bat's position to the binary value by using a V-shaped transfer function (eq.18):

$$V(v_i^{k,iter}) = \left|\frac{2}{\pi} - \arctan\left(\frac{\pi}{2} v_i^{k,iter}\right)\right| \quad (18)$$

where $v_i^{k,iter}$ is the velocity of *i*-th bat at iteration $iter$ in *k*-th dimension of the problem.

The position updating formula is represented as follows:

$$x_i^{k,iter+1} = \begin{cases} \left(x_i^{k,iter}\right)^{-1} 1 \; if \; rand < V(v_i^{k,iter+1}) \\ x_i^{k,iter} \qquad\qquad otherwise \end{cases} \quad (19)$$

In Eq.19, $x_i^{k,iter}$ indicates the position of *i*-th bat at iteration $iter$ in *k*-th dimension, $\left(x_i^{k,iter}\right)^{-1}$ is the complement of $x_i^{k,iter}$ and $rand$ is a random number taken from a uniform distribution (Xie, et al., 2019).

Eq.18 is used as the transfer function for mapping the velocities of the bats to the probabilities of flipping their position vectors' elements. Consequently, the rules in eq.19 are used to update the position vectors. The general steps of the BBA are presented in Figure 2.

[Figure 2 near here]

2.3 Proposed approach: the Modified Binary Bat Algorithm

This section presents the adaptation of the BBA to the ML problem, following the main steps of the algorithm.

2.3.1 Population and parameters initialization

For initialization purpose, a different technique is used compared to the traditional approach, in which the initial solution is a Boolean vector (0-1) obtained randomly. After fixing the number of bats (*N* solutions), a hybrid technique combining a sequence of low discrepancy numbers with the opposition-based learning technique, is used to assign the first values of these bats. The technique is developed in three steps:

1- The Halton generator is used to generate a population (*P*) of vectors whose components take on real values between 0 and 1.
2- By applying the Largest Ranked Value (LRV) technique, the generated number vectors are transformed into machine permutations. The operations to be carried out are associated to them, in the form of integer values between 1 and *M*. Thus, the initial population *P* is obtained.

3- Finally, the opposition-based learning (OBL) technique is used to determine the opposite population (*OP*) to the one obtained at step 2 above. OBL is a relatively known technique used in the context of computational intelligence (Tizhoosh, 2005) and has been applied to improve the performance of various optimization approaches, including heuristic algorithms (Ventresca & Tizhoosh, 2008). As such, it is also used in the proposed MBBA. The basic idea of this technique is that when evaluating the solution of a problem, its opposite solution should be simultaneously taken into account, to improve the chance of finding an alternative candidate solution, which could even be closer to the global optimum. The following formula is used:

$$OP_i = \max(P_i) - P_i + 1 \quad (20)$$

The population finally used to initialise the algorithm is the union of the two populations returned by steps 2 and 3:

$$Pop_{init} = \{(P) \cup (OP)\} \quad (21)$$

The proposed initialization technique is expected to improve the performance of the BBA, as during simulations carried out in Matlab® it has been observed that the computational time of the algorithm decreases considerably, thus favouring the identification of better solutions.

The initial population will therefore consist of a matrix whose rows list the *N* bats (potential solutions to the problem), and whose columns list the *O* operations that must be associated with the *M* machines. Each solution will therefore be a generic row of the matrix, composed of integer values $m_{o,i}$ ranging from 1 to *M*, reflecting the association of *o*-th operation to the *m*-th machine. For the sake of clarity, a flowchart describing the steps of the proposed initialisation technique is shown in Figure 3, while an example of the initial population is shown in Table 3.

[Figure 3 near here]

[Table 3 near here]

In the example of Table 3, the population consists of *N*=15 bats and the *O*=8 operations to be assigned to the available machines; the rows of the matrix correspond to the initial solutions of the problem.

Before starting the iterative procedure of the algorithm, some parameters should be initialized. These parameters, described in Table 2, are $A_o; r; F_{min}; F_{max}; F_o; V_o$ and $max\ iter$.

2.3.2 Fitness evaluation

At this point the iterative procedure of the algorithm begins and stops when the maximum number of iterations is reached. Using the initial population, for each solution a fitness index (i.e., the value of the objective function associated with that solution) must be evaluated. This requires the application of the set of formulae in section 2.2.1. Eq.8, in particular, is used to evaluate the system unbalance as the variance of the processing time across the machines, and to derive a fitness index of each solution.

Assigning operations to machines while minimising the variance of the total processing time associated with each machine means distributing the production activities in such a way as to level out the workload of the machines, which is expected to minimise over- and under-use of the production system. Therefore, each solution identifies a different allocation of operations to the various machines; for each solution, the variance of the total processing time of the machines to which these operations have been assigned is evaluated. This line of reasoning can be summarised into five vectors, listed and described in the example below, which refers to a simplified scenario which consists of 10 operations (O=10) and 5 machines (M=5).

Vector #1: unit operation processing time

o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}

Vector #2: Operations batch vector

o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}

Vector #3: total processing times of operations

o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
$t_1 * b_1$	$t_2 * b_2$	$t_3 * b_3$	$t_4 * b_4$	$t_5 * b_5$	$t_6 * b_6$	$t_7 * b_7$	$t_8 * b_8$	$t_9 * b_9$	$t_{10} * b_{10}$

Vector #4: allocation of operations to the machines

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
machine	3	5	1	4	2	1	4	3	5	2

Vector #5: total processing time associated with machines (T_1, ... T_5)

$\boldsymbol{m = 1}$	$T_1 = t_3 * b_3 + t_6 * b_6$
$\boldsymbol{m = 2}$	$T_2 = t_5 * b_5 + t_{10} * b_{10}$
$\boldsymbol{m = 3}$	$T_3 = t_1 * b_1 + t_8 * b$
$\boldsymbol{m = 4}$	$T_4 = t_4 * b_4 + t_7 * b_7$
$\boldsymbol{m = 5}$	$T_5 = t_2 * b_2 + t_9 * b_9$

The variance of the values of vector #5, computed as per eq.8, reflects the effectiveness of the distribution of operations between the various machines, thus providing a measure of the system unbalance. For completeness, a flowchart describing the steps of the proposed fitness evaluation is shown in Figure 4.

[Figure 4 near here]

2.3.3 Update of the best solution

On the basis of the fitness evaluation, the solution with the lowest variance will be chosen as the best solution of the problem and denoted as x_{best}^{iter}. This solution is to be compared with the best solution found so far by the algorithm, x_{best}, to see whether the iteration has improved the available best solution. Obviously, at $iter = 1$ there are no comparisons to be made because there are no previously available best solutions; hence, this step is merely a selection of the best solution (i.e., $x_{best} = x_{best}^{1}$). For the generic $iter \in]1\,; max\ iter]$, a check is made to see whether x_{best}^{iter} improves the current solution, x_{best} is updated to the new solution ($x_{best} = x_{best}^{iter}$). Otherwise, the previously available best solution will be retained, which implicitly means $x_{best} = x_{best}^{iter-1}$.

2.3.4 Population update

As mentioned above, in a machine loading problem the population consists of a matrix of *N* rows (number of solutions) and *O* columns (number of operations to be assigned); at each iteration of the algorithm, a generic cell $m_{o,i}^{iter}$ denotes the machine m to which operation o has been assigned. From iteration $iter$ to $iter + 1$, the solutions will be updated by computing the values in eqs.20 and 21:

$$f_{o,i}^{iter} = F_{min} + (F_{max} - F_{min}) * \beta \tag{20}$$

$$v_{o,i}^{iter+1} = v_{o,i}^{iter} + \left(m_{o,i}^{iter} - m_{o,i}^{iter,best}\right) * f_{o,i}^{iter} \tag{21}$$

The update of the matrix values to $m_{o,i}^{iter+1}$ will be made by comparing $v_{o,i}^{iter+1}$ and β, according to the following formula:

$$m_{o,i}^{iter+1} = \begin{cases} m_{o,i}^{iter}, \ if\ v_{o,i}^{iter+1} < \beta \\ ; \\ m_{o,i}^{iter} + 1, \ if\ v_{o,i}^{iter+1} \geq \beta \end{cases} \tag{22}$$

At each iteration the value of A_i and r_i are updated using the previous eqs.16 and 17.

Once the positions of the entire population of bats have been updated, the new fitness values will be calculated, meaning that the algorithm will restart from the "Fitness evaluation" step, unless the maximum number of iterations has been reached. If this is the case, the algorithm will stop and return as the best solution of the problem the bat whose position has the lowest variance of the total processing times.

The overall framework of the approach proposed is shown in Figure 5.

[Figure 5 near here]

3 Case study

3.1 Context

The company taken as case study has been analysed in detail by Bottani et al. (2018) in a previous publication, to which the reader is referred for further discussion. That company will be referred to as *Company A* for the sake of confidentiality and is an Italian manufacturer of plants for the food industry, which exports its products in several countries worldwide. In Italy, Company A is market leader, while it ranks second among the top producers of milling plants on a global basis.

Company A operates on an Engineer-To-Order (ETO) basis: anytime a new order is accepted, the production cycle starts with the design of the plant and proceeds through manufacturing, assembly, installation, civil works, testing of the equipment at the customer's site and staff training. The machinery department where this study was carried out (highlighted in red in Figure 6) covers an area of approximately 2,400 square meters and includes 19 machines.

[Figure 6 near here]

Production starts from raw materials (e.g. bars and rods in steel, cast iron, aluminium, brass or other metal alloy), which are converted, through several mechanical processes, into specific mechanical parts, according to the 2-D or 3-D drawing prepared by the technical division. The mechanical processes consist, mainly, in chip removal operations, carried out on automatic machines, organised in a job-shop layout. Some machines can perform a specific process only (i.e., turning, grinding and milling machines), while other more flexible ones (i.e., machining centres) can perform a large set of mechanical machining. Anyhow, to operate both types of machines require the supervision of a worker, who may also perform manual operations such as loading/unloading, set-up and start up. Typically, each worker supervises two machines, with only one exception of an employee who controls three machines. For this reason, to minimize the number of supervisors, machines are aggregated into nine groups referred to as "cells". Machines belonging to the same cell may be either identical or different, but similar in terms of functionality.

Orders are planned according to a push strategy: every week accepted jobs are released in the shop floor and, next, they are sequenced accordingly to the earliest due date (EDD) dispatching rule, with no pre-emption, i.e., priority is given to the job with the closest delivery date. Sometimes jobs may be characterized by one or more production phases and in this case a specific route into the department and on the different machines is preliminary defined for each job to be processed.

Most of the production data was retrieved from the company's information systems, in which 756 different jobs manufactured in the targeted department were found. Similarly, the number of parts to be manufactured was collected from the company's database, over a time horizon of six months (June-December 2021). In Table 4 a summary of input data is provided.

[Table 4 near here]

For the company under examination, $t_{o,j}$ and b_j are also known, as they were taken from the historical production data recorded in the company's information system. It is important to specify that the production is scheduled on 5 days with a single 8-hour shift, and on one day with two 8-hour shifts. As already mentioned, the production time span taken into account in the analysis covers 6 months (H=144 days); with these considerations, the available processing times (excluding overtime work) of each machine in the system $(t_m, \forall m = 1, \dots M)$ can be computed as follow:

$$t_m = \frac{144}{(5+1)} * [(5 * 8) + (1 * 2 * 8)] * 60 = 80{,}640\ min \quad (23)$$

Moreover, it is assumed that there are no unexpected downtimes (failures and/or extraordinary maintenance) in that period.

3.2 Scenario analysis

In this section, an analysis of two different configurations – named AS IS and TO BE – is provided. The data collected during the observation period (AS IS scenario) showed a poorly organised production state, as can be seen from the left part of Table 5. Indeed, taking t_m=80,640 minutes as benchmark, it is easy to see that numerous machines have instead worked for a significantly lower time $(m_3, m_6, m_{10}, m_{11}, m_{13}, m_{15}, m_{17}, m_{19})$. In particular, m_{17} and m_{19} appear to be very under-used compared to their potential production capacity, which in general terms means a lack of resource utilisation. Other machines, instead, have been over-used $(m_4, m_8, m_9, m_{12}, m_{14}, m_{18})$, which means that (for sure) they have worked in extra-time, involving extra-cost as well. Machines subject to over-use are obviously main prone to failures, or at least they need additional maintenance activities to ensure a sufficient useful life; on the other hand, under-utilised machines do not pay back investment made by the company to purchase them. Both situations, were possible, should therefore be avoided or limited. Other typical consequences of machine under- or over-utilization, which were also observed in the production department under examination, are the possible production delays; indeed, it manufacturing activities are not properly scheduled, it is likely that the company is not able to fulfil all the customers' requests on time. However, as Company A manufactures plants, orders often involve penalties in the case of delayed delivery of the products. A typical countermeasure to production shortfall is the usage of available (safety) stocks, which however, for being used to cope with production inefficiency, must increase significant in their amount; this ultimately leads to additional costs. Overall, the inefficiencies of the AS IS situation are reflected by the quite high system unbalance, which was computed using eq.8 and scores $2.97 * 10^9$. By the way, as it is often observed in real manufacturing systems, the schedule of working activity at Company A is mainly based on the experience of the operations manager but is not grounded on any particular optimization strategy.

In the TO BE scenario, instead, schedule of the production activities was reorganised by applying the proposed MBBA to solve the ML problem. The final goal is to minimise the system unbalance by

properly redistributing operations to machines. As already mentioned, the algorithm was implemented in Matlab® following the flowchart proposed in Figure 5. For the case under examination, the values of the parameters were chosen taking inspiration from similar examples available in literature (Gandomi, Yang, Alavi, & Talatahari, 2013), as well as by performing a preliminary series of hand-turning experiments. The parameters were finally set as follows: N=20; A_0=0.90; r_0=0.10; F_{min}=0.00; F_{max}=5.00; *max iter*=1,000. With those settings, the algorithm took 51.14 seconds to be run on a 16 gigabytes RAM, Intel Core i7 CPU laptop computer, equipped with Microsoft Windows 10 operating system.

At the end of the iterations, the algorithm returned as (best) solution an assignment of operations to machines that generated the processing times shown in the right part of Table 5. Looking at these times, it is evident that the TO BE configuration obtained using the MBBA generates a quite well-balanced production system. Indeed, almost all the machines have a total processing time close to 65,000 minutes, which is well below the available production time of 80,640 minutes, but most importantly, is quite constant across the various machines. For two machines only (m_{15}, m_{17}) the production time reaches a value of 79,200 minutes, which is close to the available production time. These machines were both underutilised in the AS IS scenario (with a peak of under-usage for m_{17}); therefore, it is likely to expect that the proposed algorithm has focused on increasing the usage of those machines, which were particularly promising from this point of view. The variance of the total processing time, which is taken as objective function, scores $1.93 * 10^7$ for that assignment. Overall, this result shows a better balance of the production system under examination, highlighted by a decrease in σ_T^2 of two orders of magnitude.

[Table 5 near here]

4 Conclusions

The ML problem in FMSs has been typically modelled as a problem that involves grappling with a wide variety of constraints and objectives. In this paper, this problem was modelled and solved with the usage of a new meta-heuristic procedure. To be more precise, this study has presented an efficient meta-heuristic approach based on the BA, to determine the job assignment to machines that minimises the system unbalance of FMSs. This index has been chosen as the objective function, and measures the variance of the total processing time, which is expected to capture the real situations in which numerous machines are typically involved. The proposed algorithm, referred to as MBBA, is an adaptation of the BBA, which easily captures the technological constraints of the problem, such as availability of tool slots and machining time.

For testing purpose, the proposed MBBA was applied to a real FMS, referring to a manufacturing company based in the North of Italy. Two scenarios were evaluated for the targeted company: the AS IS scenario, which represents the current situation of the production department, and a TO BE configuration, in which production activities are redefined thanks to the implementation of the proposed MBBA. For the TO BE scenario, an optimized job sequence was found, that allows for minimizing the

variance of the total processing time. The results of the application have therefore demonstrated that the MBBA is a viable and reliable method for solving the ML problem. Indeed, the outcomes of the TO BE scenario are overall satisfactory and show considerable improvements in term of system unbalance compared to the AS IS configuration.

Implications of these outcomes are manifold. From a scientific perspective, both the problem studied, and the algorithm were modelled in a general form; this implies that the proposed MBBA could virtually be applied to any FMS context in which a ML problem has to be studied and optimized. The modelling of the MBBA also has some strong points; in particular, the three-step initialization technique used is expected to improve the algorithm performance by decreasing the computational time significantly, which was also demonstrated during the implementation. The detailed fitness evaluation procedure is also new compared to the existing studies.

From a practical point of view, the fact that in the TO BE scenario more machines are used for the same amount of time implies that (on average), there are no under- or over-utilized machines. In particular, the algorithm proved to be effective in identifying the machines that were under-utilised in the AS IS scenario (i.e., machines 15 and 17 among others), and therefore have most potential for being used, and on focusing the optimization procedure on those machines. Overall, a more balanced workload across the various machine means that fewer failures could be expected in the production system, involving lower maintenance cost and resulting in longer life of the machines themselves. Also, better organization of the manufacturing activities should almost remove production delays and associated cost (e.g., penalties for late delivery of the finished products). Inventory holding cost can also be reduced. Indeed, stocks (and safety stocks in particular) are frequently used by companies for coping with production shortfalls; with an improved organization of manufacturing activities, these stocks will become less necessary and their amount, consequently, can be decreased, generating savings. Finally, as a general consideration, anytime the saturation level of a production plant improves, the payback time of the investment in the relating machines decreases proportionally, which is always an important parameter for companies.

Some limitations of the analysis also need to be mentioned. As a first (technical) point, the proposed MBBA, although effective for the chosen case study, could suffer from some weaknesses typical of the original algorithm (i.e., the BA). To be more precise, issues could arise when trying to solve problems of larger size, which increase the risk of not finding the global optimal solution. Applying the same proposed approach to problems of larger size is therefore recommended to test its effectiveness in these scenarios. Always from the technical point of view, the objective function and constraints set in the model perfectly fit the context under examination, i.e. the optimization of the workload balance in a FMS system, but obviously, a different context could best fit a different objective function. More in general, it is to be expected that some adaptations could be required when applying the proposed approach to a different system.

From a more general perspective, although the proposed approach proved to be effective in the context under examination, it is challenging to evaluate its performance compared to other nature-inspired algorithms that could be used for solving similar problems, and thus, comparative analyses cannot be made. But by the way, conducting such analyses goes beyond the scope of this study, which instead primarily focuses on developing a MBBA suitable to be used for solving the ML problem in FMSs and on showing its effectiveness when applied to a real case. As a matter of fact, nature-inspired algorithms are, at present, very numerous, and it is almost unfeasible to reproduce all them in one single study for comparison purpose only; also, most of them would probably need some adaptations to be used for solving the ML problem in FMSs, meaning that their application is not immediate. Even if looking at the research studies that have already applied nature-inspired algorithms for solving ML problems in FMSs, most of the previous literature lacks sufficient details for the algorithm or case study to be fully reproduced, and sometimes also lacks a detailed performance analysis, e.g. in the form of computational time or size of the problem solved; again, this prevents the possibility of making comparative evaluations. Comparative analyses are therefore left for future studies.

Similarly, future studies could start from the results of this paper, in the form of a reorganization of the production activities, and conduct a thorough cost/benefit analysis, including the cost for changing the organization of the manufacturing department and the savings that could derive from that reorganization, as briefly sketched above. As per the previous point, this analysis as well goes beyond the scope of the present paper, because, for sure, it would require collecting additional data, different from those used for testing the algorithm. Another future research direction that could be undertaken starting from this study concerns the extension of the proposed algorithm to a multi-objective domain, in which more objective functions could be simultaneously taken into account in the MBBA.

Declarations

1. No funding was received for conducting this study
2. The authors have no relevant financial or non-financial interests to disclose
3. The authors report there are no competing interests to declare.

Data availability statement

The authors confirm that the data supporting the findings of this study are available within the article or its supplementary materials.

Notes on contributors

Giorgia Casella is Assistant Professor at the Department of Engineering and Architecture of the University of Parma. After getting the bachelor's degree in engineering management in March 2014, she has achieved a Master's degree in Industrial Engineering and management in October 2016 at the same University. She got her Ph.D. in Industrial Engineering in 2022 at the University of Parma. Her

research activities concern logistics and supply chain management issues. She is author (or co-author) of thirty-three scientific publications.

Teresa Murino graduated in Mechanical Engineering and is Associate Professor in the ING-IND 17 "Industrial Mechanical Systems Engineering" disciplinary group, at the University of Naples Federico II". She teaches Operations Management, Goods and Services Production System and Industrial Logistics at Engineering Faculty. She is also Professor at "Consorzio Nettuno". She is also reviewer for Elsevier Editorials and other journal ISI indexed. The research activities are primarily concerned about the following topics: simulation modeling; maintenance strategies; supply chain management models; quick response manufacturing; sustainable production processes; location-routing and vehicle routing problem; lean service and lean production implementation.

Eleonora Bottani is Full professor of Industrial Logistics at the Department of Engineering and Architecture of the University of Parma since November 2019. She graduated (with distinction) in Industrial Engineering and Management in 2002 and got her Ph.D. in Industrial Engineering in 2006, both at the University of Parma. Her research activities concern logistics and supply chain management issues. She is author (or co-author) of >200 scientific papers (citations on Scopus>4100; H-index=32), referee for more than 60 international journals, editorial board member of five scientific journals, Associate Editor for various journals, and editor-in-chief of a scientific journal.

References

1. Abazari, A., Solimanpur, M., & Sattari, H. (2012). Optimum loading of machines in a flexible manufacturing system using a mixed-integer linear mathematical programming model and genetic algorithm. *Computers and Industrial Engineering*, 62(2), 469-478. doi:10.1016/j.cie.2011.10.013
2. Akila, S., & Christe, S. (2022). A wrapper based binary bat algorithm with greedy crossover for attribute selection. *Expert Systems with Applications*, 187. doi:10.1016/j.eswa.2021.115828
3. Arikan, M., & Erol, S. (2012). A hybrid simulated annealing-tabu search algorithm for the part selection and machine loading problems in flexible manufacturing systems. *International Journal of Advanced Manufacturing Technology*, 59(5-8), 669-679. doi:10.1007/s00170-011-3506-0
4. Azizmohammadi, R., & Mahmoudabadi, A. (2019). SMOSA optimization approach to a multi objective model for a machine tool selection and operation allocation problem in FMS. *Proceedings of the International Conference on Industrial Engineering and Operations Management*, 3279-3290.
5. Basnet, C. (2012). A hybrid genetic algorithm for a loading problem in flexible manufacturing systems. *International Journal of Production Research*, 50(3), 707-718. doi:10.1080/00207543.2010.543935
6. Berrada, M., & Stecke, K. (1986). Branch and bound approach for machine load balancing in flexible manufacturing systems. *Management Science*, 32(10), 1316-1335. doi:10.1287/mnsc.32.10.1316

7. Biswas, S., & Mahapatra, S. (2007). Machine Loading in Flexible Manufacturing System:A Swarm Optimization Approach. *Proceedings of the Eighth International Conference on Operations and Quantitative Management*, 621-628.

8. Biswas, S., & Mahapatra, S. (2008). Modified particle swarm optimization for solving machine-loading problems in flexible manufacturing systems. *International Journal of Advanced Manufacturing Technology*, 39(9-10), 931-942. doi:10.1007/s00170-007-1284-5

9. Bottani, E., Centobelli, P., Cerchione, R., Del Gaudio, L., & Murino, T. (2017). Solving machine loading problem of flexible manufacturing systems using a modified discrete firefly algorithm. *International Journal of Industrial Engineering Computations*, 8(3), 363-372. doi:10.5267/j.ijiec.2016.12.002

10. Bottani, E., Rinaldi, M., Montanari, R., Bertolini, M., & Zammori, F., (2018). An operating simulation tool for modelling and managing a job shop system. *International Journal of Service and Computing Oriented Manufacturing*, 3(4), 318-342. doi: 10.1504/IJSCOM.2018.099459.

11. Bretthauer, K., & Venkataramanan, M. (1990). Machine loading and alternate routing in a flexible manufacturing system. *Computers and Industrial Engineering*, 18(3), 341-350. doi:10.1016/0360-8352(90)90056-R

12. Chawla, V., Chanda, A., & Angra, S. (2018). Multi-load AGVs scheduling by application of modified memetic particle swarm optimization algorithm. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 40(9). doi:10.1007/s40430-018-1357-4

13. Co, H., Biermann, J., & Chen, S. (1990). A methodical approach to the flexible-manufacturing-system batching, loading and tool configuration problems. *International Journal of Production Research*, 28(12), 2171-2186. doi:10.1080/00207549008942860

14. Dhal, P., Mahapatra, S., Datta, S., & Mishra, A. (2010). An improved artificial immune system for solving loading problems in flexible manufacturing systems. *IEEM2010 - IEEE International Conference on Industrial Engineering and Engineering Management*, 585-589. doi:10.1109/IEEM.2010.5674516

15. Gamila, M., & Motavalli, S. (2003). A modeling technique for loading and scheduling problems in FMS. *Robotics and Computer-Integrated Manufacturing*, 19(1-2), 45-54. doi:10.1016/S0736-5845(02)00061-3

16. Gandomi, A., Yang, X., Alavi, A., & Talatahari, S. (2013). Bat algorithm for constrained optimization tasks. *Neural Computing and Applications*, 22(6), 1239-1255.

17. Gaur, K., Indu, & Chawla, V. (2019). Minimizing unbalance of flexible manufacturing system by genetic algorithm. *Advances in Intelligent Systems and Computing*, 741, 697-705. doi:10.1007/978-981-13-0761-4_67

18. Ginoria, S., Samuel, G., & Srinivasan, G. (2015). Optimisation of a machine loading problem using a genetic algorithm-based heuristic. *International Journal of Productivity and Quality Management*, 15(1), 36-56. doi:10.1504/IJPQM.2015.065984

19. Goswami, M., & Tiwari, M. (2006). A reallocation-based heuristic to solve a machine loading problem with material handling constraint in a flexible manufacturing system. *International Journal of Production Research*, 44(3), 569-588. doi:10.1080/00207540500266263

20. Guerrero, F., Lozano, L., Koltai, T., & Larrañeta, J. (1999). Machine loading and part type selection in flexible manufacturing systems. *International Journal of Production Research*, 37(6), 1303-1317. doi:10.1080/002075499191265

21. Gupta, J., Luong, L., & Nguyen, V. (1999). Part dispatching and machine loading in flexible manufacturing systems using central queues. *International Journal of Production Research*, 37(6), 1427-1435. doi:10.1080/002075499191337

22. Hsu, V., & De Matta, R. (1997). An efficient heuristic approach to recognize the infeasibility of a loading problem. *International Journal of Flexible Manufacturing Systems*, 9(1), 31-50. doi:10.1023/A:1007925809798

23. Huka, M., Rindler, C., & Gronalt, M. (2021). Scheduling and loading problem for multiple, identical dry kilns. *Flexible Services and Manufacturing Journal*, 33(2), 312-336. doi:10.1007/s10696-020-09386-4

24. Jaafer, A., Al-Bazoon, M., & Dawood, A. (2020). Structural topology design optimization using the binary bat algorithm. *Applied Sciences*, 10(4). doi:10.3390/app10041481

25. Kato, K., Oba, F., & Hashimoto, F. (1993). A GT-based heuristic approach for machine loading and batch formation in flexible manufacturing systems. *Control Engineering Practice*, 7(5), 845-850. doi:10.1016/0967-0661(93)90252-M

26. Kim, Y.-D. (1993). A study on surrogate objectives for loading a certain type of flexible manufacturing systems. *International Journal of Production Research*, 31(2), 381-392. doi:10.1080/00207549308956731

27. Kim, Y.-D., & Yano, C. (1994). A new branch and bound algorithm for loading problems in flexible manufacturing systems. *International Journal of Flexible Manufacturing Systems*, 6(4), 361-381. doi:10.1007/BF01324801

28. Kim, Y.-D., & Yano, C. (1997). Impact of throughput-based objectives and machine grouping decisions on the short-term performance of flexible manufacturing systems. *International Journal of Production Research*, 35(12), 3303-3322. doi:10.1080/002075497194084

29. Koşucuoğlu, D., & Bilge, Ü. (2012). Material handling considerations in the FMS loading problem with full routing flexibility. *International Journal of Production Research*, 50(22), 6530-6552. doi:10.1080/00207543.2011.653837

30. Kumar, A., Prakash, Tiwari, M., Shankar, R., & Baveja, A. (2006). Solving machine-loading problem of a flexible manufacturing system with constraint-based genetic algorithm. *European Journal of Operational Research*, 175(2), 1043-1069. doi:10.1016/j.ejor.2005.06.025

31. Kumar, N., & Shanker, K. (2000). A genetic algorithm for FMS part type selection and machine loading. *International Journal of Production Research*, 38(16), 3861-3887. doi:10.1080/00207540050176058

32. Kumar, N., & Shanker, K. (2001). Comparing the effectiveness of workload balancing objectives in FMS loading. *International Journal of Production Research*, 39(5), 843-871. doi:10.1080/00207540010002847

33. Kumar, R., Singh, A., & Tiwari, M. (2004). A fuzzy based algorithm to solve the machine-loading problems of a FMS and its neuro fuzzy petri net model. *International Journal of Advanced Manufacturing Technology*, 23(5-6), 318-341. doi:10.1007/s00170-002-1499-4

34. Kumar, V., Murthy, A., & Chandrashekara, K. (2012). A hybrid algorithm optimization approach for machine loading problem in flexible manufacturing system. *Journal of Industrial Engineering International*, 8(1). doi:10.1186/2251-712X-8-3

35. Lee, D.-H., & Kim, Y.-D. (1998). Iterative procedures for multi-period order selection and loading problems in flexible manufacturing systems. *International Journal of Production Research*, 36(10), 2653-2668. doi:10.1080/002075498192418

36. Lee, D.-H., & Kim, Y.-D. (2000). Loading algorithms for flexible manufacturing systems with partially grouped machines. *IIE Transactions (Institute of Industrial Engineers)*, 32(1), 33-47. doi:10.1023/A:1007651329917

37. Liang, M. (1993). Part selection, machine loading, and machining speed selection in flexible manufacturing systems. *Computers and Industrial Engineering*, 25(1-4), 259-262. doi:10.1016/0360-8352(93)90270-8

38. Liang, M. (1994). Integrating machining speed, part selection and machine loading decisions in flexible manufacturing systems. *Computers and Industrial Engineering*, 26(3), 599-608. doi:10.1016/0360-8352(94)90053-1

39. Liang, M., & Dutt, S. (1990). A mixed-integer programming approach to the machine loading and process planning problem in a process layout environment. *International Journal of Production Research*, 28(8), 1471-1484. doi:10.1080/00207549008942806

40. Mahmudy, W. (2015). Optimization of part type selection and machine loading problems in flexible manufacturing system using variable neighborhood search. *IAENG International Journal of Computer Science*, 42(3), 1-11.

41. Mahmudy, W., Marian, R., & Luong, L. (2013). Hybrid genetic algorithms for multi-period part type selection and machine loading problems in flexible manufacturing system. *Proceeding - IEEE CYBERNETICSCOM 2013: IEEE International Conference on Computational Intelligence and Cybernetics*, 126-130. doi:10.1109/CyberneticsCom.2013.6865795

42. Mandal, S., Pandey, M., & Tiwari, M. (2010). Incorporating dynamism in traditional machine loading problem: An AI-based optimisation approach. *International Journal of Production Research*, 48(12), 3535-3559. doi:10.1080/00207540902814306

43. Mgwatu, M. (2011). Interactive Decisions of Part Selection, Machine Loading, Machining Optimisation and Part Scheduling Sub-problems for Flexible Manufacturing Systems. *International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies*, 2, 93-109.

44. Mirjalili, S., Mirjalili, S., & Yang, X.-S. (2014). Binary bat algorithm. *Neural Computing and Applications*, 25(3-4), 663-681. doi:10.1007/s00521-013-1525-5

45. Mouwafi, M., El-Ela, A., El-Sehiemy, R., & Al-Zahar, W. (2021). Techno-economic based static and dynamic transmission network expansion planning using improved binary bat algorithm. *Alexandria Engineering Journal*. doi:10.1016/j.aej.2021.06.021

46. Mukhopadhyay, S., Midha, S., & Krishna, V. (1992). A heuristic procedure for loading problems in flexible manufacturing systems. *International Journal of Production Research*, 30(9), 2213-2228. doi:10.1080/00207549208948146

47. Mukhopadhyay, S., Singh, M., & Srivastava, R. (1998). FMS machine loading: A simulated annealing approach. *International Journal of Production Research*, 36(6), 1529-1547. doi:10.1080/002075498193156

48. Musikapun, P., & Pongcharoen, P. (2012). Solving multi-stage multi-machine multiproduct scheduling problem using bat aglorithm. *Proceedings of the 2nd International Conference on Management and Artificial Intelligence (IPEDR)*. 35, p. 98-102. Singapore: IACSIT Press.

49. Nagarjuna, N., Mahesh, O., & Rajagopal, K. (2006). A heuristic based on multi-stage programming approach for machine-loading problem in a flexible manufacturing system. *Robotics and Computer-Integrated Manufacturing*, 22(4), 342-352. doi:10.1016/j.rcim.2005.07.006

50. Nakamura, R., Pereira, L., Costa, K., Rodrigues, D., & Papa, J. Y.-S. (2012). BBA: A binary bat algorithm for feature selection. *Brazilian Symposium of Computer Graphic and Image Processing*, 291-297. doi:10.1109/SIBGRAPI.2012.47

51. Nayak, G., & Acharya, D. (1998). Part type selection, machine loading and part type volume determination problems in FMS planning. *International Journal of Production Research*, 36(7), 1801-1824. doi:10.1080/002075498192977

52. Nejad, M., Güden, H., Vizvári, B., & Barenji, R. (2018). A mathematical model and simulated annealing algorithm for solving the cyclic scheduling problem of a flexible robotic cell. *Advances in Mechanical Engineering*, 10(1). doi:10.1177/1687814017753912

53. Pandey, M. (2011). Operation Allocation in Flexible Manufacturing System Using Immune Algorithm. *Evolutionary Computing in Advanced Manufacturing*, 95-121. doi:10.1002/9781118161883.ch6

54. Ponnambalam, S., & Kiat, L. (2008). Solving Machine Loading Problem in Flexible Manufacturing Systems Using Particle Swarm Optimization. *International Journal of Industrial and Manufacturing Engineering*, 2(3), 242-247. doi:10.5281/zenodo.1075605

55. Prakash, A., Khilwani, N., Tiwari, M., & Cohen, Y. (2008). Modified immune algorithm for job selection and operation allocation problem in flexible manufacturing systems. *Advances in Engineering Software*, 39(3), 219-232. doi:10.1016/j.advengsoft.2007.01.024

56. Prakash, A., Tiwari, M., & Shankar, R. (2008). Optimal job sequence determination and operation machine allocation in flexible manufacturing systems: An approach using adaptive hierarchical ant colony algorithm. *Journal of Intelligent Manufacturing*, 19(2), 161-173. doi:10.1007/s10845-008-0071-y

57. Rahimifard, S., & Newman, S. (2000). Machine loading algorithms for the elimination of tardy jobs in flexible batch machining applications. *Journal of Materials Processing Technology*, 107(1-3), 450-458. doi:10.1016/S0924-0136(00)00719-6

58. Ram, B., Sarin, S., & Chen, C. (1990). A model and a solution approach for the machine loading and tool allocation problem in a flexible manufacturing system. *International Journal of Production Research*, 28(4), 637-645. doi:10.1080/00207549008942745

59. Ramesh, B., Mohan, V., & Reddy, V. (2013). Application of bat algorithm for combined economic load and emission dispatch. *International Journal of Electrical Engineering and Telecommunications*, 2(1), 1-9.

60. Reda, N., Hamdy, A., & Rashed, E. (2022). Multi-Objective Adapted Binary Bat for Test Suite Reduction. *Intelligent Automation & Soft Computing*, 31(2), 781-797. doi:10.32604/iasc.2022.019669

61. Reddy, K., & Reddy, N. (2017). Simultaneous Scheduling of Machines and AGVS in FMS by Using Symbiotic Organisms Search (SOS) Algorithm. *International Journal for Research in Applied Science & Engineering Technology*, 5(11), 1780-1790.

62. Roh, H.-K., & Kim, Y.-D. (1997). Due-date based loading and scheduling methods for a flexible manufacturing system with an automatic tool transporter. *International Journal of Production Research*, 35(11), 2989-3004. doi:10.1080/002075497194255

63. Sabilla, S., Sarno, R., & Effendi, Y. (2018). Optimizing time and cost using goal programming and FMS scheduling. *International Conference on Information and Communications Technology*, ICOIACT, 244-249. doi:10.1109/ICOIACT.2018.8350727

64. Santuka, R., Mahapatra, S., Dhal, P., & Mishra, A. (2015). An improved particle swarm optimization approach for solving machine loading problem in flexible manufacturing system. *Journal of Advanced Manufacturing Systems*, 14(3), 167-187. doi:10.1142/S0219686715500110

65. Sarin, S., & Chen, C. (1987). The machine loading and tool allocation problem in a flexible manufacturing system. *International Journal of Production Research*, 25(7), 1081-1094. doi:10.1080/00207548708919897

66. Sawik, T. (1998). A lexicographic approach to bi-objective loading of a flexible assembly system. *European Journal of Operational Research*, 107(3), 1801-1824. doi:10.1016/S0377-2217(97)00091-X

67. Sawik, T. (2004). Loading and scheduling of a flexible assembly system by mixed integer programming. *European Journal of Operational Research*, 154(1), 1-19. doi:10.1016/S0377-2217(02)00795-6

68. Shanker, K., & A. Srinivasulu, A. (1989). Some solution methodologies for loading problems in a flexible manufacturing system. *International Journal of Production Research*, 27(6), 1019-1034. doi:10.1080/00207548908942605

69. Shanthikumar, J., & Stecke, K. (1986). Reducing work-in-process inventory in certain classes of flexible manufacturing systems. *European Journal of Operational Research*, 26(2), 266-271. doi:10.1016/0377-2217(86)90189-X

70. Shawker, K., & Jeffrey Tzen, Y. (1985). A loading and dispatching problem in a random flexible manufacturing system. *International Journal of Production Research*, 23(3), 579-595. doi:10.1080/00207548508904730

71. Singh, R., Phogat, S., & Kanwarpal. (2017). Intelligent Approach for Balancing of Workload on Machines for Loading of Machines in Manufacturing. *International Journal of Theoretical and Applied Mechanics*, 12(1), 125-134.

72. Singh, R., Singh, R., & Khan, B. (2016). Meta-hierarchical-heuristic-mathematical- model of loading problems in flexible manufacturing system for development of an intelligent approach. *International Journal of Industrial Engineering Computations*, 7(2), 177-190. doi:10.5267/j.ijiec.2015.11.003

73. Šipoš, M., Klaić, Z., Nyarko, E., & Fekete, K. (2021). Determining the optimal location and number of voltage dip monitoring devices using the binary bat algorithm. *Energies*, 14(1). doi:10.3390/en14010255

74. Sreenivasulu, A., Venkatachalapathi, N., & Prasanthi, G. (2017). Simulation Modeling and Analysis for improvement of performance measures in a FMS. *IOSR Journal of Mechanical and Civil Engineering*, 14(2), 45-51. doi:10.9790/1684-1402034551

75. Stecke, K. (1983). Formulation and solution of non-linear integer production planning problems for flexible manufacturing systems. *Management Science*, 29, 273-288.

76. Stecke, K. (1986). A hierarchical approach to solving machine grouping and loading problems of flexible manufacturing systems. *European Journal of Operational Research*, 24(3), 369-378. doi:10.1016/0377-2217(86)90030-5

77. Stecke, K., & Morin, T. (1985). The optimality of balancing workloads in certain types of flexible manufacturing systems. *European Journal of Operational Research*, 20(1), 68-82. doi:10.1016/0377-2217(85)90285-1

78. Stecke, K., & Talbot, B. (1995). Heuristics for loading flexible manufacturing systems. *Manufacturing Research and Technology*, 23, 171-182. doi:10.1016/S1572-4417(06)80010-0

79. Stecke, K., & Talbot, F. (1983). Heuristic loading algorithms for Flexible Manufacturing System. The University of Michigan.

80. Swarnkar, R., & Tiwari, M. (2004). Modeling machine loading problem of FMSs and its solution methodology using a hybrid tabu search and simulated annealing-based heuristic approach. *Robotics and Computer-Integrated Manufacturing*, 20(3), 199-209. doi:10.1016/j.rcim.2003.09.001

81. Tiwari, M., & Vidyarthi, N. (2000). Solving machine loading problems in a flexible manufacturing system using a genetic algorithm based heuristic approach. *International Journal of Production Research*, 38(14), 3357-3384. doi:10.1080/002075400418298

82. Tiwari, M., Hazarika, B., Vidyarthi, N., Jaggi, P., & Mukhopadhyay, S. (1997). A heuristic solution approach to the machine loading problem of an FMS and its Petri net model. *International Journal of Production Research*, 35(8), 2269-2284. doi:10.1080/002075497194840

83. Tiwari, M., Kumar, S., Kumar, S., Prakash, & Shankar, R. (2006). Solving part-type selection and operation allocation problems in an FMS: An approach using constraints-based fast simulated annealing algorithm. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 36(6), 1170-1184. doi:10.1109/TSMCA.2006.878979

84. Tiwari, M., Saha, J., & Mukhopadhyay, S. (2007). Heuristic solution approaches for combined-job sequencing and machine loading problem in flexible manufacturing systems. *International Journal of Advanced Manufacturing Technology*, 31(7-8), 716-730. doi:10.1007/s00170-005-0259-7

85. Tizhoosh, H. (2005). Opposition-based learning: a new scheme for machine intelligence. *Proceedings of International Conference on Computational Intelligence for Modeling Control and Automation*, (p. 695–701).

86. Tsai, P.-W., Pan, J.-S., Liao, B.-Y., Tsai, M.-J., & Istanda, V. (2012). Bat Algorithm Inspired Algorithm for Solving Numerical Optimization Problems. *Applied Mechanics and Materials*, 148-149, 134-137. doi:10.4028/www.scientific.net/AMM.148-149.134

87. Turkcan, A., Akturk, M., & Storer, R. (2007). Due date and cost-based FMS loading, scheduling and tool management. *International Journal of Production Research*, 45(5), 1183-1213. doi:10.1080/00207540600559955

88. Tyagi, V., & Jain, A. (2008). Assessing the effectiveness of flexible process plans for loading and part type selection in FMS. *Advances in Production Engineering & Management*, 3, 27-44.

89. Umar, S.U. and Rashid, T.A. (2021). Critical analysis: bat algorithm-based investigation and application on several domains, *World Journal of Engineering*, 18(4), 606-620. doi:10.1108/WJE-10-2020-0495

90. Ventresca, M., & Tizhoosh, H. (2008). A diversity maintaining population-based incremental learning algorithm. *Information Sciences*, 178(21), 4038-4056.

91. Ventura, J., Chen, F., & Leonard, M. (1988). Loading tools to machines in flexible manufacturing systems. *Computers and Industrial Engineering*, 15(1-4), 223-230. doi:10.1016/0360-8352(88)90090-3

92. Verma, A., Shukla, N., Tiwari, M., & Shankar, R. (2007). Solving Machine Loading Problem of FMS: An Artificial Intelligence (AI) Based Random Search Optimization Approach. *Handbook of Computational Intelligence in Manufacturing and Production Management*. doi:10.4018/978-1-59904-582-5

93. Vidyarthi, N., & Tiwari, M. (2001). Machine loading problem of FMS: A fuzzy-based heuristic approach. *International Journal of Production Research*, 39(5), 953-979. doi:10.1080/00207540010010244

94. Wikarek, J., Sitek, P., & Nielsen, P. (2019). Model of decision support for the configuration of manufacturing system. *IFAC-PapersOnLine*, 52(13), 826-831. doi:10.1016/j.ifacol.2019.11.232

95. Xie, X., Qin, X., Zhou, Q., Zhou, Y., Zhang, T., Janicki, R., & Zhao, W. (2019). A novel test-cost-sensitive attribute reduction approach using the binary bat algorithm. *Knowledge-Based Systems*, 186. doi:10.1016/j.knosys.2019.104938

96. Yang, H., & Wu, Z. (2002). GA-based integrated approach to FMS part type selection and machine-loading problem. *International Journal of Production Research*, 40(16), 4093-4110. doi:10.1080/00207540210146972

97. Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. *In Nature Inspired Cooperative Strategies for Optimization* (p. 65-74).

98. Yang, X.-S. (2013). Bat algorithm: Literature review and applications. *International Journal of Bio-Inspired Computation*, 5(3), 141-149. doi:10.1504/IJBIC.2013.055093

99. Yogeswaran, M., Ponnambalam, S., & Tiwari, M. (2009). An efficient hybrid evolutionary heuristic using genetic algorithm and simulated annealing algorithm to solve machine loading problem in FMS. *International Journal of Production Research*, 47(9), 5421-5448. doi:10.1080/00207540801910429

100. Yuso, U., Khalid, M., & Khader, A. (2014). Artificial immune system for flexible manufacturing system machine loading problem. *ICIC Express Letters*, 8(3), 709-716.

101. Yusof, U., Budiarto, R., & Deris, S. (2011). Harmony Search Algorithm for Flexible Manufacturing System (FMS) Machine Loading Problem. *2011 3rd Conference on Data Mining and Optimization (DMO)*, 26-31. doi:10.1109/DMO.2011.5976500

102. Yusof, U., Budiarto, R., & Deris, S. (2012). Constraint-chromosome genetic algorithm for flexible manufacturing system machine-loading problem. *International Journal of Innovative Computing, Information and Control*, 8(3A), 1591-1609.

103. Yusof, U., Budiarto, R., & Deris, S. (2014). A hybrid of bio-inspired and musical-harmony approach for machine loading optimization in flexible manufacturing system. *International Journal of Innovative Computing, Information and Control*, 10(6), 2325-2344.

104. Zhang, X., Wang, S., Yi, L., Xue, H., Yang, S., & Xiong, X. (2018). An integrated ant colony optimization algorithm to solve job allocating and tool scheduling problem. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 232(1), 172-182. doi:10.1177/0954405416636038

Tables

Table 1. Overview of machine loading problem studies and contribution of the present study.

Authors	Approaches	Methodology	Loading objectives
(Stecke & Talbot, 1983)	Heuristics	-	Min part movements; Min system unbalance
(Stecke & Morin, 1985)	Other techniques	Single Serve Closed Queueing Network Model	Min system unbalance
(Shawker & Jeffrey Tzen, 1985)	Heuristics	-	Min system unbalance; Min number of late jobs
(Berrada & Stecke, 1986)	Mathematical	Branch and bound approach	Min system unbalance
(Stecke, 1986)	Other techniques	Hierarchical approach	Max throughput
(Shanthikumar & Stecke, 1986)	Other techniques	Dynamic approach	Min system unbalance
(Sarin & Chen, 1987)	Mathematical	Mathematical model	Min total cost
(Ventura, Chen, & Leonard, 1988)	Heuristics	-	Min makespan
(Shanker & A. Srinivasulu, 1989)	Mathematical	Branch and backtrack	Max throughput; Min system unbalance
(Bretthauer & Venkataramanan, 1990)	Mathematical	Integer linear programming	Max weighted sums of operations
(Co, Biermann, & Chen, 1990)	Mathematical	Mixed integer linear programming	Min system unbalance
(Liang & Dutt, 1990)	Mathematical	Mixed integer linear programming	Min production cost
(Ram, Sarin, & Chen, 1990)	Heuristics	Fast heuristic algorithm	Max throughput
(Mukhopadhyay, Midha, & Krishna, 1992)	Heuristics	-	Max throughput; Min system unbalance
(Kato, Oba, & Hashimoto, 1993)	Heuristics	-	Min tools required; Max machine utilization rate
(Kim, 1993)	Other techniques	Due-Date Based Loading method	Max throughput
(Liang, 1993)	Mathematical	Non-linear programming	Max throughput
(Liang, 1994)	Mathematical	Non-linear programming	Max throughput; Min production cost
(Kim & Yano, 1994)	Mathematical	New branch and bound algorithm	Max throughput
(Stecke & Talbot, 1995)	Heuristics	-	Min handling parts; Min system unbalance
(Hsu & De Matta, 1997)	Mathematical	Mixed integer linear programming and Lagrangian-based	Min production cost
(Kim & Yano, 1997)	Other techniques	Queueing Network Model	Max Throughput; Min system unbalance; Min Makespan
(Roh & Kim, 1997)	Other techniques	Due-Date Based Loading method	Min total tardiness
(Tiwari, Hazarika, Vidyarthi, Jaggi, & Mukhopadhyay, 1997)	Heuristics	-	Max throughput; Min system unbalance
(Lee & Kim, 1998)	Other techniques	Iterative procedure	Min earliness; Min tardiness
(Mukhopadhyay, Singh, & Srivastava, 1998)	Meta-heuristics	Simulated Annealing	Min system unbalance
(Nayak & Acharya, 1998)	Heuristics	Mathematical programming and heuristics	Max throughput; Max routing flexibility
(Sawik, 1998)	Mathematical	Integer linear programming and lexicographic technique	Min system unbalance; Min transfer time between machines
(Guerrero, Lozano, Koltai, & Larrañeta, 1999)	Mathematical	Mixed integer linear programming	Min system unbalance
(Gupta, Luong, & Nguyen, 1999)	Other techniques	Dispatching Approach	Min makespam; Min average flow time; Min tardiness
(Kumar & Shanker, 2000)	Meta-heuristics	Genetic Algorithm	Max throughput; Max machine utilization rate
(Lee & Kim, 2000)	Heuristics	-	Min of maximum workload
(Rahimifard & Newman, 2000)	Other techniques	Combined Machine Loading Algorithms	Min production time; Min production cost

(Tiwari & Vidyarthi, 2000)	Meta-heuristics	Genetic Algorithm	Max throughput; Min system unbalance
(Kumar & Shanker, 2001)	Mathematical	Mixed integer programming	Min system unbalance
(Vidyarthi & Tiwari, 2001)	Heuristics	Fuzzy Based Heuristic approach	Max throughput; Min system unbalance
(Yang & Wu, 2002)	Meta-heuristics	Genetic Algorithm-based integrated approach	Min system unbalance; Min movements of pieces
(Gamila & Motavalli, 2003)	Mathematical	Mixed integer programming	Min production time
(Kumar, Singh, & Tiwari, 2004)	Other techniques	Fuzzy Based Algorithm	Max throughput; Min system unbalance
(Sawik, 2004)	Mathematical	Mixed integer programming	Min production time
(Swarnkar & Tiwari, 2004)	Meta-heuristics	Simulated Annealing and Tabu Search	Max throughput; Min system unbalance
(Kumar, Prakash, Tiwari, Shankar, & Baveja, 2006)	Meta-heuristics	Constraint based Genetic Algorithm	Min system unbalance; Max filling tool warehouse
(Nagarjuna, Mahesh, & Rajagopal, 2006)	Heuristics	Multi-stage	Min system unbalance
(Goswami & Tiwari, 2006)	Heuristics	-	Min system unbalance
(Tiwari, Kumar, Kumar, Prakash, & Shankar, 2006)	Meta-heuristics	Constraint based Fast Simulated Annealing	Max throughput; Min system unbalance
(Biswas & Mahapatra, 2007)	Meta-heuristics	Particle Swarm Optimization	Min system unbalance
(Verma, Shukla, Tiwari, & Shankar, 2007)	Meta-heuristics	Ant Colony Optimization	Max throughput; Min system unbalance
(Tiwari, Saha, & Mukhopadhyay, 2007)	Heuristics	-	Max throughput; Min system unbalance
(Turkcan, Akturk, & Storer, 2007)	Meta-heuristics	Genetic Algorithm	Min production cost; Min production delays
(Prakash, Khilwani, Tiwari, & Cohen, 2008)	Meta-heuristics	Modified Immune Algorithm	Max throughput; Min system unbalance
(Prakash, Tiwari, & Shankar, 2008)	Meta-heuristics	Adaptive Hierarchical Ant Colony Optimization	Max throughput; Min system unbalance
(Biswas & Mahapatra, 2008)	Meta-heuristics	Modified Particle Swarm Optimization	Min system unbalance
(Ponnambalam & Kiat, 2008)	Meta-heuristics	Particle Swarm Optimization	Max throughput; Min system unbalance
(Tyagi & Jain, 2008)	Meta-heuristics	Genetic Algorithm	Min system unbalance
(Yogeswaran, Ponnambalam, & Tiwari, 2009)	Meta-heuristics	Genetic Algorithm and Simulated Annealing	Max throughput; Min system unbalance
(Mandal, Pandey, & Tiwari, 2010)	Meta-heuristics	Genetic Algorithm and Simulated Annealing	Max throughput; Min system unbalance
(Dhal, Mahapatra, Datta, & Mishra, 2010)	Meta-heuristics	Modified Artificial Immune System	Max throughput; Min system unbalance
(Mgwatu, 2011)	Mathematical	Linear mathematical programming	Max throughput; Min makespan
(Pandey, 2011)	Meta-heuristics	Immune Algorithm	Max throughput; Min system unbalance
(Abazari, Solimanpur, & Sattari, 2012)	Mathematical	Linear mathematical programming	Min system unbalance
(Kumar, Murthy, & Chandrashekara, 2012)	Meta-heuristics	Genetic Algorithm and Particle Swarm Optimization	Max throughput; Min system unbalance
(Basnet, 2012)	Meta-heuristics	Hybrid Genetic Algorithm	Min system unbalance
(Arikan & Erol, 2012)	Meta-heuristics	Simulated Annealing and Tabu Search	Min system unbalance
(Yusof, Budiarto, & Deris, 2012)	Meta-heuristics	Constraint Chromosome Genetic Algorithm	Max throughput; Min system unbalance
(Koşucuoğlu & Bilge, 2012)	Meta-heuristics	Genetic Algorithm based mathematical programming	Min total handling distance
(Mahmudy, Marian, & Luong, 2013)	Meta-heuristics	Real Coded Genetic Algorithm and Variable Neighborhood Search	Max throughput; Min system unbalance
(Yusof, Budiarto, & Deris, 2014)	Meta-heuristics	Hybrid Bio- inspired and Musical Harmony Approach	Max throughput; Min system unbalance
(Yuso, Khalid, & Khader, 2014)	Meta-heuristics	Artificial immune system	Max throughput; Min system unbalance
(Santuka, Mahapatra, Dhal, & Mishra, 2015)	Meta-heuristics	Modified Particle Swarm Optimization	Max throughput; Min system unbalance

(Ginoria, Samuel, & Srinivasan, 2015)	Meta-heuristics	Genetic Algorithm	Max throughput; Min system unbalance
(Mahmudy, 2015)	Meta-heuristics	Variable Neighborhood Search	Max throughput; Min system unbalance
(Singh, Singh, & Khan, 2016)	Heuristics	Meta-hierarchical-heuristic-mathematical model	Min system unbalance
(Bottani, Centobelli, Cerchione, Del Gaudio, & Murino, 2017)	Meta-heuristics	Modified Discrete Firefly Algorithm	Max throughput; Min system unbalance
(Reddy & Reddy, 2017)	Meta-heuristics	Symbiotic Organisms Search	Min makespan
(Singh, Phogat, & Kanwarpal, 2017)	Heuristics	Intelligent	Min system unbalance
(Sreenivasulu, Venkatachalapathi, & Prasanthi, 2017)	Other techniques	Simulation Modeling and Analysis	Min makespan
(Chawla, Chanda, & Angra, 2018)	Meta-heuristics	Modified Memetic Particle Swarm Optimization Algorithm	Min total handling distance; Min waiting times
(Nejad, Güden, Vizvári, & Barenji, 2018)	Mathematical	Mathematical model	Optimisation of scheduling strategies
(Sabilla, Sarno, & Effendi, 2018)	Other techniques	Goal Programming	Min production time; Min production cost
(Zhang, et al., 2018)	Meta-heuristics	Max-Min Ant Colony Optimization	Min makespan
(Azizmohammadi & Mahmoudabadi, 2019)	Meta-heuristics	Simulated Annealing based Multi Objective Algorithm	Tool allocation and selection
(Gaur, Indu, & Chawla, 2019)	Meta-heuristics	Genetic Algorithm	Min system unbalance
(Wikarek, Sitek, & Nielsen, 2019)	Mathematical	Mixed integer linear programming	Integration of several sub-problems; Reduction of complexity
(Huka, Rindler, & Gronalt, 2021)	Heuristics	-	Min tardiness
Present study	*Meta-heuristics*	*Modified Binary Bat algorithm*	*Min system unbalance*

Table 2. Nomenclature used throughout the paper.

Symbol	Description	Measurement unit
Indexes		
Subscripts		
j: $\mathbf{1 \le j \le J}$	Jobs	-
m: $\mathbf{1 \le m \le M}$	Machines	-
o: $\mathbf{1 \le o \le O}$	Operations	-
i: $\mathbf{1 \le i \le N}$	Bat population	
Superscripts		
iter: $\mathbf{1 \le iter \le max\ iter}$	Iteration	
Decision variables		
x_j	-	-
$x_{j,m}$	-	-
$x_{o,j,m}$	-	-
Parameters		
b_j	Batch size of job *j* (and consequently of operation *o*)	-
H	Length of planning horizon	[Days]
t_m	Available time on machine *m*	[Min]
$t_{o,j}$	Processing time needed to process operation *o* of job *j*	[Min]
$t^r_{o,j,m}$	Time remaining on machine *m* after allocation of operation *o* of job *j*	[Min]
T_m	Total processing times of machine *m*	[Min]
T_j	Total processing times for each job *j*	[Min]
μ_T	Mean of the processing times across the machines	[Min]
σ^2_T	Variance of the processing times across the machines (system unbalance)	[Min^2]
U_m	Set of tools needed for machine *m*	-
$U_{o,j,m}$	Tools required by machine *m* to process operation *o* of job *j*	-
$U^r_{oj,m}$	Tool slots remaining on machine *m* after allocation of operation *o* of job *j*	-
A^{iter}	Average loudness of the emitted sound	-
A_o	Initial intensity index	-
A_i	Intensity of the bat *i*	-

r	Emission speed index	-
r_i	Emission speed index of bat *i*	-
r_o	Initial velocity index	-
f_i	Ultrasound frequency of the bat *i*	-
$x_i, x_i^{iter}, x_i^{iter+1}$	Location of bat *i*	-
$v_i, v_i^{iter}, v_i^{iter+1}$	Velocity of bat *i*	-
F_{min}; F_{max}	Minimum and maximum ultrasound frequency	-
F_o	Initial ultrasound frequency	-
V_o	Initial bat speed	-
β	Random parameter ∈ [0,1]	-
x_{best}^{iter}	Best solution found by the algorithm at iteration $iter$	-
x_{best}	Global best solution	
ε	Randomly value ∈ $[-1,1]$	-
α	Algorithmic parameters (0<α<1)	-
φ	Algorithmic parameters (φ >0)	-
max iter	Maximum number of iterations	-
x_{new}	New solution	-
x_{old}	Old solution	-
P_i	Population of bat *i*	-
Pop_{init}	Population used to initialise the algorithm	-
$f_{o,i}^{iter}$	Ultrasound frequency of bat *i* to process operation *o* at iteration *iter*	-
$v_{o,i}^{iter+1}$	Velocity of bat *i* to process operation *o* at iteration *iter+1*	-
$m_{o,i}^{iter}$	Matrix values to iteration *iter*	-

Table 3. Example of initial population.

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
i_1	$m_{1,1}$	$m_{2,1}$	$m_{3,1}$	$m_{4,1}$	$m_{5,1}$	$m_{6,1}$	$m_{7,1}$	$m_{8,1}$
i_2	$m_{1,2}$	$m_{2,2}$	$m_{3,2}$	$m_{4,2}$	$m_{5,2}$	$m_{6,2}$	$m_{7,2}$	$m_{8,2}$
i_3	$m_{1,3}$	$m_{2,3}$	$m_{3,3}$	$m_{4,3}$	$m_{5,3}$	$m_{6,3}$	$m_{7,3}$	$m_{8,3}$
i_4	$m_{1,4}$	$m_{2,4}$	$m_{3,4}$	$m_{4,4}$	$m_{5,4}$	$m_{6,4}$	$m_{7,4}$	$m_{8,4}$
i_5	$m_{1,5}$	$m_{2,5}$	$m_{3,5}$	$m_{4,5}$	$m_{5,5}$	$m_{6,5}$	$m_{7,5}$	$m_{8,5}$
i_6	$m_{1,6}$	$m_{2,6}$	$m_{3,6}$	$m_{4,6}$	$m_{5,6}$	$m_{6,6}$	$m_{7,6}$	$m_{8,6}$
i_7	$m_{1,7}$	$m_{2,7}$	$m_{3,7}$	$m_{4,7}$	$m_{5,7}$	$m_{6,7}$	$m_{7,7}$	$m_{8,7}$
i_8	$m_{1,8}$	$m_{2,8}$	$m_{3,8}$	$m_{4,8}$	$m_{5,8}$	$m_{6,8}$	$m_{7,8}$	$m_{8,8}$
i_9	$m_{1,9}$	$m_{2,9}$	$m_{3,9}$	$m_{4,9}$	$m_{5,9}$	$m_{6,9}$	$m_{7,9}$	$m_{8,9}$
i_{10}	$m_{1,10}$	$m_{2,10}$	$m_{3,10}$	$m_{4,10}$	$m_{5,10}$	$m_{6,10}$	$m_{7,10}$	$m_{8,10}$
i_{11}	$m_{1,11}$	$m_{2,11}$	$m_{3,11}$	$m_{4,11}$	$m_{5,11}$	$m_{6,11}$	$m_{7,11}$	$m_{8,11}$
i_{12}	$m_{1,12}$	$m_{2,12}$	$m_{3,12}$	$m_{4,12}$	$m_{5,12}$	$m_{6,12}$	$m_{7,12}$	$m_{8,12}$
i_{13}	$m_{1,13}$	$m_{2,13}$	$m_{3,13}$	$m_{4,13}$	$m_{5,13}$	$m_{6,13}$	$m_{7,13}$	$m_{8,13}$
i_{14}	$m_{1,14}$	$m_{2,14}$	$m_{3,14}$	$m_{4,14}$	$m_{5,14}$	$m_{6,14}$	$m_{7,14}$	$m_{8,14}$
i_{15}	$m_{1,15}$	$m_{2,15}$	$m_{3,15}$	$m_{4,15}$	$m_{5,15}$	$m_{6,15}$	$m_{7,15}$	$m_{8,15}$

Table 4. Input data of the case study.

Parameter	Numerical value	Measurement Unit
j	756	-
m	19	-
o	942	-
U_m	20	-
H	144	Days

Table 5. Results as a function of the total processing time of the machines ($m = 1, \dots 10$) and relating variance (system unbalance).

AS IS scenario				TO BE scenario			
m_1	88,120[min]	m_{11}	26,740 [min]	m_1	65,381 [min]	m_{11}	65,285 [min]

m_2	40,470 [min]	m_{12}	170,000 [min]	m_2	65,459 [min]	m_{12}	65,431 [min]
m_3	23,000 [min]	m_{13}	17,030[min]	m_3	65,218 [min]	m_{13}	65,219 [min]
m_4	116,040 [min]	m_{14}	115,410 [min]	m_4	65,308 [min]	m_{14}	64,952 [min]
m_5	36,070 [min]	m_{15}	37,760 [min]	m_5	65,200 [min]	m_{15}	79,200 [min]
m_6	31,490 [min]	m_{16}	41,910 [min]	m_6	65,290 [min]	m_{16}	65,229 [min]
m_7	39,250 [min]	m_{17}	15,600 [min]	m_7	65,306 [min]	m_{17}	79,200 [min]
m_8	170,080 [min]	m_{18}	108,640 [min]	m_8	65,231 [min]	m_{18}	65,396 [min]
m_9	150,000 [min]	m_{19}	12,270 [min]	m_9	65,229 [min]	m_{19}	65,291 [min]
m_{10}	28,340 [min]			m_{10}	65,386 [min]		
	$\sigma_T^2 = 2.97 * 10^9$				$\sigma_T^2 = 1.93 * 10^7$		

Table Caption

Table 2. Overview of machine loading problem studies and contribution of the present study.

Table 2. Nomenclature used throughout the paper.

Table 3. Example of initial population.

Table 4. Input data.

Table 5. Results as a function of the total processing times of the machines ($m=1,\ldots,10$) and relating variance (system unbalance).

Figures

Bat algorithm

Step 1. Initialize the bat population

Step 2. Define the pulse frequency

Step 3. Initialize the pulse rate r_0 *and the loudness* A_0

While $(iter < max\ iter)$

Step 4. Generate new solution by adjusting the frequency and updating velocities and locations/solutions [Eqs.7-9]

If $(rand > r_i)$

Step 5. Select a solution among the best solutions

Step 6. Generate a local solution around the selected best solution x_{best}^{iter}

end if

Step 7. Generate a new solution by flying randomly [Eq.10]

If $(\mathrm{rand} < \mathrm{A_i}\ \&\ \mathrm{f_{x_i}} < \mathrm{f_x})$

Step 8. Accept the new solution

Step 9. Increase r_i *and reduce* $\mathrm{A_i}$

end if

Step 10. Rank the bat and find the current global best solution x_{best}

end while

Step 11. Postprocess results and visualization

Figure 1. Pseudo-code of the bat algorithm (adapted from Yang X.-S., 2010).

Binary Bat algorithm

Step 1. Initialize the bat population

Step 2. Define pulse frequency

Step 3. Initialize pulse rate r *and the loudness*A_0

While $(iter < max\ iter)$

Step 4. Adjust frequency and update velocities and positions/solutions [Eqs.13-14]

If *(rand>r_i)*

Step 5. Select a solution among the best solutions randomly; change some of the dimensions of position vector with some of the dimensions of the selected best solution

end if

Step 6. Generate a new solution by flying randomly

If $(\text{rand} < A_i \ \& \ f_{x_i} < f_{x^{iter+1}})$

Step 7. Accept the new solutions

Step 8. Increase r_i and reduce A_i

end if

Step 9. Rank the bats and find the current best solution x_{best}

end while

Figure 2. Pseudo-code of the binary bat algorithm (adapted from Mirjalili, Mirjalili, & Yang, 2014).

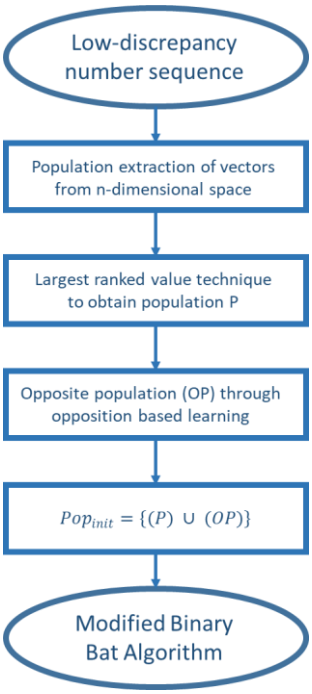

Figure 3. initialization process of the modified binary bat algorithm.

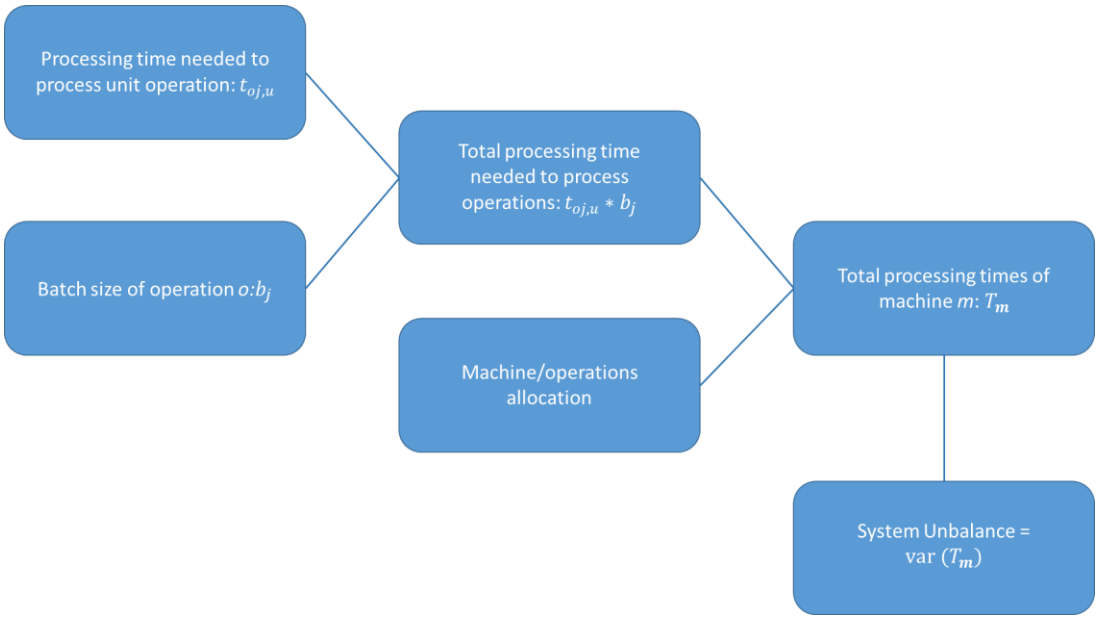

Figure 4. Flowchart of the fitness evaluation process.

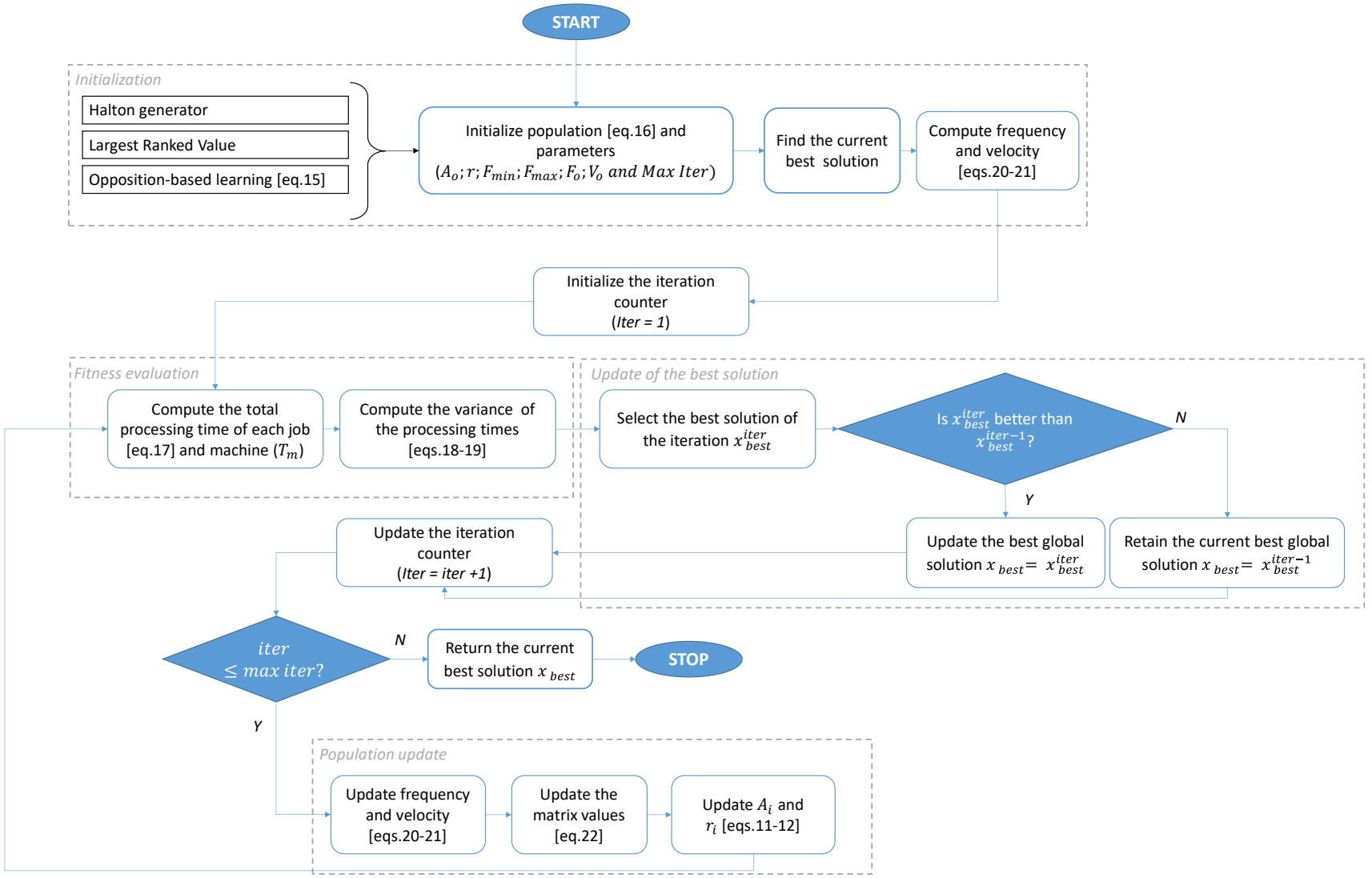

Figure 5. Flowchart of the proposed bat algorithm.

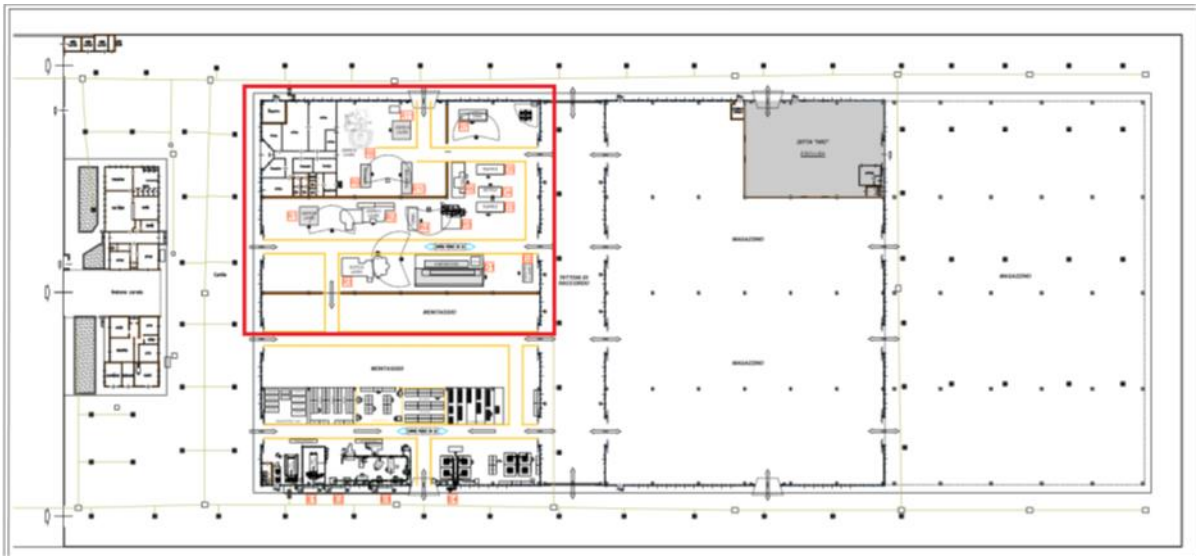

Figure 6. Layout of Company A. Red line denotes the machinery department.

Figure Caption

Figure 1. Pseudo-code of the bat algorithm (adapted from Yang X.-S., 2010).

Figure 2. Pseudo-code of the binary bat algorithm (adapted from Mirjalili, Mirjalili, & Yang, 2014).

Figure 3. initialization process of the modified binary bat algorithm.

Figure 4. Flowchart of the fitness evaluation process.

Figure 5. Flowchart of the proposed bat algorithm.

Figure 6. Layout of Company A. Red line denotes the machinery department.