Hybrid Differential Evolution Algorithm and Adaptive Large Neighborhood Search to Solve Parallel Machine Scheduling to Minimize Energy Consumption in Consideration of Machine-Load Balance Problems

Rujapa Nanthapodej 1, Cheng-Hsiang Liu 2,*, Krisanarach Nitisiri 3© and Sirorat Pattanapairoj 3

Abstract: Environmental and economic considerations create a challenge for manufacturers. The main priorities for production planning in environmentally friendly manufacturing industries are reducing energy consumption and improving productivity by balancing machine load. This paper focuses on parallel machine scheduling to minimize energy consumption (PMS_ENER), which is an indicator of environmental sustainability when considering machine-load balance problems. A mathematical model was formulated to solve the proposed problem and tested using a set of problem groups. The findings indicated that the mathematical model could find an optimal solution within a limited calculation time for small problems. For medium and large problems, the mathematical model could also find the optimal solution within a limited calculation time, but worse than all metaheuristics. However, finding an optimal solution for a larger problem is time-consuming. Thus, a novel method, a hybrid differential evolution algorithm with adaptive large neighborhood search (HyDE-ALNS), is presented to solve large-scale PMS_ENER. The new mutation and recombination formula for the differential evolution (DE) algorithm proposed in this article obtained promising results. By using the HyDE-ALNS, we improved the solution quality by 0.22%, 7.21%, and 12.01% compared with a modified DE (MDE-3) for small, medium, and large problems respectively. In addition, five new removal methods were designed to implement in ALNS and achieve optimal solution quality.

Keywords: green scheduling; adaptive large neighborhood search; hybrid differential evolution algorithm; energy consumption; machine load balance; parallel machine scheduling

1. Introduction

Greenhouse gases trap heat and consequently cause global warming. The largest source of greenhouse gases is human activity, particularly from burning fossil fuels for generating electricity and heat, and for transportation. Especially in the industrial sector, there is a release of waste gases such as CO₂ into the atmosphere, which causes air pollution while consuming a large amount of energy. The main sources of energy used in power facilities and for electricity generation are natural gas, coal, liquids, and renewables [1]. It is clear that the topic of energy consumption is of interest to various industrial sectors and a sustainable development agenda is urgent for production systems. Growing energy consumption has led to damaging environmental effects and has resulted in permissible limits on production companies [2,3]. Managers are under increasing pressure to reduce CO₂ emissions amid growing anxiety about climate change and global warming. This
burden will become much greater in the future, owing to rising energy prices arising both from likely carbon taxes and policies, as well as from rising energy demand in developing countries [4]. Nowadays, the industrial sector not only focuses on improving production efficiency, but also on energy-saving issues. In particular, the price of electricity is important for industrial competitiveness, as electricity usually accounts for a significant proportion of the total energy costs in manufacturing. Such environmental and economic considerations are a tremendous challenge for manufacturers, who must decrease energy consumption and save on energy costs while also becoming more environmentally friendly.

Since machines are used in various manufacturing processes, and the final products are normally processed by machines both directly and indirectly, machines are the main energy-consuming units in a factory. Reducing the energy consumption of machines would significantly improve manufacturing sustainability. However, most enterprises maintain modern and advanced machines alongside outdated ones. The modern ones usually have higher operating speeds, but consume more power. There are many potential energy reduction approaches for manufacturing plants, such as developing more energy-efficient machines and processes. In the case of production lines, energy efficiency is highly dependent on production planning and scheduling. It is easy to improve energy efficiency by job sequencing or machine scheduling, since they do not require machine modification, changing the plant layout, or making a large investment [1,2,4–7].

In real manufacturing environments, most workstations have multiple machines. Thus, production scheduling is particularly important in modern factories that require planning and sequencing jobs for machines, since mass production relies on a large number of machines working in parallel. Parallel machines can process several jobs simultaneously and parallel machine scheduling sequences jobs when similar types of machines are available. When a particular machine can process different jobs at different rates, the situation is considered to involve parallel machines. Attempting to minimize some objective functions of interest has led to various sequencing and/or processing restrictions such as due dates, start times, and machine capacities [8–18]. Thereby, solutions or algorithms constructed to schedule parallel machines are crucial, because when they are implemented in relation to parallel machine problems, they can generate sufficient solutions for production improvement [8,9].

Generally, the objective of parallel machine scheduling is to minimize the makespan and completion time to complete jobs as quickly as possible and to maximize machine efficiency. However, paying too much attention to either completion time or energy cost may cause an imbalanced workload for machines, leading to bottlenecks and untimely machine failures. For instance, attempting to minimize the completion time by assigning too many jobs to the fastest machine and leaving others idle is inefficient. Bottlenecks in production can be reduced by utilizing all machines equally as much as possible [10,11,15].

The scheduling problems can be classified as optimization problems involving production efficiency, expense, and performance. Many methods are used to solve the scheduling problems, e.g., exact algorithms (algorithms that always provide an optimal solution or mathematical model) [1,2,7,9,10]; heuristic algorithms [8] (partial solution search algorithms); metaheuristic algorithms (methods or heuristics designed to find or select a heuristic that may supply an adequately good solution) [1,2,4,5,7–10]; and simulation methods. However, considering the costs of energy and environmental damage, green scheduling models have gained more attention from researchers [1,7,14,17]. Considering the discussion above, many researchers and production companies should be aware of the importance of energy efficiency. The parallel machine scheduling problem as a concept involving energy consumption, including machine load balance, has rarely been studied in previous research. Therefore, this scheduling is not only a natural bi-objective optimization problem, but also an NP-hard problem (non-deterministic polynomial-time hardness problem) for which it is difficult to obtain an optimal solution [7,10,19]. The differential evolution (DE) algorithm is appropriate for solving this sort of problem because it can find the non-dominated solutions in a single run and is also successfully applied in many
fields of optimization problems [20–26]. Furthermore, metaheuristics based on local search methods such as the adaptive large neighborhood search (ALNS) have been successfully applied to solve many combinatorial optimization problems [13,27]. This has inspired us to develop a parallel machine scheduling model and propose a modified differential evolution (MDE) algorithm and a hybrid differential evolution algorithm with adaptive large neighborhood search (HyDE-ALNS) to handle this type of problem.

In this study, we considered a parallel machine scheduling problem with the energy consumption under machine load balance (PMS_ENER). The minimization of both makespan and total energy costs is an important performance indicator for companies in terms of production efficiency and environmental sustainability. The purpose of this work was to develop an MDE and a HyDE-ALNS to obtain the bi-objective near the Pareto front. Based on the features of the problem and the evolutionary algorithm, the proposed MDE and HyDE-ALNS apply a new encoded structure, which can convert discrete optimization problems into continuous optimization problems, and adopt MDE and HyDE-ALNS using three different strategies to enhance the exploitation ability.

The remainder of the paper is organized as follows. In Section 2, we review the literature related to parallel scheduling, considering energy consumption and machine workload balance, including applications of the DE and ALNS. In Section 3, we provide a formal definition of the problem and offer a mathematical model for the considered problem. The proposed MDE and HyDE-ALNS with multiple operators for handling the scheduling problem are described in Section 4. The experimental results of our proposed MDE and HyDE-ALNS are presented and analyzed in Section 5. In Section 6, conclusions and suggestions for several future research areas are provided.

2. Literature Review

Several studies focus on the parallel machine scheduling problem to increase production systems efficiently; different variants of this kind of problem have been investigated, e.g., job-splitting [18,22], sequence-dependent setup times [10,18,25], and load balancing [27–30]. In this study, we reviewed the literature related to our research from two directions: the parallel machine scheduling pertaining to workload balancing, and the parallel machine scheduling pertaining to energy consideration.

Parallel machine scheduling problems have been extensively examined in previous studies [31–37]. Most of the research has focused on minimizing the makespan, which is the maximum completion time. Taking the machine workload balancing problem into account, the ratio of the difference between maximum completion time on a given machine and individual completion time on a given machine is defined as the relative workload imbalance. The aim of machine load balancing in parallel machine scheduling is to minimize relative workload imbalance [28]. Workload balancing has been studied in many environments. For example, in semiconductor manufacturing, a common production bottleneck is the photolithography process. Enforcing a strict load-balancing constraint, referring to whole machines sharing the same workload, has been shown to improve productivity. An automated stepper load balance allocation system was developed by Miwa et al. [29] to improve productivity in the photolithography process and maintain a balanced load distribution of tool constraint layers. The findings showed that the developed system outperformed the conventional manual allocation method by reducing the deviation in total load based on the processing time. These studies demonstrated that workload balancing is relatively important in machine scheduling, since imbalance in workload sequencing caused machines to work in atypically, leading to unexpected completion times and energy consumption.

A growing number of corporations in the Industry 4.0 era are running parallel applications in the Cloud that lead to some issues such as slowness, imbalanced time load, and low resource utilization. Zhang et al. [30] proposed an optimized strategy of a particle genetic algorithm based on time load balancing to utilize resources and reduce resource wastage in the Cloud environment. The experimental results demonstrated that the proposed
algorithm outperformed the standard PSO-GA algorithm in term of search optimization, computing time, time load balance. In the parallel-machine field, Keskinturk et al. [10] presented the strategy of minimizing the average relative imbalance through sequence-dependent setup times; an ant colony optimization metaheuristic performed better than another compared method in solving the workload balance of the parallel machine scheduling problem with a sequence-dependent setup. Ouazene et al. [38] demonstrated the performance of a mixed-integer linear programming model and genetic algorithm for solving joint load balancing and total tardiness minimization in identical parallel machine scheduling. In the context of scheduling, machine load balancing problems on parallel machines were addressed previously. However, none of these studies have taken energy consumption into consideration in workload balancing [29–31].

Since environmental and energy sustainability are becoming critical issues, various energy concerns (i.e., energy consumption, energy efficiency, carbon footprint, and energy cost) are considered in production scheduling problems. There are also several studies on energy-efficient production systems. For instance, Mouzon et al. [39] developed operational methods such as dispatching rules and a mathematical model to minimize the total completion time and energy consumption of manufacturing equipment in production scheduling. Angel et al. [40] obtained feasible solutions within the bounded expected value by employing a randomized approximation algorithm for energy consumption scheduling on unrelated parallel machines with the average weighted completion time. Sobottka et al. [41] presented the development and evaluation of a digital method for multi-criteria optimization, considering traditional business goals and energy efficiency, for production scheduling in metal casting manufacturing. The developed method, featuring hybrid (discrete and continuous) simulation-based multi-objective optimization (a multistage hybrid heuristic and metaheuristic), was employed in a heat treatment process which required batch ordering and sequencing/scheduling on parallel machines and considering complex restrictions. The results indicated that the enumerative heuristic and specific GA outperformed the traditional GA.

CO₂ emissions are strongly correlated to energy consumption. Liu [42] studied scheduling problems related to the environment where the goal of minimizing CO₂ emissions was considered with total weighted tardiness (TWT). The experimental results showed that the proposed ε-archived genetic algorithm (ε-AGA) performed better than the nondominated sorting-based genetic algorithm II (NSGA-II) in solving two batch scheduling problems. In their subsequent article, Liu and Huang [4] demonstrated both the effectiveness of an adaptive multi-objective genetic algorithm (AMGA) in converging to the true Pareto-optimal set over the NSGA-II in consideration of economic-related criteria (total weighted tardiness) and environmental-related criterions (carbon footprint and peak power) of scheduling on a batch-processing machine with dynamic job arrivals. In relation to another low-carbon parallel machine scheduling problem, Pan et al. [43] presented the advantages of a novel imperialist-competitive algorithm (ICA) for minimizing total tardiness, total energy consumption, and CO₂ emissions.

In the context of energy cost, a number of previous works have attempted to minimize the energy cost during production processes when energy prices and electricity prices are allowed to vary with time. For instance, Fang et al. [1] presented a mixed-integer programming formulation for optimizing the operating schedule of a flow shop that considered both productivity and energy-related criteria in the simple case. The results showed that it was only feasible to obtain the optimal solution for small-size problems (6 jobs), even when using commercial optimization software tools. Zeng [17] investigated a uniform parallel machine scheduling problem under time-dependent or time-of-use electricity tariffs when the electricity price changed every hour within a day. A bi-objective mixed integer linear programming model was first formulated for this problem. Then, the proposed approach, an insertion algorithm, was employed to minimize the total electricity cost and the number of process machines. Computational results revealed that the proposed method could find high-quality solutions for large-size problems. Zhou et al. [9] proposed
a multi-objective differential evolution algorithm (MODDE) for solving the parallel batch processing machine scheduling problem in the case of dynamic job arrivals and a time-of-use pricing form. The aim was to simultaneously minimize both the makespan (an indicator of production efficiency) and the total electricity cost (a measure of environmental sustainability). The numerical experimental results illustrated that the proposed algorithm successfully obtained superior quality Pareto solutions.

Considering the abovementioned research, production managers can take the models and algorithms into a decision-making trade-off between production and energy efficiency, since the objective of this paper is to minimize energy consumption in consideration of machine load balance for parallel machine scheduling, which is considered as an optimization problem. Although a mathematical model as an exact method is able to provide the optimal solution, it requires more time to compute even small test instances, and is unable to complete finding the solutions in larger test instances. Therefore, the appropriated methods used to find sufficient solutions for this kind of problem in shorter computational times are metaheuristic algorithms, e.g., genetic algorithms (GA), Adaptive large neighborhood search (ALNS) and differential evolution (DE).

Differential evolution (DE) is a type of evolutionary algorithm in continuous search space that has been used for function optimization [20]. The DE algorithm has attracted significant attention since it was first introduced by Storn and Price [21]. There are a variety of industrial applications of the DE algorithm for solving optimization problems. For instance, the DE algorithm has been implemented to solve a number of production scheduling problems. Wang et al. [22] applied hybrid differential evolution (HDE) for scheduling problems with splitting jobs. A global search method with block mutation and block crossover was proposed. The results showed that the proposed HDE obtained a better solution than the traditional DE. The findings indicated that the proposed HDE algorithm was feasible and efficient. Zhou et al. [23] developed a hybrid DE algorithm into discrete DE algorithm (DDE) by proposing discrete DE operators to minimize the makespan in scheduling problems for uniform parallel machines with arbitrary job sizes, non-identical capacities, and different speeds. The DDE algorithm was proposed for solving large-scale problems. In this algorithm, individuals are represented as discrete job sequences and job permutation and crossover operators were designed based on this representation. The experimental results showed that the proposed DDE algorithm was the superior method for solving large-scale problems in terms of solution quality and robustness. Wu and Che [24] proposed a memetic differential evolution (MDE) algorithm for an energy-efficient bi-objective unrelated parallel machine scheduling problem in order to minimize makespan and total energy consumption. Efficient speed adjusting and a job-machine swap heuristic were proposed and integrated into the algorithm as local search approaches by an adaptive meta-Lamarckian learning strategy. The computational results demonstrated that the proposed MDE successfully obtained an efficient solution compared with other methods. Furthermore, a procedure to reduce the total energy cost of a schedule without compromising the makespan was presented by Li et al. [25]. They studied a parallel machine scheduling problem with different color families, sequence-dependent setup times, and machine eligibility restrictions in the dyeing process in a textile factory. Since the dyeing optimization problem is NP-hard, the HDE algorithm was proposed to solve real-world instances. In this proposed HDE algorithm, a special encoding and decoding structure was modeled to deal with the machine-eligibility constraints. Chaos theory was adopted to determine the parameters of the proposed DE algorithm. The experimental results confirmed that the proposed HDE outperformed the other DE in terms of solution quality and computation time. Kusoncum et al. [16] proposed a traditional DE with embedded heuristics to obtain near-optimal solutions for realistically sized machine scheduling, with capacitated machine restrictions and sequencing independent setup time included in the problem. The results showed that, during the simulation, the proposed method tended to find a new optimal solution, while the local search-based heuristics were trapped at some local optima and there was a lack of search intensification in the DE. The studies of
mentioned metaheuristic algorithms demonstrated that the hybrid and modified versions of the DE algorithm were successfully implemented in various optimization problems. Thus, this study investigates an additional metaheuristic to develop the DE performance by hybridization: the adaptive large neighborhood search.

Adaptive large neighborhood search (ALNS), a metaheuristic based on local search methods, has had significant attention and has been recently applied in various research areas. For example, Khamsing [44] presented a solution to the family tourism route problem by considering daily time windows. The study demonstrated that modified adaptive large neighborhood search (MALNS) method using the four destructions approach featured partial solution removal in order to generate new solutions; the four reconstruction approach was used for the insertion of the repairing solution was applied and successfully found the optimal solution for travel routing. In the context of production scheduling and assignment problems in a broiler farm case study, Praseeratasang [27] developed an algorithm using the ALNS method in order to find an appropriate solution by using a shorter time period, which enabled the farms to achieve maximum profit. In a subsequent article, Praseeratasang [45] also proposed the ALNS method to solve a specific production planning and labor assignment problem in a pig farm case study. The results revealed that the proposed method could increase profits compared with the traditional production plan. Later, the newly developed destruction and reconstruction methods of the ALNS metaheuristics were presented by Pitakaso [46]. The concepts of DE and ALNS were applied to solve a mechanical harvester assignment and routing problem with time windows in order to maximize the total working area under a sharing infield resource system. For an unrelated parallel machine scheduling problem with consideration of setup time (UPMPS-ST), Cota [47] demonstrated a successful adapted removal and insertion probability by applying an ALNS metaheuristic and learning automata (LA) to solve the UPMPS-ST problem. In a subsequent article by Cota [35], the main focus was on green machine scheduling, maximizing production, and machine efficiency. Multi-objective extensions of the ALNS metaheuristic were proposed with LA to improve the search process and solve the large-scale sample size efficiently. The results indicated that the decomposition strategy was not only profitable for evolutionary algorithms, but was also a capable method to extend ALNS for solving multi-objective problems. Considering how the abovementioned research is constructively responding to global climate change, but managers should consider how their decisions affect the environment.

In our review of previous studies, we found there were many types of problem objectives such as makespan, tardiness, load-balancing, and energy consumption; the constraints were starting time, due dates, and load capacity. Solving methods included mathematical models, heuristics, and metaheuristics that were studied in the literature. Therefore, we can state that the research on energy efficiency and workload balancing is truly crucial in terms of its theoretical and practical value. In this paper, we first attempt to develop a mathematical model to minimize the total energy consumption in consideration of the machine load balance for the parallel machine problem. Then, a hybrid of two metaheuristics, DE and ALNS, is presented as Ade1-3 and HyDE-ALNS-1-3 for improving the search efficiency. Five removal methods and one repair method are proposed to further improve the algorithm.

3. Mathematical Model Formulation

This paper considers the problem of energy-aware scheduling with the constraint of machine load balance (PMS_ENER) for parallel machines. This is in order to improve energy consumption and production efficiency, which are indicators of environmental sustainability [9]. Minimizing the total electricity cost is also considered, as the objective of this study includes the consideration of the completion times of machines with machine load balance constraints. In this context, a set of \(I\) jobs \(\{j_1, j_2, \ldots, j_I\}\) was scheduled on parallel \(M\) machines \(\{m_1, m_2, \ldots, M\}\). In this study, we assumed that this problem consisted entirely of deterministic parameters, and no interfering parameters during processing.
Each job $J_i$ has a deterministic processing time $p_{jm}$ on each machine. The jobs may be assigned to any machine. Due to the operating characteristics, each machine requires a different energy consumption $e_{jm}$. All jobs are available at time zero. The completion time of job $J_i$ is denoted by $C_{jm}$, which is the time when the processing of the job has been completed on machine $M$. After starting the process, no idle time can be inserted into the schedule, and there can be no preemption. In addition, only one job can be processed at a time on machine $M$.

Based on these assumptions, a mathematical model was presented to formulate this problem in order to minimize the energy consumption and machine load imbalance. In other words, we intended to minimize the difference between the workload of the bottleneck machine and the workload of the fastest machine.

**Mathematical Formulation**

Based on the characteristics of the PMS_ENER problem, a mathematical model is constructed in this section. The following are the details of indices, parameters, decision variables, objective functions, and constraints.

**Indices:**
- $i, j$ Number of jobs; $i = 1, 2, \ldots, I$ when $i$ or $j = 0$, 0 represents the dummy node, which is always produced first in each machine
- $n, m$ Number of machines; $m = 1, 2, \ldots, M$

**Parameters:**
- $I$ Total number of jobs
- $M$ Total number of machines
- $p_{jm}$ Processing time of job $j$ on machine $m$ (min)
- $e_{jm}$ Energy consumption for producing job $j$ on machine $m$ (THB); Thai baht; 1 THB: 0.032 USD
- $B$ Cost per hour of processing or all machines
- $U$ Allowed load difference between machines
- $A_{jm}$ Job-machine restriction; $A_{jm} = 1$ when job $j$ can produce on machine $m$

**Decision Variable:**
- $X_{ijm} \begin{cases} 1 & \text{when job } i \text{ is immediately produced before job } j \text{ in machine } m \\ 0 & \text{Otherwise} \end{cases}$
- $Y_{jm} \begin{cases} 1 & \text{when job } i \text{ is assigned to machine } m \\ 0 & \text{Otherwise} \end{cases}$
- $S_{jm}$ Start time of job $j$ on machine $m$
- $C_{jm}$ Completion time of job $j$ on machine $m$
- $D_{mn}$ Percentage of different completion time between machine $m$ and $n$
- $L_m$ Workload of machine $M$

**Objective function:**

$$\text{Min } z = \sum_{m=1}^{M} \sum_{j=1}^{I} e_{jm} Y_{jm} + \sum_{m=1}^{M} B \times \text{Max}_{j=1}^{I} C_{jm}$$ (1)

Subject to:

$$\sum_{j=1}^{I} X_{ijm} \leq 1 \ \forall \ m$$ (2)

$$\sum_{m=1}^{M} \sum_{i=1, i \neq j}^{I} X_{ijm} = 1 \ \forall \ j$$ (3)

$$\sum_{j=1, i \neq j}^{I} X_{ijm} \leq Y_{im} \ \forall \ i, \ m$$ (4)

$$\sum_{i=1, i \neq j}^{I} X_{ijm} = Y_{jm} \ \forall \ j, \ m$$ (5)
The objective function in Equation (1) is to minimize the energy consumed to produce all jobs and the production overhead cost per working hour. Equations (2) and (3) confirm that all jobs must be assigned, to at most, one machine and the starting of all assigned jobs must be with job 0 (a dummy job). Equation (4) is the relationship constraint of \( X_{ijm} \) and \( Y_{im} \). Equation (5) ensures that job \( i \) will be processed before job \( j \) in machine \( m \) only when job \( j \) is assigned to machine \( m \). Equation (6) ensures that a job can be assigned to at most one machine. Equation (7) calculates the starting time of job \( j \) in machine \( m \) and Equation (8) is the calculation of the completion time of job \( j \). Equations (9) and (10) define the starting and completion times of the dummy job. Equation (11) calculates the machine load, while Equations (12) and (13) ensure that the completion time of the last job on machine \( n \) will not differ from machine \( m \) by more than the allowed load difference between machines. Equation (14) is a job machine restriction.

4. The Proposed Methods

As mentioned above, when the size of the problems becomes larger and they become more complicated, they may not lend themselves to exact solution methods. In this paper, we present HyDE-ALNS, a hybrid of two metaheuristics—differential evolution (DE) algorithms and adaptive large neighborhood search (ALNS). HyDE-ALNS is introduced for improving search efficiency and also for solving the PMS ENER. HyDE-ALNS is composed of five steps: (1) generate a set of initial solutions, (2) perform a mutation process, (3) perform a recombination process, (4) perform ALNS, and (5) perform a selection process. Steps (2) to (5) were iteratively performed until the termination condition was met. The algorithm framework is shown in Figure 1 and the procedures can be explained stepwise as follows.
4.1. Generate an Initial Solution

In this article, indirect encoding was used to solve the PMS_ENER and a set of vectors was randomly generated. Real number coding was used to operate in HyDE-ALNS and five example vectors are shown in Table 1.

Table 1. An example of five vectors used in hybrid differential evolution algorithm with adaptive large neighborhood search (HyDE-ALNS).

| Position | Vector | Job's Vector | Machine's Vector |
|----------|--------|--------------|------------------|
|          |        | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | A  | B  | C  |
| 1        | 0.19   | 0.11| 0.74| 0.27| 0.51| 0.51| 0.79| 0.39| 0.29| 0.25| 0.75| 0.33| 0.26 |
| 2        | 0.62   | 0.03| 0.14| 0.02| 0.11| 0.71| 0.37| 0.01| 0.43| 0.88| 0.52| 0.39| 0.62 |
| 3        | 0.74   | 0.75| 0.77| 0.91| 0.84| 0.24| 0.90| 0.74| 0.21| 0.18| 0.93| 0.26| 0.06 |
| 4        | 0.50   | 0.20| 0.61| 0.76| 0.41| 0.31| 0.27| 0.03| 0.03| 0.50| 0.49| 0.12| 0.82 |
| 5        | 0.66   | 0.33| 0.36| 0.34| 0.67| 0.95| 0.05| 0.72| 0.47| 0.65| 0.00| 0.31| 0.07 |
The vectors, which are shown in Table 1, are composed of 13 positions. The first 10 positions are the elements of the vector that represent jobs 1 to 10. The last three positions represent three parallel machines: A, B, and C. The details of the decoding methods are as follows.

4.1.1. The Decoding Methods

The decoding method shown in this section includes four steps.

1. Apply the rank order value (ROV) separately between job and machine vectors. The ROV of jobs is called the sequence of jobs in position $i$ ($SJ_i$), and the sequence of machines in position $m$ is called $SM_m$.

2. Assign the first job in position 1 of $SJ_i$ to machine 1 of $SM_m$, assign the second job in position 2 of $SJ_i$ to machine in position 2 of $SM_m$, and so on ($i + 1$ to machine $m + 1$) until $m = M$. In the context of job machine restriction, job $j$ is assigned to machine $m$ only when job $j$ is allowed to be produced in machine $m$. If job $j$ is not allowed to be produced in machine $m$, job $j$ is assigned to the next machine in the line.

3. Continue assigning jobs in position $I + 1$ to the machine with the minimum total processing time among all $M$ machines until all jobs are assigned to a machine.

4. Calculate the completion time and energy consumption of the result in step 3.

4.1.2. An Example of the Decoding Method

An example scheduling of 10 jobs with three machines is provided in Table 2. $P_{jm}$ the processing time of job $i$ on machine $m$, and $E_{jm}$ is the energy consumption of machine $m$ for operating job $i$.

| Job | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-----|---|---|---|---|---|---|---|---|---|----|
| $P_{jm}$ | M/C | m |
| A | 20 | 29 | 17 | 21 | 27 | 21 | 16 | 24 | 19 | 21 |
| B | 30 | 23 | 20 | 21 | 19 | 20 | 27 | 16 | 17 | 22 |
| C | 19 | 20 | 30 | 21 | 24 | 18 | 30 | 29 | 22 | 28 |
| $E_{jm}$ | M/C | m |
| A | 43 | 14 | 35 | 41 | 17 | 10 | 31 | 12 | 45 | 19 |
| B | 20 | 41 | 38 | 49 | 17 | 36 | 34 | 10 | 13 | 40 |
| C | 10 | 39 | 33 | 25 | 11 | 17 | 33 | 37 | 42 | 28 |

Step 1. Apply ROV to the job vector and machine vector separately from the smallest value to the largest value; the result of ROV is shown in Table 3.

| Vector | Job-Machine | Job | Machine |
|--------|-------------|-----|---------|
| Vector before ROV | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A | B | C |
| Value in position | 0.19 | 0.11 | 0.74 | 0.27 | 0.51 | 0.51 | 0.79 | 0.39 | 0.29 | 0.25 | 0.75 | 0.33 | 0.26 |

| Vector | Job-Machine | Job | Machine |
|--------|-------------|-----|---------|
| Vector after ROV | 2 | 1 | 10 | 4 | 9 | 8 | 5 | 6 | 3 | 7 | C | B | A |
| Value in Position | 0.11 | 0.19 | 0.25 | 0.27 | 0.29 | 0.39 | 0.51 | 0.51 | 0.74 | 0.79 | 0.26 | 0.33 | 0.75 |

Step 2. Assign jobs 2, 1, and 10 to machines C, B, and A, respectively, then calculate the total processing time of each machine.

Step 3. Assign job 4 to machine C, since it has the lowest total processing time; afterwards, jobs 9, 8, 5, 6, 3, and 7 are assigned to machines A, B, A, C, B, and B, respectively, with the same conditions.
In this plan, the production sequence of jobs on machine C is 2-4-6, with 1-8-3-7 on machine B, and 10-9-5 on machine A. The energy consumption is 186 baht and the makespan is 93 min. Since the minimum labor cost is assumed to be 450 baht per h or 7.5 baht per min, the cost of production is 622.5 baht. Therefore, the total cost of this plan is THB 886.5. The result of this assignment is shown in Table 4.

**4.2. Perform a Mutation Process**

In the mutation process, the target vector is transformed into a mutant vector \(v_{ij,G}\). In this study, we modified the mutation formula from Qin et al. [26] to increase the exploration capability of the DE, as presented by Equations (15) and (16):

\[
v_{ij,G} = x_{r1j,G} + F(x_{r2j,G} - x_{r3j,G})
\]

(15)

\[
v_{ij,G} = \begin{cases} v_{ij,G} & \text{when } CR \leq \text{rand} \\ x_{ij,G} & \text{when } CR > \text{rand} \end{cases}
\]

(16)

where \(v_{ij,G}\) is the mutant vector \(i\) in position \(j\) and with iteration \(G\), \(x_{rj,G}\) is the current target vector, \(x_{r2j,G}\) and \(x_{r3j,G}\) are the other target vectors, and \(x_{r4j,G}\) are the new randomly generated data. \(f(X_{ijk})\) is the objective function of the target vector. \(F\) is the scaling factor, which is a self-adaptive parameter and is set to 0.8 [47].

**4.3. Perform a Recombination Process**

In this process, the trial vector was generated. \(U_{ij,G}\) is the trial vector of vector \(i\) in position \(j\) with iteration \(G\) and \(\text{rand}\) is a random number between 0 and 1. The transforming of a mutant vector to a trial vector involves Equations (17) and (18), where \(CR\) is the crossover rate and is set to 0.6.

\[
U_{ij,G} = \begin{cases} \text{when } CR \leq \text{rand} \\ \text{when } CR > \text{rand} \end{cases}
\]

(17)

\[
U_{ij,G} = \begin{cases} v_{ij,G} & \text{when } CR \leq \text{rand} \\ x_{ij,G} & \text{when } CR > \text{rand} \end{cases}
\]

(18)

**4.4. Apply ALNS to Improve the Solution**

After performing the recombination process, the decoding method is applied to the trial vector and then ALNS is employed to improve the current solution quality (initial solution obtained from the trial vector’s solution). The ALNS used in this article includes three steps: (1) perform the destruction procedure, (2) perform the reconstruction procedure, and (3) update the required information.

**4.4.1. Destruction Procedures**

In this section, the destruction procedure is used to disassemble the solution obtained from the trial vector so it becomes incomplete, which causes the solution to move to other search areas and, therefore, a new solution is obtained. In this paper, there are five steps in the destruction method: (1) N-random removal, (2) N-shortest greedy removal, (3) N-longest greedy removal, (4) N-two-tails removal, and (5) N-jobs-string removal. In each method, there are principles and steps of destruction, as follows.
N-Random Removal

The principle of N-random removal destruction is shown in Algorithm 1.

Algorithm 1 N-random removal

1. Randomly select a value of N that lies between 2 and I (the number of jobs)
2. B = I; I = [Sequence of all jobs]
3. L = {}
4. While |L| < N do
   4.1 Random i from B
   4.2 I ← I ∩ {i}
   4.3 L ← L ∪ {i}
5. Return L

From Algorithm 1, we create list I first, which is the sequence of all jobs. For example, the last solution shown in Table 4, and N with a value of 4. Thus, the randomly selected four jobs are removed from list I, and we can obtain L {1, 3, 8, 4}. This set of L needs to have one repair method performed afterwards.

N-Shortest Greedy Removal

This destruction method is executed to remove the value that looks ideal at the moment (greedy strategy) from the set of solutions. Algorithm 2 shows the procedure of N-shortest greedy removal.

Algorithm 2 N-shortest greedy removal

1. Randomly select a value of N that lies between 2 and I (the number of jobs)
2. B = I; I = [Sequence of all jobs]
3. L = {}
4. While |L| < N do
   4.1 Calculate the attractiveness of selection to remove job i by using the formula below (Equation (19))
   4.2 Randomly select job i from set B using the roulette wheel selection method
   4.3 L = L ∪ {i}
   4.4 B = B ∩ {i}
5. Return L

\[ O_i = \frac{r_i \times \frac{A}{\text{Min}_m}=i \Pi_{m=1}^A}{\sum_{k=1}^{r_k \times \frac{A}{\text{Min}_m}=i \Pi_{m=1}^A}} \]  \tag{19}

\( O_i \) is the probability of selecting job \( i \) from set \( L \). \( r_i \) is a random number for job \( i \), while \( A \) is a predefined constant. \( P_{im} \) is the processing time of job \( i \) in machine \( m \).

N-Longest Greedy Removal

The procedure of N-longest greedy removal is shown in Algorithm 3.

Algorithm 3 N-longest greedy removal

1. Randomly select a value of N that lies between 2 and I (the number of jobs)
2. B = I; I = [Sequence of all jobs]
3. L = {}
4. While |L| < N do
   4.1 Calculate the attractiveness of selection to remove job i by using the formula that follows (Equation (20))
   4.2 Randomly select job i from set B using the roulette wheel selection method
   4.3 L = L ∪ {i}
   4.4 B = B ∩ {i}
5. Return L
The N-jobs-string removal algorithm is shown in Algorithm 5.

Best Insertion

The principles used to select the repair method are as follows. We present three repair methods: (1) best insertion, (2) random insertion, and (3) cyclic insertion. The N-longest greedy removal is similar to the N-shortest greedy removal, but the difference is that the selection formula to remove jobs prefers a longer processing time. The remaining step is the same as the N-shortest greedy removal.

N-Two-Tails Removal

The N-two-tails removal algorithm is shown in Algorithm 4.

N-Jobs-String Removal

This method of destruction destroys the value of the considered solution based on a random selection of the position of the job sequence before removing the job from list \( I \). The N-jobs-string removal algorithm is shown in Algorithm 5.

4.4.2. Repairing Procedure

The complete solution is obtained after using the repair method. In this paper, we present three repair methods: (1) best insertion, (2) random insertion, and (3) cyclic insertion. The principles used to select the repair method are as follows.

Best Insertion

Best insertion is a method used to repair the solution. The job in list \( I \) is put into the available machine by considering the processing time of the job on each machine. Then, we determine which machine will be employed to process that job. The best insertion algorithm is shown in Algorithm 6.
Algorithm 6 Best insertion
1. \( B = L \{a, b, c, d, ..., Z\} \)
2. While \(|B| > 0\) do
   Insert job in position \(a\) into the machine that currently has the lowest energy consumption among all \(M\) machines (except the machine that it has been removed from).

For example, if we have a list of jobs \(\{2, 10, 8, 5\}\), producing job 2 would require 14, 41, and 39 energy units in machines A, B, and C, respectively. Since job 2 has been removed from machine C, we only have machines A and B to choose from. Therefore, job 2 will be assigned to machine A due to it requiring the lowest energy to produce. After that, we continuously execute jobs 10, 8, and 5 with the same mechanism until all jobs in \(B\) have been reassigned.

Random Insertion

Random insertion is a method used to repair an incomplete solution by finding a random machine to operate the job. Algorithm 7 shows the random insertion algorithm.

Algorithm 7 Random insertion
1. \( B = L \)
2. While \(|B| > 0\) do
   Insert the job in \(B\) list into a randomly selected machine that is not the machine that the job has been removed from.

For example, job 2 has been removed from machine C, so we only have two choices, machines A or B. Therefore, machines A or B will be randomly selected to operate job 2.

Cyclic Insertion

The cyclic insertion is shown in Algorithm 8. For example, since job \(B = \{2, 10, 8, 5\}\) is removed from machine list \(K = \{C, A, B, A\}\), jobs 2, 10, 8, and 5 are reassigned to be executed by machines A, B, A, and C, respectively.

Algorithm 8 Cyclic insertion
1. \( B = L \), Create machine list \(K\), which includes the list of machines the jobs in list \(B\) have been removed from; list \(K' = \) the list of machines that can be replaced cyclically or machine list \(K\)
i.e., \(B = \{2, 10, 8, 5\}, K = \{C, A, B, A\}, K' = \{A, C, A, B\}\)
2. While \(|B| > 0\) do
   Assign job from list \(B\) into the machine in list \(K'\)

4.5. Perform Solution Acceptance

If the new solution obtained after the destruction and reconstruction procedures performs better than the original solution, the new solution is automatically accepted to be the next current solution. A worse solution is sometimes accepted according to some probability functions. In this study, we apply Equation (21) as an acceptance function, where \(G^{\text{mix}}\) is the predefined maximum number of iterations:

\[
\text{acceptance probability} = 1 - \exp\left[\left[\frac{f^{\left(g_{i}(G+1)\right)} - f^{\left(g_{i}(G)\right)}}{f^{\left(g_{i}(G)\right)}} + (G - G^{\text{mix}})\right] \right]^2. \tag{21}
\]
4.6. Perform the Selection Procedure

This procedure is used to generate the target vectors for the next iteration. It is the vector selection step in DE that compares the cost of assignment (fitness function) with the cost from the target and trial vectors from the mutation procedure using Equation (22).

\[
x_{i,j,G+1} = \begin{cases} 
    u_{i,j,G+1} & \text{if } f(u_{i,j,G+1}) \leq f(x_{i,j,G}) \\
    x_{i,j,G} & \text{otherwise}
\end{cases}
\]  

(22)

In this procedure, if the cost of the assignment from the trial vector is less than or equal to the cost of assignment from the target vector, then the trial vector is selected and collected for the next population. In contrast, if the cost of the assignment given by the trial vector is greater than the target vector, then the target vector is collected from this population into the target vector for the next iteration. We repeat the steps in Sections 4.2–4.6 until the best answer is acquired. From the explanation in Sections 4.1–4.6, the proposed metaheuristics procedure is shown in Figure 2.

![Figure 2. Pseudocode of the proposed metaheuristics.](image)

5. Computational Framework and Results

The proposed metaheuristics were programed in C++ and simulated on a computer (Intel (R) Core i7-3520M CPU @ 2.90 GHz Ram 8.00 GB). We tested our algorithms with three groups of test instances: small, medium, and large. The details of the test instances are shown in Table 5.
Table 5. Details of the test instances.

| Group Test Instance | Number of Test Instances | $P_{jm}$ (min) | Number of Job | Number of Machine | $E_{jm}$ (THB) | $U$ | $B$ (THB) |
|---------------------|--------------------------|----------------|--------------|------------------|----------------|-----|---------|
| Small               | 12                       | 4–20           | 5–15         | 3–5              | 30–75          | 5–20| 10–100  |
| Medium              | 12                       | 4–20           | 20–28        | 5–7              | 30–75          | 5–20| 10–100  |
| Large               | 12                       | 4–20           | 70–130       | 5–7              | 30–75          | 5–20| 10–100  |
| Case study          | 1                        | 4–20           | 187          | 8                | 30–75          | 5–20| 10–100  |

In Table 5, there are 37 tested instances, composed of 12 each of small, medium, and large instances and one case study. For small test instances, the proposed methods were compared with the results obtained from the Lingo program (exact method). For medium and large test instances, the proposed methods were compared with the lower bound, which was generated by Lingo v.11 within 72 h of computation time. $E_{jm}$ is the energy used to produce job $j$ in machine $m$ (THB). $P_{jm}$ is the production time of job $j$ and machine $m$ (min) and is converted into monetary units (THB). $U$ is the percentage of gap difference for machine load balance. $B$ is a constant number (baht), which is set to be the production overhead cost per hour in the factory. The results for the proposed methods were compared with the results for the original differential evolution algorithm (DE) and the original adaptive large neighborhood search (ALNS). Details of the proposed methods and the compared methods are shown in Table 6.

Table 6. Details and comparison of the proposed methods.

| Algorithm          | Detail                     | Mutation Formula | Recombination Formula | Selection Formula |
|--------------------|----------------------------|------------------|-----------------------|-------------------|
| DE                 | Traditional DE             | 14               | 16                    | 19                |
| MDE-1              | Modified DE-1              | 14               | 17                    | 19                |
| MDE-2              | Modified DE-2              | 15               | 16                    | 19                |
| MDE-3              | Modified DE-3              | 15               | 17                    | 19                |
| ALNS               | Adaptive large neighborhood search | -               | -                     | -                 |
| HyDE-ALNS-1        | Hybrid DE and ALNS-1       | 14               | 17                    | 19                |
| HyDE-ALNS-2        | Hybrid DE and ALNS-2       | 15               | 16                    | 19                |
| HyDE-ALNS-3        | Hybrid DE and ALNS-3       | 15               | 17                    | 19                |

Table 6 shows the eight algorithms that were tested in the mentioned datasets. Traditional DE and ALNS were compared with the proposed methods, which are MDE-1-3 and HyDE-ALNS-1-3. The differences between MDE-1-3 and HyDE-ALNS 1-3 are the different mutation and recombination formulas, as shown in Table 6. The simulation was executed five times and the optimal solution among all was selected as the solution, as shown in Tables 7–14.

The first experiment was executed with the small and medium test instances. The stopping criterion for the Lingo program was the point at which it found the optimal solution. The optimal solution and computational time were collected. The stopping criteria for all proposed methods were set as 10% of the lowest computational time for the Lingo program. In this case, the lowest computational time to find the optimal solution for the Lingo program was 651 s, so the stopping criteria of all proposed methods were set to 65.1 s. Five runs were executed for each proposed method; the optimal solution is shown in Table 7 and the results of the statistical tests are shown in Table 8.
Table 7. Computational results of small test instances.

| No | Job | \( m \) | Obj. (THB) | Com. Time (s) | DE | ALNS | MDE-1 | MDE-2 | HyDE-ALNS-1 | HyDE-ALNS-3 | HyDE-ALNS-3 |
|----|-----|-----|--------|-----------|----|------|------|------|-------------|-------------|-------------|
| S-1 | 5   | 3   | 472    | 872       | 472| 472  | 472  | 472  | 472         | 472         | 472         |
| S-2 | 6   | 3   | 528    | 651       | 528| 528  | 528  | 528  | 528         | 528         | 528         |
| S-3 | 7   | 3   | 635    | 2056      | 635| 635  | 635  | 635  | 635         | 635         | 635         |
| S-4 | 8   | 3   | 735    | 1843      | 735| 735  | 735  | 735  | 735         | 735         | 735         |
| S-5 | 8   | 4   | 649    | 4561      | 649| 649  | 649  | 649  | 649         | 649         | 649         |
| S-6 | 10  | 4   | 834    | 86,400    | 834| 834  | 834  | 834  | 834         | 834         | 834         |
| S-7 | 11  | 4   | 840    | 86,400    | 858| 848  | 840  | 851  | 851         | 840         | 840         |
| S-8 | 12  | 4   | 826    | 86,400    | 826| 840  | 834  | 843  | 826         | 826         | 826         |
| S-9 | 12  | 5   | 842    | 20,787    | 842| 842  | 842  | 842  | 842         | 842         | 842         |
| S-10| 12  | 5   | 631    | 21,894    | 651| 631  | 640  | 647  | 640         | 631         | 631         |
| S-11| 14  | 5   | 845    | 86,400    | 845| 889  | 845  | 845  | 845         | 845         | 845         |
| S-12| 15  | 5   | 921    | 48,890    | 949| 934  | 934  | 921  | 921         | 921         | 921         |

Number of times that the proposed method can find the optimal solution (NTO) 8 8 9 9 10 12 12 12

%NTO 66.7% 66.7% 75% 75% 83.3% 100% 100% 100%

Table 8. Statistical testing for small test instances.

| Method | Lingo | DE | ALNS | MDE-1 | MDE-2 | MDE-3 | HyDE-ALNS-1 | HyDE-ALNS-2 | HyDE-ALNS-3 |
|--------|-------|----|------|------|------|------|-------------|-------------|-------------|
|        |       | 0.056 | 0.090 | 0.088 | 0.168 | 0.000 | 0.000       | 0.000       | 0.000       |
|        |       | 0.674 | 0.250 | 0.250 | 0.094 | 0.168 | 0.168       | 0.168       | 0.168       |
|        |       | 0.204 | 0.106 | 0.106 | 0.090 | 0.090 | 0.090       | 0.090       | 0.090       |
|        |       | 0.204 | 0.106 | 0.106 | 0.090 | 0.090 | 0.090       | 0.090       | 0.090       |
|        |       | 0.078 | 0.106 | 0.106 | 0.090 | 0.090 | 0.090       | 0.090       | 0.090       |
|        |       | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000       | 0.000       | 0.000       |

Table 9. Computation time of the proposed method that obtained the optimal solution.

| No | Optimal Solution | Com. Time (s) | DE | ALNS | MDE-1 | MDE-2 | MDE-3 | HyDE-ALNS-1 | HyDE-ALNS-2 | HyDE-ALNS-3 |
|----|-----------------|-------------|----|------|------|------|------|-------------|-------------|-------------|
| S-1| 472             | 872         | 62.8| 61.4 | 64.3 | 56.7 | 54.8 | 57.5        | 57.1        | 55.9        |
| S-2| 528             | 651         | 68.9| 70.7 | 75   | 65.5 | 62.5 | 69.3        | 69.9        | 64.5        |
| S-3| 635             | 2056        | 78.4| 79.1 | 74.9 | 75.1 | 72.3 | 65          | 64.5        | 62.9        |
| S-4| 735             | 1843        | 85.8| 82.5 | 104.4| 75.2 | 72.8 | 95.5        | 95.8        | 90.3        |
| S-5| 649             | 4561        | 112.8| 120.6| 130.6| 105.5| 102.3| 112.9       | 114.3       | 105.8       |
| S-6| 834             | 86,400      | N/A | 133.9| 145.9| 128.3| 120.5| 122.6       | 124.5       | 120.7       |
| S-7| 840             | 86,400      | N/A | 64.3 | N/A  | N/A  | N/A  | 131.3       | 131.1       | 128.6       |
| S-8| 826             | 86,400      | 150.8| N/A | N/A | N/A | N/A | 139.4       | 141.4       | 143.3       |
### Table 9. Cont.

| No | Lingo | Original Method(s) | Proposed Methods(s) |
|----|-------|--------------------|---------------------|
|    | Optimal Solution | Com. Time (s) | DE | ALNS | MDE-1 | MDE-2 | MDE-3 | HyDE-ALNS-1 | HyDE-ALNS-2 | HyDE-ALNS-3 |
| S-9 | 842 | 20,787 | 152.3 | 155.7 | 147.1 | 148.9 | 145.6 | 140.8 | 140.9 | 132.8 |
| S-10 | 631 | 21,894 | N/A | 161.3 | N/A | N/A | 142.3 | 142.2 | 141.3 |
| S-11 | 845 | 86,400 | 160.1 | N/A | 154.4 | 152.8 | 150.7 | 139.9 | 140.8 | 138.7 |
| S-12 | 921 | 86,400 | N/A | N/A | N/A | 154.3 | 150.8 | 57.5 | 57.1 | 55.9 |
| Average computational time (s) | 43,981.09 | 115.59 | 114.83 | 112.08 | 113.20 | 112.99 | 110.77 | 108.22 | 106.95 |

### Table 10. Statistical tests of computational times for small test instances.

|       | DE  | ALNS | MDE-1 | MDE-2 | MDE-3 | HyDE-ALNS-1 | HyDE-ALNS-2 | HyDE-ALNS-3 |
|-------|-----|------|-------|-------|-------|-------------|-------------|-------------|
| Lingo | 0.006 | 0.006 | 0.006 | 0.006 | 0.006 | 0.006 | 0.006 | 0.006 |
| DE    | 0.146 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| ALNS  | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| MDE-1 | 0.638 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| MDE-2 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| MDE-3 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| HyDE-ALNS-1 | 0.037 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| HyDE-ALNS-2 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

### Table 11. Computational results for medium test instances.

| No | Job | m | Exct Method | Traditional Methods | Proposed Methods |
|----|-----|---|-------------|--------------------|------------------|
|    |     |   | Lingo v.11 | DE | ALNS | MDE-1 | MDE-2 | MDE-3 | HyDE-ALNS-1 | HyDE-ALNS-2 | HyDE-ALNS-3 |
| M-1 | 20 | 5 | 1682 | 1411 | 1402 | 1368 | 1377 | 1302 | 1269 | 1258 | 1180 |
| M-2 | 20 | 6 | 1559 | 1428 | 1411 | 1357 | 1365 | 1349 | 1295 | 1289 | 1268 |
| M-3 | 22 | 5 | 1572 | 1597 | 1602 | 1569 | 1551 | 1523 | 1466 | 1452 | 1410 |
| M-4 | 22 | 6 | 1648 | 1633 | 1604 | 1578 | 1564 | 1550 | 1503 | 1494 | 1460 |
| M-5 | 23 | 5 | 1694 | 1792 | 1811 | 1786 | 1767 | 1733 | 1692 | 1673 | 1628 |
| M-6 | 23 | 6 | 1780 | 1698 | 1702 | 1665 | 1653 | 1602 | 1564 | 1538 | 1497 |
| M-7 | 25 | 5 | 1951 | 1867 | 1836 | 1783 | 1774 | 1746 | 1670 | 1649 | 1577 |
| M-8 | 25 | 6 | 1997 | 1883 | 1845 | 1794 | 1683 | 1679 | 1604 | 1610 | 1564 |
| M-9 | 26 | 6 | 1965 | 1817 | 1840 | 1788 | 1791 | 1784 | 1659 | 1655 | 1618 |
| M-10 | 28 | 5 | 1963 | 1872 | 1889 | 1806 | 1796 | 1788 | 1748 | 1752 | 1713 |
| M-11 | 28 | 6 | 2074 | 1898 | 1894 | 1887 | 1873 | 1845 | 1785 | 1794 | 1756 |
| M-12 | 28 | 7 | 1985 | 1953 | 1904 | 1881 | 1889 | 1867 | 1844 | 1813 | 1788 |
Table 12. Objective function statistical test for medium test instances.

| Algorithm | Lingo | DE   | ALNS  | MDE-1 | MDE-2 | MDE-3 | HyDE-ALNS-1 | HyDE-ALNS-2 | HyDE-ALNS-3 |
|-----------|-------|------|-------|-------|-------|-------|-------------|-------------|-------------|
| Lingo     | 0.011 | 0.008| 0.001 | 0.001 | 0.000 | 0.000 | 0.000       | 0.000       | 0.000       |
| DE        | 0.242 | 0.001| 0.001 | 0.000 | 0.000 | 0.000 | 0.000       | 0.000       | 0.000       |
| ALNS      | 0.000 | 0.001| 0.000 | 0.000 | 0.000 | 0.000 | 0.000       | 0.000       | 0.000       |
| MDE-1     | 0.135 | 0.001| 0.000 | 0.000 | 0.000 | 0.000 | 0.000       | 0.000       | 0.000       |
| MDE-2     | 0.001 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000       | 0.000       | 0.000       |
| MDE-3     | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000       | 0.000       | 0.000       |
| HyDE-ALNS-1 | 0.018 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000       | 0.000       | 0.000       |
| HyDE-ALNS-2 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000       | 0.000       | 0.000       |

From the computational results shown in Tables 7 and 8, we see that all hybrid methods (HyDE-ALNS-1, HyDE-ALNS-2, and HyDE-ALNS-3) produced the same results in terms of the optimal solution obtained from the Lingo program. The original DE and ALNS found an optimal solution 8 out of 12 times of the test instances (66.7%), while the modified version of DE using different mutation and recombination formulas (MDE-1 and MDE-2) found an optimal solution 9 out of 12 times of the test instances (75%); when using
the proposed mutation and recombination formula, MDE-3 found an optimal solution 10 out of 12 times of the test instances (83.3%).

In the next experiment, we used the small test instances. We ran all proposed methods until the Lingo program found the optimal solution. The computation time to find the optimal solution was recorded. If an optimal solution was not found within 24 h, the computation was terminated and recorded as N/A. The computation time that each method took to find the optimal solution is recorded in Table 9.

The results in Tables 9 and 10 reveal that the original DE had a longer computation time than MDE and ALNS to find the optimal solution. The hybrid of MDE and ALNS required a lower computation time when searching for the optimal solution than the original version of MDE. The average computation times of the original methods (DE and ALNS), MDE, and hybrid metaheuristics of MDE and ALNS are shown in Figure 3.

As shown in Figure 3a, the hybrid method obtained the optimal solution the fastest, followed by MDE, and then by original methods such as DE and ALNS. Figure 3 reveals the effect of applying the modified mutation and recombination formulas (Equations (15) and (17)) to the tested methods, while Equations (14) and (16) are the original mutation and recombination formulas. Figure 3b illustrates that using the modified versions of both the mutation and recombination formulas together required the lowest computation time to find the optimal solution. However, a mixed version of modified and original formulas obtained the optimal solution faster than using both original versions (mutation and recombination formulas) together.

The third experiment was tested with the medium dataset, which was composed of 12 test instances. We randomly selected 20 to 28 jobs and the number of machines was set to 5–7. Lingo was executed for 48 h. The computation time of all metaheuristics was set with 30 min as the termination condition. The simulation was executed five times. The optimal solution is recorded in Tables 11 and 12.

Tables 11 and 12 show that the optimal solution found after 48 h was worse than that of all proposed metaheuristics. While DE, ALNS, MDE-1, and MDE-2 did not perform significantly differently, all hybrid methods performed significantly differently from the other proposed methods. Thus, the best method for the medium test instances was HyDE-ALNS-3, which included Equations (15) and (17) as the mutation and recombination processes, respectively.

The fourth experiment tested the large random test instances set, which was composed of 12 test instances. The number of jobs was randomly selected from 70 to 130 and the number of machines was set to 5–7 (L-1 to L-12). In this set of test instances, we also tested
a case study that contained 187 jobs and eight machines (C-1). Lingo was executed for 72 h. The computation time of all metaheuristics was set with 45 min as the termination condition. The simulation was executed for five replications, and the optimal solution was recorded. Tables 13 and 14 display the statistical results.

Tables 13 and 14 indicate that hybrid methods outperformed all other methods, including the original version. HyDE-ALNS-3 was the best method, as with the medium test instances. HyDE-ALNS-3 employed Equations (15) and (17) as the mutation and recombination formulas. Therefore, we can conclude that these new formulas are superior to the original version.

The percentage difference of all proposed methods with the results obtained from Lingo was calculated using Equation (23):

\[
\% \text{diff} = \left( \frac{F_{\text{opt}} - F_H}{F_{\text{opt}}} \right) \times 100\% ,
\]

where \( F_{\text{opt}} \) is the solution obtained from Lingo during the predefined computation time (48 h) and \( F_H \) is the solution obtained from the proposed metaheuristics.

Figure 4 shows the percentage differences between all proposed methods and the results obtained from Lingo. It is evident that the proposed hybrid methods (HyDE-ALNS-1-3) perform better when the problem size is larger. The percentage differences between the solution obtained from Lingo and the proposed hybrid methods for small, medium, and large problem sizes were as follows: for HyDE-ALNS-1 they were 0, 12.58, and 13.29; for HyDE-ALNS-2 they were 0, 13.16, and 13.75; and for HyDE-ALNS-3 they were 0, 15.54, and 16.72, respectively. This indicates that the larger the problem size, the better the performance of the hybrid methods.

\[\text{Figure 4. Average percentage differences between the proposed metaheuristics and Lingo.}\]

In the last experiment, the effectiveness was examined by a percentage allowance: we allowed each machine load to be different (%GAP), as calculated by Equation (24). HyDE-ALNS-3 was selected to be representative of all proposed methods due to it being the optimal method from the results in Tables 10–14. In this context, the number after the name HyDE-ALNS was the maximum %GAP allowance; for instance, HyDE-ALNS-2.0 refers to each machine being allowed to have a maximum 2% different load from other machines. The experiment was executed with the medium test instances (12) and the results are shown in Table 15.
where \( F_r \) is the solution obtained from the maximum percentage allowance of HyDE-ALNS-3 and \( F_p \) is the solution obtained from the proposed methods from maximum percentage allowance.

| Instance | Objective Function (THB) |
|----------|--------------------------|
| M-1      | 1342 1342 1248 1180 1180 1180 1180 |
| M-2      | 1440 1440 1362 1268 1268 1190 1190 |
| M-3      | 1539 1539 1539 1410 1410 1410 1410 |
| M-4      | 1553 1553 1553 1460 1460 1386 1386 |
| M-5      | 1756 1756 1756 1628 1628 1573 1573 |
| M-6      | 1559 1559 1559 1559 1497 1497 1497 |
| M-7      | 1676 1676 1676 1577 1577 1510 1510 |
| M-8      | 1679 1679 1602 1618 1618 1594 1594 |
| M-9      | 1717 1717 1694 1618 1618 1694 1694 |
| M-10     | 1803 1803 1713 1713 1713 1688 1688 |
| M-11     | 1859 1859 1756 1756 1756 1756 1756 |
| M-12     | 1890 1890 1853 1788 1788 1788 1788 |

For the experiment shown in Table 15, we calculated the percentage of different load allowances from Equation (22) and applied ANOVA. In this context, %GAP affected the energy used and the \( p \)-value was equal to 0.000, which means that the level of %GAP affected the objective function. The average percentage difference of each %GAP is shown in Figure 5. We can see that the more %GAP machine load allowed, the lower the objective function obtained.

6. Conclusions

In the context of sustainable manufacturing, green machine scheduling is a problem that requires balancing energy efficiency, reducing the energy consumption of machines, and improving productivity. In this paper we proposed novel methods, called the hybrid differential evolution algorithm and adaptive large neighborhood search (HyDE-ALNS), to solve the problem of parallel machine scheduling in order to minimize energy consumption in consideration of machine load balance. In this study we attempted to improve the local
search method performance and the obtained solution quality by adjusting and developing two original algorithms, DE and ALNS. Thus, three modified DE and three hybrid DE and ALNS algorithms were proposed in this article: MDE-1-3 and HyDE-ALNS-1-3.

We first formulated a mathematical model to find the solution using Lingo v.11. The outcomes and computation times of the proposed algorithms were compared with Lingo and executed with small and medium problem sizes. The numerical results illustrated that all hybrid methods (HyDE-ALNS-1-3) provided the same result in terms of the optimal solution acquired from Lingo. Then, new mutation and recombination procedures were proposed for modifying the differential evolution algorithm (DE). The findings of this modified DE (MDE) led to a promising performance in terms of the outcomes and running times. As shown in the results, the proposed methods obtained optimal solutions within 2, 30, and 45 min, depending on the problem size, while Lingo required 24, 48, and 72 h to execute the small, medium and large problem sizes, respectively. Therefore, not only did it improve the solution quality of the original DE by 0.91%, 4.95%, and 3.34% for small, medium, and large test instances, respectively, but it also required less computation time to achieve a superior solution.

Furthermore, we proposed a hybrid method of DE with ALNS (HyDE-ALNS). This proposed hybrid method is composed of DE and some of the ALNS operations, applying destruction and repair operations of ALNS into DE after the recombination process to improve the solution quality. Applying ALNS to the HyDE-ALNS, five newly designed removal methods were presented. These were (1) N-removal; (2) N-shortest greedy removal; (3) N-longest greedy removal; (4) N-two-tails removal; and (5) N-jobs-string removal, including one new repair method, cyclic insertion. A comparison of these proposed procedures, enabling all proposed hybrid methods (HyDE-ALNS-3), generated a better solution quality of the best modified DE (MDE-3) by 0.22%, 6.01%, and 9.26% for small, medium, and large test instances, respectively, and also required less computation time to find a better solution.

This study showed the advantages of HyDE-ALNS include a lower computation time to find a solution and solution quality equal to that of Lingo for medium and large problem sizes. This indicates that the larger the problem size, the better the performance of HyDE-ALNS. Additionally, in this study, the problem of machine load balance was considered together with the minimizing of energy consumption in parallel machine scheduling. In our experiments, we found that the %GAP machine load balance significantly influenced the total energy consumption in the production system; as a result, the more %GAP machine load that was allowed, the lower the objective function that was obtained.

As long as machines are the main energy consumers in the manufacturing industry, reducing the energy consumption of machines can significantly promote manufacturing sustainability. Therefore, keeping the focus on reducing the energy used in production plants by developing different kinds of efficient optimization methods is still an attractive strategy. In addition, further considerations such as job priority, job tardiness, and makespan control will be interesting topics for future research to address. Furthermore, in order to reflect more realistic industrial manufacturing environments, an uncertainty scenario should be considered in future research by applying variability in demand, machining times, or breakdowns. We can also study the performance of the proposed method compared with different approaches such as simulation and artificial intelligence.

**Author Contributions:** Conceptualization, R.N. and C.-H.L.; methodology, R.N.; software, S.P.; validation, C.-H.L., S.P., and K.N.; formal analysis, R.N.; investigation, K.N.; writing—original draft preparation, R.N.; writing—review and editing, K.N.; project administration, C.-H.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.
Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fang, K.; Uhan, N.; Zhao, F.; Sutherland, J.W. A New Approach to Scheduling in Manufacturing for Power Consumption and Carbon Footprint Reduction. *J. Manuf. Syst.* 2011, 30, 234–240. [CrossRef]

2. Yin, L.; Li, X.; Lu, C.; Gao, L. Energy-Efficient Scheduling Problem Using an Effective Hybrid Multi-Objective Evolutionary Algorithm. *Sustainability* 2016, 8, 1268. [CrossRef]

3. Pitakaso, R.; Sethanan, K.; Theeravririya, C. Variable Neighborhood Strategy Adaptive Search for Solving Green 2-Echelon Location Routing Problem. *Comput. Electron. Agric.* 2020, 173, 105406. [CrossRef]

4. Liu, C.-H.; Huang, D.-H. Reduction of Power Consumption and Carbon Footprints by Applying Multi-Objective Optimisation via Genetic Algorithms. *Int. J. Prod. Res.* 2013, 52, 337–352. [CrossRef]

5. Zhang, Z.; Wu, L.; Peng, T.; Jia, S. An Improved Scheduling Approach for Minimizing Total Energy Consumption and Makespan in a Flexible Job Shop Environment. *Sustainability* 2019, 11, 179. [CrossRef]

6. Fysikopoulos, A.; Pastras, G.; Alexopoulos, T.; Chryssolouris, G. On a Generalized Approach to Manufacturing Energy Efficiency. *Comput. Oper. Res.* 2014, 73, 1437–1452. [CrossRef]

7. Kones, I.; Levin, A. A Unified Framework for Designing EPTAS for Load Balancing on Parallel Machines. *Algorithmica* 2019, 81, 3025–3046. [CrossRef]

8. Lin, Y.K.; Pfund, M.E.; Fowler, J.W. Heuristics for Minimizing Regular Performance Measures in Unrelated Parallel Machine Scheduling Problems. *Comput. Oper. Res.* 2011, 38, 901–916. [CrossRef]

9. Zhou, S.; Li, X.; Du, N.; Pang, Y.; Chen, H. A Multi-Objective Differential Evolution Algorithm for Parallel Batch Processing Machine Scheduling Considering Electricity Consumption Cost. *Comput. Oper. Res.* 2018, 96, 55–68. [CrossRef]

10. Keskinurk, T.; Yildirim, M.B.; Barut, M. An Ant Colony Optimization Algorithm for Load Balancing in Parallel Machines with Sequence-Dependent Setup Times. *Comput. Oper. Res.* 2012, 39, 1225–1235. [CrossRef]

11. Kones, I.; Levin, A. A Unified Framework for Designing EPTAS for Load Balancing on Parallel Machines. *Algorithmica* 2019, 81, 3025–3046. [CrossRef]

12. Sethanan, K.; Jamrus, T. Hybrid Differential Evolution Algorithm and Genetic Operator for Multi-Trip Vehicle Routing Problem with Backhauls and Heterogeneous Fleet in the Beverage Logistics Industry. *Comput. Ind. Eng.* 2020, 146, 106571. [CrossRef]

13. Theeravririya, C.; Sirirak, W.; Praseeratasang, N. Location and Routing Planning Considering Electric Vehicles with Restricted Distance in Agriculture. *World Electr. Veh. J.* 2020, 11, 61. [CrossRef]

14. Liu, C.-H.; Nanthapodej, R.; Hsu, S.-Y. Scheduling Two Interfering Job Sets on Parallel Machines under Peak Power Constraint. *Prod. Eng.* 2018, 12, 611–619. [CrossRef]

15. Jirarisirlerd, G.; Pitakaso, R.; Sethanan, K.; Kaewman, S.; Sirirak, W.; Kosacka-Olejnik, M. Simple Assembly Line Balancing Problem Type 2 by Variable Neighborhood Strategy Adaptive Search: A Case Study Garment Industry. *J. Open Innov. Technol. Mark. Complex.* 2020, 6, 21. [CrossRef]

16. Kusoncum, C.; Sethanan, K.; Pitakaso, R.; Hartl, R.F. Heuristics with Novel Approaches for Cyclical Multiple Parallel Machine Scheduling in Sugarcane Unloading Systems. *Int. J. Prod. Res.* 2020, 58, 1–19. [CrossRef]

17. Zeng, Y.; Che, A.; Wu, X. Bi-Objective Scheduling on Uniform Parallel Machines Considering Electricity Cost. *Eng. Optim.* 2017, 50, 19–36. [CrossRef]

18. Sethanan, K.; Wisittipanich, W.; Wisittipanit, N.; Nitisiri, K.; Moonsri, K. Integrating Scheduling with Optimal Sublot for Parallel Machine with Job Splitting and Dependent Setup Times. *Comput. Ind. Eng.* 2019, 137, 106095. [CrossRef]

19. Aubry, A.; Rossi, A.; Espinouse, M.-L.; Jacomino, M. Minimizing Setup Costs for Parallel Multi-Purpose Machines under Load-Balancing Constraint. *Eur. J. Oper. Res.* 2008, 187, 1115–1125. [CrossRef]

20. Eltaeib, T.; Mahmood, A. Differential Evolution: A Survey and Analysis. *Appl. Sci.* 2018, 8, 1945. [CrossRef]

21. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* 1997, 11, 341–359. [CrossRef]

22. Wang, W.-L.; Wang, H.-Y.; Zhao, Y.-W.; Zhang, L.-P.; Xu, X.-L. Parallel Machine Scheduling with Splitting Jobs by a Hybrid Differential Evolution Algorithm. *Comput. Oper. Res.* 2013, 40, 1196–1206. [CrossRef]

23. Zhou, S.; Liu, M.; Chen, H.; Li, X. An Effective Discrete Differential Evolution Algorithm for Scheduling Uniform Parallel Batch Processing Machines with Non-Identical Capacities and Arbitrary Job Sizes. *Int. J. Prod. Econ.* 2016, 179, 1–11. [CrossRef]

24. Wu, X.; Che, A. A Memetic Differential Evolution Algorithm for Energy-Efficient Parallel Machine Scheduling. *Omega* 2019, 82, 155–165. [CrossRef]

25. Li, D.; Wang, J.; Qiang, R.; Chiong, R. A Hybrid Differential Evolution Algorithm for Parallel Machine Scheduling of Lace Dyeing Considering Colour Families, Sequence-Dependent Setup and Machine Eligibility. *Int. J. Prod. Res.* 2020, 58, 1–17. [CrossRef]

26. Qin, A.K.; Huang, V.L.; Suganthan, P.N. Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization. *IEEE Trans. Evol. Comput.* 2009, 13, 398–417. [CrossRef]

27. Praseeratasang, N.; Pitakaso, R.; Sethanan, K.; Kosacka-Olejnik, M.; kaewman, S.; Theeravririya, C. Adaptive Large Neighborhood Search to Solve Multi-Level Scheduling and Assignment Problems in Broiler Farms. *J. Open Innov. Technol. Mark. Complex.* 2019, 5, 37. [CrossRef]
28. Rajakumar, S.; Arunachalam, V.P.; Selladurai, V. Workflow Balancing in Parallel Machines through Genetic Algorithm. *Int. J. Adv. Manuf. Technol.* 2006, 33, 1212–1221. [CrossRef]

29. Miwa, T.; Nishihara, N.; Yamamoto, K. Automated Stepper Load Balance Allocation System. *IEEE Trans. Semicond. Manuf.* 2005, 18, 510–516. [CrossRef]

30. Zhang, Y.; Hou, S.; Chang, L. Optimization of Particle Genetic Algorithm Based on Time Load Balancing for Cloud Task Scheduling in Cloud Task Planning. *Int. J. Perform. Eng.* 2018. [CrossRef]

31. Jouhari, H.; Lei, D.; Al-qaress, M.A.A.; Elaziz, M.A.; Ewees, A.A.; Farouk, O. Sine-Cosine Algorithm to Enhance Simulated Annealing for Unrelated Parallel Machine Scheduling with Setup Times. *Mathematics* 2019, 7, 1120. [CrossRef]

32. Lee, J.-H.; Jang, H. Uniform Parallel Machine Scheduling with Dedicated Machines, Job Splitting and Setup Resources. *Sustainability* 2019, 11, 7137. [CrossRef]

33. Zhang, B.; Wu, D.; Song, Y.; Liu, K.; Xiong, J. A Novel Fast Parallel Batch Scheduling Algorithm for Solving the Independent Job Problem. *Appl. Sci.* 2020, 10, 460. [CrossRef]

34. Jouhari, H.; Lei, D.; Al-qaress, M.A.A.; Elaziz, M.A.; Damaševičius, R.; Korytkowski, M.; Ewees, A.A. Modified Harris Hawks Optimizer for Solving Machine Scheduling Problems. *Symmetry* 2020, 12, 1460. [CrossRef]

35. Cota, L.P.; Guimarães, F.G.; Ribeiro, R.G.; Meneghini, L.R.; de Oliveira, F.B.; Souza, M.J.F.; Siarry, P. An Adaptive Multi-Objective Algorithm Based on Decomposition and Large Neighborhood Search for a Green Machine Scheduling Problem. *Swarm Evol. Comput.* 2019, 51, 100601. [CrossRef]

36. Shim, S.-O.; Park, K. Technology for Production Scheduling of Jobs for Open Innovation and Sustainability with Fixed Processing Property on Parallel Machines. *Sustainability* 2016, 8, 904. [CrossRef]

37. Arnaout, J.-P. A Worm Optimization Algorithm to Minimize the Makespan on Unrelated Parallel Machines with Sequence-Dependent Setup Times. *Ann. Oper. Res.* 2019, 285, 273–293. [CrossRef]

38. Ouazene, Y.; Hnaien, F.; Yalaoui, F.; Amodeo, L. The Joint Load Balancing and Parallel Machine Scheduling Problem. *Oper. Res. Proc.* 2011, 497–502. [CrossRef]

39. Mouzon, G.; Yildirim, M.B.; Twomey, J. Operational Methods for Minimization of Energy Consumption of Manufacturing Equipment. *Int. J. Prod. Res.* 2007, 45, 4247–4271. [CrossRef]

40. Angel, E.; Bampis, E.; Kacem, F. Energy Aware Scheduling for Unrelated Parallel Machines. Available online: https://ieeexplore.ieee.org/abstract/document/668361 (accessed on 27 February 2021). [CrossRef]

41. Sobottka, T.; Kamhuber, F.; Heinzl, B. Simulation-Based Multi-Criteria Optimization of Parallel Heat Treatment Furnaces at a Casting Manufacturer. *Int. J. Manuf. Mater. Process.* 2020, 4, 94. [CrossRef]

42. Liu, C.-H. Approximate Trade-off between Minimisation of Total Weighted Tardiness and Minimisation of Carbon Dioxide (CO2) Emissions in Bi-Criteria Batch Scheduling Problem. *Int. J. Comput. Integr. Manuf.* 2013, 27, 759–771. [CrossRef]

43. Pan, Z.; Lei, D.; Zhang, Q. A New Imperialist Competitive Algorithm for Multiobjective Low Carbon Parallel Machines Scheduling. *Math. Probl. Eng.* 2018, 2018, 1–13. [CrossRef]

44. Khamsing, N.; Chindapraset, K.; Pitakaso, R.; Sirirak, W.; Theeravigiya, C. Modified ALNS Algorithm for a Processing Application of Family Tourist Route Planning: A Case Study of Buriram in Thailand. *Computation* 2021, 9, 23. [CrossRef]

45. Prasertatsang, N.; Pitakaso, R.; Sethanan, K.; Kaewman, S.; Golinska-Dawson, P. Adaptive Large Neighborhood Search for a Production Planning Problem Arising in Pig Farming. *J. Open Innov. Technol. Mark. Complex.* 2019, 5, 26. [CrossRef]

46. Pitakaso, R.; Sethanan, K. Adaptive Large Neighborhood Search for Scheduling Sugarcane Inbound Logistics Equipment and Machinery under a Sharing Infield Resource System. *Comput. Electron. Agric.* 2019, 158, 313–325. [CrossRef]

47. Cota, L.P.; Guimarães, F.G.; de Oliveira, F.B.; Souza, M.J.F. An Adaptive Large Neighborhood Search with Learning Automata for the Unrelated Parallel Machine Scheduling Problem. Available online: https://ieeexplore.ieee.org/abstract/document/7969312 (accessed on 26 February 2021).