Metaheuristics for Energy-Efficient No-Wait Flowshops: A Trade-off Between Makespan and Total Energy Consumption

Damla Yüksel
Department of Industrial Engineering
Yasar University
Izmir, Turkey
damla.yuksel@yasar.edu.tr

Mehmet Fatih Taşgetiren
Department of International Logistics Management
Yasar University
Izmir, Turkey
fatih.tasgetiren@yasar.edu.tr

Levent Kandiller
Department of Industrial Engineering
Yasar University
Izmir, Turkey
levent.kandiller@yasar.edu.tr

Quan-Ke Pan
School of Mechatronic Engineering and Automation
Shanghai University
Shanghai, China
panquanke@shu.edu.cn

Abstract— **No-wait flowshop scheduling problem (NWFSP) is a well-known strongly NP-hard problem, where in-process waiting is not allowed between any two consecutive machines in such a way that once a job is started, subsequent processing must be carried out on all machines until completion. In this paper, we propose an energy-efficient NWFSP in order to investigate the trade-off between makespan and total energy consumption. The energy-efficient NWFSP aims to seek to obtain Pareto solution sets to minimize the makespan and the total energy consumption conflicting with each other. Unlike the classical NWFSP, there are different speed levels for each job on machines and the processing times of jobs can differ according to the assigned speed levels. Therefore, we modify the formulation of NWFSP by introducing a speed scaling strategy in order to approximate Pareto solution sets, i.e., non-dominated solution sets. In this paper, we propose a mixed-integer linear programming model (MILP), an energy-efficient variable block insertion heuristic (EE-VBIH), an energy-efficient iterated greedy algorithm (IG) and an energy-efficient & IG-ALL) to solve the energy-efficient NWFSP. Extensive computational analyses on Taillard's benchmark suite show that the proposed algorithms are very effective for approximating Pareto solution sets.**

Keywords— no-wait flowshop scheduling problem, energy-efficient scheduling, metaheuristics, multi-objective optimization

I. INTRODUCTION

Unlike the traditional permutation flowshop scheduling problem (PFSP), its variant, so-called no-wait flowshop scheduling problem (NWFSP), is very common in many industries, such as the chemicals, plastics, metals, electronics, pharmaceuticals, and food-processing industries [1], [2]. Due to the technical reasons, in-process waiting is not allowed between any two consecutive machines in these industries in such a way that once a job is started on the first machine, processing of jobs without interruption must be continuously carried out on all machines until completion. The problem is strongly NP-hard for three or more machines [3]. For the NWFSP with the makespan criterion, an excellent paper is presented in [4] where they developed two mat-heuristics to solve almost all benchmark suites in the literature to optimality. In addition, a very detailed literature review on NWFSP with the makespan criterion can be found in [4]. In this study, we consider makespan together with a total energy consumption criterion for the energy-efficient NWFSP. In this paper, we develop an energy-efficient NWFSP in order to investigate the trade-off between the makespan and the total energy consumption. The proposed energy-efficient NWFSP seeks to obtain Pareto solutions sets by minimizing the makespan and the total energy consumption, simultaneously, which are conflicting with each other. Unlike the classical NWFSP, we employ speed scaling strategy, where there are different speed levels for each job on machines and processing time of a job can differ according to the assigned speed levels. Therefore, we modify the formulation of NWFSP by employing a job-based speed scaling strategy, in which machines can be operated at different speed levels, which correspond to different energy consumption levels. The trade-off between the processing time and the energy consumption emerges with a result of different speed levels, as higher speed levels consume higher energy while decreasing the processing times. On the contrary, lower speed levels consumes less energy while increasing the processing times.

High-energy consumption is a vital problem for manufacturing companies because of the increase in fuel prices and negative environmental impacts such as global warming and CO_2 emissions. For these reasons, industries seek energy-efficient scheduling [5]. A comprehensive review of energy-efficient scheduling was presented [6]. As a pioneering study, the turn-off strategy was suggested in [7] and they concluded that a substantial amount of energy can be saved when the machine is turned off during idle times. Later on, they applied the turn off strategy to the single machine scheduling problem in order to minimize the total energy consumption and the total tardiness in [8]. Turn off strategy was also applied for the flexible flowshop problem [9]. Even though the turn off strategy saves energy, it is not practical in some production environments as it might ruin the service life of some machines. Hence, a speed scaling strategy was developed for the energy-efficient permutation flowshop scheduling problem (PFSP) by considering the operation speed as an independent variable that can vary to improve the energy efficiency in [5]. In addition, a mixed-integer programming formulation was proposed for the PFSP with makespan minimization with a constraint on peak

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

power consumption [10]. Speed scaling strategy was also proposed for the PFSP with the objectives of the total carbon emissions and makespan in [11]. They developed a modified multi-objective iterated greedy (IG) algorithm. Speed scaling strategy was also implemented on the two-machine sequence-dependent PFSP in [12]. Later on, they extended their study to a genetic algorithm (GA) for the same problem in [13]. An energy-efficient evolutionary algorithm was proposed for the single machine scheduling problem by [14], while a backtracking algorithm was proposed for the energy-efficient PFSP by [15]. Furthermore, an energy-efficient job shop scheduling problem was presented in [16]. A multi-level approach was proposed in [17] while energy-efficient dynamic scheduling was presented in [18]. An energy-efficient genetic algorithm for the job shop scheduling problem was also developed in [19]. An energy-efficient green PFSP was presented in [20] whereas an energy-efficient single machine total weighted tardiness problem with sequence-dependent setup times was proposed in [21] More recently, a memetic differential evolution algorithm for energy-efficient parallel machine scheduling is proposed in [22]. Lastly, very recent studies on speed scaling strategy have been performed for NWFSPs to minimize the total tardiness and the total energy consumption in the study of [23] and for hybrid flow shops to minimize the makespan and the total energy consumption in the study of [24]. However, to the best of our knowledge, the aforementioned energy-efficient consideration has not been employed for NWFSPs in the literature to minimize the makespan and the total energy consumption simultaneously. Hence, in this study, we initially propose a novel mixed-integer linear programming model (MILP) to fill the gap in the literature. Resulting from the NP-hardness nature of the problem, an energy-efficient variable block insertion heuristic algorithm (EE-VBIH), an iterated greedy algorithm (EE-IG) and a variant of EE-IG, that is, an iterated greedy algorithm with a local search (EE-IG$_{ALL}$) for the energy-efficient NWFSP problem is proposed. Extensive computational analysis on Taillard's benchmark suite show that EE-VBIH algorithm provide better approximations of the Pareto solution sets than the other algorithms from the point of cardinality, quality and diversity of the generated non-dominated solution sets.

The rest of the paper is organized as follows. The energy-efficient mixed-integer linear programming model and meta-heuristic formulation of NWFSP are given in Section II. In Section III, the details of all meta-heuristic algorithms are presented. In Section IV, the computational results are reported for both mathematical models and algorithms. Finally, in Section V, concluding remarks are presented with possible future research extensions.

II. Problem Formulation

In this paper, we study an energy-efficient NWFSP with two conflicting objectives of the makespan and the total energy consumption. The goal is to obtain Pareto solutions sets that optimize these two performance measures. Unlike the standard NWFSP, there is a finite and discrete set of l processing speed levels $L = \{1, .., l\}$ for each machine. Therefore, the processing time of a job may vary based on the assigned speed level. Due to the job-based speed-scaling structure, we assume that each job is processed with the same speed level through its route in the NWFSP. A similar total energy consumption (TEC) calculation from the two-machine PFSP model by [12] is employed. To solve the energy-efficient NWFSP, a mixed-integer linear programming approach and three metaheuristic algorithms are employed. Hence, the problem notations needed for all proposed approaches are given below in Table I.

TABLE I. Parameters and Decision Variables

Parameters	
P_{ir}	Processing time of job i on machine r
s_l	Speed factor of processing speed level l
λ_l	Conversion factor for processing speed level l
φ_r	Conversion factor for idle time on machine r
τ_r	Power of machine (kW) r
Q	A very large number
Decision Variables	
y_{irl}	1 if job i is processed at speed l on machine r, 0 otherwise.
C_{ir}	Completion time of job i on machines r
D_{ik}	1 if job i is scheduled any time before job k, 0 otherwise $(i < k)$
θ_r	Idle time on machine r
C_{max}	Maximum completion time (makespan)
TEC	Total energy consumption (kWh)

The mixed-integer linear programming model for the energy-efficient NWFSP is provided below:

Minimize C_{max} (1), Minimize TEC (2)

Subject to;

$$C_{i1} \geq \sum_{l \in L} \frac{P_{i1} * y_{i1l}}{s_l} \quad \forall i \in N \tag{3}$$

$$C_{ir} - C_{i,r-1} \geq \sum_{l \in L} \frac{P_{ir} * y_{irl}}{s_l} \quad \forall i \in N, \forall r \in M: r \geq 2 \tag{4}$$

$$C_{ir} - C_{kr} + Q * D_{ik} \geq \sum_{l \in L} \frac{P_{ir} * y_{irl}}{s_l} \quad \forall i \in N : k > i, \forall r \in M \tag{5}$$

$$C_{ir} - C_{kr} + Q * D_{ik} \leq Q - \sum_{l \in L} \frac{P_{kr} * y_{krl}}{s_l} \quad \forall i \in N : k > i, \forall r \in M \tag{6}$$

$$C_{max} \geq C_{iM} \quad \forall i \in N \tag{7}$$

$$C_{ir} - C_{i,r-1} \leq \sum_{l \in L} \frac{P_{ir} * y_{irl}}{s_l} \quad \forall i \in N, \forall r \in M: r \geq 2 \tag{8}$$

$$\sum_{l \in L} y_{irl} = 1 \quad \forall i \in N, \forall r \in M \tag{9}$$

$$y_{irl} = y_{i,r+1,l} \quad \forall i \in N, \forall r \in M: r < M, \forall l \in L \tag{10}$$

$$\theta_r = C_{max} - \sum_{i \in N} \sum_{l \in L} \frac{P_{ir} * y_{irl}}{s_l} \quad \forall r \in M \tag{11}$$

$$TEC = \sum_{i \in N} \sum_{r \in M} \sum_{l \in L} \frac{P_{ir} * \tau_r * \lambda_l}{60 s_l} y_{irl} + \sum_{r \in M} \frac{\varphi_r * \tau_r * \theta_r}{60} \tag{12}$$

$$y_{irl} \in \{0,1\} \quad \forall i \in N, \forall r \in M, \forall l \in L \tag{13}$$

$$C_{ir} \quad \forall i \in N, \forall r \in M \tag{14}$$

$$D_{ik} \in \{0,1\} \quad \forall i, k \in N : k > i \tag{15}$$

The makespan and the total energy consumption are minimized by the objective functions (1) and (2), respectively. The completion time of each job must be at least its processing time on machine 1 is provided by constraint (3). Constraint (4) provides that the difference between the completion time of a

job on any consecutive machines is at least greater than or equal to the processing time of the job. The precedence relation of jobs is provided by constraints (5) and (6). The makespan that is the maximum of completion times is computed by constraint (7). The no-wait restriction is preserved by constraint (8) together with the constraint (4). In other words, the completion time of a job on any machine plus its processing time on the consecutive machine must be equal to the completion time of that job on the consecutive machine. One speed level selection for each job is provided by constraint (9) and then each job will possess the same speed level on each machine is procured by constraint (10). The idle times on all machines are computed by constraint (11). Lastly, the total energy consumption is computed in kilowatt-hour by constraint (12) as provided by [12]. The binary variables and the sign restrictions are given in constraints (13), (14), and (15). It is significant to mention that the provided model is an expansion of Manne's PFSP model [25] with the addition of the no-wait restriction and the consideration of the total energy consumption in the objective function as the second objective. In other words, the energy efficiency idea incorporates the idle time and the total energy consumption computations as well as the speed level considerations as constraints.

From the point of heuristic approaches, single objective NWFSP with makespan criterion can be defined as follows: A set $N = \{J_1, J_2.., J_n\}$ of n jobs is to be processed on m machines, such that the same permutation will be used on all machines. All jobs and machines are available for processing at time zero. Each job $i\ (i = 1, 2, .., n)$ has a nonnegative processing time P_{ir} on machine $r\ (r = 1, 2, .., m)$. Each job can be processed at most by one machine, and every machine can process no more than one job at a time. No job queue is allowed at any machine, except for the first one. Once the processing of a job is started on the first machine, it must be processed continuously on all machines until completion. To satisfy the no-wait constraint, the processing time of the job on the first machine may be delayed to make sure that it does not need to wait for processing on subsequent machines. The aim is to find a feasible schedule $\pi = \{\pi_1, \pi_2, .., \pi_n\}$ for the n jobs such that the makespan is minimized, where $\pi_k (k = 1,2, .., n)$ denotes the job assigned to position k. Due to the no-wait constraint, the difference between the completion times of two consecutive jobs depends entirely on those two consecutive jobs [26], [27], and independent of their positions in the sequence and the sequence of other jobs. For this reason, the delay matrix $D_{(n+1)xn}$ can be pre-calculated and provides all $d([\pi_{k-1}, \pi_k])$ values for any schedule π as follows:

$$d([\pi_{k-1}, \pi_k]) = P_{\pi_{k-1},1} + \max\left\{0, \max_{2\le k\le m}\left\{\sum_{h=2}^{k} P_{\pi_{k-1},h} - \sum_{h=1}^{k-1} P_{\pi_k,h}\right\}\right\} \quad \forall k = 2, .., \mathrm{N} \tag{16}$$

Thus, the makespan (C_{max}) calculation can be easily done very rapidly as follows:

$$C_{max}(\pi) = \sum_{k=2}^{N} d_{\pi_{k-1},\pi_k} + \sum_{r=1}^{M} P_{\pi_N,r} \tag{17}$$

To convert the single objective NWFSP with makespan into an energy-efficient bi-objective NWFSP, speed levels should be considered in equations (16) and (17) as follows:

$$d\left([\pi_{k-1}, \pi_k][l_{\pi_{k-1}}, l_{\pi_k}]\right) = \frac{P_{\pi_{k-1},1}}{s_{l_{\pi_{k-1}}}} + \max\left\{0, \max_{2\le k\le m}\left\{\sum_{h=2}^{k} \frac{P_{\pi_{k-1},h}}{s_{l_{\pi_{k-1}}}} - \sum_{h=1}^{k-1} \frac{P_{\pi_k,h}}{s_{l_{\pi_k}}}\right\}\right\}$$

$$\forall k = 2, \ldots, N, \forall l \in \{1,2,3\} \tag{18}$$

$$C_{max}(\pi) = \sum_{k=2}^{N} d([\pi_{k-1}, \pi_k][l_{k-1}, l_k]) + \sum_{r=1}^{M} \frac{P_{\pi_N,r}}{s_{l_{\pi_k}}} \tag{19}$$

The idle time on each machine i can be calculated as follow:

$$\theta_r = C_{max}(\pi) - \sum_{k=1}^{N} \frac{P_{\pi_k,r}}{s_{l_{\pi_k}}} \qquad \forall r = 1, \ldots, M \tag{20}$$

Finally, the total energy consumption can be calculated as follows:

$$TEC = \sum_{k=1}^{N}\sum_{r=1}^{M} \frac{P_{\pi_k,r} * \tau_r * \lambda_{l_{\pi_k}}}{60 s_{l_{\pi_k}}} + \sum_{r=1}^{M} \frac{\vartheta_r * \tau_r * \theta_r}{60} \tag{21}$$

Now, the problem becomes an energy-efficient NWFSP with minimization of both the makespan and the total energy consumption, simultaneously in order to approximate a Pareto front solution set (i.e., non-dominated solution set). In multi-objective minimization problems, three main concepts are used to define relations between different solutions to the problem.

- **Dominance relation:** A feasible solution $\vec{r}$ dominates another feasible solution $\vec{t}$ if the two following conditions are satisfied (denoted as $\vec{r} \succ \vec{t}$):
  - $\forall p \in 1, .., P; f_p(\vec{r}) \le f_p(\vec{t})$
  - $\exists p \in 1, .., P; f_p(\vec{r}) < f_p(\vec{t})$

  A feasible solution $\vec{r}$ weakly dominates another feasible solution $\vec{t}$ (denoted as $\vec{r} \succcurlyeq \vec{t}$) if:
  - $\forall p \in 1, .., P; f_p(\vec{r}) \le f_p(\vec{t})$
- **Non-dominated set (X^*):** Amongst a set of solutions (X), the non-dominated set of solutions (X^*) are the solutions that are not dominated by any element of set X.
- **Pareto-optimal set**: The non-dominated (Pareto-optimal) solution set of the entire feasible search space is called as the Pareto-optimal set.

III. Meta-Heuristic Approaches

Due to the computational difficulty of the mathematical model, energy-efficient metaheuristics, namely, an energy-efficient variable block insertion heuristic algorithm (EE-VBIH), an energy-efficient iterated greedy algorithm (EE-IG) and an energy-efficient iterated greedy algorithm with optimization of partial solutions (EE-IG$_{ALL}$) are presented for the NWFSP where the makespan and the total energy consumption criteria are the objectives. In the literature, EE-VBIH, EE-IG, and EE-IG$_{ALL}$ are employed on the energy-efficient single machine scheduling problem by the studies provided by [21] and [26] and on the permutation flowshop scheduling problem by the study of [27]. To the best of our knowledge, there is no any study regarding the metaheuristic applications on NWFSPs with the makespan and the total energy consumption consideration. Hence, the concept of energy-

efficient meta-heuristics, which are provided in the literature for different scheduling problems, are redesigned for the NWFSPs and employed in this study. In addition, we did not employ the NSGA-II algorithm for this problem since its underperformance compared with other algorithms has explicitly shown in [23].

A. Solution Representation and Initial Population

In this study, we propose a job-based speed-scaling strategy for the energy-efficient NWFSP. Therefore, a multi-chromosome structure is developed, which includes a permutation of n jobs and a speed vector of three levels corresponding to fast, normal, and slow speed levels, respectively. The solution representation for an individual x_i is given in Fig. 1.

$x_i(\pi, l)$	π	3	2	1	4	5	…	n
	v	1	3	2	1	2	…	3

Fig. 1. Individual Solution Representation

In Fig. 1, the individual $x_i(\pi_{i,k}, l_{i,k})$ represents a solution where job $\pi_{i,1} = 3$ has a fast speed level ($l_{i,1} = 1$); job $\pi_{i,2} = 2$ has a slow speed level ($l_{i,2} = 3$) and so on. Note that the same speed vector is used in all machines. In all proposed algorithms, the initial population with size NP is constructed as follows. Firstly, an initial solution π^0 is constructed by the $FRB5$ heuristic [30] in Fig. 2, which is an extension of well-known NEH heuristic [31]. In the first phase, the sum of the processing times on all machines is calculated for each job. Then, the jobs are sorted in decreasing order to obtain δ. In the second phase, the first job in δ is selected to establish a partial solution. The remaining jobs in δ are inserted in the partial solution one by one. After each iteration, a local search is applied to the partial solution. Insertion local search is applied as long as the partial solution is improved. After having inserted all jobs, a complete solution π^0 is obtained.

$\delta = DecreasingOrder(\sum_{r=1}^{m} P_{ir})$
$\pi_1 = \delta_1$
$for\ i = 2\ to\ n\ do$
  $\pi^0 = InsertJobInBestPosition(\pi_i, \delta_i)$
  $\pi^0 = ApplyPartialLocalSearch(\pi^0, f(\pi_i))$,
$end\ for$
$return\ \pi^0\ with\ n\ jobs$

Fig. 2. FRB5 Constructive Heuristic

To obtain a diversified initial population, we fix the population size to $NP = 1$ with the initial solution π^0 and devote the 25% of the total CPU time to EE-VBIH, EE-IG, and EE-IG$_{ALL}$ algorithms by fixing all speed levels to $l_i = 2$, where $i \in \{1, .., n\}$. We run algorithms with normal speed level for each job and obtain an individual π_{best}. Then, we fix the population size to $NP = 100$. The first three individuals are generated by assigning fast, normal and slow speed levels to each job in the π_{best}. Then, the rest of the population is constructed by assigning random speed levels to each job in π_{best}. Note that with the same permutation, a different solution can be obtained by changing the speed level of any job in the permutation. The archive set ϕ is initially empty and will be updated with non-dominated solutions from the initial population.

B. Energy-Efficient Variable Block Insertion Heuristic Algorithm (EE-VBIH)

Variable block insertion heuristic algorithms are recently presented in the literature by [21], [32]–[34]. In this paper, we propose an energy-efficient VBIH (EE-VBIH) algorithm as follows. EE-VBIH algorithm establishes the initial population as mentioned in the previous section. As seen in Fig. 3, EE-VBIH randomly removes a block bS of jobs with their speed levels from individual x_i and they are stored in $x_{i,B}$. The remaining jobs and the speed levels are also stored in $x_{i,D}$. Then, an insertion local search as presented in Fig. 4 is applied to the partial solution $x_{i,D}$. Before block insertion moves, random speed levels are assigned to jobs in $x_{i,B}$ by $x_{i,B}(l_{i,B}) = rand()\%3$. Then, it carries out $n - bs + 1$ block insertion moves. In other words, the block $x_{i,B}$ is inserted in the partial solution $x_{i,D}$, sequentially. Again, it should be noted that dominance rule ($\succ$) in multi-objective optimization will be used when two solutions are compared. A complete solution is obtained by selecting the non-dominated solution among $n - bS + 1$ solutions after the block is inserted into last position of partial solution $x_{i,D}$. Finally, and the identical insertion local search is again applied to the complete solution obtained after block insertion moves. If the new solution x^* dominates the individual x_i, it is replaced by x^* and the archive set ϕ is updated. Then, the block size is incremented by one. Above process is repeated until it reaches maximum block size bS_{max}.

$Set\ NP = 100;\ bS_{max} = 8$
$Construct\ initial\ population\ with\ size\ NP$
$Compute\ delay\ matrix\ d_{ij}$
$While\ (NotTermination)\ do$
  $for\ i = 1\ to\ NP\ do$
    $bS = 2$
    do
      $x_{i,B}(\pi_{i,B}, l_{i,B}) = Remove\ a\ block\ x_{i,B}\ from\ x_i$
      $x_{i,B}(l_{i,B}) = rand()\%3$
      $x_{i,D}(\pi_{i,D}, l_{i,D}) = PartialLocalSearch(x_{i,D})$
      $x^*(\pi^*, l^*) = Insert\ block\ x_{i,B}\ in\ x_{i,D}$
      $x^*(\pi^*, l^*) = LocalSearch(x^*(\pi^*, l^*))$
      $if\ (x^* \succ x_i)\ then\ do$
        $x_i = x^*,$
        $Update\ the\ archive\ set\ \phi\ with\ x^*$
      $endif$
      $bS = bS + 1$
    $while\ (bS \leq bS_{max})$,
  $endfor$
  $for\ i = 1\ to\ NP\ do$
    $x^*(\pi^*, l^*) = \text{Crossover}(l_i, l_k)$
    $if\ (x^* \succ x_i)\ then\ do$
      $x_i = x^*,$
      $Update\ the\ archive\ set\ \phi\ with\ x^*$
    $endif$
  $endfor$
  $for\ i = 1\ to\ NP\ do$
    $x_i(\pi_i, l_i) = Mutation\ (l_i)$
  $endfor$
$endwhile,$
$return\ \phi$

endprocedure

Fig. 3. EE-VBIH Algorithm

Regarding local search, we employ a very effective insertion local search in all algorithms proposed, which is given in Fig. 4. For each position j, we remove jth job and its speed level as (π^*, l^*) from solution x_i and assign a random speed level to jth job. Then, we insert (π^*, l^*) into n positions. A non-dominated solution $x^*(\pi^*, l^*)$ from n insertion is obtained. If it dominates individual x_i, it is replaced by x^*.

for $j = 1$ *to* n *do*
  $(\pi^*, l^*) = Remove\ \ job\ and\ speed\ from\ x_i\ at\ position\ j$
  $l^* = rand()\%3$
  $x^*(\pi^*, l^*) = InsertInDominatingPosition(x_i, (\pi^*, l^*))$
  if $(x^* \succ x_i)$ *then do*
    $x_i = x^*\}$
  endif
endfor
return x_i
endprocedure

Fig. 4. Insertion Local Search

The EE-VBIH algorithm is extremely effective for makespan minimization. However, more energy-efficient schedules can be generated by employing a uniform crossover operator by using only the speed levels. The same permutation is kept for each individual, and a uniform crossover is performed only on the speed levels as follows: For each individual x_i in the population, another individual is selected from population randomly, say x_k, A new solution is generated either by taking the speed level from either x_i or x_k, depending on the crossover probability. We generate a new solution by making a uniform crossover as follows:

$$x^*(\pi^*, l^*) = \begin{cases} l_{i,j} & if\ r_{i,j} \le CR[i] \\ l_{k,j} & otherwise \end{cases} \quad j \in 1,.,n \qquad (22)$$

where $r_{i,j}$ is a uniform random number in $U(0,1)$. $CR[i]$ is the crossover probability, drawn from the unit normal distribution $N(0.5,0.1)$, with mean 0.5 and standard deviation 0.1. If $x^*(\pi^*, l^*)$ dominates x_i $(x^* \succ x_i)$, x_i is replaced by x^* and the archive set ϕ is updated. This is repeated for all individuals in the population. After crossover local search, we also mutate the speed levels of jobs in individuals in the population with a small mutation probability as follows:

$$x_i(\pi_{ij}, l_{ij}) = \begin{cases} l_{ij} = \text{rand}()\%3 & if\ r_{ij} \le MR[i] \\ l_{ij} & otherwise \end{cases} \quad j \in 1,..,n \ (23)$$

where $r_{i,j}$ is a uniform random number in $U(0,1)$. $MR[i]$ is the mutation probability, drawn from unit normal distribution $N(0.05,0.01)$ with mean 0.05 and a standard deviation of 0.01.

C. EE-IG and EE_IG$_{ALL}$ Algorithms

Iterated greedy (IG) algorithms have mainly four components, namely, initial solution, destruction and construction procedure, local search, and acceptance criterion [35]. Recently, a new IG$_{ALL}$ algorithm is presented [36] with excellent results for the PFSP with makespan minimization. The difference between IG$_{ALL}$ and traditional IG is that IG$_{ALL}$ applies an ***additional local search to partial solutions after destruction***, which substantially enhances solution quality.

In the EE_IG variants, dS jobs with their speed levels are removed from an individual x_i and stored in $x_{i,R}$. The remaining jobs are also stored in $x_{i,D}$. In the EE_IG, Each job in $x_{i,R}$ is inserted into $x_{i,D}$ for the construction of the final solution. However, in the EE_IG$_{ALL}$, an insertion local search in Fig. 4 is applied to the partial solution $x_{i,D}$. Before construction, random speed levels are assigned to jobs in $x_{i,R}$ by $x_{i,R}(l_{i,R}) = rand()\%3$. Regarding construction, each job and speed level in $x_{i,R}$ is inserted into $x_{i,D}$. As the problem is multi-objective one, the dominance rule $(\succ)$ is used when two solutions are compared. Note that, partial solutions are assessed based on the partial dominance rule. Finally, a complete solution is obtained by selecting the non-dominated solution among n solutions after the last removed job and speed level is inserted for n positions. Finally, the insertion of the local search in Fig. 4 is again applied to the complete solution obtained after construction. If the new solution x^* dominates the individual x_i, it is replaced by x^* and the archive set ϕ is updated. In addition, uniform crossover and mutation applied to the population as in the EE-VBIH algorithm. The EE_IG and EE_IGALL algorithms are given in Fig. 5 and 6. Again, note that the local search application to partial solutions which differentiates IG$_{ALL}$ from IG is illustrated in bold in the pseudocode as presented in Fig. 6.

Set $dS = 4; NP = 100$
Construct initial population with size NP
Compute delay matrix d_{ij}
While $(NotTermination)$ *do*
  for $i = 1$ *to NP do*
    $x_{i,R}(\pi_{i,R}, l_{i,R}) = Destruction(x_i)$
    $x_{i,R}(l_{i,R}) = rand()\%3$
    $x^*(\pi^*, l^*) = Construction(x_{i,R})$
    $x^*(\pi^*, l^*) = LocalSearch(x^*(\pi^*, l^*))$
    if $(x^* \succ x_i)$ *then do*
      $x_i = x^*$,
      Update the archive ϕ *with* x^*
    endif
  endfor
  for $i = 1$ *to NP do*
    $x^*(\pi^*, l^*) = \text{Crossover}(l_i, l_k)$
    if $(x^* \succ x_i)$ *then do*
      $x_i = x^*$,
      update the archive ϕ *with* x^*
    endif
  endfor
  for $i = 1$ *to NP do*
    $x_i(\pi_i, l_i) = Mutation\ (v_i)$
  endfor
endwhile
return ϕ
endprocedure

Fig. 5. EE-IG Algorithm

$Set\ dS = 4; NP = 100$
$Construct\ initial\ population\ with\ size\ NP$
$Compute\ delay\ matrix\ d_{ij}$
$While\ (NotTermination)\ do$
  $for\ i = 1\ to\ NP\ do$
    $x_{i,R}(\pi_{i,R}, l_{i,R}) = Destruction(x_i)$
    $x_{i,R}(l_{i,R}) = rand()\%3$
    $\boldsymbol{x_{i,D}(\pi_{i,D}, l_{i,D}) = PartialLocalSearch(x_{i,D})}$
    $x^*(\pi^*, l^*) = Construction(x_{i,R})$
    $x^*(\pi^*, l^*) = LocalSearch(x^*(\pi^*, l^*))$
    $if\ (x^* \succ x_i)\ then\ do$
      $x_i = x^*,$
      $Update\ the\ archive\ \phi\ with\ x^*$
    $endif$
  $endfor$
  $for\ i = 1\ to\ NP\ do$
    $x^*(\pi^*, l^*) = \text{Crossover}(l_i, l_k)$
    $if\ (x^* \succ x_i)\ then\ do$
      $x_i = x^*,$
      $update\ the\ archive\ \phi\ with\ x^*$
    $endif$
  $endfor$
  $for\ i = 1\ to\ NP\ do$
    $x_i(\pi_i, l_i) = Mutation\ (v_i)$
  $endfor$
$endwhile$
$return\ \phi$
$endprocedure$

Fig. 6. EE-IG$_{ALL}$ Algorithm

D. The Archive Set

In the EE-VBIH, EE-IG, and EE-IG$_{ALL}$ algorithms, we use an archive set ϕ to store non-dominated solutions. When a new non-dominated solution is obtained, it is added to the archive set ϕ and any member dominated by the new non-dominated solution is removed.

IV. Computational Results

To evaluate the performance of the proposed algorithms, the benchmark suite of Taillard [37] is employed, where it originally includes 12 groups of problems (10 instances in each group) with changing sizes from 20 jobs and 5 machines, to 500 jobs and 20 machines. Since solving these instances are computationally hard, 30 small instances are generated where it contains 5x5, 5x10, and 5x20 set of instances. These instances are generated by cropping the first 5 jobs of 20x5, 20x10, and 20x20 set of instances. The energy-related parameters used in TEC calculation are taken as L= {1 (fast), 2 (normal), 3 (slow)}, v_l={1.2,1,0.8}, λ_l={1.5,1,0.6}, φ_j=0.05, and τ_j=60 kW, which are taken from [12]. The augmented ε-constraint method [38] is employed on the mixed-integer linear programming model where the epsilon value is reserved as 10^{-2}. Then, the small instances are run in IBM ILOG CPLEX Optimization Studio (version 12.8) on a Core i7, 2.60 GHz, 8 GB RAM computer. The all energy-efficient algorithms were coded in C++ on Microsoft Visual Studio 2013, and all instances were solved on a Core i5, 3.20 GHz, 8 GB RAM computer. 30 replications were made for each instance. In each replication, the algorithms were run for $25nm$ milliseconds for small instances and $50nm$ milliseconds for large instances. To show the conflict between the total energy consumption and the makespan, an example of Pareto optimal set is represented by addressing the 5x5_01 instance in Fig. 7.

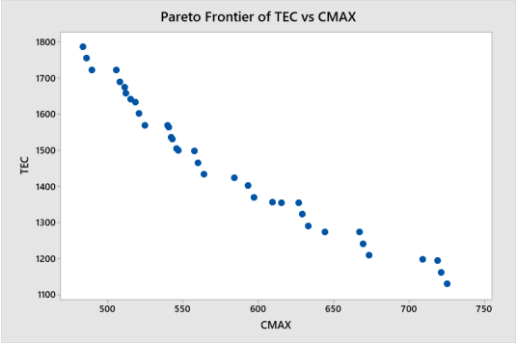

Fig. 7. The Pareto Optimal Set of 5x5_01 Instance

We have used the following performance metrics to evaluate the solution quality of the EE-VBIH, EE-IG, and EE-IG$_{ALL}$ algorithms comparing with the optimal solutions obtained by the augmented ε-constraint method.

- **Ratio of the Pareto-optimal solutions found:**

  $R_{p_Z} = |Z \cap P|/|P|,$

- **Inverted generational distance** [39]**:**

  $IGD_Z = \sum_{v \in P} d(v, Z)/|P|,$ where the minimum Euclidean distance between two solutions is denoted as $d(v, Z)$

- **Distribution Spacing** [40]**:**

  $DS_Z = \left[\frac{1}{|Z|}\sum_{i \in Z}(d_i - \bar{d})^2\right]^{1/2}/\bar{d},$ where $\bar{d} = \sum_{i \in Z} d_i/|Z|$ and d_i indicates the minimum Euclidean distance between solution i and its closest neighbor in Z. The solutions in Z are said to be uniformly dispersed whenever the low spacing values are being encountered.

It is a fact that Z relates to the non-dominated solution set of the heuristic algorithms and therewithal K, L, and M are redefined for the non-dominated solution set of the EE-VBIH, EE-IG and EE-IG$_{ALL}$ algorithms, respectively, to differentiate the heuristic algorithms. The performance metrics of comparison for all algorithms on small instances are reported in Table II.

TABLE II. Computational Results of Algorithms on Small Instances in Terms of Ratio of Pareto Optimal Solutions Found, Inverted General Distance and Distibution Spacing

Instance Set	R_p			IGD			DS		
	K	L	M	K	L	M	K	L	M
5x5	1.000			0.000			0.623		
5x10	1.000			0.000			0.817		
5x20	1.000			0.000			0.835		
Average	**1.000**			**0.000**			**0.758**		

As seen from Table II, all the algorithms reach the optimal solutions found by the mathematical model at a 100% level. Also, the 0 IGD values mean that exactly the same Pareto

optimal solution is obtained by all metaheuristics. Furthermore, the low spacing value (0.758) indicates that the points in the Pareto optimal set are uniformly dispersed Then, due to the computational complexity of the multi-objective problem, we used only the first 60 instances of Taillard's instances [37] as large instances, namely, 20x5, 20x10, 20x20, 50x5, 50x10 and 50x20. We used the distribution spacing performance metric for large instances, as well, whereas the following performance measures are additively considered to evaluate the solution quality of the EE-VBIH, EE-IG, and EE-IG_{ALL} algorithms. Once again, Z refers to the non-dominated solution set of the heuristic algorithms (EE-VBIH, EE-IG, or EE-IG_{ALL}).

- **Cardinality:**
  Number of non-dominated solutions found: $R_p = |Z|$.
- **Coverage of Two Sets** [35] **:**
  $C(Z,T) = |\{t \in T;\ \exists z \in Z: z \succcurlyeq t\}|/|T|$, where $C(Z,T)$ equals 1 if some solutions of Z weakly dominate all solutions of T.

Table III summarizes the results for the heuristic algorithms on large instances (20x5, 20x10, 20x20, 50x5, 50x10, and 50x20) in terms of cardinality, distribution spacing, and coverage metrics. As shown in Table III, the EE-VBIH algorithm outperforms on the EE-IG and EE-IG_{ALL} algorithms in terms of the cardinality of the non-dominated solutions. EE-VBIH, EE-IG, and EE-IG_{ALL} algorithms can find 83.00, 58.10, 63.10 number of non-dominated solutions on average overall instances, respectively. Hence, the EE-VBIH algorithm can find 1.42 times more non-dominated solutions than EE-IG and 1.31 times more non-dominated solutions than EE-IGALL. Regarding the distribution spacing metric, the EE-VBIH again surpasses both remaining algorithms. The distribution spacing values of the EE-VBIH, EE-IG, and EE-IGALL algorithms are 1.067, 1.223, and 1.621, respectively. These values indicate that all algorithms are good at finding the uniformly distributed non-dominated set of solutions. However, the solutions in EE-VBIH are distributed more uniformly than the solutions of the EE-IG and EE-IG_{ALL} algorithm due to lower DS value. In terms of coverage measure, 58.4% of the solutions of EE-IG are weakly dominated by some solutions of EE-VBIH; 64.2% of the solutions of EE-IGALL are weakly dominated by some solutions of EE-VBIH, as seen from Table III. Hence, the set of solutions obtained by the EE-VBIH algorithm can dominate 58.4% and 64.2% of the solutions obtained by EE-IG and EE-IGALL, respectively. To conclude, the EE-VBIH algorithm outperforms on the other two algorithms concerning cardinality, dispersion, and coverage metrics.

TABLE III. COMPUTATIONAL RESULTS OF ALGORITHMS ON LARGER INSTANCES IN TERMS OF CARDINALITY, DISTIBUTION SPACING AND COVERAGE

Instance	|K|	|L|	|M|	DS_K	DS_L	DS_M	C(K,L)	C(L,K)	C(K,M)	C(M,K)	C(L,M)	C(M,L)
20x5	82.30	102.30	76.80	0.785	0.859	0.846	0.122	0.851	0.272	0.676	0.850	0.211
20x10	62.40	65.20	63.60	0.873	0.834	0.855	0.401	0.533	0.478	0.512	0.577	0.432
20x20	54.60	48.80	54.40	0.876	1.048	1.050	0.505	0.458	0.462	0.514	0.492	0.503
50x5	125.80	60.70	81.20	1.324	1.067	2.067	0.721	0.154	0.827	0.096	0.612	0.313
50x10	99.20	45.80	67.00	1.554	1.595	2.580	0.840	0.080	0.869	0.099	0.330	0.517
50x20	73.90	26.70	35.50	0.992	1.932	2.325	0.916	0.062	0.946	0.054	0.272	0.379
Average	**83.00**	**58.30**	**63.10**	**1.067**	**1.223**	**1.621**	**0.584**	**0.356**	**0.642**	**0.325**	**0.522**	**0.393**

V. CONCLUSION

In this paper, we propose an energy-efficient NWFSP in order to investigate a trade-off between the makespan and the total energy consumption. The purpose of the energy-efficient NWFSP is to obtain Pareto solution sets to minimize the make-span and total energy consumption. Unlike the traditional NWFSP, we introduce different speed levels for each job on machines, and processing times of jobs can be different according to assigned speed levels. Therefore, a mixed-integer linear programming model formulation is proposed and the heuristic formulation of NWFSP is modified by introducing a speed scaling strategy in order to approximate non-dominated solution sets. Due to the NP-hardness nature of the problem, three metaheuristics are proposed (EE-VBIH, EE-IG, and EE-IG_{ALL}). Extensive computational analyses on Taillard's benchmark suite show that proposed algorithms are very effective for approximating Pareto solution sets. Among three algorithms, EE-VBIH outperforms than the EE-IG and EE-IG_{ALL} in terms of both cardinality and quality. For future work, various multi-objective algorithms such as a multi-objective genetic algorithm with non-dominating sorting and crowded distance can be developed to compare and assess the performance of both algorithms. Several objective functions can be analyzed within the energy-efficiency scope or this scope can be implemented on other scheduling problems such as blocking-flowshop or no-idle flowshop scheduling problems.

REFERENCES

[1] T. Aldowaisan and A. Allahverdi, "New heuristics for m-machine no-wait flowshop to minimize total completion time," *Omega*, vol. 32, no. 5, pp. 345–352, Oct. 2004.

[2] S. U. Sapkal and D. Laha, "A heuristic for no-wait flow shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 68, no. 5–8, pp. 1327–1338, Sep. 2013.

[3] H. Röck and Hans, "The Three-Machine No-Wait Flow Shop is NP-Complete," *J. ACM*, vol. 31, no. 2, pp. 336–345, Mar. 1984.

[4] S.-W. Lin and K.-C. Ying, "Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics," *Omega*, vol. 64, pp. 115–125, Oct. 2016.

[5] K. Fang, N. Uhan, F. Zhao, and J. W. Sutherland, "A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction," *J. Manuf. Syst.*, vol. 30, no. 4, pp. 234–240, Oct. 2011.

[6] C. Gahm, F. Denz, M. Dirr, and A. Tuma, "Energy-efficient scheduling in manufacturing companies: A review and research

framework," *Eur. J. Oper. Res.*, vol. 248, no. 3, pp. 744–757, Feb. 2016.

[7] G. Mouzon, M. B. Yildirim, and J. Twomey, "Operational methods for minimization of energy consumption of manufacturing equipment," *Int. J. Prod. Res.*, vol. 45, no. 18–19, pp. 4247–4271, Sep. 2007.

[8] G. Mouzon and M. B. Yildirim, "A framework to minimise total energy consumption and total tardiness on a single machine," *Int. J. Sustain. Eng.*, vol. 1, no. 2, pp. 105–116, Jun. 2008.

[9] M. Dai, D. Tang, A. Giret Boggino, M. Salido Gregorio, and W. Li, "Energy-efficient scheduling for a flexible flow-shop using improved genetic-simulated annealing algorithm," *Robot. Comput. Integr. Manuf.*, no. 29, pp. 418–429, 2013.

[10] K. Fang, N. A. Uhan, F. Zhao, and J. W. Sutherland, "Flow shop scheduling with peak power consumption constraints," *Ann. Oper. Res.*, vol. 206, no. 1, pp. 115–145, Jul. 2013.

[11] J. Y. Ding, S. Song, and C. Wu, "Carbon-efficient scheduling of flow shops by multi-objective optimization," *Eur. J. Oper. Res.*, vol. 248, no. 3, pp. 758–771, Feb. 2016.

[12] S. A. Mansouri, E. Aktas, and U. Besikci, "Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption," *Eur. J. Oper. Res.*, vol. 248, no. 3, pp. 772–788, Feb. 2016.

[13] S. A. Mansouri and E. Aktas, "Minimizing Energy consumption and makespan in a two-machine flowshop scheduling problem," *J. Oper. Res. Soc.*, vol. 67, no. 11, pp. 1382–1394, 2016.

[14] L. Yin, X. Li, C. Lu, and L. Gao, "Energy-Efficient Scheduling Problem Using an Effective Hybrid Multi-Objective Evolutionary Algorithm," *Sustainability*, vol. 8, no. 12, p. 1268, Dec. 2016.

[15] C. Lu, L. Gao, X. Li, Q. Pan, and Q. Wang, "Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm," *J. Clean. Prod.*, vol. 144, no. 144, pp. 228–238, Feb. 2017.

[16] R. Zhang and R. Chiong, "Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption," *J. Clean. Prod.*, vol. 112, pp. 3361–3375, 2016.

[17] J. Yan, L. Li, F. Zhao, F. Zhang, and Q. Zhao, "A multi-level optimization approach for energy-efficient flexible flow shop scheduling," *J. Clean. Prod.*, vol. 137, pp. 1543–1552, Nov. 2016.

[18] D. Tang, M. Dai, M. A. Salido, and A. Giret, "Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization," *Comput. Ind.*, vol. 81, pp. 82–95, Sep. 2016.

[19] M. A. Salido, J. Escamilla, A. Giret, and F. Barber, "A genetic algorithm for energy-efficiency in job-shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 85, no. 5–8, pp. 1303–1314, Jul. 2016.

[20] H. Öztop, M. Fatih Tasgetiren, D. Türsel Eliiyi, and Q.-K. Pan, "Green Permutation Flowshop Scheduling: A Trade-off- Between Energy Consumption and Total Flow Time," in *Huang DS., Gromiha M., Han K., Hussain A. (eds) Intelligent Computing Methodologies. ICIC 2018. Lecture Notes in Computer Science, vol 10956.*, Springer, Cham, 2018, pp. 753–759.

[21] M. F. Tasgetiren, H. Öztop, U. Eliiyi, D. T. Eliiyi, and Q. K. Pan, "Energy-efficient single machine total weighted tardiness problem with sequence-dependent setup times," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 10954 LNCS, pp. 746–758.

[22] X. Wu and A. Che, "A memetic differential evolution algorithm for energy-efficient parallel machine scheduling," *Omega (United Kingdom)*, vol. 82, pp. 155–165, Jan. 2019.

[23] D. Yüksel, M. F. Tasgetiren, L. Kandiller, and L. Gao, "An Energy-Efficient Bi-Objective No-Wait Permutation Flowshop Scheduling Problem To Minimize Total Tardiness And Total Energy Consumption," *Comput. Ind. Eng.*, vol. in Press, 2020.

[24] H. Öztop, M. F. Tasgetiren, L. Kandiller, D. T. Eliiyi, and L. Gao, "Ensemble of metaheuristics for energy-efficient hybrid flowshops: Makespan versus total energy consumption," *Swarm Evol. Comput.*, vol. 54, p. 100660, May 2020.

[25] A. S. Manne, "On the Job-Shop Scheduling Problem," *Oper. Res.*, vol. 8, no. 2, pp. 219–223, Apr. 1960.

[26] D. A. Wismer, "Solution of the Flowshop-Scheduling Problem with No Intermediate Queues," *Oper. Res.*, vol. 20, no. 3, pp. 689–697, Jun. 1972.

[27] Q.-K. Pan, M. F. Tasgetiren, and Y.-C. Liang, "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem," *Comput. Oper. Res.*, vol. 35, no. 9, pp. 2807–2839, Sep. 2008.

[28] M. F. Tasgetiren, U. Eliiyi, H. Öztop, D. Kizilay, and Q. K. Pan, "An energy-efficient single machine scheduling with release dates and sequence-dependent setup times," in *GECCO 2018 Companion - Proceedings of the 2018 Genetic and Evolutionary Computation Conference Companion*, 2018, pp. 145–146.

[29] H. Öztop, M. F. Tasgetiren, D. T. Eliiyi, and Q. K. Pan, "Green Permutation Flowshop Scheduling: A Trade- off- Between Energy Consumption and Total Flow Time," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 10956 LNAI, pp. 753–759.

[30] S. Farahmand Rad, R. Ruiz, and N. Boroojerdian, "New High Performing Heuristics for Minimizing Makespan in Permutation Flowshops," *Omega*, pp. 37(2):331–45, 2009.

[31] M. Nawaz, E. E. Enscore, and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, Jan. 1983.

[32] M. F. Tasgetiren, Q. K. Pan, Y. Ozturkoglu, and A. H. L. Chen, "A memetic algorithm with a variable block insertion heuristic for single machine total weighted tardiness problem with sequence dependent setup times," in *2016 IEEE Congress on Evolutionary Computation, CEC 2016*, 2016, pp. 2911–2918.

[33] M. Tasgetiren, Q.-K. Pan, D. Kizilay, and K. Gao, "A Variable Block Insertion Heuristic for the Blocking Flowshop Scheduling Problem with Total Flowtime Criterion," *Algorithms*, vol. 9, no. 4, p. 71, Oct. 2016.

[34] M. F. Tasgetiren, Q. K. Pan, D. Kizilay, and M. C. Velez-Gallego, "A variable block insertion heuristic for permutation flowshops with makespan criterion," in *2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings*, 2017, pp. 726–733.

[35] R. Ruben and T. Stützle, *A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem*. Ithaca: Shaker., 2007.

[36] J. Dubois-Lacoste, F. Pagnozzi, and T. Stützle, "An iterated greedy algorithm with optimization of partial solutions for the makespan permutation flowshop problem," *Comput. Oper. Res.*, vol. 81, pp. 160–166, May 2017.

[37] E. Taillard, "Benchmarks for basic scheduling problems," *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 278–285, Jan. 1993.

[38] G. Mavrotas, "Effective implementation of the ε-constraint method in Multi-Objective Mathematical Programming problems," *Appl. Math. Comput.*, vol. 213, no. 2, pp. 455–465, Jul. 2009.

[39] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Vol. 5, pp. 79-104)*, 2nd Ed. New York: Springer, 2002.

[40] K. C. Tan, C. K. Goh, Y. J. Yang, and T. H. Lee, *Evolving Better Population Distribution and Exploration in Evolutionary Multi-Objective Optimization*, vol. 171, no. 2. North-Holland, 2006.