Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 6756588, 16 pages
https://doi.org/10.1155/2021/6756588

Research Article

Hyperheuristic Based Migrating Birds Optimization Algorithm for a Fairness Oriented Shift Scheduling Problem

Gözde Alp and Ali Fuat Alkaya

Department of Computer Engineering, Engineering Faculty, Marmara University, 34722 Istanbul, Turkey

Correspondence should be addressed to Gözde Alp; gozde.alp@marmara.edu.tr

Received 14 September 2021; Revised 12 October 2021; Accepted 18 October 2021; Published 2 November 2021

Academic Editor: Guoqiang Wang

The purpose of this paper is twofold. First, it introduces a new hybrid computational intelligence algorithm to the optimization community. This novel hybrid algorithm has hyperheuristic (HH) neighborhood search movements embedded into a recently introduced migrating birds optimization (MBO) algorithm. Therefore, it is called HHMBO. Second, it gives the necessary mathematical model for a shift scheduling problem of a manufacturing company defined by including the fairness perspective, which is typically ignored especially in manufacturing industry. Therefore, we call this complex optimization problem fairness oriented integrated shift scheduling problem (FOSSP). HHMBO is applied on FOSSP and is compared with the well-known simulated annealing, hyperheuristics, and classical MBO algorithms through extended computational experiments on several synthetic datasets. Experiments demonstrate that the new hybrid computational intelligence algorithm is promising especially for large sized instances of the specific problem defined here. HHMBO has a high exploration capability and is a promising technique for all optimization problems. To justify this assertion, we applied HHMBO to the well-known quadratic assignment problem (QAP) instances from the QAPLIB. HHMBO was up to 14.6% better than MBO on converging to the best known solutions for QAP benchmark instances with different densities. We believe that the novel hybrid method and the fairness oriented model presented in this study will give new insights to the decision-makers in the industry as well as to the researchers from several disciplines.

1. Introduction

Workforce scheduling has been a subject of continued research and commercial interest in several disciplines due to its important practical applications within the context of intelligent systems. Workforce scheduling is a concept that embraces a variety of scheduling problems, also referred to as manpower scheduling and personnel scheduling in the literature. Constructing efficient and equitable schedules is a challenging issue requiring time and labor cost for the companies with high number of employees.

The real world problem we tackled aims to provide fair personnel work schedules for a large-scale manufacturing company that works 24 hours and seven days a week. Personnel requirements for each day and shift differ and are updated periodically. Employee requirement changes every four weeks, regularly. On the other hand, there is a high rate of employee circulation. Large numbers of new employees are recruited and large numbers of employees leave at the end of each period. Therefore, possible maximum level of fairness within each planning period is sought. There are also some essential legal regulations for employee schedules which make the problem much more complicated. The problem is described in detail in Section 3.

The presented problem has a sophisticated solution space. We have envisioned that exploring the solution space by more than one neighborhood search method could contribute to finding promising results. Hyperheuristics is a widely known way of applying multiple heuristic techniques. On the other hand, recently proposed migrating birds optimization algorithm is receiving growing attention from the researchers, due to the ability of providing good quality solutions over a wide range of instances of a problem. As a result, we wanted to integrate the exploitation capability of

MBO with HH's exploration capability for solving the problem. Consequently, we introduce a novel technique that has hyperheuristic movements embedded in the migrating birds optimization algorithm (HHMBO). To the best of our knowledge, there has been no investigation about hybridizing hyperheuristics with MBO in the literature. HHMBO is compared with the well-known algorithms, and it is shown that the proposed hybrid algorithm is promising especially for large sized instances of FOSSP.

HHMBO obtains very successful results on the FOSSP model. Hence, we believe HHMBO is also promising for the other combinatorial optimization problems. Therefore, we aimed to measure its performance on a widely tackled problem's benchmark instances. Based on this idea, we have implemented HHMBO on the quadratic assignment problem (QAP). As a result, HHMBO is again found to be very successful on converging to the best known solutions for QAP instances with different densities. We should note that the data as well as the source codes of the programming framework can be found at http://mimoza.marmara.edu.tr/falkaya/research.htm.

The contribution of this study is threefold: (i) defining and validating the mathematical model of the fairness oriented shift scheduling problem (FOSSP), (ii) proposing a new hybrid metaheuristic algorithm that is composed of hyperheuristic movements embedded in migrating birds optimization (HHMBO), and (iii) showing the superiority of HHMBO through computational experiments conducted on both FOSSP dataset and well-known QAP benchmark instances.

The remainder of the manuscript is organized as follows: Literature review is presented in Section 2, and problem description details are presented in Section 3. Proposed solution techniques are described in detail in Section 4. Experimental setup is demonstrated in Section 5. Results are presented and investigated in Section 6. Section 7 concludes the paper and comprises suggestions for further research.

2. Literature Review

Many scientific studies have been examined in order to locate the problem that we have dealt with. Ernst et al. [1] offered a guide study that consisted of an annotated bibliography of personnel scheduling. There are also many review studies that extensively examine the workforce scheduling literature [2–6]. Brucker et al. [7] categorized the shift scheduling problems. Baker classified workforce scheduling problems into three types [8]: shift scheduling problem, days-off scheduling problem, and tour scheduling problem. Shift scheduling involves selecting a set of the most suitable shifts from a (large) pool of candidate shifts on a single day by satisfying employee requirement in each time slot. On the other hand, the main concern in days-off scheduling is to determine the off-work days for each worker over the rostering horizon rather than to assign the worker particular shifts on working days. In contrast to days-off scheduling, tour scheduling chooses off days for the workers and decides shift types for each of their working days over the planning horizon. Mathematical model of tour scheduling problem is summarized and problem classification is presented by Alfares [9]. Erhard et al. [4] presented the first review study that focuses on quantitative methods for physician scheduling literature. They indicated in their study that it is more common to include fairness aspects for physician scheduling domain than to ignore them.

Many different workforce scheduling problem models are presented up to now. A new mathematical formulation for mine shift scheduling is presented by Seifi et al. [10]. Brunner and Stolletz [11] prepared a schedule for the employees in check-in counters at airport. Atlason et al. [12] presented a shift assignment study in which planning horizon is divided into short periods. Morris and Showalter [13] presented a rewarding study that clarifies the usage of set covering formulation for shift scheduling, days-off scheduling, and tour scheduling problems ([14–16]). Alfares [17] proposed a specific type of days-off scheduling, solved by proposed solution technique. Altner et al. [18] presented a days-off scheduling formulation. Lau [19] introduced a special tour scheduling problem.

The model we offer in this study associates tour scheduling problem and employee assignment problem and incarnates them as a single problem by overseeing the fairness issue. We call this integrated problem fairness oriented shift scheduling problem (FOSSP), in which schedules are constructed for employees during the planning period where the objective is providing fairness among all workers. The problem we are tackling is undertaken by solving the tour scheduling and employee assignment problems consecutively in [20]. However, obtaining solutions to subproblems gives suboptimal results. Therefore, an integrated solution approach is necessary. The mathematical model of FOSSP is built from scratch and then the model is implemented with a solver. However, the solver finds the optimum result but with an exponential growth in run time, due to the NP-Hard nature of the problems. Therefore, heuristics and/or metaheuristic approaches that provide reasonable solutions for the problem are needed.

Many metaheuristic based solution techniques have been applied for solving the workforce scheduling problems so far. They include artificial bee colony algorithm [21], particle swarm optimization [22], migrating birds optimization [23], genetic algorithm (GA) [24], an adaptive multiple crossover GA [25], and a modified differential evolution (DE) algorithm ([26–29]). A tabu-search hyperheuristics solution for nurse scheduling problem is presented in Burke et al.'s work [30]. Pan et al. [31] proposed a hybrid heuristic, combining tabu search and large neighborhood search techniques. Hernández-Leandro et al. [32] presented a Lagrangian relaxation and a metaheuristic for the multiactivity shift scheduling problem. A hybrid discrete water wave optimization algorithm is presented for scheduling problems [33].

Combinatorial optimization problems have a sophisticated solution space and a single improvement technique is insufficient to obtain promising results. More than one neighborhood search method with different characteristics may be promising to obtain more efficient exploration in the solution space. One widely known way of applying multiple heuristic techniques is hyperheuristics. Hyperheuristics

(HH) is a metaheuristic technique formally defined as a heuristic to select and manage a set of low level heuristics. Many HH variations for optimization and scheduling problems have been defined up to now, such as tensor based hyperheuristic for nurse scheduling [34], column based hyperheuristic for bus driver scheduling problem [35], tabu-search hyperheuristics [30], and a simulated annealing hyperheuristic [36]. Hyperheuristics are also integrated with swarm based computational intelligence techniques such as hybrid particle swarm optimization [37], ant colony optimization based hyperheuristic [38], and hyperheuristic based artificial bee colony algorithm [39]. On the other hand, Duman et al. [40] recently proposed a new metaheuristic algorithm named migrating birds optimization (MBO). They applied their algorithm to quadratic assignment problems (QAP) and proved its efficiency.

3. Problem Description

In this section, we try to define and clarify the proposed and tackled problem. Firstly, the properties and limitations of the problem are explained. Then, cost measurement method is described. Lastly, the mathematical model of FOSSP is presented. The terms time period, time slot, and working slot are used interchangeably throughout this manuscript.

3.1. Properties and Limitations of the Problem. The main motivation behind defining this new problem is obtaining a fair distribution of working slots (time periods) among fixed number of employees during the specified planning period. Production factory operates incessantly 24 hours a day, seven days a week. There are totally three shifts in each day; those are day shift, evening shift, and night shift. Employee requirement list contains weekly employee demand for each day of the week and each shift of a day. The planning horizon which is specified as four weeks in our case may consist of several weeks. Employee requirement list is updated by management department at the end of each period, and same the requirement list is valid for all weeks in planning horizon. Multiweek employee schedules are prepared supplying personnel demand. That is, the output schedule must satisfy the number of workers needed for each day and shift, besides ensuring a fair distribution between employees along the given weeks.

Additionally, there are some essential legal regulations for employee schedules. Some of those rules may be listed as follows; if an employee has night shift on Sunday, (s)he cannot be assigned to day shift on the following Monday; if an employee has weekend off, he/she cannot be off on adjacent Monday and an employee cannot work more than six consecutive days. These additional legal constraints increase the complexity of the problem. Under these constraints, the aim is to provide a fair schedule distribution among employees throughout the planning period. Employees are assigned to a specific stationary shift type during one week and each employee has two off days. However, for each worker, shift types and off days may change from one week to another.

3.2. Determination of Schema Costs. A tour or a schema may be defined by a seven-lettered string containing D, E, N, or X which mean day shift, evening shift, night shift, and off day, respectively, and each letter corresponds to the days of the week starting from Monday up to Sunday. An example schema string is "XXEEEEE" which means Monday and Tuesday are off and the week continues with five successive evening shifts. Thus, there occur exactly 63 different schemata with three shift types and two days off rules.

Three shifts in a day (namely, day, evening, and night) and seven days of the week make a total of 21 time slots, starting from Monday day shift and ending with Sunday night shift. In the factory, day shift is between 07:00 and 15:00, evening shift is between 15:00 and 23:00, and night shift is between 23:00 and 07:00. Each employee is assigned to one of those shifts each week; additionally each employee has two off days during a week.

Taking individual employee preferences into account while modeling the problem was not appropriate, as it would make the already complex problem unsolvable. Hereby, the preferability of time zones has been determined by a survey conducted with the employees. We had classified the off days according to their desirability. We have identified five off day types: (i) weekend off, (ii) Friday-Saturday off, (iii) Sunday-weekday off, (iv) consecutive weekday off, and (v) separate weekday off. For determining the quality of a schema, we assigned numerical values to off day types and shift types. Based on the survey results obtained, enumerations for shift types are determined as one, two, and three for day, evening, and night shifts, respectively. Similarly, off day types, weekend off, Friday-Saturday off, Sunday-weekday off, consecutive weekday off, and separate weekday off, are enumerated from one to five, respectively (Table 1).

Each schema (out of 63 schemata) has a predetermined cost, obtained by the multiplication of its off day type enumeration and shift type enumeration. An illustrative cost calculation example for some of the schemata is given in Table 2. One can easily recognize that more preferable schemata have lower cost when the schema cost calculation is examined. So, the objective of the problem turns out to be minimizing the cost. As an example, "DDDDDXX" is the most preferred and lowest cost tour.

A small sample requirement input for four employees is illustrated in Table 3. Schema ID is schema identifier; a schema may be identified by a number between one and 63. FOSSP scheduling decides schema usage quantities by caring for the employee requirements in each time period. The requirements for Monday day shift and evening shift are zero and no employees are assigned to day shift on Monday. The requirement for Monday night shift is one and one employee will be assigned to night shift on Monday with schema ID 63. FOSSP output satisfies employee requirements by selecting the most suitable schema with the required usage quantity.

Since the requirements are the same for all weeks during the planning period (four weeks in our case), the same usage quantity of each schema must be used for each week. The sample output of FOSSP is given in Table 4; schema ID 1 is assigned to employee one and employee

Table 1: Shift type and off day type enumeration.

Type	Cost
Day shift	1
Evening shift	2
Night shift	3
Weekend off	1
Friday-Saturday off	2
Sunday-weekday off	3
Consecutive weekday off	4
Separate weekday off	5

Table 2: Some examples regarding schema cost computation.

Schema	String	c (s)
21	DDDDDXX	1 (1 ∗ 1)
42	EEEEEXX	2 (1 ∗ 2)
63	NNNNNXX	3 (1 ∗ 3)
8	DXDXDDD	5 (5 ∗ 1)
29	EXEXEEE	10 (5 ∗ 2)
50	NXNXNNN	15 (5 ∗ 3)

Table 3: Sample requirement input.

Time slot ID	Definition	Requirement
1	Monday day shift	0
2	Monday evening shift	0
3	Monday night shift	1
4	Tuesday day shift	1
5	Tuesday evening shift	1
6	Tuesday night shift	1
7	Wednesday day shift	1
8	Wednesday evening shift	1
9	Wednesday night shift	1
10	Thursday day shift	2
11	Thursday evening shift	1
12	Thursday night shift	1
13	Friday day shift	2
14	Friday evening shift	0
15	Friday night shift	1
16	Saturday day shift	2
17	Saturday evening shift	1
18	Saturday night shift	0
19	Sunday day shift	2
20	Sunday evening shift	1
21	Sunday night shift	0

three in week one, two other employees in week two, and so on. Schema ID 2 is not assigned to anyone. Schema ID 25 is assigned to one employee and schema ID 63 is assigned to one employee in each week. The problem assigns the same schemas with same quantities to different employees for given number of weeks.

3.3. Fairness Oriented Shift Scheduling Problem Model. The scheduling problem we handled in this study was modeled as two consecutive problems in previous studies. Tour scheduling problem and employee assignment problem [20] are jointly sufficient to define this particular scheduling issue. Note that tour scheduling problem reduces to set covering problem [13] which is NP-Hard. Similarly employee assignment problem reduces to the generalized assignment problem [14] which is also shown to be NP-Hard. Since FOSSP corresponds to the combination of both problems, it is also NP-Hard. Solving the problem as two subproblems offers suboptimal solutions; however, whole parts of the issue are significant and we need to find a solution for an integrated single problem. FOSSP is the first sample that put those problems together and solves them as a unique problem in literature, to the best of our knowledge. In the following, we provide the necessary notation and then the mathematical formulation of FOSSP.

Table 4: Schema and employee assignments for four weeks.

Employee ID	Week 1	Week 2	Week 3	Week 4
1	1	1	25	63
2	63	25	1	1
3	1	63	1	25
4	25	1	63	1

x_{ws} is the number of times schema s is used for each week w,

S is the set of schemas (remember that a schema is a seven-lettered string denoting the shift types and off days; each schema is denoted by a number from 1 to 63),

T represents the time slots of week (there are totally 21 time slots in a week; each day contains three shifts (planning time slot), and seven days comprise 21 time slots similarly; each time slot is denoted by a number from 1 to 21),

E is the set of employees,

W is the set of weeks,

c_s is the cost of schema s,

r_t is the requirement (number of required workers) at time slot t,

$$A_{ews} = \begin{cases} 1, & \text{if employee } e \text{ assigned to schema } s \text{ in week } w, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

A_{ews} is the assignment variable and shows the assignments of each employee e for each week w and for each schema s. A is equal to one if an employee e is assigned to schema s in week w, and it is equal to zero otherwise,

$$b_{st} = \begin{cases} 1, & \text{if schema } s \text{ covers times lot } t, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

b_{st} indicates the time slot coverage of each schema. If schema s comprises time slot t then b is equal to one and it is equal to zero otherwise. $M = \sum_s \sum_w \sum_e A_{ews} * c_s/|E|$ is the average cost value for all employees, calculated by total of schema assignment number and schema cost multiplication, divided by the number of employees,

$N_e = \sum_w \sum_s A_{ews} * c * c_s$ is the all weeks' total cost value. This value is calculated separately for each employee. If employee e is assigned to schema s during week w, A_{ews} variable is equal to one and schema cost is added to the total. The sum of all weeks' schema costs is calculated and divided by total number of planning week number W. So average cost value is calculated for each employee,

$D_e = (M - N_e)^2$ is the sum of squared distances of each employee individual total N_e and the general mean total M. The sum of square deviations which measures the pairwise differences is one of the inequality indices widely applied in the literature [41, 42],

$$\text{minimize} \sum_e D_e, \tag{3}$$

$$\sum_s A_{ews} = 1, \quad \text{for } \forall e, w, \tag{4}$$

$$\sum_e A_{ews} = x_{ws}, \quad \text{for } \forall w, s, \tag{5}$$

$$\sum_s x_{ws} b_{st} \geq r_t, \quad \text{for } \forall w, t. \tag{6}$$

The objective is minimizing the squared distances of each employee so that maximum fairness is sought. Constraint (4) means that each employee is assigned to only one schema in each week. Constraint (5) ensures that the total number of employees assigned to schema s for each week is equal to x_{ws} (number of total employee assignments in each week and for each schema). Constraint (6) ensures that requirement in each time slot t (i.e., the required number of employees) is satisfied.

$$A_{ews} * b_{s(21)} + A_{e(w+1)(s_1)} * b_{(s_1)1} \leq 1, \quad \text{for } \forall e, w, s, s1, \tag{7}$$

$$\sum_{k=16}^{21} A_{ews} * b_{sk} + \sum_{k=1}^{3} A_{e(w+1)(s1)} * b_{(s_1)k} \geq 1, \quad \text{for } \forall e, w, s, s1, \tag{8}$$

$$\sum_{k=3q+1}^{21} A_{ews} * b_{sk} + \sum_{k=1}^{3q} A_{e(w+1)(s_1)} * b_{(s_1)k} \leq 6, \quad \text{for } q = 0 \ldots 6 \text{ for } \forall e, w, s, s1, \tag{9}$$

$$s, s1 \in S, \quad t, k \in T, \quad e \in E, \quad w, w+1 \in W. \tag{10}$$

Variables $w + 1$ and $s1$ are used in constraints five to seven. $w + 1$ demonstrates the following week and $s1$ corresponds to the schema assignment of the following week.

Additional constraints (equations (7)–(9)) may be explained as follows: Constraint five carries out the rule; if employee has night shift on Sunday, (s)he cannot be assigned to day shift on adjacent Monday. Time slot 21 corresponds to Sunday night shift and time slot 1 corresponds to Monday day shift (Table 3). This constraint prevents assigning any two employees to adjacent shifts for consecutive weeks w and $w + 1$. Another legal regulation is that if an employee has weekend off, he/she cannot be off on Monday (the first day of the next week), implemented by constraint six. Time slot numbers from 16 to 21 embrace weekend period, while time slots 1 to 3 correspond to three shifts of Monday. Total assignment for those three days must be greater than or equal to one. Hence, if someone had weekend off, that person will work on Monday of the next week ($w + 1$), or in other words an employee cannot be off for three consecutive days. Constraint seven enforces the rule; an employee cannot work for more than six days consecutively. If q is equal to zero, it checks only the first week; if q is equal to one, it calculates the total for six days of first week starting from Tuesday and one day (Monday) from second week and so on.

The proposed model is implemented using GAMS software. Tests are done in GAMS for small dataset and the results are demonstrated in Section 6. However, for the large-scale real-world requirements, we have implemented heuristic based solution techniques and developed a novel hybrid technique described in detail in the next section.

4. Proposed Solution Methods

In this section, the solution methods are clarified. Initial individual generation process for FOSSP, hyperheuristics, simulated annealing, migrating birds optimization algorithms, and a novel hybrid algorithm, which is one of the most important contributions of this study, are explained, respectively.

4.1. Initial Individual Generation for FOSSP. Recall from the small example that the ultimate output of FOSSP is a matrix containing schema numbers assigned to employees for each week. Therefore, a solution (also called an individual) to a problem instance will refer to a feasible output in the remainder of the manuscript. Again, recall that, due to the nature of the problem, same set of schema numbers are used in each week with different permutations. A sample FOSSP individual is illustrated in Table 5 for ten employees and four weeks. Employee number one is assigned to schema number five in week one, employee number two is assigned to schema twenty in week one, and so on.

We want to remind that requirement list is updated periodically before preparing the schedules and the requirements are the same for all weeks in planning horizon. Initial individual creation progress of FOSSP is as follows: Schemas are assigned to employees for the first week satisfying the requirements in each time interval. The schedules of subsequent weeks (week two, week three, etc.) are constructed by shuffling assigned schema values in week one.

Table 5: Sample four-week, ten-employee FOSSP individual.

Week	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10
1	5	20	30	35	40	53	58	8	18	27
2	8	18	27	30	35	40	53	58	5	20
3	8	18	27	53	58	40	30	35	5	20
4	40	53	27	30	35	8	18	58	5	20

Additional controls for meeting the requirements and controlling official rules are also taken into account.

4.2. Hyperheuristics. Hyperheuristics (HH) is a neighborhood search technique used for selecting, applying, and managing low-level heuristics and it is widely used in solving combinatorial optimization problems [43]. Pseudocode of the HH is presented in Algorithm 1. HH has two important functionalities, one of which is heuristic selection mechanism and the other is the move acceptance mechanism. Heuristic selection mechanism is responsible for selecting a low-level heuristic within the pool. Move acceptance mechanism decides whether to accept new solutions or not. In our implementation, simple random heuristic selection, random permutation heuristic selection, and adaptive searching heuristic selection strategies are used. Only Improvement and Monte Carlo acceptance strategies are exploited as acceptance strategies. Now let us explain the low-level heuristics applied for tackled problems, respectively.

4.2.1. Low-Level Heuristics for FOSSP. There are totally four low-level heuristics for FOSSP in this study. Each heuristic checks the additional constraints specified in the model.

Heuristic 1 (H1): H1 selects a random schema among the ones that are already assigned to an employee. Another random schema is selected with tournament selection with size five. Five schemas are selected randomly among 63 schemas, and one of those five schemas is chosen as second schema. Tournament selection increases the possibility of choosing a lower cost schema.

Heuristic 2 (H2): H2 randomly selects two employees for each week and swaps their schema assignment values.

Heuristic 3 (H3): H3 selects two employees randomly for each week and inserts the schema value in subsequent location to the next place in prior employee location.

Heuristic 4 (H4): H4 selects two employees randomly for each week and inverts schema assignment values between these employee locations.

4.2.2. Low-Level Heuristics for QAP. Four low-level heuristics are implemented on QAP. Those may be explained basically as follows:

Swap: Swap heuristic simply selects two points in a QAP individual and relocates their places.

Insert: Two random points are selected within an individual and the subsequent one is placed right after the first random point. All points between selected points move one step to the right.

Inverse: Two random points are selected within an individual. All assignments between the selected points are reversed.

Scramble: Two random points are selected on an individual. Assignments between the selected points are scrambled.

4.2.3. Heuristic Selection Strategies. Selecting the low-level heuristics requires a strategy. We applied three different heuristic selection strategies. These are simple random (SR), random permutation (RP), and adaptive searching (AS).

(i) SR randomly selects one of the low-level heuristics and applies it only once.

(ii) RP firstly constructs a permutation of low-level heuristics and applies each one once according to the permutation.

(iii) AS is a smart selection mechanism that we designed in this study. Each low-level heuristic has an equal initial score (Ri) and those scores are updated during the run. If heuristic improves the solution, its score is increased as much as the change quantity (Cq); if it worsens the solution, its score is decreased as much as Cq. A successful heuristic's score may not exceed predefined upper boundary (Ub); analogously an abortive heuristic's score may not be less than the predefined lower boundary (Lb). The heuristic with greater score has more probability to be selected. It is applied once and the scores are updated according to its performance.

4.2.4. Acceptance Strategies. Once a low-level heuristic is selected and applied, the acceptance of new solution also requires a strategy. We applied two strategies; these are Only Improvement (OI) and Monte Carlo (MC).

(i) OI acceptance strategy only accepts better solutions.

(ii) MC acceptance strategy accepts better solutions certainly. Additionally, it accepts worse solutions with a small probability (Monte Carlo constant) in order not to get stuck at the local optima.

4.3. Simulated Annealing. Simulated annealing (SA) is a well-known local search algorithm proposed by Kirkpatrick et al. [44], inspired from the annealing process of metal work. A starting temperature T is determined. The larger this value is, the more inferior solutions are encouraged. After a predetermined number of iterations, T is set to T/a. When a is large, the temperature decrease ratio is large and the acceptance of inferior solutions become less likely at a greater rate. When temperature becomes less than a predefined value, SA stops (the stopping condition). The number of iterations at each temperature is limited by R and

```
Initialization i = 0
Initial candidate solution S_0
Final solution S_f
sNew = S_0, S_f = S_0
while (i < maxIterations) Select heuristic according to heuristic selection strategy
   sNew = applySelectedHeuristic (S_f)
   if (sNew is accepted according to move acceptance strategy)
      S_f = S_new
   end if
   i++
end while
return S_f
```

Algorithm 1: Pseudocode of HH.

greater values of R correspond to slower cooling; that is, more neighbor solution trial is occurring when there is a greater likelihood of inferior exchanges being accepted. In our implementation, R value changes and increases by an initially specified b value. When a random neighbor is generated, it is compared with the previous solution. If it is better than the previous one, it is accepted. If it is worse than the previous one, it may still be accepted with a small probability. This probability is calculated based on the difference of old and new solution quality and the temperature. The pseudocode of SA is given in Algorithm 2.

4.4. Migrating Birds Optimization. Migrating birds optimization (MBO) is a neighborhood search technique proposed by Duman et al. [40], inspired from the V formation flight of the migrating birds. It has been recently proposed but successfully applied in various research areas such as task allocation problem [45], flow shop scheduling problem [46–48], U-shaped assembly line balancing problems with workers assignment [49], and continuous functions [50].

The pseudocode of MBO is given in Algorithm 3. Similar to birds of a V shape migration, some initial solutions organize a V formation including one leading solution and some followers. In the flock of solutions, a limited number of neighboring solutions for each main solution are generated. The neighboring solutions of each initial solution are evaluated and if there are any improvements among them, that initial solution is replaced by the solution provided by the most improved neighbor. Then, each main solution is tried to be improved further by the help of its neighbors. This means that each solution will share some of its unused neighbors to the next (behind) main solution. Therefore, except the leading solution, the other main solutions of the flock have the chance to be improved by one of the neighbors of the main solution in front of them. The procedure is repeated a number of times. Then, the leading solution moves to the end of the flock and one of its followers becomes the new leader. The same procedure is done and repeated for the new flock. The algorithm continues until a number of iterations are reached. Finally, the best solution of the flock is introduced as the solution of the MBO algorithm.

4.5. A New Hybrid Algorithm: HHMBO. Now, we present a novel hybrid technique by integrating various hyperheuristic techniques with the migrating birds optimization algorithm. To the best of our knowledge, this work is the first attempt of hybridizing MBO with hyperheuristics.

MBO is a metaheuristic algorithm that explores new solutions by single step heuristics as illustrated in Figure 1. Neighbor generation technique is fixed and does not change throughout the execution of the algorithm. Therefore the performance of classical MBO is much dependent on the neighbor generation technique. However, giving chance more than one heuristic with different characteristics to discover search space may increase its exploration capability. Based on this idea, we integrated the recently presented MBO algorithm with the hyperheuristic principles. Instead of a single discovery method, we took the advantage of exploiting a set of low-level heuristics in order to construct the neighbor solutions in the population.

Neighborhood generation process of hyperheuristics embedded migrating bird optimization (HHMBO) is presented in Figure 2. The neighbor solution generation part of MBO is hybridized by HH characteristics, which is capable of managing heuristic pool. A set of heuristic selection strategies are handled for choosing the low-level heuristics. In our implementation, we have applied simple random, random permutation, and adaptive searching heuristic selection strategies. The new neighboring solution is accepted or rejected according to the current move acceptance specifications. We have implemented Only Improvement and Monte Carlo move acceptance strategies. HHMBO operates in different ways according to the adjusted heuristic selection strategy and the move acceptance strategy. As a result, the HHMBO types that emerged in the application can be listed as follows: simple random Only Improvement (SR_OI), simple random Monte Carlo (SR_MC), random permutation Only Improvement (RP_OI), random permutation Monte Carlo (RP_MC), adaptive searching Only Improvement (AS_OI), and adaptive searching Monte Carlo (AS_MC). HHMBO generates neighbor solutions by one of those hyperheuristic variations. The pseudocode of the algorithm is given in Algorithm 4.

HHMBO offers a collaboration of hyperheuristics and migrating birds optimization algorithm. HH provide an

```
Generate a random initial solution and indicate it as current solution, cs, initialize
temperature T
Best solution(bs) = cs
while termination condition is not satisfied
  for R times
    Obtain a neighbor solution, ns of cs
    delta = cost of ns−cost of cs
    if delta < 0
      cs = ns
    else if random() < e^(|delta|/T)
      cs = ns
    if cost of ns < cost of bs
      bs = ns
  end for
    T = T/a
    R = R * b
end while
```

Algorithm 2: Pseudocode of SA.

```
Generate nob initial solutions randomly and place them on a hypothetical V formation arbitrarily
while termination condition is not satisfied
  for nof times
    Try to improve leading solution by generating and evaluating non neighbors of it
    for each solution s_i in the flock (except leader)
    Try to improve s_i by evaluating (non-olf) neighbors of it and olf unused best neighbors from the solution in the front
    end for
  end for
  Move the leader solution to the end and forward one of the solutions following it to the leader position
end while
```

Algorithm 3: Pseudocode of MBO.

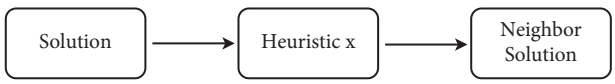

Figure 1: Neighborhood generation in classical MBO.

extensive exploration capability due to their ability to quickly adapt according to the problem instance at hand. MBO exploits the leader solution more detailed each time and the leader is updated in each generation. HHMBO takes the advantage of exploring the solution space by the exploration talent of HH in addition to the exploitation skills of MBO. The properties of the HHMBO which distinguish it from the other metaheuristic approaches are listed as follows:

(i) A number of solutions running in parallel

(ii) The benefit mechanism among the solutions

(iii) Exploiting the leader solution more than the others in each generation

(iv) Exploring the search space by multiple operators where the best strategy is determined by HH

5. Experimental Setup

In this section, we provide the details of computational experiment settings. Firstly the dataset used in experiments is described. Then the parameter settings of MBO, HH, SA, and HHMBO are explained. Lastly, the benchmark dataset is clarified.

GAMS is used to verify and test the mathematical model. All algorithm implementations are done using Java programming language on Eclipse IDE running on Intel i7-5500U CPU and 16 GB memory computer operated by Windows 10 operating system. Besides the basic constraints on the formulation presented, additional legal regulation constraints have also been applied while implementing each solution technique.

5.1. Data Preparation. We had a limited amount of real data provided by our collaborator company. However, we needed a great amount of experimental data to achieve a fair comparison among solution techniques we had implemented. Therefore, a simple application is prepared, which generates synthetic data, which have the general characteristics of the real data. Requirements are selected randomly

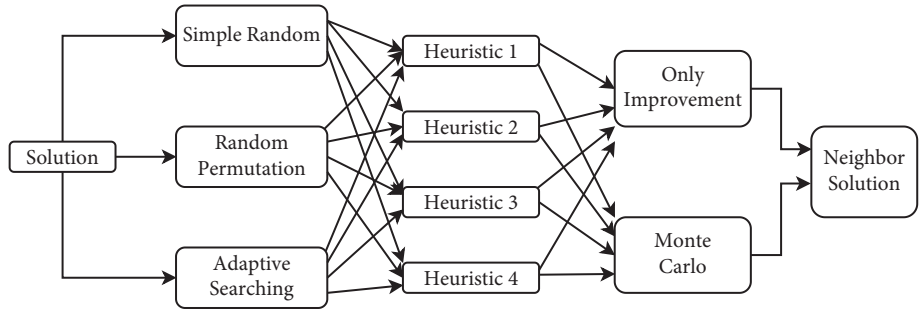

Figure 2: Neighborhood generation mechanism in HHMBO.

Generate *nob* initial solutions randomly and place them V formation
Determine the heuristic selection strategy (Simple random, random permutation or adaptive searching)
Determine the move acceptance strategy (Only improvement or monte carlo)
while termination condition is not satisfied
  for *nof* times
    Try to improve leading solution by generating and evaluating *non* neighbors of it applying selected low level heuristic
    for each solution s_i in the flock (except leader)
      Try to improve s_i by evaluating (*non-olf*) neighbors of it and *olf* unused best neighbors from the solution in the front
      Accept or reject the neighbor solution according to current move acceptance strategy.
    end for
  end for
  Move the leader solution to the end and forward one of the solutions following it to the leader position
end while

Algorithm 4: Pseudocode of HHMBO.

among a certain range, where lower and upper boundaries are calculated based on employee number. A real input dataset for 200 employees and sample synthetic requirement data values are illustrated in Table 6. The employee numbers are shown in the top row and they vary from 30 to 500. This illustration comprises a sample requirement input for varied number of employees. First column of the table contains the time slot IDs; each one corresponds to a day and a shift type. The requirements in each time slot for a company with 200 (real), 30, 50, 100, 150, 200, and 500 employees are given in the subsequent columns.

5.2. Parameter Settings. MBO and SA perform a unique heuristic technique for improving solutions. Since we have four different heuristic techniques, we have conducted an experiment to understand which of the four operators in the system is compatible with which algorithm. In order to assess a fair comparison among all applied techniques, we set the stopping criterion of the algorithms based on the number of instances (noi) created while each algorithm is running. This experiment is done on each of the datasets, while the number of instances (noi) is set to 10,000 and each case is repeated 50 times. Heuristic operators are listed from the best performing one to the least performing one in Table 7. H1 through H4 correspond to heuristic one through heuristic four, respectively, which are explained in Section 4.2.1. Swap, Insert, Inverse, and Scramble are applied on QAP and explained in Section 4.2.2.

In order to get the best performance from the algorithms for FOSSP, we need to use their best performing values. These best performing values are discovered with fine-tuning experiments. The number of solutions that algorithm can generate while surfing in the solution space is limited to 50,000 for each algorithm. The experiments are repeated ten times and the results acquired from the average performances are represented in Tables 8 and 9. Best parameter values are emphasized in bold font.

5.3. Experimental Setup of Benchmark Data. Apart from the specific problem defined in this study, computational experiments are also conducted with QAP to measure the success of the HHMBO. Computational experiments presented have been done using QAPLIB benchmark dataset, where QAP instances are available together with their best known solutions [51]. Fine-tuning experiments are handled to find out the best performing values. The number of solutions that the algorithm can generate while navigating the solution space is limited to 50,000. The experiments were repeated ten times and the results from the average performances are shown in Table 10, with the best parameter values being in bold.

6. Results and Discussion

In this section, we present and analyze the results of computational experiments. GAMS results are presented firstly. Then, the results obtained by heuristic solution

Table 6: Sample real and synthetic inputs.

ID	Time slot meaning	Real data 200	Synthetic 30	Data 50	100	150	200	500
1	Monday day shift	58	3	12	15	33	12	36
2	Monday evening shift	36	8	11	2	16	12	30
3	Monday night shift	50	8	7	23	9	42	76
4	Tuesday day shift	58	5	6	20	28	10	2
5	Tuesday evening shift	36	9	10	4	17	7	60
6	Tuesday night shift	50	9	7	1	1	34	41
7	Wednesday day shift	58	6	9	13	25	29	94
8	Wednesday evening shift	36	5	1	16	26	4	40
9	Wednesday night shift	5	3	4	14	15	40	3
10	Thursday day shift	58	6	8	6	28	40	88
11	Thursday evening shift	36	3	11	16	25	27	31
12	Thursday night shift	50	7	5	19	30	12	78
13	Friday day shift	58	8	7	5	33	2	103
14	Friday evening shift	36	1	4	14	16	17	89
15	Friday night shift	50	8	6	17	6	26	48
16	Saturday day shift	2	6	10	2	14	24	80
17	Saturday evening shift	5	3	11	1	24	1	96
18	Saturday night shift	5	2	2	5	15	10	79
19	Sunday day shift	5	1	13	1	1	39	55
20	Sunday evening shift	49	1	11	23	26	35	56
21	Sunday night shift	36	5	10	22	14	43	40

Table 7: Sample real and synthetic inputs.

FOSSP	Low-level heuristics	QAP	Low-level heuristics
MBO	**H2**, H4, H3, H1	MBO	**Swap**, Insert, Inverse, Scramble
SA	**H1**, H3, H4, H1	SA	**Swap**, Insert, Inverse, Scramble

Table 8: Parameter fine-tuning experiment results for FOSSP.

Parameter	MBO	HHMBO					
		SR_OI	SR_MC	RP_OI	RP_MC	AS_OI	AS_MC
nob	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**
non	**3**, 5, 7	3, 5, **7**	3, **5**, 7	3, **5**, 7	3, **5**, 7	3, **5**, 7	**3**, 5, 7
nof	5, **10**	**5**, 10	**5**, 10	5, **10**	**5**, 10	5, **10**	**5**, 10
olf	**1**, 2, 3	**1**, 2, 3	**1**, 2, 3	1, **2**, 3	**1**, 2, 3	**1**, 2, 3	1, **2**, 3
Monte Carlo constant			0.0005, **0.001**, 0.005		0.0005, 0.001, **0.005**		0.0005, 0.001, **0.005**

Table 9: Parameter fine-tuning experiment results for FOSSP.

Parameter	SA	Parameter	Adaptive Searching
T	**50.000**, 75.000, 100.000	*Ri*	10, **15**, 20
A	**1.1**, 1.2, 1.3	*Cq*	0.5, 0.8, **1**
R	10, **20**, 30	*Ub*	**40**,50,60,100
b	**1.1**, 1.2, 1.3	*Lb*	**0**, 5, 10

techniques on FOSSP are presented. Lastly, experimental outcomes on QAPLIB are presented.

6.1. GAMS Results. The GAMS utilizing CPLEX solver is used to verify and test the mathematical model of FOSSP. If the schedule is completely fair and all assignments are scrupulous for each employee, then fitness is equal to zero. The result gets away from zero, while the solution wanders away from fairness.

Solver tests are done for datasets with 10, 20, 30, and 60 employees and results are illustrated in Table 11 in terms of cost and execution time. FOSSP is capable of finding optimal value (zero) for employee numbers 10 to 30. FOSSP produces results closer to optimal for employee number 60. As the size of the problem gets larger, complexity increases. Obviously, execution time increases beyond linear behavior as the number of employees increases. Model attains border line of maximum iteration number in GAMS. Execution is terminated on maximum iteration number; hence, the

Table 10: Parameter fine-tuning experiment results for QAP.

Parameter	MBO	HHMBO						Parameter	Adaptive searching
		SR_OI	SR_MC	RP_OI	RP_MC	AS_OI	AS_MC		
nob	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	*Ri*	10, 15, **20**
non	**3**, 5, 7	**3**, 5, 7	3, **5**, 7	**3**, 5, 7	3, **5**, 7	**3**, 5, 7	**3**, 5, 7	*Cq*	0.5, **0.8**, 1
nof	5, **10**	5, **10**	**5**, 10	**5**, 10	5, **10**	5, **10**	**5**, 10	*Ub*	40,50,**60**,100
olf	**1**, 2, 3	**1**, 2, 3	**1**, 2, 3	**1**, 2, 3	**1**, 2, 3	**1**, 2, 3	**1**, 2, 3	*Lb*	**0**, 5, 10
Monte Carlo constant			0.0005, **0.001**, 0.005		**0.0005**, 0.001, 0.005		**0.0005**, 0.001, 0.005		

Table 11: GAMS results for FOSSP datasets.

Number of employees	Cost values	Execution time (minutes)
10	0	12.824
20	0	500.334
30	0	968.884
60	3.34	988.58

Table 12: Average results when noi is N^2.

Employees number	MBO	SA	HHMBO						HH					
			SR_OI	SR_MC	RP_OI	RP_MC	AS_OI	AS_MC	SR_OI	SR_MC	RP_OI	RP_MC	AS_OI	AS_MC
30	21.81	33.67	16.95	**13.13**	13.79	20.16	19.77	19.67	20.91	21.24	19.88	22.48	20.17	19.99
50	29.79	59.82	24.59	**15.72**	22.80	21.72	30.48	36.84	27.78	29.94	30.83	30.33	42.69	38.23
100	52.13	161.54	44.99	44.33	**39.99**	51.48	63.09	68.60	46.07	48.96	54.99	110.15	135.29	108.24
150	62.11	292.00	61.51	67.18	**57.00**	63.09	98.14	101.93	91.42	178.29	133.70	135.24	208.15	244.27
200	70.12	367.25	67.99	73.20	**67.67**	71.78	143.12	141.09	241.61	260.86	315.59	209.68	248.82	350.38
500	287.60	926.11	301.62	254.06	**247.60**	257.94	499.58	460.58	962.85	939.88	588.76	765.77	712.36	912.62

Table 13: Standard deviations of the experiment.

Employees number	MBO	SA	HHMBO						HH					
			SR_OI	SR_MC	RP_OI	RP_MC	AS_OI	AS_MC	SR_OI	SR_MC	RP_OI	RP_MC	AS_OI	AS_MC
30	2.58	2.55	2.05	2.32	2.07	2.13	2.03	**2.01**	3.67	3.51	3.65	3.96	3.96	2.89
50	1.67	1.87	1.59	1.65	**1.52**	1.64	**1.52**	1.58	2.15	3.02	2.33	2.83	2.78	3.36
100	1.78	2.00	1.95	1.80	**1.71**	1.88	1.75	1.77	3.51	3.16	3.68	3.44	3.41	3.28
150	1.73	1.92	1.92	1.69	**1.52**	1.77	1.90	1.89	2.99	3.33	3.31	3.18	3.46	3.20
200	1.96	1.75	2.09	2.12	**1.48**	2.01	2.09	2.01	2.96	2.85	2.71	3.05	2.63	2.75
500	1.27	1.28	1.43	1.52	**1.22**	1.32	1.37	1.34	1.99	2.11	2.05	1.91	2.15	2.14

execution time values are the total minutes required for maximum iteration in each case.

GAMS is capable of solving the model, but it is poor in terms of execution time, especially for large-scale problems. Therefore, we needed to implement and use heuristic based solution techniques for obtaining reasonable results within acceptable run time.

6.2. Results Obtained by Utilizing Synthetic FOSSP Dataset. In this subsection, results of heuristic based solution techniques on FOSSP are presented. If N is the number of employees, we limited the run time with N^2 iterations in order to assess a fair comparison among the applied techniques. This corresponded to 0.9 *s* for a problem of size 30 and to 34 *s* for a problem of size 100 on the specified machine. Consequently, as the complexity of the problem increases, the effort to be spent on its resolution also increases.

Solution techniques are tested for numbers of employees of 30, 50, 100, 150, 200, and 500. We have generated 100 sample synthetic inputs for each case. Average numerical cost values gained from the ten runs are demonstrated in Table 12. For all numbers of employees, HHMBO variants discover the best results. For numbers of employees of 30 and 50, HHMBO SR_MC variant outperforms the others by at least 5% on average. When the number of employees increases to 100, 150, 200, and 500, HHMBO RP_OI variant outperforms the other techniques.

When the performances are analyzed in accordance with the increase of the number of employees, it is apparent that MBO and HHMBO variants retain their accomplishment as complexity rises. However, MBO is 66%, 90%, 30%, 9%, and 4% worse on average than the best performing HHMBO variant for numbers of employees of 30, 50, 100, 150, and 200, respectively. It is clear that both algorithms are successful in spite of increasing problem complexity. However,

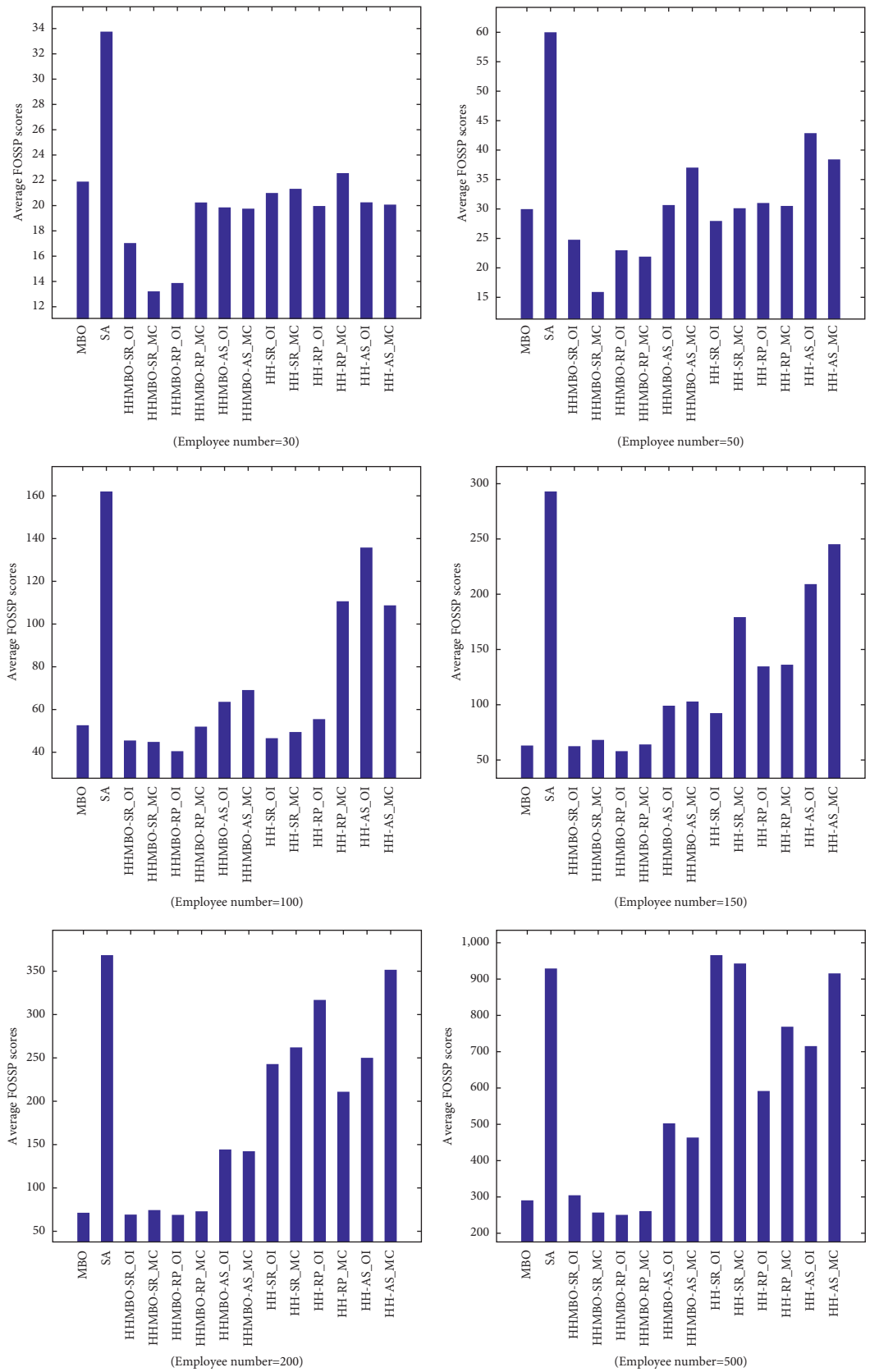

Figure 3: The performance of each algorithm for each number of employees.

Table 14: The t-test results of HHMBO-RP_OI and the state-of-the-art algorithms on FOSSP.

Algorithm 1 ↔ Algorithm 2	30	50	100	150	200	500
HHMBO-RP_OI ↔ MBO	$1.3E-02$	$9.9E-02$	$3.1E-04$	$1.7E-02$	$5.2E-05$	$3.7E-03$
HHMBO-RP_OI ↔ SA	$1.6E-05$	$3.8E-04$	$5.7E-05$	$6.9E-06$	$4.1E-08$	$4.9E-04$
HHMBO-RP_OI ↔ HH-SR_OI	$2.4E-06$	$7.4E-05$	$1.1E-08$	$3.4E-12$	$2.2E-11$	$2.7E-22$
HHMBO-RP_OI ↔ HH-SR_MC	$9.7E-06$	$1.5E-04$	$2.4E-10$	$3.3E-11$	$4.0E-15$	$2.2E-19$
HHMBO-RP_OI ↔ HH-RP_OI	$2.6E-05$	$1.3E-02$	$1.9E-11$	$1.4E-11$	$5.8E-14$	$2.7E-23$
HHMBO-RP_OI ↔ HH-RP_MC	$2.8E-04$	$1.9E-02$	$6.4E-10$	$7.7E-09$	$3.9E-12$	$1.4E-21$
HHMBO-RP_OI ↔ HH-AS_OI	$4.1E-03$	$3.8E-02$	$1.7E-05$	$4.9E-12$	$7.8E-16$	$1.4E-17$
HHMBO-RP_OI ↔ HH-AS_MC	$6.7E-04$	$1.2E-03$	$2.9E-06$	$6.0E-12$	$2.2E-15$	$6.1E-18$

Table 15: The percentage deviations of heuristic results from the best solution.

Problem	Density	Size	BKS	MBO	HHMBO					
					SA_OI	SR_MC	RP_OI	RP_MC	AS_OI	AS_MC
esc64a	3.17	64	116	0.0	0.0	0.0	58.6	58.6	0.0	0.0
tai64c	4.13	64	1,855,928	0.0	0.0	0.0	11.4	10.1	0.0	0.1
chr22a	8.68	22	6,156	9.6	16.8	13.3	41.2	45.8	8.1	7.1
esc16j	9.38	16	8	0.0	0.0	0.0	0.0	25.0	0.0	0.0
chr20a	9.50	20	2,192	31.3	55.3	56.6	147.7	132.6	27.9	20.1
chr20b	9.50	20	2,298	25.7	35.9	47.5	67.6	119.4	17.8	36.3
chr18a	10.49	18	11,098	54.2	71.0	81.6	197.2	190.1	53.8	55.2
esc16i	11.72	16	14	0.0	0.0	0.0	28.6	14.3	0.0	0.0
chr15a	12.44	15	9,896	22.6	12.7	40.4	76.2	116.9	35.8	30.1
chr15b	12.44	15	7,990	38.2	70.5	37.7	193.3	168.5	33.4	51.4
chr15c	12.44	15	9,504	53.8	39.2	62.9	95.4	96.6	39.2	42.0
chr12c	15.28	12	11,156	14.9	28.1	14.3	60.4	46.0	18.8	8.5
chr12a	15.28	12	9,552	20.5	37.5	19.7	60.2	58.5	19.8	5.7
esc16d	16.41	16	16	0.0	0.0	0.0	12.5	0.0	0.0	0.0
esc16e	16.41	16	28	0.0	0.0	0.0	14.3	14.3	0.0	0.0
esc16g	16.41	16	26	0.0	0.0	0.0	15.4	23.1	0.0	0.0
esc32d	17.58	32	200	0.0	3.0	4.0	37.0	35.0	0.0	0.0
esc32c	25.59	32	642	0.0	0.0	0.0	13.7	14.0	0.0	0.0
esc32h	27.54	32	438	0.0	2.3	2.3	26.5	21.0	0.0	0.0
esc16a	29.69	16	68	0.0	0.0	0.0	8.8	8.8	0.0	0.0
scr15	37.33	15	51,140	7.0	11.3	7.0	29.4	28.1	6.9	9.2
esc16c	39.84	16	160	0.0	0.0	0.0	5.0	8.8	0.0	0.0
esc16b	71.88	16	292	0.0	0.0	0.0	0.0	0.0	0.0	0.0
lipa20b	89.25	20	27,076	11.2	14.8	17.2	22.0	21.9	9.8	11.0
esc16h	89.84	16	996	0.0	0.0	0.0	1.6	0.0	0.0	0.0
lipa20a	90.25	20	3,683	2.2	3.0	3.0	4.5	4.5	2.4	2.7
had12	91.67	12	1,652	0.4	0.7	0.1	3.4	2.4	1.0	0.8
rou12	91.67	12	235,528	3.0	4.7	2.8	9.8	9.0	5.7	4.9
rou15	93.33	15	354,210	5.1	7.7	6.8	13.7	13.4	5.0	4.8
nug16b	93.75	16	1,240	3.9	6.9	3.4	12.9	13.7	4.5	4.4
els19	94.74	19	17,212,548	4.7	8.7	5.5	39.7	47.3	1.8	2.7
lipa40b	94.81	40	476,581	5.3	21.0	19.6	26.0	26.3	3.5	6.5
lipa40a	95.06	40	31,538	1.3	2.0	2.0	2.9	2.8	1.6	1.6
lipa50b	95.88	50	1,210,244	17.9	21.8	21.7	25.3	25.5	8.3	11.2
tai60b	98.33	60	608,215,054	1.1	2.2	2.7	40.2	41.0	0.6	1.1
sko100a	99.00	100	152,002	0.7	5.9	5.7	13.9	13.9	0.7	2.6
bur26b	100.00	26	3,817,852	0.1	0.5	0.3	2.5	3.0	0.0	0.1
bur26a	100.00	26	5,426,670	0.1	0.4	0.4	2.8	1.8	0.1	0.2
bur26c	100.00	26	5,426,795	0.0	0.2	0.2	3.1	3.1	0.0	0.1
bur26e	100.00	26	5,386,879	0.0	0.3	0.4	3.4	3.6	0.0	0.1
bur26g	100.00	26	10,117,172	0.0	0.1	0.4	2.6	3.0	0.1	0.0
Average				8.2	11.8	11.7	34.9	35.9	**7.5**	7.8

HHMBO variants obtain slightly better results than MBO in each condition.

The best performing HHMBO variant is 156%, 281%, 304%, 412%, and 443% better than SA for numbers of employees of 30, 50, 100, 150, and 200, respectively. As the complexity of the problem increases, the performance of SA decreases and the difference between the two algorithms increases exponentially. Best performing HHMBO variant is 51%, 77%, 15%, 60%, and 210% better than the best performing HH variant for numbers of employees of 30, 50, 100, 150, and 200, respectively. The performance of HH is admissible for numbers of employees of 30 to 150, whereas when the number of employees increases to 200, HH performs exponentially worse.

For the experiment in which the number of employees is 500, the total iteration number determined with N^2 corresponds to 250,000. That is a reasonably big number of chances to search through the solution space. For this challenging case, the best performing HHMBO variant (RP_OI) is 274% better than SA, 138% better than HH, and 16% better than MBO.

HHMBO (RP_OI) obtained significantly better results in complex FOSSP instances. RP_OI applies each of the four problem specific heuristics consecutively due to the permutation order. Each heuristic is applied to an equal number of instances. Each heuristic has a particular ability to discover search space. Accomplishment of HHMBO (RP_OI) exposes that when the algorithm is given a great search ability, using heuristics equal numbers of times helps us to find the best results. HHMBO is successful on catching values close to global optimum. HHMBO provides diversification along the search space by applying a variety of heuristics alternately; that is why it is promising for solving combinatorial optimization problems.

The standard deviation results gained by FOSSP experiments are presented in Table 13 to demonstrate the robustness of the algorithms. It is observed that HHMBO-RP_OI is more robust than its rivals. We can say that the HHMBO variants produce slightly better standard deviation results than their HH variants. This can be explained by the design of the algorithms as follows. The HHMBO variants wander within the solution space by exploiting some prioritized (leader) solutions more than HH do, which inherently makes them more robust.

The average results of each algorithm are illustrated in Figure 3 for each number of employees. When the number of employees is equal to 30, all algorithms produce close results to each other. As the number of employees increases, the distinction between the algorithms involved in the experiment becomes more prominent. HHMBO-RP_OI produced the best results for the most challenging experiment, when the number of employees is equal to 500. HHMBO versions figure out lower (better) results when they are compared with their HH versions.

We have also applied t-test to check if the differences are statistically significant. The best performing HHMBO-RP_OI and all other algorithms in the experiment are tested and the statistical results obtained by two-tailed t-test are given in Table 14. We observed that the p value for pairwise t-tests among the best performing proposed variant HHMBO-RP_OI and the state-of-the-art algorithms is 0.004% on average. The results imply that HHMBO-RP_OI is indeed statistically different from other algorithms.

6.3. Results on Benchmark Problems in QAPLIB. After observing that HHMBO variants get very successful results for the FOSSP instances, we wanted to see their ability in converging optimal solutions on a widely used problem. We applied HHMBO on QAP, where MBO is firstly applied and tested. If N is the size of a QAP instance, we limited the run time with N^3 iterations. This corresponded to 0.01 s for a problem of size 12 and to 48 s for a problem of size 100 on the specified machine.

We have utilized 41 QAPLIB instances with different densities. Density corresponds to the percentage of nonzero entries in the flow matrix. The instance names, densities, size of the instances, best known solutions (BKS), and the percentage deviations of heuristic results from BKS are given in Table 15. Problem instances are listed from the sparse one to the dense one. Only the minimum costs obtained in 10 runs are considered, since the aim is to check the ability of HHMBO in finding BKS.

The results indicate that AS_IO variant of HHMBO is accomplished on catching promising solutions. HHMBO (AS_OI) is up to 14.6% better than MBO on converging to the BKS. According to the average percentage deviation gained from 41 instances, HHMBO (AS_OI) converges 7.5% to BKS, while MBO is 8.2% close to it. HHMBO (AS_OI) performs 0.7% better than MBO on average. Best performing variant of this experiment is adaptive searching Only Improvement HHMBO. Adaptive searching strategy exploits low-level heuristics depending on their performances. This adaptive feature ensures choosing the most beneficial low-level heuristic for each specific case during the run. It has been observed from the experiments conducted on QAPLIB benchmark problems that HHMBO consistently allows finding better results, albeit slightly.

7. Conclusion

In this study, we introduced a novel hybrid computational intelligence algorithm to the literature and the mathematical model of a shift scheduling problem of a manufacturing company defined by including the fairness perspective, which is typically ignored especially in manufacturing industry. The novel algorithm is designed by embedding hyperheuristic (HH) improvements in the migrating bird optimization (MBO) algorithm, called HHMBO. In this way, the exploitation capability of MBO is improved and the solution space is explored in a more diversified manner.

The problem that we tackled is called fairness oriented shift scheduling problem (FOSSP). The model of FOSSP is firstly verified with a solver, where the solver is capable of catching global optima with large execution time even for small instances. Therefore, exploitation of heuristic based techniques was an apparent need. In line with this need, we applied HHMBO together with simulated annealing (SA),

hyperheuristics (HH), and classical migrating bird optimization (MBO). We have conducted computational experiments in order to assess the performance of the algorithms on synthetic data. Results show the superiority of HHMBO-RP_OI and it is observed that it outperforms its rivals by 40% on average for FOSSP. Besides, it is seen that HHMBO-RP_OI definitely offers better solutions in large problem instances, which is a desired property in practice and gap in the literature. We believe that this good performance is due to integrating the exploitation capability of HH with MBO's exploration capability.

To justify the superiority of HHMBO over the other algorithms, we used QAP instances from the QAPLIB. This experiment was significant to understand whether the achievement of the algorithm is specific to the particular problem. We conducted computational experiments on 41 QAPLIB instances with assorted densities. According to the results of this experiment, AS_OI variant of the new hybrid algorithm could achieve up to 14.6% better results than MBO.

There are several interesting directions for future research. The approach may be applied on different types of optimization problems to highlight its wide applicability and generality. An example may be the test suites given in [52]. Furthermore, a comparative study may be conducted on swarm based hybrid algorithms. A multiobjective variation of HHMBO may be developed and may be applied on a set of multiobjective problems. Another research direction is to help solve employee transportation issues while constructing weekly schedules.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Marmara University Scientific Research Committee under the Project ID FEN-C-DRP-131217-0671.

References

[1] A. T. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens, and D. Sier, "An annotated bibliography of personnel scheduling and rostering," *Annals of Operations Research*, vol. 127, no. 1–4, pp. 21–144, 2004.

[2] P. De Causmaecker, P. Demeester, G. V. Berghe, and B. Verbeke, "Analysis of real-world personnel scheduling problems," in *Proceedings of the 5th International Conference on Practice and Theory of Automated Timetabling*, pp. 183–197, Pittsburgh, PA, USA, August 2004.

[3] M. Defraeye and I. Van Nieuwenhuyse, "Staffing and scheduling under nonstationary demand for service: a literature review," *Omega*, vol. 58, pp. 4–25, 2016.

[4] M. Erhard, J. Schoenfelder, A. Fügener, and J. O. Brunner, "State of the art in physician scheduling," *European Journal of Operational Research*, vol. 265, no. 1, pp. 1–18, 2018.

[5] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering: a review of applications, methods and models," *European Journal of Operational Research*, vol. 153, no. 1, pp. 3–27, 2004.

[6] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck, "Personnel scheduling: a literature review," *European Journal of Operational Research*, vol. 226, no. 3, pp. 367–385, 2013.

[7] P. Brucker, R. Qu, and E. Burke, "Personnel scheduling: models and complexity," *European Journal of Operational Research*, vol. 210, no. 3, pp. 467–473, 2011.

[8] K. R. Baker, "Workforce allocation in cyclical scheduling problems: a survey," *Journal of the Operational Research Society*, vol. 27, no. 1, pp. 155–167, 1976.

[9] H. K. Alfares, "Survey, categorization, and comparison of recent tour scheduling literature," *Annals of Operations Research*, vol. 127, no. 1–4, pp. 145–175, 2004.

[10] C. Seifi, M. Schulze, and J. Zimmermann, "A new mathematical formulation for a potash-mine shift scheduling problem with a simultaneous assignment of machines and workers," *European Journal of Operational Research*, vol. 292, no. 1, pp. 27–42, 2021.

[11] J. O. Brunner and R. Stolletz, "Stabilized branch and price with dynamic parameter updating for discontinuous tour scheduling," *Computers & Operations Research*, vol. 44, pp. 137–145, 2014.

[12] J. Atlason, M. A. Epelman, and S. G. Henderson, "Call center staffing with simulation and cutting plane methods," *Annals of Operations Research*, vol. 127, no. 1–4, pp. 333–358, 2004.

[13] J. G. Morris and M. J. Showalter, "Simple approaches to shift, days-off and tour scheduling problems," *Management Science*, vol. 29, no. 8, pp. 942–950, 1983.

[14] F. F. Easton and D. F. Rossin, "Equivalent alternate solutions for the tour scheduling problem," *Decision Sciences*, vol. 22, no. 5, pp. 985–1007, 1991.

[15] F. F. Easton and D. F. Rossin, "A stochastic goal program for employee scheduling," *Decision Sciences*, vol. 27, no. 3, pp. 541–568, 1996.

[16] R. Stolletz, "Operational workforce planning for check-in counters at airports," *Transportation Research Part E: Logistics and Transportation Review*, vol. 46, no. 3, pp. 414–425, 2010.

[17] H. K. Alfares, "An efficient two-phase algorithm for cyclic days-off scheduling," *Computers & Operations Research*, vol. 25, no. 11, pp. 913–923, 1998.

[18] D. S. Altner, E. K. Mason, and L. D. Servi, "Two-stage stochastic days-off scheduling of multi-skilled analysts with training options," *Journal of Combinatorial Optimization*, vol. 38, no. 1, pp. 111–129, 2019.

[19] H. C. Lau, "On the complexity of manpower shift scheduling," *Computers & Operations Research*, vol. 23, no. 1, pp. 93–102, 1996.

[20] G. Alp and A. F. Alkaya, "Improving the quality of personnel scheduling by incorporating fairness," *International Journal of Modeling and Optimization*, vol. 9, no. 2, pp. 97–101, 2019.

[21] A. Yurtkuran, B. Yagmahan, and E. Emel, "A novel artificial bee colony algorithm for the workforce scheduling and balancing problem in sub-assembly lines with limited buffers," *Applied Soft Computing*, vol. 73, pp. 767–782, 2018.

[22] M. Günther and V. Nissen, "Application of particle swarm optimization to the British telecom workforce scheduling problem," vol. 4, 2013.

[23] M. N. Janardhanan, Z. Li, and P. Nielsen, "Model and migrating birds optimization algorithm for two-sided assembly line worker assignment and balancing problem," *Soft Computing*, vol. 23, no. 21, pp. 11263–11276, 2019.

[24] H. Algethami and D. Landa-Silva, "Diversity-based adaptive genetic algorithm for a workforce scheduling and routing problem," in *Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1771–1778, IEEE, Donostia, San Sebastián, Spain, June 2017.

[25] H. Algethami, A. Martínez-Gavara, and D. Landa-Silva, "Adaptive multiple crossover genetic algorithm to solve workforce scheduling and routing problem," *Journal of Heuristics*, vol. 25, no. 4, pp. 753–792, 2019.

[26] W.-L. Liu, Y.-J. Gong, W.-N. Chen, Z. Liu, H. Wang, and J. Zhang, "Coordinated charging scheduling of electric vehicles: a mixed-variable differential evolution approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 5094–5109, 2019.

[27] K. Sethanan and R. Pitakaso, "Improved differential evolution algorithms for solving generalized assignment problem," *Expert Systems with Applications*, vol. 45, pp. 450–459, 2016.

[28] F. Zhao, L. Zhao, L. Wang, and H. Song, "An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion," *Expert Systems with Applications*, vol. 160, Article ID 113678, 2020.

[29] S. Zhou, L. Xing, X. Zheng, N. Du, L. Wang, and Q. Zhang, "A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times," *IEEE transactions on cybernetics*, vol. 51, no. 3, 2019.

[30] E. K. Burke, G. Kendall, and E. Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," *Journal of Heuristics*, vol. 9, no. 6, pp. 451–470, 2003.

[31] S. Pan, M. Akplogan, N. Touati, L. Létocart, R. Wolfler Calvo, and L.-M. Rousseau, "A hybrid heuristic for the multi-activity tour scheduling problem," *Electronic Notes in Discrete Mathematics*, vol. 69, pp. 333–340, 2018.

[32] N. A. Hernández-Leandro, V. Boyer, M. A. Salazar-Aguilar, and L.-M. Rousseau, "A matheuristic based on Lagrangian relaxation for the multi-activity shift scheduling problem," *European Journal of Operational Research*, vol. 272, no. 3, pp. 859–867, 2019.

[33] F. Zhao, L. Zhang, Y. Zhang, W. Ma, C. Zhang, and H. Song, "A hybrid discrete water wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion," *Expert Systems with Applications*, vol. 146, Article ID 113166, 2020.

[34] S. Asta, E. Özcan, and T. Curtois, "A tensor based hyper-heuristic for nurse rostering," *Knowledge-Based Systems*, vol. 98, pp. 185–199, 2016.

[35] H. Li, Y. Wang, S. Li, and S. Li, "A column generation based hyper-heuristic to the bus driver scheduling problem," *Discrete Dynamics in Nature and Society*, vol. 201510 pages, 2015.

[36] F. Garza-Santisteban, R. Sánchez-Pámanes, L. A. Puente-Rodríguez et al., "A simulated annealing hyper-heuristic for job shop scheduling problems," in *Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 57–64, IEEE, Wellington, New Zealand, June 2019.

[37] M. Alinia Ahandani, M. T. Vakil Baghmisheh, M. A. Badamchi Zadeh, and S. Ghaemi, "Hybrid particle swarm optimization transplanted into a hyper-heuristic structure for solving examination timetabling problem," *Swarm and Evolutionary Computation*, vol. 7, pp. 21–34, 2012.

[38] L. Chen, H. Zheng, D. Zheng, and D. Li, "An ant colony optimization-based hyper-heuristic with genetic programming approach for a hybrid flow shop scheduling problem," in *Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC)*, pp. 814–821, IEEE, Sendai, Japan, May 2015.

[39] V. Pandiri and A. Singh, "A hyper-heuristic based artificial bee colony algorithm for k-interconnected multi-depot multi-traveling salesman problem," *Information Sciences*, vol. 463-464, pp. 261–281, 2018.

[40] E. Duman, M. Uysal, and A. F. Alkaya, "Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem," *Information Sciences*, vol. 217, pp. 65–77, 2012.

[41] W. Jang, H. H. Lim, T. J. Crowe, G. Raskin, and T. E. Perkins, "The Missouri lottery optimizes its scheduling and routing to improve efficiency and balance," *Interfaces*, vol. 36, no. 4, pp. 302–313, 2006.

[42] W. Y. Szeto and H. K. Lo, "Transportation network improvement and tolling strategies: the issue of intergeneration equity," *Transportation Research Part A: Policy and Practice*, vol. 40, no. 3, pp. 227–243, 2006.

[43] E. K. Burke, M. Gendreau, M. Hyde et al., "Hyper-heuristics: a survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[44] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[45] D. Öz and I. Öz, "Scalable parallel implementation of migrating birds optimization for the multi-objective task allocation problem," *The Journal of Supercomputing*, vol. 77, no. 3, pp. 2689–2712, 2021.

[46] Y. Han, J.-Q. Li, D. Gong, and H. Sang, "Multi-objective migrating birds optimization algorithm for stochastic lot-streaming flow shop scheduling with blocking," *IEEE Access*, vol. 7, pp. 5946–5962, 2018.

[47] T. Meng, Q.-K. Pan, J.-Q. Li, and H.-Y. Sang, "An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem," *Swarm and Evolutionary Computation*, vol. 38, pp. 64–78, 2018.

[48] A. Sioud and C. Gagné, "Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times," *European Journal of Operational Research*, vol. 264, no. 1, pp. 66–73, 2018.

[49] Z. Zhang, Q. Tang, D. Han, and Z. Li, "Enhanced migrating birds optimization algorithm for u-shaped assembly line balancing problems with workers assignment," *Neural Computing & Applications*, vol. 31, no. 11, pp. 7501–7515, 2019.

[50] A. F. Alkaya, R. Algin, Y. Sahin, M. Agaoglu, and V. Aksakalli, "Performance of migrating birds optimization algorithm on continuous functions," in *Proceedings of the International Conference in Swarm Intelligence*, pp. 452–459, Springer, Hefei, China, October 2014.

[51] R. E. Burkard, S. E. Karisch, and F. Rendl, "Qaplib–a quadratic assignment problem library," *Journal of Global Optimization*, vol. 10, no. 4, pp. 391–403, 1997.

[52] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization," Technical Report, National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, 2017.