# Hybrid Metaheuristics for the Unrelated Parallel Machine Scheduling to Minimize Makespan and Maximum Just-in-Time Deviations

Chiuh-Cheng Chyu*, Wei-Shung Chang
Department of Industrial Engineering and Management,
Yuan-Ze University, Jongli 320, Taiwan

***Abstract*—This paper studies the unrelated parallel machine scheduling problem with three minimization objectives – makespan, maximum earliness, and maximum tardiness (MET-UPMSP). The last two objectives combined are related to just-in-time (JIT) performance of a solution. Three hybrid algorithms are presented to solve the MET-UPMSP: reactive GRASP with path relinking, dual-archived memetic algorithm (DAMA), and SPEA2. In order to improve the solution quality, min-max matching is included in the decoding scheme for each algorithm. An experiment is conducted to evaluate the performance of the three algorithms, using 100 (jobs) x 3 (machines) and 200 x 5 problem instances with three combinations of two due date factors – tight and range. The numerical results indicate that DAMA performs best and GRASP performs second for most problem instances in three performance metrics: HVR, GD, and Spread. The experimental results also show that incorporating min-max matching into decoding scheme significantly improves the solution quality for the two population-based algorithms. It is worth noting that the solutions produced by DAMA with matching decoding can be used as benchmark to evaluate the performance of other algorithms.**

***Keywords-Greedy randomized adaptive search procedure; memetic algorithms; multi-objective combinatorial optimization; unrelated parallel machine scheduling; min-max matching***

## I. INTRODUCTION

In production scheduling, management concerns are often multi-dimensional. In order to reach an acceptable compromise, one has to measure the quality of a solution on all important criteria. This concern has led to the development of multi-criterion scheduling [1]. During scheduling, consideration of several criteria will provide the decision maker with a more practical solution. In production scheduling, objectives under considerations often include system utilization or makespan, total machining cost or workload, JIT related costs (earliness and tardiness penalties), total weighted flow time, and total weighted tardiness. The goal of total weighted flow time is to lower the work-in-process inventory cost during the production process, while the goal of just-in-time is to minimize producer and customer dissatisfactions towards delivery due dates.

Parallel machine models are a generalization of single machine scheduling, and a special case of flexible flow shop. Parallel machine models can be classified into three cases: identical, uniform, and unrelated (UPMSP). In the UPMSP case, machine *i* may finish job 1 quickly but will require much longer with job 2; on the other hand, machine *j* may finish job 2 quickly but will take much longer with job 1. In practice, UPMSPs are often encountered in production environments; for instance, injection modeling and LCD manufacturing [2], wire bonding workstation in integrated-circuit packaging manufacturing [3], etc. Moreover, many manufacturing processes are flexible flow shops (FFS) which are composed of UPMSP at each stage: PCB assembly and fabrication [4-6], ceramic tile manufacturing Ruiz and Maroto [7]. Jungwattanakit et al. [8] proposed a genetic algorithm (GA) for FFS with unrelated parallel machines and a weighted sum of two objectives – makespan and number of tardy jobs. The numerical results indicate that the GA outperforms dispatching rule-based heuristics. Davoudpour and Ashrafi [9] employed a greedy random adaptive search procedure (GRASP) to solve the FFS with a weighted sum of four objectives.

Over the years, UPMSPs with a single objective have been widely studied. For a survey of parallel machine scheduling on various objectives and solution methods, we refer to Logendran et al. [10] and Allahverdi et al. [11]. In contrast, there are relatively few studies on UPMSPs considering multiple objectives. T'kindt *et al*. [12] studied an UPMSP glass bottle manufacturing, with the aim of simultaneously optimizing workload balance and total profit. Cochran et al. [13] introduced a two-phase multi-population genetic algorithm to solve multi-objective parallel machine scheduling problems. Gao [14] proposed an artificial immune system to solve the UPMSPs to simultaneously minimizing the makespan, total earliness and tardiness penalty. For further references regarding multicriteria UMPSPs, refer to Hoogeveen [1].

In this paper, we consider a multi-objective unrelated parallel machine scheduling problems aiming to simultaneously minimize three objectives – makespan, maximum earliness, and maximum tardiness. Hereafter we shall refer to this problem as MET-UPMSP, where the latter two objectives are used to evaluate the just-in-time performance of a schedule.

This paper is organized as follows: Section 2 describes the problem MET-UPMSP; Section 3 presents the algorithms for MET-UPMSP; Section 4 introduces several performance metrics and analyzes experimental results; Section 5 provides concluding remarks.

## II. PROBLEM DESCRIPTION

The MET-UPMSP has the following features: (1) the problem contains *M* unrelated parallel machines and *J* jobs; (2) each job has its own due date, and may also have a different processing time depending on the machine assigned; (3) each machine is allowed to process one job at a time, where the processing is non-preemptive; (4) setup times are job sequence- and machine-dependent. The following are notations and mathematical model for the MET-UPMSP.

### A. Notations:

$m$: machine index, $m = 1,\ldots, M$
$j$: job index, $j = 1,\ldots, J$
$p_{jm}$: processing time of job $j$ on machine $m$
$s_{ijm}$: setup time of job $j$ following job $i$ on machine $m$
$d_j$: due date of job $j$

### B. Decision variables:

$x_{ijm} = 1$ if both jobs $i$ and $j$ are processed on machine $m$, and job $i$ immediately precedes job $j$; otherwise, $x_{ijm} = 0$.
$C_j$ = completion time of job $j$
$E_j$ = earliness of job $j$; $E_j = max\{0, d_j - C_j\}$
$T_j$ = tardiness of job $j$; $T_j = max\{0, C_j - d_j\}$
$C_{max}$ = production makespan
$E_{max}$ = maximum earliness
$T_{max}$ = maximum tardiness

### C. Mathematical model:

$$Minimize\ (f_1, f_2, f_3) = (C_{max}, E_{max}, T_{max}) \quad (1)$$

*s.t.*

$$\sum_{i=0, i\neq j}^{J} \sum_{m=1}^{M} x_{ijm} = 1 \quad (2)$$

$$\sum_{j=1}^{J} x_{0jm} = 1 \quad m = 1, \ldots, M \quad (3)$$

$$C_j - \left(C_i + p_{jm} + s_{ijm}\right) + M_{big}\cdot\left(1 - x_{ijm}\right) \geq 0$$

$$i, j = 0,1,\ldots, J, j \neq i; m = 1, \ldots, M \quad (4)$$

$$C_{max} \geq C_j \quad j = 1,\ldots, J \quad (5)$$

$$T_{max} \geq T_j;\ T_j \geq C_j - d_j;\ T_j \geq 0;\ C_j \geq 0 \quad j = 1,\ldots, J \quad (6)$$

$$E_{max} \geq E_j;\ E_j \geq d_j - C_j;\ E_j \geq 0 \quad j = 1,\ldots, J \quad (7)$$

$$x_{ijm} = 0,1 \quad i, j = 0, 1,\ldots, J; i \neq j; \mu = 1,\ldots, M \quad (8)$$

In the model, equation (1) shows the three objectives. Constraint set (2) restricts job sequence and machine assignment. Constraint set (3) ensures that each machine has the first job. Constraint set (4) specifies the relationships between the finish and start times of jobs processed on the same machine, where $M_{big}$ is a sufficiently large number; the inequality is invalid if jobs $i$ and $j$ are not processed on the same machine and/or job $i$ does not immediately precede job $j$. Constraint (5) specifies that the production makespan must not be smaller than the finish time of any job. Constraint set (6) defines the tardiness of a job and the maximum tardiness of all jobs. Constraint set (7) defines the earliness of a job and the maximum earliness among all jobs. Constraint set The MET-UPMSP is strongly NP-hard since the single machine scheduling problem with the objective of minimizing makespan, $1 \mid s_{jk} \mid C_{max}$, is strongly NP-hard.

## III. SOLVING MET-UPMSP

We present three algorithms to solve MET-UPMSP: GRASP (greedy randomized adaptive search procedure) [15-17], dual-archived memetic algorithm (DAMA), and SPEA2 [18]. To enhance the solution quality, min-max matching is included in the decoding scheme for each generated solution.

### A. GRASP

We present three algorithms to solve MET-UPMSP: GRASP (greedy randomized adaptive search procedure) [15-17], dual-archived memetic algorithm (DAMA), and SPEA2 [18]. To enhance the solution quality, min-max matching is included in the decoding scheme for each generated solution.

$$g_j(i,m,t) = 1/p_{jm}\cdot exp\{-max\left(0, \frac{D - p_{jm} - t}{k_1\cdot\bar{p}_m}\right)\}\cdot$$

$$exp\{-max\left(0, \frac{d_j - p_{jm} - t}{k_1\cdot\bar{p}_m}\right)\}\cdot exp\{-\frac{s_{ijm}}{k_2\cdot\bar{s}_m}\}\cdot$$

$$exp\{-max\left(0, \frac{p_{jm} - t - d_j}{k_1\cdot\bar{p}_m}\right)\} \quad (9)$$

Where $p_{jm}$ is the processing time of job $j$ on machine $m$, $d_j$ is the due date of job $j$, $\bar{p}_m$ and $\bar{s}_m$ are the average processing time of the remaining jobs if they are processed on machine $m$, $k_1$ is the due-date related scaling parameter and $k_2$ the setup time related scaling parameter. $D$ is the estimated makespan $(\beta\cdot\bar{s} + \bar{p})\cdot\mu$, where $\mu$ is the total number of jobs divided by the total number of machines, $\bar{s}$ is the mean setup time, and $\beta = 0.4 + 10/\mu^2 - \eta/7$. The parameters $k_1$ and $k_2$ can be regarded as functions of three factors: (1) the due date tightness factor $\tau$; (2) the due date range factor $R$; (3) the setup time severity factor $\eta = \bar{s}/\bar{p}$.

$k_1 = 4.5 + R$ for $R \leq 0.5$ and $k_1 = 6 - 2R$ for $R \geq 0.5$

$k_2 = \tau/(2\sqrt{\eta})$

#### 1) Construction of the RCL

The greedy functions defined above are the larger the better. At any GRASP iteration step, a job $j$ is selected using roulette method from the restricted candidate list (RCL), in which each element has a greedy function value within the interval, $[g_{min} + (1 - \alpha)\cdot g_{max},\ g_{max}]$, where $g_{min} = Min_{j\in C}\{g_j\}$, $g_{max} = Max_{j\in C}\{g_j\}$.

#### 2) Reactive GRASP

In the construction phase, reactive GRASP is used, rather than basic GRASP. Prais and Ribeiro [21] showed that using a single fixed value for RCL parameter $\alpha$ often hinders finding a high-quality solution, which could be found if another value

was used. Another drawback of the basic GRASP is the lack of learning from previous searches. In our Reactive GRASP, a set of parameter $\alpha$ values {0.05, 0.1, 0.3, 0.5} is chosen. Originally, each $\alpha_i$ value is used to find constructive solutions for a predetermined number of times. Let *Nd** be the current largest nadir distance, and $A_i$ the current average nadir distance for $\alpha_i$. Define $q_i = A_i / Nd^*$. Then the probability of $\alpha_i$ being chosen is $p_i = q_i / \sum_{k=1}^{4} q_k$.

An experimental result indicates that reactive GRASP outperforms basic GRASP for any fixed $\alpha$ value in {0.05, 0.1, 0.3, 0.5}. In the experiment, three instances of problem size 200 x 5 were generated for each of the three due date parameters: ($\tau$, $R$) = (0.2, 0.8), (0.5, 0.5), and (0.8, 0.2). Each instance has ten replication runs, and each run has 25 restarts, each of which performs 30 local search iterations. Afterward, the average nadir distance of the ten replication runs for each instance is computed, and then the average and standard deviation of results. The result shows that the average nadir distance of the reactive GRASP is larger than that of basic GRASP for MET-UPMSP.

*3) Nadir distance*

The nadir point in the objective space is computed as follows:

$C_{max}^{nd} = \text{Max}\left\{\sum_{k=1}^{[J/M]}(p(\delta_m^1(k)\right) + s\left(\delta_m^2(k)\right) | m = 1, \dots, M\}$ , where $[J/M]$ is the smallest integer which is not smaller than $J/M$, $\delta_m^1$ is the sequence that ranks all job processing times on machine $m$ in decreasing order, $p(\delta_m^1(k))$ is the $k$-th largest processing time for machine $m$, $\delta_m^2$ is the sequence that ranks all job setup times on machine $m$ in decreasing order, $s\left(\delta_m^2(k)\right)$ is the $k$-th sequence setup time.

$E_{max}^{nd} = \max\{d_j | j = 1, \dots, M\} - \min\{p_{jm} | j = 1,\dots, J;\ m = 1,\dots, M\} - \min\{s_{ijm} | i, j = 1,\dots, J;\ m = 1,\dots, M\}$, which is the maximum job due date less the shortest processing time and smallest setup time.

$T_{max}^{nd} = D - \min\{d_j | j = 1, \dots, J\}$, where $D$ is the estimated makespan.

The nadir distance of a solution with objective vector $\underline{\boldsymbol{a}}$ is defined as the Euclidean distance between $\underline{\boldsymbol{a}}$ and nadir point. The neighborhood solution will replace current solution is the nadir distance of the former is greater than that of the latter.

*4) Local Search*

Given a current solution (CS), a neighborhood solution (NS) is generated as follows:

In the CS, select the group-machine pair having the smallest nadir distance, and randomly select another group from the remaining groups. Each group first determines the number of jobs based on a random integer from [1, 0.25·$J/M$]; then randomly select a job set from the two groups for swapping. For each single-machine scheduling, apply 3-opt local search for a number of times. To determine whether NS will replace CS, the following rule is used:

If NS dominates CS, set CS = NS; if CS dominates NS, leave CS unchanged; if NS and CS do not dominate each other, then set the one with a larger nadir distance to be the CS.

To enhance the local search improvement on solution quality, min-max matching is employed. The following describes this matching technique for a partition of jobs $\{G_k: k = 1,\dots, M\}$.

Step 0: Set $S = \varnothing$.

Step 1: For each group-machine pair, $\{G_j, M_k\}$, apply 3-opt to obtain a local optimal solutions with respect to nadir distance, and then compute the corresponding three objectives. Thus, we can obtain an $M$ by $M$ matrix where each element has three objective values $(C_{max}, E_{max}, T_{max})$.

Step 2: Apply min-max matching to each individual objective in the matrix. Let $C_{max}^L$, $E_{max}^L$, $T_{max}^L$ be the corresponding optimal values; let $C_{max}^U$, $E_{max}^U$, and $T_{max}^U$ be the maximum values for the three objectives, respectively. Let $SC = \{C_{max} | C_{max}^L \le C_{max} \le C_{max}^U\}$, $SE = \{E_{max} | E_{max}^L \le E_{max} \le E_{max}^U\}$, and $ST = \{T_{max} | T_{max}^L \le T_{max} \le T_{max}^U\}$.

Step 3: For each configuration of $(C_{max}, E_{max}, T_{max})$ with $C_{max} \in SC, E_{max} \in SE, T_{max} \in ST$, assign a very large value to the cells $(f_1, f_2, f_3)$ in the matrix where $f_1 > C_{max}$, $f_2 > E_{max}$, and $f_3 > T_{max}$. Apply maximum cardinality matching to the resulting matrix. If the maximum matrix is equal to $M$, then set $S = S \cup \{(C_{max}, E_{max}, T_{max})\}$.

Step 4: Compare all elements in $S$ based on Pareto domination. Let $P$ be the set of all non-dominated elements in $S$. Output the set $P$.

*5) Path-Relinking*

The iterative two-phase process of GRASP aims to generate a set of diversified Pareto local optimal solutions that will be stored in an archive. In the final phase, path relinking is applied using these Pareto local optimal solutions to further refine the solution quality. At each iteration, an initiating solution and a guiding solution are drawn from the current archive to perform a PR operation.

Let $Q$ be the number of solutions in the archive. Thus, there are $Q$-1 adjacent solutions. For each pair of adjacent solutions ($x_i$, $x_{i+1}$), backward and forward relinking search procedures will be applied. For each relinking path, a sequence of {1/$p$, 2/$p$, …, $p$-1/$p$} is selected and one point crossover operation is performed based on the position of the encoding list at 1/$p$, …, $p$-1/$p$. Each bi-directional path relinking search will calculate 2($p$-1) solutions. The choice for $p$ will be determined by the number of solutions used in performance comparison of the three algorithms. In GRASP, $p$ is set to 5.

An experiment is conducted to determine the parameter settings for (number of restarts, number of PRs). The experiment tests three problem instances with ($\tau$, $R$) = (0.8, 0.2). Each instance has four combination levels on (number of restarts, number of PRs), and each level is solved with 10 replications. Each restart and each PR will generate 100 solutions. The four combination levels will be compared using the average nadir distance based on 2,000 solutions for each replication. The experimental results indicate that (restart, PR) = (15, 5) and (10, 10) yield approximately the same average

nadir distance. Thus, policy (15, 5) is selected for our GRASP in solving the MET-UPSMP.

### B. Dual-Archived Memetic Algorithm (DAMA)

DAMA is a variant of SPEA2. It differs from SPEA2 in three aspects: (1) population evolves with two archives – elite and inferior, using competitive strategy to produce the population of next generation; (2) fuzzy C-means [22] is applied to maintain archive size; (3) min-max matching is included in decoding scheme. The proposed parallel archived evolutionary algorithm is termed memetic algorithm since min-max matching will serve as an effective local search to improve solution quality for decoding scheme.

#### 1) Encoding and decoding schemes

DAMA and SPEA2 adopt random key list (RKL) as their encoding scheme. For each RKL, the integral value of a cell represents the group to which the job is assigned, and the decimal value ranks job processing order. Fig. 1 presents an example of RKL for 7 jobs on two machines. In the example, the initial processing sequence of jobs for the first group $G_1$ = {5, 2, 6, 3}, and for $G_2$ = {4, 1, 3} according to their decimal values in the RKL. Then the 3-opt local refinement is applied to generate a neighborhood solution for each group-machine pair using nadir distance to decide the current representative solution. The procedure is repeated until a pre-specified number of 3-opt operations have been reached. For the 3-opt local search process of group $G_k$ with machine $m$, nadir point is defined as follows:

$$\text{For } C_{max},\ C_{nadir} = \sum_{j\in G_k} p_{jm} + \sum_{i,j\in G_k} Max\{s_{ijm}\} + \varepsilon \quad (10)$$

$$\text{For } E_{max},\ E_{nadir} = \sum_{j\in G_k} Max\{0, d_{jm} - (p_{jm} + s_{0jm})\} + \varepsilon \quad (11)$$

$$\text{For } T_{max},\ T_{nadir} = C_{nadir} - Min_{j\in G_k}\{d_{jm}\} \quad (12)$$

| job | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| RKL | 2.67 | 1.32 | 1.92 | 2.28 | 1.25 | 1.68 | 2.88 |

Figure 1. Random key list encoding scheme

Besides maintaining one efficient archive ($EA_t$) at each generation $t$ to assist in algorithm convergence, the DAMA uses an inefficient archive ($IA_t$) to prevent premature convergence, and enable the memetic algorithm to explore solutions in an extensive space. At each generation, two parallel memetic procedures collectively produce the subsequent population: one procedure applies memetic operation (recombination followed by min-max matching) on the union of $GP_t$ and $EA_t$, and the other procedure applies memetic operation to the union of $GP_t$ and $IA_t$. In the recombination operation, each cell of the child will take the value from parent 1 if the sum of the two parents' decimal values in the same cells exceeds one; otherwise, it will take the value from parent 2. The following illustrates the DAMA algorithm.

Step1 Initialization: Randomly generate initial population $GP_0$; decode $GP_0$ and compute respectively the first and the last non-dominated front, $F_1(GP_0)$ and $F_L(GP_0)$; set $EA_0 = F_1(GP_0)$, $IA_0 = F_L(P_0)$, $r_0$; set $U_1$ , $U_2$, and $U_3$ as the worst of $f_1$, $f_2$, and $f_3$ in $IA_0$ respectively; set $t = 0$.

Step 2 Fitness assignment: Calculate fitness values of individuals in ($GP_t \cup EA_t$) and ($GP_t \cup IA_t$), respectively.

Step 3 Generate population $GP_{t+1}$:

Step 3.1 Perform crossover on ($GP_t \cup EA_t$): Produce [$r \cdot N$] offspring from ($GP_t \cup EA_t$) by crossover operation using binary tournament for mating selection. Decode each offspring.

Step 3.2 perform ($GP_t \cup IA_t$): Produce $N$ – [$r \cdot N$] offspring from ($GP_t \cup EA_t$) by the same method in Step 3.1.

Step 4: Update of $EA_{t+1}$ and $IA_{t+1}$

Step 4.1: Compute $F_1(GP_{t+1})$ and copy into $EA_t$; update $EA_{t+1}$. If $|EA_{t+1}| > \bar{N}$, trim $EA_{t+1}$ to size $\bar{N}$ by FCM.

Step 4.2: Compute $F_L(GP_{t+1})$ and copy it into $IA_t$, update $IA_{t+1}$. If If $|IA_{t+1}| > \bar{N}$, trim $IA_{t+1}$ to size $\bar{N}$ by FCM.

Step 5: Compute $r_{t+1}$ according to the following equation.

$r_{t+1}$ = |(crossover on ($GP_t \cup EA_t$)$r_t \cap F_1(GP_{t+1})$| /(|$F_1(GP_{t+1})$|+$\rho$)

Step 6: $t = t+1$; if $t = T$, proceed to Step 7; otherwise, return to Step 2.

Step 7: If the number of restarts is not over, proceed to Step 0; otherwise, output global non-dominated set A from all $EA_T$.

#### 2) Fitness assignment

Generally, the fitness assignment for ($GP_t \cup EA_t$) follows SPEA2 [18] on minimization problems, and the fitness assignment for ($GP_t \cup IA_t$) follows SPEA2 on maximization problems. The fitness assignment considers domination and diversity factors. For DAMA, a modification is made on diversity measure because the problem under study is discrete.

## IV. NUMERICAL RESULTS

An experiment was conducted to investigate the performance of the proposed algorithms. All algorithms were coded in Visual Studio C++.NET 2008, and implemented on a computer with Intel (R) core (TM) i5-2400@3.1 GHz and 4 GB DDR3.

### A. Parameter settings

Population and archive sizes of DAMA and SPEA2 are $N$ = 20, $\bar{N}$= 20, maximum iterations = 100, no. of restarts = 7. The competitive ratio of DAMA is $r_0$ = 0.9. For GRASP, we set (no. restart, no. PR) = (15, 5). All algorithms were executed 10 replications for each instance. The performances of algorithms with min-max matching are compared based on the same number of matching iterations. Finally, the effect of including min-max matching in the decoding scheme will also be discussed.

### B. Generating test instances

Two problem sizes are considered in this experiment: 100 (jobs) x 3 (machines), and 200 x 5. We shall refer to the former as large size and the latter as moderate size. For each problem size, three test sets each consisting of three instances, were generated according to Lee and Pinedo [19]. Each test instance is denoted by four characters: *AB0n*. The first character *A*

represents problem size: "L" for large and "M" for moderate. The second character *B* represents due date tightness: "L" for loose due date factors ($\tau$, $R$) = (0.2, 0.8), "M" for moderate ($\tau$, $R$) = (0.5, 0.5), and "T" for tight ($\tau$, $R$) = (0.8, 0.2). Finally, the last two characters *0n* represent the problem instance index. The larger the problem size, the more complex the problem; the tighter the due date factors, the more difficult the problem. Thus, "LM" problems will be the easiest to solve and "LT" problems will be the most difficult. Table 1 shows the data sets.

TABLE I. TEST INSTANCES INFORMATION

| Problem size | test instances with ($\tau$, $R$) | | |
|---|---|---|---|
| | (0.2, 0.8) | (0.5, 0.5) | (0.8, 0.2) |
| 100 x 3 | ML01-03 | MM01-03 | MT01-03 |
| 200 x 5 | LL01-03 | LM01-03 | LT01-03 |

### C. Performance metrics

When developing an algorithm to solve multi-objective optimization problems, diverse evaluation techniques are required to measure algorithm performance. Generally speaking, performance metrics are classified into three categories: Proximity, Diversity, and both. The following are several metrics used in our research.

#### 1) Proximity

This metric evaluates the total distance between the local Pareto optimal front generated by an algorithm and globally Pareto-optimal front. We consider a commonly used proximity metric, GD (generational distance).

$GD(A) = \sum_{i \in A} d_i / |A|$, where $A$ is the set of non-dominated solutions generated by algorithm, $|A|$ is the number of solutions, and $d_i$ is the distance of objective values of solution $i$ to the nearest Pareto front point.

#### 2) Diversity

Diversity is another important characteristic for measuring the quality of a non-dominated set. One popular metric for diversity is *Spread* [23], which calculates a relative minimum distance between local Pareto-optimal front elements. This metric also considers the extent of the spread and requires a reference Pareto front set $P_r$ to be computed. For three-objective problems, *Spread* will be computed using minimum spanning tree which involves three shortest distances from the local Pareto-optimal front elements A to the three planes.

$Spread(A) = (\sum_{i=1}^{3} d_i^e + \sum_{j \in A} |d_j - \bar{d}|)/(\sum_{i=1}^{3} d_i^e + |A| \cdot \bar{d})$, where $\sum_{i=1}^{3} d_i^e$ is the shortest distance from $A$ to X-Y, X-Z, and Y-Z planes, $\sum_{i \in A} d_i$ is the total distance of the minimum spanning tree for A, and $\bar{d}$ is the mean distance counting all |A| + 2 arcs.

#### 3) Proximity and diversity

Zitzler and Thiele [24] introduced a hypervolume (HV) metric which can measure both proximity and diversity. A nadir point is required to calculate the HV metric. It is clear to observe that if point $\underline{a}$ dominates point $\underline{b}$, then the volume of $\underline{a}$ must be greater than that of $\underline{b}$. Let $A = \{\underline{a}_1, \ldots, \underline{a}_q\}$. The better the quality of A in proximity and diversity, the larger the HV of *A*.

HV = volume $(\bigcup_{i=1}^{|A|} h_i)$, where $h_i$ is the hypercube of $\underline{a}_i$ in *A*.

For three-objective case, the following formula can be applied to calculate HV for *A*. Let $v(h_i)$ be the volume of $\underline{a}_i$.

$$\mathrm{HV}(A) = \sum_{i=1}^{|A|} v(h_i) - \sum_{i \neq j} v(h_i \cap h_j) + \cdots + (-1)^{|A|+1} \cdot v(\cap_{i=1}^{|A|} h_i) \quad (13)$$

The calculation will be time-consuming if the set *A* contains a large number of elements. In our algorithms, the archive size is limited to 20. The computation time is acceptable.

HVR(*A*) (hypervolume rate) is defined as $\mathrm{HV}(A)/\mathrm{HV}(P_r)$, where $P_r$ is the reference Pareto front set obtained by comparing the local non-dominated solutions produced by all algorithms.

### D. Performance comparisons

TABLEs II and III present the HVR performance of SPEA2, DAMA, and GRASP on medium- and large-sized problem instances. In the tables, the symbol "$\beta$" in $\beta(\gamma)$ represents the performance where the min-max matching technique is not used, and "$\gamma$" represents the performance where matching technique is applied. For example, the values 37.4(63.8) located in ML column and DAMA(M) row of TABLE II indicate that HVR is 37.4% for DAMA without matching-based decoding, and HVR is improved to 63.8% for DAMA with matching. From TABLEs II and III, SPEA2 and DAMA with matching-based decoding considerably improve solution quality. However, GRASP does not reveal much advantage when matching is applied. For example, in MT instances, SPEA2 improves HVR from 27.5% to 84.9%, DAMA from 28.2% to 88.6%, but GRASP only from 56.5% to 59.7%.

GRASP performs best among all algorithms without matching, and there is little improvement for GRASP without matching. This indicates that GRASP is able to produce high quality solutions. However, for SPEA2 and DAMA, the effect of matching is significant, particularly for tight due-date instances. In summary, DAMA with matching (DAMA_M) is superior to the others in terms of HVR metric.

TABLE II. HVR (%) OF ALGORITHMS ON 100 X 3 TEST SETS

| | ML | MM | MT |
|---|---|---|---|
| SPEA2 (M) | 35.4 (59.7) | 37.7 (61.1) | 27.5 (84.9) |
| DAMA (M) | 37.4 (63.8) | 37.5 (71.4) | 28.2 (88.6) |
| GRASP (M) | 56.3 (58.0) | 46.0 (50.0) | 56.5 (59.7) |

TABLE III. HVR (%) OF ALGORITHMS ON 200 X 5 TEST SETS

| | LL | LM | LT |
|---|---|---|---|
| SPEA2 (M) | 35.3 (56.0) | 33.9 (55.3) | 31.7 (84.3) |
| DAMA (M) | 35.8 (60.2) | 29.3 (60.3) | 30.3 (85.2) |
| GRASP (M) | 71.2 (73.2 | 52.9 (63.0) | 66.9 (68.7) |

TABLEs IV-V display GD performance of the algorithms. For 100 x 3 instances (TABLE VI), DAMA_M performs best for all three types of instances. GRASP_M is little better than GRASP, but both perform well for ML. For 200 x 5 instances, GRASP_M performs best for LL and LM instances. However, for LT instances, DAMA_M is superior in GD performance. From the entries of TABLE VII, we can conclude that SPEA2_M, DAMA_M, GRASP, and GRASP_M produce local solutions which are close to the reference set. The value behind the sign "±" is standard deviation.

TABLE IV. GD PERFORMANCE OF ALGORITHMS ON 100 X 3 TEST SETS

| | ML | MM | MT |
|---|---|---|---|
| SPEA2 | 3.8±1.1E-02 | 2.6±0.8E-02 | 9.2±2.2E-02 |
| SPEA2_M | 1.1±0.27E-02 | 9.4±4.1E-03 | 1.1±0.36E-02 |
| DAMA | 5.3±1.8E-02 | 2.2±0.4E-02 | 7.6±1.4E-02 |
| DAMA_M | 4.5*±1.1E-03 | 5.3*±0.85E-03 | 5.1±1.7E-03* |
| GRASP | 9.5±2.5E-03 | 1.8±0.76E-02 | 7.4±1.2E-02 |
| GRASP_M | 7.7±3.0E-03 | 1.4±0.74E-02 | 5.0±1.1E-02 |

TABLE V. GD PERFORMANCE OF ALGORITHMS ON 200 X 5 TEST SETS

| | LL | LM | LT |
|---|---|---|---|
| SPEA2 | 3.6±0.69E-02 | 2.6±0.73E-02 | 3.2±0.76E-02 |
| SPEA2_M | 8.4±1.2E-03 | 4.9±0.7E-03 | 7.0±1.4E-03* |
| DAMA | 5.0±0.42E-02 | 3.2±1.6E-02 | 4.3±1.6E-02 |
| DAMA_M | 9.6±1.4E-03 | 4.8±1.1E-03 | 1.1±0.81E-02 |
| GRASP | 3.8±2.7E-03 | 4.3±1.2E-03 | 2.0±0.69E-02 |
| GRASP_M | 2.6*±1.5E-03 | 2.3±0.4E-03* | 1.4±0.31E-02 |

TABLEs VI and VII present the *Spread* performance of the algorithms. *Spread* measures the diversity of the local solutions generated by an algorithm. A small *Spread* value indicates that the local solutions are more uniformly distributed. For 100 x 3 instances, DAMA_M generates more evenly distributed local solutions than the other algorithms. GRASP_M performs second best. For 200 x 5 instances, DAMA_M is superior to the others. In contrast, SPEA2_M performs next and generates Spread values closest to the best for every type of instances. From the entries of TABLEs VI and VII, we observe that using matching decoding will produce better distributed local solutions than not using. The gap of the *Spread* values is significant when problem size increases.

## V. CONCLUSION

Parallel machine scheduling are often observed in production environment, and the goal that production management wishes to achieve is often multi-fold. This paper studies unrelated parallel machine scheduling problems with three minimization objectives: makespan, maximum earliness, and maximum tardiness.

Three algorithms are presented to solve this problem: GRASP, DAMA, and SPEA2. Our numerical results indicate that GRASP outperforms the other two algorithms without the min-max matching technique, but the performance improvement is not significant when the min-max matching is used. In contrast, the two population-based algorithms, SPEA2 and DAMA, including min-max matching in the decoding scheme will significantly improve the solution quality. Although the DAMA with matching-based decoding scheme requires more computation time, it will produce high quality solutions, which can be used as comparison standard to evaluate the performance of other algorithms.

TABLE VI. *SPREAD* OF ALGORITHMS ON 100 X 3 TEST SETS

| | ML | MM | MT |
|---|---|---|---|
| SPEA2 | 0.88 ± 0.05 | 0.82 ± 0.08 | 0.86 ± 0.08 |
| SPEA2_M | 0.72 ± 0.03 | 0.55 ± 0.06 | 0.64 ± 0.07 |
| DAMA | 0.91 ± 0.06 | 0.86 ± 0.07 | 0.81 ± 0.01 |
| DAMA_M | 0.63* ± 0.11 | 0.64 ± 0.03 | 0.50* ± 0.03 |
| GRASP | 0.73 ± 0.08 | 0.63* ± 0.06 | 0.72 ± 0.04 |
| GRASP_M | 0.66 ± 0.11 | 0.56 ± 0.07 | 0.69 ± 0.01 |

TABLE VII. *SPREAD* OF ALGORITHMS ON 200 X 5 INSTANCES

| | LL | LM | LT |
|---|---|---|---|
| SPEA2 | 1.00 ± 0.07 | 0.97 ± 0.07 | 0.90 ± 0.09 |
| SPEA2_M | 0.60 ± 0.02 | 0.52 ± 0.04 | 0.84 ± 0.14 |
| DAMA | 0.96 ± 0.04 | 0.94 ± 0.12 | 0.89 ± 0.04 |
| DAMA_M | 0.61* ± 0.03 | 0.49* ± 0.01 | 0.79* ± 0.05 |
| GRASP | 0.83 ± 0.06 | 0.70 ± 0.07 | 0.89 ± 0.02 |
| GRASP_M | 0.70 ± 0.15 | 0.68 ± 0.08 | 0.81 ± 0.06 |

## ACKNOWLEDGMENT

This work was supported by the National Science Council of Taiwan under grant NSC 99-2221-E-155-029.

## REFERENCES

[1] H. Hoogeveen, "Multicriteria scheduling," Eur. J. Oper. Res., vol. 167, iss.3, pp. 592-623, 2005.

[2] J. F. Chen, "Scheduling on unrelated parallel machines with sequence- and machine-dependent setup times and due-date constraints," Int. J. Adv. Manuf. Technol., vol.44, iss.11-12, pp. 1204-1212, 2009.

[3] D. Yang, "An evolutionary simulation-optimization approach in solving parallel-machine scheduling problem – A case study," Comput. Ind. Eng., vol.56, iss.3, pp. 1126-1136, 2009.

[4] D. Alisantoso, L. P. Khoo, and P. Y. Jiang, "An immune algorithm approach to the scheduling of a flexible PCB flow shop," Int. J. Adv. Manuf. Technol., vol.22, iss.11-12, pp. 819-827, 2003.

[5] J. C. Hsieh, P. C. Chang, and L. C. Hsu, "Scheduling of drilling operations in printed circuit board factory," Comput. Ind. Eng., vol.44, iss.3, pp. 461-473, 2003.

[6] L. Yu, H. M. Shih, M. Pfund, W. M. Carlyle, and J. W. Fowler, "Scheduling of unrelated parallel machines- An application to PWB manufacturing," IIE Trans., vol.34, iss.11, pp. 921-931, 2004.

[7] R. Ruiz, and C. Maroto, "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility," Eur. J. Oper. Res., vol.169, iss.3, pp. 781-800, 2006.

[8] J. Jungwattanakit, M. Reodecha, P. Chaovalitwongse, and F. Werner, "Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria," Int. J. Adv. Manuf. Technol., vol.37, iss.3-4, pp. 354-370, 2008.

[9] H. Davoudpour, and M. Ashrafi, "Solving multi-objective SDST flexible flow shop using GRASP algorithm," Int. J. Adv. Manuf. Technol., vol.44, iss.7-8, pp. 737-747, 2009.

[10] R. Logendran, B. McDonell, and B. Smucker, "Scheduling unrelated parallel machines with sequence-dependent setups," Comput. Oper. Res., vol.34, iss.11, pp. 3420-3438, 2007.

[11] C. T. N. Allahverdi, T. C. E. Ceng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," Eur. J. Oper. Res., vol.187, iss.3, pp. 985-1032, 2008.

[12] V. T'kindt, J. C. Billaut, and C. Prouse, "Solving a bicriteria scheduling problem on unrelated parallel machines occurring in the glass bottle industry," Eur. J. Oper. Res., vol.135, iss.1, pp. 42-49, 2001.

[13] J. K. Cochran, S. M. Horng, and J. W. Fowler, "A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines," Comput. Oper. Res., vol.30, iss.7, pp. 1087-1102, 2003.

[14] J. Q. Gao, "A novel artificial immune system for solving multiobjective scheduling problems subject to special process constraint," Comput. Ind. Eng., vol.58, iss.4, pp. 602-609, 2010.

[15] T. A. Feo, and M. G. C. Resende, "Greedy randomized adaptive search procedures," J. Global. Optim., vol.6, iss.2, pp. 109-134, 1995.

[16] M. G. C. Resende, and C. C. Ribeiro, "Greedy randomized adaptive search procedures," in: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer, pp. 219-249, 2003.

[17] V. A. Armentano, and M. F. de Franca Filho "Minimizing total tardiness in parallel machine scheduling with setup times: An adaptive memory-based GRASP approach," Eur. J. Oper. Res., vol.183, iss.1, pp. 100-114, 2007.

[18] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm, Technical report," Comput. Eng. Netw. Lab. (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.

[19] Y. H. Lee, and M. Pinedo, "Scheduling jobs on parallel machines with sequence-dependent setup times," Eur. J. Oper. Res., vol.100, iss.3, pp. 464-474, 1997.

[20] M. Pinedo, Scheduling Theory, Algorithms and Systems (2nd edition), Prentice-Hall, Inc., A Simon & Schuster Company Englewood Cliffs, New Jersey, p 36, 2008.

[21] M. Prais, and C. C. Ribeiro, "Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment," INFORMS J. Comput., vol.12, iss.3, pp. 164-176, 2000.

[22] J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York, 1981.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," IEEE Trans. Evol. Comput., vol.6, iss.2, pp. 182-197, 2002.

[24] E. Zitzler, and L. Thiele, "Multiobjective optimization using evolutionary algorithms—A comparative case study," 5th Int. Conf. Parallel Problem Solving from Nature (PPSN-V), In: A. E. Eiben, T. B¨ack, M. Schoenauer, H. P. Schwefel (Eds). Berlin, Germany: Springer-Verlag, Lecture Notes in Computer Science, vol.1498, pp. 292–301, 1998.

## AUTHORS PROFILE

Chiuh-Cheng Chyu is currently an associate professor of the department of Industrial Engineering and Management at Yuan-Ze University. His current research interests are in the areas of applied operations research, multiple criteria decision-making, scheduling, and meta-heuristics for combinatorial optimization problems.

Wei-Shung Chang obtained his PhD degree from the Department of Industrial Engineering and Management at Yuan-Ze University, Chung-Li, Taiwan. His research interests include meta-heuristics for production scheduling and combinatorial optimization problems.