www.palgrave-journals.com/jors/

Fast and effective algorithms for the liquid crystal display module (LCM) scheduling problem with sequence-dependent setup time

SH Chung*, WL Pearn and YT Tai

National Chiao Tung University, Hsinchu, Taiwan, ROC

The liquid crystal display module scheduling problem (LCMSP) is a variation of the classical parallel machines scheduling problem, which has many real-world applications, particular, in the thin film transistor liquid crystal display (TFT-LCD) manufacturing industry. In this paper, we present a case study on the LCMSP, which is taken from a final liquid crystal display module (LCM) shop floor in a TFT-LCD industry. For the case we investigated, the jobs are clustered by their product types and the machine setup time is sequentially dependent on the product types of the jobs processed on the machine. In LCMSP, the objective is to maximize the total profit subject to fulfilling contracted quantities without violating the due date and machine capacity restrictions. The LCMSP can be modelled as a multi-level optimization problem. The sub-problem of LCMSP can be transformed into the vehicle routing problem with time window (VRPTW). One can therefore solve the LCMSP efficiently using existing VRPTW algorithms. We present two new algorithms based on the savings algorithms with some modifications to accommodate the LCMSP. Based on the characteristics of the LCM process, a set of test problems is generated covering most of the real-world applications for test purposes. Computational results and performance comparisons show that the proposed algorithms solved the LCMSP efficiently and near-optimally.

Journal of the Operational Research Society (2009) **60,** 921–933. doi:10.1057/palgrave.jors.2602604
Published online 4 June 2008

Keywords: parallel machines scheduling; profit; sequence dependence setup time

Introduction

The liquid crystal display module scheduling problem (LCMSP) is a variation of the classical parallel-machine scheduling problem considered by So (1990), Shin and Leon (2004), and Jeong *et al* (2001), which has many real-world applications, particularly, in the thin film transistor liquid crystal displays (TFT-LCD) manufacturing industry. The major three stages of processes for the TFT-LCD manufacturing include TFT array fabrication, LCD assembly, and liquid crystal display module (LCM). The LCM is the final stage of the whole process. It assembles customer-specified components into the cells, which are received from the LCD assembly stage. In the LCM stage, the varied contracted quantities, due date, product profits, and long setup time of critical resources (printed circuit board bonding) result in the determination of jobs and setups sequence becomes more difficult. Therefore, developing fast and efficient scheduling algorithms to maximize the profit without violating customer due date and enhance the utilization of the critical resources is essential.

Correspondence: SH Chung, Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu, Taiwan, ROC. E-mail: shchung@mail.nctu.edu.tw

In this paper, we consider a case study on the LCMSP in the LCM factory with sequence dependence setup time and multiple profits. The case we investigated is taken from a module assembly process of a TFT-LCD factory in Hsinchu science-based industrial park, Taiwan. For the case we investigated, the jobs are clustered by their product types, which must be processed on any parallel machines and be completed before the due date. Setup times between two consecutive jobs of different product types on the same machine are sequentially dependent. The job processing time may vary, depending on the product type of the jobs processed on. Furthermore, the hybrid market consists of contract and spot markets in TFT-LCD factories. The contract jobs, associated with contracted prices and due dates, have been placed. The remaining capacity is to be sold into the spot market. The further spot prices are uncertain and high price volatility due to today's fierce competitive environment. Since the LCMSP involved different product profits, with constraints on sequence dependence setup time, product-type-dependent processing time, due date and machine capacity, it is more difficult to solve than the classical parallel machine scheduling problem.

So (1990) considered a simple version of the LCMSP with sequence-independent setup time. He presented three

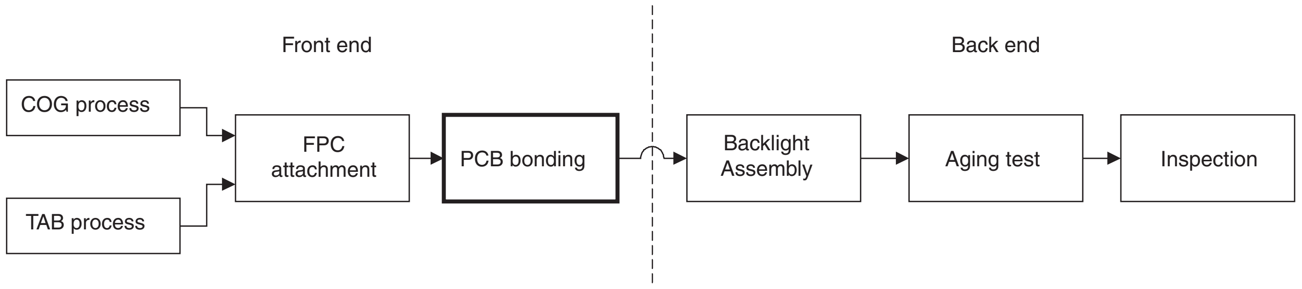

Figure 1 The six steps in the LCM process.

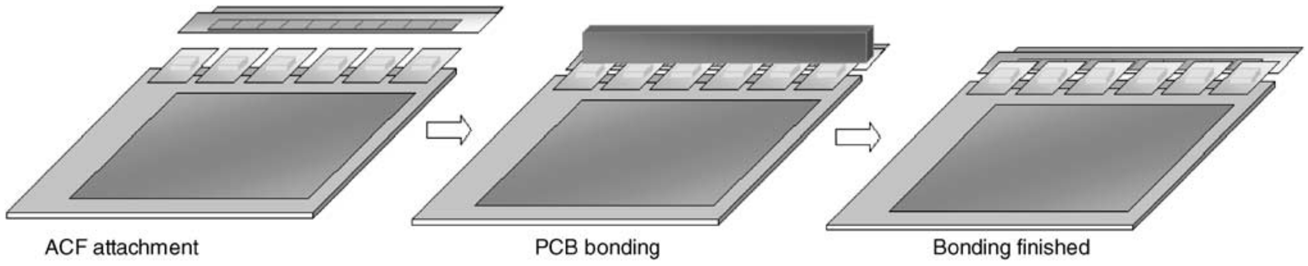

Figure 2 PCB bonding illustration.

heuristics to solve the problem approximately. However, the setup time in his model was only characterized by two magnitudes, minor and major. He only tackled a problem involving minor setup times between jobs of the same family and major setup times between jobs of different groups. Further, he did not consider the due date restriction, which is essential in practical factories. Tovia *et al* (2004) considered a simple version of the LCMSP, which is a parallel machine scheduling problem with the objective of maximizing throughput. Tovia *et al* (2004) also presented a mathematical programming model and a rule-based heuristic approach. Unfortunately, their model only includes processing time without considering setup time and due date restrictions, which may not reflect the real situations accurately.

Few works have considered the scheduling problem in TFT-LCD plants. Jeong *et al* (2001) addressed the LCD assembly process, presented mathematical models, and proposed two heuristic algorithms considering mean flow time and demands fulfillment. Unfortunately, their model does not consider the due date restriction and setup time that are essential and critical in real-world situations. Shin and Leon (2004) discussed an analogous version of the LCMSP, which considers family setup time and due date. They proposed two heuristic approaches based on the MULTIFIT method and tabu search to solve the LCMSP approximately. We note that Shin and Leon (2004) have simplified the model by considering family setup time ignoring the fact that the setup times are sequentially dependent on hundreds of product types of job processed. However, none of these investigations considered sequence-dependent setup time and job profits under contract and spot market simultaneously.

The LCMSP we investigated can be modelled as a multi-level optimization problem. At the first level, we schedule the contract jobs to minimize the total setup time without violating machine capacity and customer due date restrictions, which can be solved using algorithms for vehicle routing problem with time windows (VRPTW) (see Pearn *et al*, 2002a,b, 2004a,b). At the second level, we apply a greedy concept to choose a subset of spot jobs then insert into the schedules constructed in the first level. In this paper, we use basic technologies for VRPTW algorithms including parallel, generalized savings algorithms and provide two modifications to solve the LCMSP efficiently. To further analyse the impact of the problem characteristics on the performance of those savings algorithms, we provide a set of test problems, considering the workload level of contract jobs, tightness of due dates, setup time variation, and variation of profit ratio. Exact solutions are used here as convenient reference points for evaluating the accuracy and effectiveness of our heuristic algorithms. The computational experiments and comparisons demonstrate the performance of the three phases of the modified parallel savings algorithm (PSA) outperform the other algorithms.

LCM process and problem description

LCM process

The TFT-LCD applications include monitors, notebook PCs, mobile phones, portable DVDs, LCD TVs, and many others. Such applications are hundreds of product types of job processed with sizes ranging from 1.6 to 46 inches. The manufacturing process of the LCM assembly generally consists of front end and back end two main segments. It contains the following six processes: (1) COG (chip on glass)/TAB (tape automated bonding) process, (2) flexible printed circuit board (FPC) attachment, (3) printed circuit

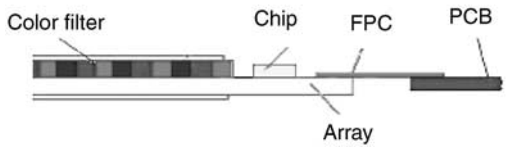

Figure 3 COG structure.

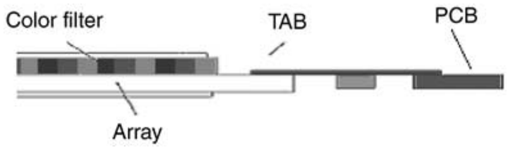

Figure 4 TAB structure.

board (PCB) bonding, (4) back light assembly, (5) aging test, (6) inspection, as presented in Figure 1. For the case investigated, the critical resource is PCB bonding, which has the longest setup time and using anisotropic conductive film (ACF) to connect the PCB and FPC, as illustrated in Figure 2.

The processes of PCB bonding are determined by the types of mount technology of LCD drive IC and controller IC, namely, the COG (chip on glass) and TAB (tape automated bonding). The COG is a technology that mounts the LCD driver to the contact edge of the LCD glass, which is depicted in Figure 3. The TAB is the LCD driver or controller electronics are encapsulated in a thin film, like package, with metal leads extension from the IC chips, which is depicted as Figure 4. The process is also called OLB (Outer Lead Bonding). In general, large size LCD applications (ranging from 15 to 46 inches) adopt the TAB structure and small size LCD applications (ranging from 1.6 to 6 inches) adopt the COG structure.

Setup time between different technologies in PCB bonding is complicated and time consuming. The longest setup time may consume six hours. The setup activities of PCB bonding include the following: cool down temperature, replacement of the appropriate mold, and the rising to a suitable temperature and voltage. Furthermore, setup times between two consecutive jobs of different product types on the same PCB bonding machine are sequentially dependent. In a similar product family, the setup activity may adjust the temperature and voltage. Different LCD panel sizes must replace the mold.

LCMSP

The scheduling problem in this paper can be stated as follows. Let machine group $M = \{m_k | k = 1, 2, \ldots, K\}$, containing K machines, and product types $R = \{R_i | i = 1, 2, \ldots, I\}$, containing I clusters of jobs. Because the hybrid contract and spot sales environment is considered, one product type could be divided into two subsets R_i^P and R_i^q. The contract job set can be shown as $R_i^P = \{r_{ij}^P | j = 1, 2, \ldots, J_i^{\text{contract}}\}$, and the spot sales job set can be shown as $R_i^q = \{r_{ij}^q | j = J_i^{\text{contract}} + 1, J_i^{\text{contract}} + 2, \ldots, J_i\}$. Term J_i^{contract} and J_i is the number of contract jobs and total jobs of product type R_i, respectively. Each job in their associated product type is a candidate to be processed without preemption on a set of parallel machines K.

Table 1 The job information for the seven-job example

Job ID	*Product type*	*Processing time*	*Profit*	*Due date*	*Contract/ spot*
r_{A1}	R_A	21	50	80	C*
r_{A2}	R_A	21	40	100	S†
r_{A3}	R_A	21	40	100	S
r_{B1}	R_B	25	60	93	C
r_{B2}	R_B	25	50	100	S
r_{C1}	R_C	28	63	100	C
r_{C2}	R_C	28	63	60	C

*Term C indicates the contract job.
†Term S indicates the spot job.

Table 2 Setup times required for switching one product type to another for R_A, R_B, and R_C

	To			
From	*U*	R_A	R_B	R_C
U	–	15	15	15
R_A	0	0	10	7
R_B	0	8	0	5
R_C	0	3	16	0

Each job r_{ij} carries with it processing time denoted by p_{ij}, and profit denoted by $profit_{ij}$, where $i = \{1, 2, \ldots, I\}$ and $j = \{1, 2, \ldots, J_i\}$. For each job r_{ij}, b_{ij} represent the ready time of the job to be processed on a machine and e_{ij} represent the latest starting time to process job r_{ij}, which relates to the due dates d_{ij} and can be computed as $e_{ij} = d_{ij} - p_{ij}$. A setup time is incurred in the different product types. When job r_{ij} immediately succeeds job $r_{i'j'}$ on machine m_k, a setup time $s_{ii'}$ happens. The setup time is sequentially dependent on the product types of the jobs processed on the machine. The objective is to maximize the total profit without violating contracted due date and machine capacity, W.

An illustrative example

Consider the following example with two machines and seven jobs clustered into three product types. Table 1 displays the product type, processing time, job profit, due date, and contracted status for each job. Furthermore, the setup times are incurred for switching one product type to another for the three product types R_A, R_B, and R_C. Table 2 shows the required setup times and the term U denotes that the machine is in idle status. The capacity is set to 95 for each machine in the illustrative example.

The objective in the illustrative example is to find a schedule for the subset of jobs, which satisfies the due date restrictions without violating the constraints of machine capacity, while maximizing the total profits.

An integer programming formulation

We formulate the MILP model for the LCMSP. The applicability of the MILP model associated with a sequence-dependent setup time of parallel machine scheduling has been demonstrated by Pearn *et al* (2002b). We modify the objective and some constraints of MILP model to accommodate the LCMSP.

Let x_{ijk} be the variable indicating whether job r_{ij} is scheduled on machine m_k, with $x_{ijk}=1$ if job r_{ij} is scheduled to be processed on machine m_k, and $x_{ijk}=0$ otherwise. Let $y_{iji'j'k}$ be the precedence variable, where $y_{iji'j'k}$ should be set to 1 if job $r_{i'j'}$ is scheduled following job r_{ij} (not necessarily directly), and where $y_{iji'j'k}=0$ otherwise. Let $z_{iji'j'k}$ be the direct-precedence variable, with $z_{iji'j'k}=1$ if job $r_{i'j'}$ is scheduled immediately following job r_{ij} on machine m_k, and $z_{iji'j'k}=0$ otherwise. Further, the starting processing time t_{ijk} should not be less than the ready time and not be greater than the latest starting time e_{ij}. The exact formulation for the LCMSP is as follows.

$$\text{Maximize } Z=\sum_{k=1}^{K}\sum_{i=1}^{I}\sum_{j=1}^{J_i} x_{ijk}\,\mathit{profit}_{ij} \tag{1}$$

subject to

$$\sum_{k=1}^{K} x_{ijk}=1 \quad \text{for } i=1,2,\ldots,I,\quad j=1,2,\ldots,J_i^{\text{contract}} \tag{2}$$

$$\sum_{k=1}^{K} x_{ijk}\leqslant 1 \quad \text{for } i=1,2,\ldots,I,\quad j=J_i^{\text{contract}}+1,\ J_i^{\text{contract}}+2,\ldots,J_i \tag{3}$$

Capacity constraints:

$$\sum_{i=0}^{I}\sum_{j=1}^{J_i} x_{ijk}p_{ij}+\sum_{i=0}^{I}\sum_{j=1}^{J_i}\left(\sum_{i'=0}^{I}\sum_{j'=1}^{J_{i'}} z_{iji'j'k}\,s_{ii'}\right)\leqslant W \quad \text{for all } k \tag{4}$$

Due date constraints:

$$t_{ijk}\geqslant b_{ij}x_{ijk} \quad \text{for all } i,j,k \tag{5}$$

$$t_{ijk}\leqslant e_{ij}x_{ijk} \quad \text{for all } i,j,k \tag{6}$$

Precedence constraints:

$$t_{ijk}+p_{ij}+s_{ii'}-t_{i'j'k}+Q(y_{iji'j'k}-1)\leqslant 0 \quad \text{for all } i,j,k \tag{7}$$

$$t_{ijk}+p_{ij}+s_{ii'}-t_{i'j'k}+Q(y_{iji'j'k}+z_{iji'j'k}-2)\geqslant 0 \quad \text{for all } i,j,k \tag{8}$$

$$(y_{iji'j'k}+y_{i'j'ijk})-Q(x_{ijk}+x_{i'j'k}-2)\geqslant 1 \quad \text{for all } i,j,k \tag{9}$$

$$(y_{iji'j'k}+y_{i'j'ijk})+Q(x_{ijk}+x_{i'j'k}-2)\leqslant 1 \quad \text{for all } i,j,k \tag{10}$$

$$(y_{iji'j'k}+y_{i'j'ijk})-Q(x_{ijk}+x_{i'j'k})\leqslant 0 \quad \text{for all } i,j,k \tag{11}$$

$$(y_{iji'j'k}+y_{i'j'ijk})-Q(x_{i'j'k}+x_{ijk}+1)\leqslant 0 \quad \text{for all } i,j,k \tag{12}$$

$$(y_{iji'j'k}+y_{i'j'ijk})-Q(x_{ijk}+x_{i'j'k}+1)\leqslant 0 \quad \text{for all } i,j,k \tag{13}$$

$$y_{iji'j'k}\geqslant z_{iji'j'k} \quad \text{for all } i,j,k \tag{14}$$

$$\sum_{i=0}^{I}\sum_{j=1}^{J_i} x_{ijk}-\sum_{r_{ij}\neq r_{i'j'}} z_{iji'j'k}=1 \quad \text{for all } k \tag{15}$$

$$y_{iji^*j^*k}+z_{iji^*j^*k}-Q(y_{iji^*j^*k}+z_{iji^*j^*k}-2)-Q(y_{iji'j'k}-z_{iji'j'k}-1)\geqslant 2 \quad \text{for all } i,j,k \tag{16}$$

Binary variables:

$$x_{ijk}\in\{0,1\} \quad \text{for all } i,j,k \tag{17}$$

$$y_{iji'j'k}\in\{0,1\} \quad \text{for all } i,j,k \tag{18}$$

$$z_{iji'j'k}\in\{0,1\} \quad \text{for all } i,j,k \tag{19}$$

The objective function (1) states that the total profit is to be maximized over all machines. Constraint (2) ensures that each contract job is scheduled on one machine exactly. Constraint (3) ensures that each spot job is scheduled on at most one machine. Constraint (4) is the capacity constraint, which forces the sum of processing time and setup time for each machine within available capacity. Constraints (5) and (6) state that the starting time t_{ijk} for each job r_{ij} scheduled on machine m_k should not be less than the earlier starting time b_{ij} and not be greater than the latest starting time e_{ij}. Constraints (7) and (8) are the starting time constraints. The time of the following job starts after the proceeding job and related setup is complete. As usual, Q is a 'sufficiently large' positive number, so that constraints (9)–(13) are satisfied for $y_{iji'j'k}=0$ or 1. Constraints (9)–(13) are the precedence constraints and constraints (14)–(16) are the direct constraints. These constraints state their sequence relation. In constraint (16), variable $z_{iji^*j^*k}$ states that there is a job $r_{i^*j^*}$ existing after job r_{ij} when $y_{iji'j'k}=1$ and $z_{iji'j'k}=0$. Constraints (17)–(19) indicate that x_{ijk}, $y_{iji'j'k}$, and $z_{iji'j'k}$ are binary integer variables. The total number of variables is $2N_I^2K$, and the total number of equations is $N_I^3K-(5/2)N_I^2K-(3/2)N_IK+N_I+2K$, where $N_I=\sum_{i=1}^{I}J_i$.

Algorithms for the LCMSP

In this section, we present two heuristic procedures to solve the LCMSP efficiently. Some basic technologies have been used for developing algorithms for machine scheduling. We modify the conventional savings algorithms to minimize the total setup time for the contract jobs and apply the greedy concept for the spot jobs to enhance the total profits. We first review two savings algorithms, the PSA proposed by Golden (1977) and the (GSA) provided by Christofides *et al* (1979). We then present two modified algorithms, which are referred to as the three-phase modified parallel savings algorithm (MPSA_TP) and the three-phase modified generalized savings algorithm (MGSA_TP).

Parallel savings algorithm

Golden (1977) proposed the PSA to solve the TSP approximately. The PSA, initially calculates the savings of all pairs of jobs and sorts those savings in descending order. The PSA creates the multiple of K machines simultaneously at the initial stage, where K is the number of machines. Note that a selected pair of jobs is feasible if it does not violate the machine capacity constraints. The PSA then searches downward from the savings list, for a job which could be merged into one of current K schedules (at the first endpoint or last endpoint) with the most savings. The algorithm terminates when no more jobs can be inserted. Algorithm details are presented as follows.

Step 1: (Initialization) Calculate the savings $SA_{ii'}$, defined as the following, for all pairs of two jobs associated with product type R_i and $R_{i'}$, where U denotes the idle status.

$$SA_{ii'} = s_{Ui} + s_{i'U} - s_{ii'} \tag{20}$$

Step 2: Sort the savings list in descending order of magnitude.

Step 3: (Schedule initial construction) Choose the first K pairs of jobs on the list satisfying the machine capacity constraints, to start K new schedules simultaneously.

Step 4: Starting from the top of the savings list. Find the first feasible pair on the list to add to one of the two ends of a currently constructed schedule with the most savings.

Step 5: The chosen jobs form a feasible machine schedule. Repeat Step 4 until all schedules are full and cannot be expanded.

Generalized savings algorithm

In contrast to the PSA, which constructs a multiple of K machine schedules simultaneously at the initial stage, the GSA proposed by Christofides *et al* (1979) creates one schedule at a time and considers not only the end points but also positions between two consecutive jobs when inserting a new job into the current partial schedule. Besides, the insertion costs are calculated for every unscheduled job at every possible position. The chosen job, which maximizes savings while minimizing insertion costs, is used to avoid the algorithm to create a new schedule on another machine with a high setup time.

New algorithms

To effectively apply these technologies, we modify them to develop fast and effective algorithms. The two new algorithms essentially consist of three phases. Phase I applies network algorithms to schedule sub-problem of contract jobs with minimum total setup time. Phase II inserts spot jobs near the same constructed product type so as no extra setup time is needed. Phase III sorts the remaining spot jobs with profit ratio (ie job profit is divided by job processing time) and chooses a subset of high profit ratio jobs which are then inserted into the constructed schedules sequentially until all machine capacities are full. It is noted, however, that the contract jobs in the current constructed schedule should be pushed backward when a spot job is inserted into this schedule in Phase II or Phase III. Thus, the contract jobs, following the inserted spot job, should be re-checked according to their due dates to meet customer deadlines. The new solution procedures can be described as follows.

Three-phase modified parallel savings algorithm (MPSA_TP)

The MPSA_TP calculates savings by adding two terms, profit ratio and time window restrictions. The profit ratio term is added in the savings calculation in order to choose the pairs of jobs with higher profit ratios as the first job pairs for higher savings values than the pairs of jobs with lower ones. By doing this, the jobs with higher profit ratios are forced to be processed earlier than other jobs with lower ones. Furthermore, the jobs with the same profit ratios are forced to be processed closer to each other than the other jobs. The other added term, time window restrictions, takes job r_{ij} whose latest starting time (e_{ij}) is earlier than a later job ($e_{i'j'}$) and tends to place it before another job ($r_{i'j'}$). Time window restrictions have been used by Pearn *et al* (2004a); however, they only considered the value $0 \leqslant \gamma_1 \leqslant 1$ and did not systematically examine the parameter with the value $\gamma_1 \geqslant 1$. In this paper, three parameters, α_1, β_1, and γ_1, and the three ranges $0 \leqslant \alpha_1 \leqslant 1$, $0 \leqslant \beta_1 \leqslant 1$, and $0 \leqslant \gamma_1 \leqslant 2$, are added to the savings function to weight the 'savings term', 'profit ratio term', and 'time window restrictions term', respectively. Parameter α_1 represents weight of savings, which results from the setup time between two jobs r_{ij} and $r_{i'j'}$ in a job pair. It can help the savings term to avoid a long setup time being incurred. Parameter β_1 is used to weight the summation of the profit ratios calculated involving two jobs r_{ij} and $r_{i'j'}$ in a pair in order to achieve higher profits. Finally, parameter γ_1 can assist the time windows restrictions term to prevent the jobs being processed after their due date in order to enhance customer satisfaction. Term W represents the machine predetermined capacity.

Phase I (Modified savings algorithm)

Step 1: Calculate the savings, $PSA_{r_{ij}r_{i'j'}}$, for all pair of jobs r_{ij} and $r_{i'j'}$, where U denotes the idle status. $s_{ii'}$, is the required setup time for switching product type R_i to type $R_{i'}$. Terms $profit_{ij}$, p_{ij}, and e_{ij} represent job profit, job processing time, and latest starting time, respectively, to process job r_{ij}.

$$PSA_{r_{ij}r_{i'j'}} = \begin{cases} 0 & \text{if } PSA_{r_{ij}r_{i'j'}} < 0 \text{ or } i = i',\ j = j' \\ \alpha_1(s_{Ui} + s_{i'U} - s_{ii'}) + \beta_1\left(\dfrac{profit_{ij}}{p_{ij}} + \dfrac{profit_{i'j'}}{p_{i'j'}}\right) + 10W\left(\dfrac{\gamma_1}{e_{ij}} - \dfrac{(2-\gamma_1)}{e_{i'j'}}\right) & \text{otherwise} \end{cases}$$

Step 2: Sort the savings and create a list in descending order of magnitude.
Step 3: Choose the first feasible K pairs of jobs on the list satisfying the machine capacity and due date constraints and remove these pairs from the savings list. Then, start the K new schedules simultaneously.
Step 4: Starting from the top of the remaining savings list, find the first feasible pair on the list and check which job of the pair is able to be added to one of the two ends of a currently constructed schedule.
Step 5: The chosen jobs form a feasible machine schedule. Repeat Step 4 of Phase I until all the contract jobs are scheduled.

Phase II (spot jobs assigned without extra setup)

Step 1: Calculate the profit ratio for the spot jobs.
Step 2: Sort the profit ratio and create a profit ratio list in descending order of magnitude.
Step 3: Choose the job with highest profit ratio as the job to be inserted. Find the machine, which has scheduled the jobs with the same product type as the job to be inserted. Whether the job is inserted or not, it should be removed from the profit ratio list.
Step 4: If the machine for insertion is found, then insert the chosen job by the side of the job with same product type without violating the machine capacity and due date constraints.
Step 5: Repeat Steps 3 and 4 of Phase II until no more jobs can be found in the profit ratio list.

Phase III (Remaining spot jobs assigned with extra setup)

Step 1: Calculate the profit ratio for the remaining spot jobs.
Step 2: Sort the profit ratio in descending order of magnitude.
Step 3: Choose the top job on the list as the job to be inserted.
Step 4: Schedule the job by applying the cheapest insertion algorithm (Rosenkrantz *et al*, 1977) sequentially to construct a feasible schedule without violating machine capacity and due date restrictions. Whether the job is inserted or not, it should be removed from the profit ratio list.
Step 5: The algorithm will terminate with no job in the profit ratio list. Otherwise, return to Steps 3 and 4 of Phase III.

To illustrate how the MPSA_TP algorithm may be applied, we consider the LCMSP example with two machines and three product types described in the previous section entitled 'LCM process and problem description'. In Phase I, the savings value for each pair between two contract jobs is calculated by Step 1 and shown in Table 3 while the values of α_1, β_1, and γ_1, are set to 0.5, 0.05, and 1.5, respectively. The savings are sorted in descending order of magnitude using Step 2, and the first two feasible pairs (12th pair and 4th pair) are chosen and scheduled in the two machines initially in Step 3. After processing using the algorithm and by repeating Steps 4 and 5 until no more contract jobs can be found, two schedules are constructed: job r_{C2} precedes job r_{C1} directly on Machine 1 and job r_{B1} precedes job r_{A1} directly on Machine 2.

In Phase II, the three spot jobs (r_{A2}, r_{A3}, and r_{B2}) are inserted into the two constructed schedules without violating machine capacity and due date constraints. In Step 1 of this phase, the profit ratios of the three jobs are computed: 1.9, 1.9, and 2 for job r_{A2}, job r_{A3}, and job r_{B2}, respectively. The profit ratios are sorted using Step 2. It was found that r_{B2} is the highest ratio and is, therefore, chosen and removed from the profit ratio list in Step 3. Machine 2 has the same product type as job r_{B2}; therefore, job r_{B2} can be inserted into Machine 2 in Step 4. However, the insertion of job r_{B2} into the two possible positions (preceding or following job r_{B1}

Table 3 The value of each pair of contract jobs

Pair ID	*Job pair*	*Values of savings*	*Pair ID*	*Job pair*	*Values of savings*
1	(r_{A1}, r_{B1})	14.75	7	(r_{C1}, r_{A1})	19.14
2	(r_{A1}, r_{C1})	18.29	8	(r_{C1}, r_{B1})	10.89
3	(r_{A1}, r_{C2})	13.57	9	(r_{C1}, r_{C2})	16.21
4	(r_{B1}, r_{A1})	22.76	10	(r_{C2}, r_{A1})	33.28
5	(r_{B1}, r_{C1})	24.54	11	(r_{C2}, r_{B1})	25.02
6	(r_{B1}, r_{C2})	19.83	12	(r_{C2}, r_{C1})	35.06

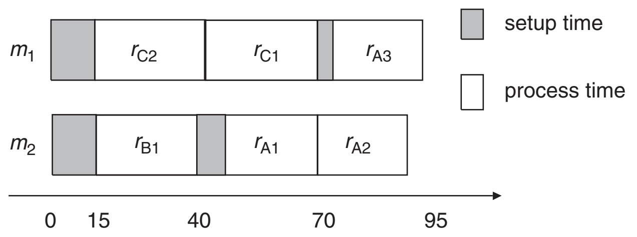

Figure 5 Gantt chart for the example problem.

immediately) will cause the job r_{A1} to violate its due date. Hence, job r_{B2} is not inserted into this machine. Then, Step 3 of the algorithm is repeated and the other spot job r_{A2} is considered. In Step 4, Machine 2 is the inserted machine. Job r_{A2} is scheduled following job r_{A1} without any extra setup time and without violating any restrictions. Furthermore, by repeating Step 3, the third spot job r_{A3} is chosen but is not suitable for insertion due to its violation of due date and machine capacity.

Finally, in Phase III, the remaining spot jobs (r_{B2} and r_{A3}) are considered and scheduled although extra setup times are required. The profit ratios of the two spot jobs are computed and sorted in descending order in Step 1 and Step 2 of Phase III, respectively. In Step 3, job r_{B2} is chosen and removed from the list, but it cannot be inserted into the two machines. However, after processing is repeated in Step 3, the other job, r_{A3}, is inserted into Machine 1 using the cheapest insertion algorithm in Step 4. Owing to the absence of more jobs on the profit ratio list and fullness of the two machines, the final solution from the PSA_TP algorithm is therefore 316 and is depicted in Figure 5.

Three-phase modified generalized savings algorithm (MGSA_TP)

The MGSA_TP constructs the schedules sequentially in contrast to the MPSA_TP which creates a multiple of K machine schedules simultaneously. In addition, the difference in Phase I with MPSA_TP is that MPSA_TP considers only two end points when merging a new job into the current partial schedule, while MGSA_TP considers not only the end points but also the positions between two consecutive jobs when merging a new job into the current partial schedule. The savings function is expressed as follows:

$$GSA_{r_{ij}r_{i'j'}} = \begin{cases} 0 & \text{if } GSA_{r_{ij}r_{i'j'}} < 0 \text{ or } i = i',\ j = j' \\ \alpha_2(s_{Ui} + s_{i'U} = s_{ii'}) + \beta_2\left(\dfrac{profit_{ij}}{p_{ij}} + \dfrac{profit_{i'j'}}{p_{i'j'}}\right) + 10W\left(\dfrac{\gamma_2}{e_{ij}} - \dfrac{(2-\gamma_2)}{e_{i'j'}}\right) & \text{otherwise} \end{cases}$$

Based on the new savings function, the initial partial schedule can be selected from the top of the savings list. In addition, schedules can be expanded based on parameters δ_1, δ_2, and formulas (21)–(24) (designed by Christofides *et al*, 1979). Let PS be the current schedule, $PS = (u_U, \ldots, u_{g-1}, u_g, \ldots, u_G, u_{U'})$, where u_U and $u_{U'}$ are virtual jobs (machine idle status). The insertion costs are computed using formulas (21)–(22) for every unscheduled job r_{ij} at every possible position of PS. Let $\lambda(u_{g-1}, r_{ij}, u_g)$ be the additional setup cost when job r_{ij} is inserted between position $(g-1)$ and g in schedule PS. Let λ^* $(u_g - 1, r_{ij}, u_g)$ be the minimal insertion cost value.

$$\lambda(u_{g-1}, r_{ij}, u_g) = s_{u_{g-1}i} + s_{iu_g} - \delta_1 \times s_{u_{g-1}u_g}, \quad 1 \leqslant \delta \leqslant 2 \tag{21}$$

$$\lambda^*(u_{g-1}, r_{ij}, u_g) = \min_{g=1,\ldots,G}[\lambda(u_{g-1}, r_{ij}, u_g)] \tag{22}$$

Job r_{ij} is chosen, which maximizes the savings $\sigma^*(u_{g-1}, r_{ij}, u_g)$ while minimizing the insertion cost $\lambda^*(u_{g-1}, r_{ij}, u_g)$, and which avoids the algorithm to create a new schedule on another machine with a high setup time $\delta_1 \times s_{Ui}$. Furthermore, in addition to taking into account machine capacity, the due date constraints of all jobs must also be examined for violation before a job is inserted. The procedure is repeated until all schedules are full and cannot be expanded. Phase II and Phase III are same as the corresponding phases of the MPSA_TP algorithm.

$$\sigma(u_{g-1}, r_{ij}, u_g) = \delta_2 \times s_{Ui} - \lambda^*(u_{g-1}, r_{ij}, u_g), \quad 1 \leqslant \delta_2 \leqslant 2 \tag{23}$$

$$\sigma^*(u_{g-1}, r_{ij}, u_g) = \max[\sigma(u_{g-1}, r_{ij}, u_g)] \tag{24}$$

The same illustrative example described in the previous section entitled 'LCM process and problem description' with two machines and three product types is also used to illustrate Phase I of MGSA_TP. First, the savings are computed when the values of α_1, β_1, and γ_1, are set to 0.5, 0.05, and 1.5, respectively. The savings values for MGSA_TP are the same as those obtained by MPSA_TP, as shown in Table 3. In contrast to the MPSA_TP algorithm, one job pair (r_{C2}, r_{C1}) with the largest savings is chosen and scheduled on Machine 1 initially. Second, the other two contract jobs (r_{A1} and r_{B1}) should be considered with their insertion and savings cost using formulas (21)–(24) in order to decide which one should be inserted into the current partial schedule. For the insertion costs, there are three possible positions for each candidate contract job on the current partial schedule, $PS = (u_U, r_{C2}, r_{C1}, u_{U'})$. Therefore, six insertion costs are computed while $\delta_1 = 2$ and $\delta_2 = 1$ and are presented in Table 4. In Table 4, minimal insertion costs obtained for r_{A1} and r_{B1} are -8 and -10, respectively. The following savings costs are then obtained, $\sigma(u_{g-1}, r_{A1}, u_g) = 1 \times 15 - (-8) = 23$ and $\sigma(u_{g-1}, r_{B1}, u_g) = 1 \times 15 - (-10) = 25$ to r_{A1} and r_{B1}, respectively. Therefore, $\sigma^*(u_{g-1}, r_{ij}, u_g) = \max[23, 24] = 25$, then the job r_{B1} is chosen.

Table 4 The insertion cost of each job at every possible position

Job ID	*Possible insertion positions*	*Insertion cost*	*Values*
r_{A1}	$\lambda(u_U, r_{A1}, r_{C2})$	$S_{UA} + S_{AC} - 2 \times S_{UC}$	−8
r_{A1}	$\lambda(u_{C2}, r_{A1}, r_{C1})$	$S_{CA} + S_{AC} - 2 \times S_{CC}$	10
r_{A1}	$\lambda(u_{C1}, r_{A1}, r_{U'})$	$S_{CA} + S_{AU'} - 2 \times S_{CU'}$	3
r_{B1}	$\lambda(u_U, r_{B1}, r_{C2})$	$S_{UB} + S_{BC} - 2 \times S_{UC}$	−10
r_{B1}	$\lambda(u_{C2}, r_{B1}, r_{C1})$	$S_{CB} + S_{BC} - 2 \times S_{CC}$	21
r_{B1}	$\lambda(u_{C1}, r_{B1}, r_{U'})$	$S_{CB} + S_{BU'} - 2 \times S_{CU'}$	16

Third, job r_{B1} is the candidate job and will be inserted on Machine 1 preceding r_{C2}. However, this insertion would cause r_{C2} and r_{C1} to be out of their due dates. Therefore, job r_{B1} is not inserted in the machine. As in this example there is only one other possible contract job (r_{A1}), the process needs to be repeated to determine if it is suitable for insertion. However, it was found that job r_{A1} also cannot be inserted in Machine 1 because of due date and machine capacity constraints. Therefore, the algorithm steps are repeated on Machine 2, jobs r_{B1} and r_{A1} are scheduled on Machine 2 as they were found not to violate any restrictions. Phase I of MGSA_TP is terminated when all contract jobs are scheduled.

Test problems design

For the purpose of testing and comparing the performance of the proposed two new algorithms on various LCMSP with different data characteristics, we generate a set of 24 problems based on a case taken from an LCM factory located on the science-based industrial park in Taiwan. For the case investigated, jobs of 26 product types contain contract and spot jobs. The jobs are scheduled to five identical parallel machines. The contract jobs must be completed on the parallel machines before the given due dates. The machine capacity is set to 4320 min, which is set to equal to the planning period (three days). 'Minute' here is used as the time unit for the job process time, setup time, due date, and machine capacity.

The structure and data of the generated test problems are generated covering most real-world applications. These characteristics include: (1) workload level of contract jobs, (2) tightness of due dates, (3) setup time variation, and (4) variation of (contract/spot) profit ratio. These problem sizes range from low workload level of contract jobs, loose due date tightness, small setup time variation, low variation of (contract/spot) profit ratio, to high workload level of contract jobs, tight due date tightness, high setup time variation, and high variation of (contract/spot) profit ratio.

Workload level of contract jobs

In the real production environment, different workload levels of contract jobs result from different market demand or sale seasons, such as hot seasons in electronic industries. Therefore, we need to evaluate the impact of different workload levels of contract jobs on the performance of the solution algorithms. Owing to varied workload levels of contract jobs, the number of contract jobs is different. Let ES be the estimated setup time required in the problem. $AVG_i[S_{ii'}]$ is the average setup time from product type R_i to other types. And finally, $AVG[S_{iU}]$ is the average setup time to switch to idle status, $AVG[S_{Ui}]$ be the average setup time from idle status to process. The estimated setup time can be expressed as follows.

$$ES = K(AVG[S_{Ui}] + AVG[S_{iU}]) + \frac{(I-1)}{I}\sum_{all\ i} AVG_i[S_{ii'}] \qquad (25)$$

The workload calculation formula in our investigation can be expressed in Equation (26). In the 24 testing problems, each problem contains 120 jobs carrying specific contract jobs and spot jobs. Taking problem 4, 5, and 6 (see Table 5) for example, when the number of contract jobs is 25, 50, and 75, the workload level of contract jobs will be 42, 64, and 86%, respectively. Problem 6 is used to illustrate how the calculation of workload level be applied, a setup time matrix is presented in Table A1 and detailed job information is shown in Table A2 (see the Appendix). The average setup times required for switching from a job with idle status to process ($AVG[S_{Ui}]$) is 120 min, while the reverse ($AVG[S_{iU}]$) only requires 0 min. The average setup time requires for switching from all contract jobs of product type R_i to type $R_{i'}$ ($AVG[S_{ii'}]$) is equal to 3496.9 min. Therefore, the estimated setup time is 3962.4 min. Furthermore, the total processing time of the contract jobs in Problem 6 is 14 451 min. The workload level of Problem 6 is then obtained using Equation (26) when the number of machines (K) is 5 and each machine capacity (W) is 4320 min.

$$\text{Workload level} = \frac{ES + \sum_i^I \sum_j^J p_{ij}}{K \times W} \times 100\% \quad \text{for } i = 1, 2, \ldots, I \text{ and } j = 1, 2, \ldots, J_i^{\text{contract}} \qquad (26)$$

Tightness of due dates

To analyse the impact of the tightness of due dates on the performance of scheduling algorithms, we include two levels of the tightness of due dates. Here, we apply the tightness index formula proposed by Pearn *et al* (2004a) and Equation (25) to estimate the setup time. In the tight situation, the number of jobs during the due dates of day 1 and day 2 is greater than day 3. In the loose situation, the number of jobs during the due date of day 1 would be less than the number of jobs during the due dates of day 2 and day 3.

Setup time variation

In LCMSP, we reduce setup time by scheduling contract and spot jobs without violating the contracted due dates. Thus, the setup time is one of the critical factors for increasing the impact of results. However, the setup time could be varied because of different considerations of setup operations. For

Table 5 Summarized information of 24 problems

Problem number	*Tightness of due date*		*Workload level of contract jobs*			*Setup time variation*		*Variation (contract/spot) profit ratio*	
Parameter	*Tight*	*Loose*	*Low*	*Middle*	*High*	*Small*	*Large*	*Small*	*Large*
1	*		*				*		*
2	*			*	*		*		*
3	*								*
4	*		*					*	
5	*				*			*	
6	*							*	
7		*	*						*
8		*			*				*
9		*							*
10		*	*					*	
11		*			*			*	
12		*						*	
13	*		*			*			*
14	*				*	*			*
15	*					*			*
16	*		*			*		*	
17	*				*	*		*	
18	*					*		*	
19		*	*			*			*
20		*			*	*			*
21		*				*			*
22		*	*			*		*	
23		*			*	*		*	
24		*				*		*	

instance, the cool down and the rapid rising of temperature and voltage are good examples of differing conditions. In our test, we include two levels of setup time variation. The high setup time variation is 10519.4 and the low setup time variation is 3990.3.

Variation of (contract/spot) profit ratio

The contract/spot profit ratio is the division of the contract job unit profit by the corresponding spot job unit profit for each of the product types. The variation of the contract/spot profit ratio is to analyse the variance among different product types. In the real-world application, the profit ratio among different product types should be varied owing to the market competition. The high variation of contract/spot profit ratio is set to 0.01 and the low variation of contract/spot profit ratio is set to 0.001.

The problem sets of the four considered factors have 24 different problems. The problem lists with the four different factors are listed as Table 5. Take problem 6 for example, the setup time matrix is presented in Table A1 and the detailed job information is shown in Table A2 (see the Appendix).

Computational results

To solve the LCMSP case, two heuristic algorithms are coded in Virtual Basic 6.0 programming language, and run on a Pentium IV 3.2 GHz PC. We first experiment with the two new algorithms on 10 small size of LCMSP, where the optimal solutions are available. The size of the problems range from 10 to 15 total jobs, 6 to 10 contract jobs, and 950 to 1650 min machine capacity with various workload levels of contract jobs. For these problems, we write a C + + programming code to generate the constraints and variables of the models. In addition, we solve them using the IP software CPLEX 7.5 to obtain optimal solutions. The computational results are displayed in Table 6.

Table 6 indicates that the heuristic solutions receive eight optimal solutions (out of 10) in term of total profits. The average gap between the optimality and two heuristic algorithms, MGSA_TP and MPSA_TP, is 1.4 and 0.5%, respectively. The average run times (in CPU seconds) for the two heuristic algorithms are indeed significantly faster than the run time of optimality.

For large size of the LCMSP, solving the optimal solutions using integer programming model is computationally inefficient. Therefore, in the following, we test the performance of the two new algorithms on the 24 problems described in the previous section entitled 'Test problems design'. We obtained 72 computational results, which include 48 results for MPSA_TP with two types of parameter combination (MPSA_TP denotes MPSA_TP with $\lambda_1 = 0.5$, $\beta_1 = 0.05$, and $\gamma_1 = 1.5$; MPSA_TP2 denotes MPSA_TP with $\lambda_1 = 0.5$, $\beta_1 = 0$, and $\gamma_1 = 1.5$). MPSA_TP2 represents that the profit ratio adding item of the savings calculation is not considered.

Table 6 A comparison between the optimal solutions and two heuristic algorithms

Prob. no.	*J*	J^{contract}	*W*	*Optimal value*	*CPU (s)*	*MGSA_TP*		*MPSA_TP*	
						Profit	*CPU*	*Profit*	*CPU*
1	10	6	1000	444 000	501.64	444 000	0.031	444 000	0.016
2	11	6	950	444 000	405.09	444 000	0.031	444 000	0.016
3	12	5	1100	514 000	451.89	514 000	0.031	514 000	0.016
4	12	8	1075	557 000	129.75	557 000	0.031	557 000	0.016
5	13	6	1000	488 000	574.39	452 000	0.047	471 000	0.031
6	13	6	1100	537 000	224.61	537 000	0.031	537 000	0.016
7	14	8	1300	660 000	5920.83	660 000	0.047	660 000	0.031
8	14	7	1200	597 000	1219.03	555 000	0.047	584 000	0.031
9	15	9	1440	670 000	13430.4	670 000	0.048	670 000	0.032
10	15	10	1650	760 000	61640.1	760 000	0.063	760 000	0.031

Underlines indicate the optimal solutions.

Table 7 Performance comparison in the three algorithms

	Profit						
Parameter / *Prob. no.*	*MGSA_TP* $\alpha_2 = 0.3$ $\beta_2 = 0.2$ $\gamma_2 = 0.4$ $\delta_1 = 2$ $\delta_2 = 1$	*MPSA_TP* $\alpha_1 = 0.5$ $\beta_1 = 0.05$ $\gamma_1 = 1.5$	*MPSA_TP2* $\alpha_1 = 0.5$ $\beta_1 = 0$ $\gamma_1 = 1.5$	*Prob. no.*	*MGSA_TP* subtracted from MPSA_TP	*MPSA_TP2* subtracted from MPSA_TP	*MPSA_TP2* subtracted from MGSA_TP
1	**5 056 000**	**5 150 500**	4 930 500	1	94 500*	220 000*	125 500*
2	5 385 000	**5 470 000**	5 351 000	2	85 000*	119 000*	34 000*
3	5 796 500	6 009 500	6 038 500	3	213 000*	−29 000	−242 000
4	5 776 000	**5 910 500**	5 614 500	4	134 500*	296 000*	161 500*
5	5 791 000	**5 914 000**	5 776 000	5	123 000*	138 000*	15 000*
6	5 986 500	6 259 500	6 298 500	6	273 000*	−39 000	−312 000
7	5 099 000	**5 193 000**	5 013 000	7	94 000*	180 000*	86 000*
8	5 440 000	**5 584 000**	5 527 000	8	144 000*	57 000*	−87 000
9	**6 020 000**	5 997 000	5 743 000	9	−23 000	254 000*	277 000*
10	5 826 000	**5 905 000**	5 815 000	10	79 000*	90 000*	11 000*
11	5 915 000	**6 209 000**	5 999 000	11	294 000*	210 000*	−84 000
12	**6 236 000**	6 227 000	5 933 000	12	−9000	294 000*	303 000*
13	**5 102 000**	5 073 500	4 973 000	13	−28 500	100 500*	129 000*
14	5 499 000	**5 563 000**	5 428 000	14	64000*	135 000*	71 000*
15	**6 037 000**	6 028 500	5 927 500	15	−8500	101 000*	109 500*
16	5 786 000	**5 855 500**	5 717 000	16	69 500*	138 500*	69 000*
17	5 959 000	**6 147 000**	5 866 000	17	188 000*	281 000*	93 000*
18	**6 277 000**	6 273 500	6 147 500	18	−3500	126 000*	129 500*
19	5 016 000	**5 192 000**	5 063 000	19	176 000*	129 000*	−47 000
20	5 577 000	**5 640 000**	5 537 000	20	63 000*	103 000*	40 000*
21	6 006 000	**6 027 000**	6 013 500	21	21 000*	13 500*	−7500
22	5 726 000	**5 952 000**	5 835 000	22	226 000*	117 000*	−109 000
23	6 014 000	**6 052 000**	5 999 000	23	38 000*	53 000*	15 000*
24	6 224 000	**6 267 000**	6 253 500	24	43 000*	13 500*	−29 500

Result in bold is the best solution among the three algorithms.
Result with * is better than the former.

Table 7 displays the detailed comparison within the two types of algorithms. It denotes the number of better solutions comparing to the two proposed heuristic procedures and the three pairs of subtraction among three different results. Each cell represents the value by subtracting the former from the latter and the cell with the symbol * indicates that the former is better. In comparing the three algorithms, the test results showed that:

(1) The MPSA_TP received 19 better solutions (out of 24) than the MGSA_TP. The average improvement between the 19 improved problems of MPSA_TP comparing with MGSA_TP in terms of profit is 2.2%.

Table 8 Results in means with different problem characteristic groups

Algorithm		*MGSA_TP*	*MPSA_TP*	*MPSA_TP2*
Parameter		$\alpha_2 = 0.3, \beta_2 = 0.02, \gamma_2 = 0.4$ $\delta_1 = 2, \delta_2 = 1$	$\alpha_1 = 0.5, \beta_1 = 0.05,$ $\gamma_1 = 1.5$	$\alpha_1 = 0.5, \beta_1 = 0,$ $\gamma_1 = 1.5$
	n	*Mean*	*Mean*	*Mean*
Total	24	5 731 250	5 829 167*	5 699 958
Tightness DD = Tight	12	5 704 250	5 804 583*	5 672 333
Tightness DD = Loose	12	5 758 250	5 853 750*	5 727 583
Contracted workload = Low	8	5 423 375	5 529 000*	5 370 125
Contracted workload = Middle	8	5 697 500	5 822 375*	5 685 375
Contracted workload = High	8	6 072 875	6 136 125*	6 044 375
Setup timevariation = Small	12	5 768 583	5 839 250*	5 730 000
Setup time variation = Large	12	5 693 917	5 819 083*	5 669 917
Variation of profit ratio = Small	12	5 502 792	5 577 333*	5 462 083
Variation of profit ratio = Large	12	5 959 708	6 081 000*	5 937 833

Result with * is best solution among the three algorithms.

Table 9 Performance comparisons of the three algorithms (24 problems)

	MGSA_TP	*MPSA_TP*	*MPSA_TP2*
Average rank among the three algorithms	2.125	1.25	2.625
Number of problems receiving the best solutions	5	17	2
Average run times CPU seconds	5.11	11.9	8.08

(2) In the MPSA_TP algorithm group, the MPSA_TP received 22 better solutions (out of 24) than MPSA_TP2. The profit ratio term, in the savings calculation of MPSA_TP, indeed improves the solutions.

By computing the mean of the solutions generated by each algorithm in total profit for the 24 problems, we could compare the performances among these algorithms. To further analyse the performance of those algorithms on problems with different characteristics, we grouped the results with the four problem factors and factor levels, which is shown in Table 8. Since the factors such as tightness of due date, setup time variation, and variation of profit ratio, contain two levels of values, these groups each include 12 results. Because the factor of workload level of contract jobs contains three levels of values, these groups include eight results each.

In Table 8, MPSA_TP outperforms MGSA_TP and MPSA_TP2 on all nine groups. Therefore, we can say the performance of the MPSA_TP is better than the MGSA_TP and MPSA_TP2 stably in varied situations.

Finally, we compared performances generated by the three algorithms, which is presented in Table 9, with respect to (1) average rank among the three algorithms, (2) number of problems receiving the best solutions, and (3) average run times in CPU seconds on a Pentium IV 3.2 GHz PC.

The results, displayed in Table 9, indicate that the run times of the three algorithms are quite short for solving those problems containing five machines and 120 jobs with different problem characteristics. In particular, the MPSA_TP significantly outperformed the other algorithms.

Conclusions

In this paper, we considered the LCMSP with sequence dependence setup time and multiple product profits to sequence jobs on identical parallel machines, which has many real-world applications, particular, in the TFT-LCD manufacturing industry. For the LCMSP case investigated, the jobs are clustered by 26 product types, which are processed on five identical parallel machines and must be completed before the due dates. The LCMSP involves constraints on multiple product profit, sequence dependence setup times, due date, product-type-dependent processing time, and machine capacity, it is more difficult to solve than the classical parallel machines scheduling problem. In this paper, we investigated two network algorithms and developed two modifications to solve the LCMSP efficiently. To test the performances of those algorithms, a set of test problems was designed. The design of test problems involves four critical factors including the workload level of contract jobs, tightness of due date, setup time variation, and variation of profit ratio. The computational test results reveal that the MPSA_TP perform better than the MPSA_TP2 and MGSA_TP stably in varied situations. All proposed algorithms solve the large-scale LCMSP effectively.

References

Christofides N, Mingozzi A, Toth P and Sandi C (1979). *Combinatorial Optimization*. John Wiley & Sons: New York,

Golden B (1977). Evaluate a sequential vehicle routing algorithm. *AIIE Trans* **9**: 204–208.

Jeong B, Kim SW and Lee YJ (2001). An assembly scheduler for TFT LCD manufacturing. *Comput Ind Eng* **41**: 37–58.

Pearn WL, Chung SH and Yang MH (2002a). The wafer probing scheduling problem (WPSP). *J Opl Res Soc* **53**: 864–874.

Pearn WL, Chung SH and Yang MH (2002b). Minimizing the total machine workload for the wafer probing scheduling problem. *IIE Trans.* **34**: 211–220.

Pearn WL, Chung SH, Yang MH and Chen YH (2004a). Algorithms for the wafer probing scheduling problem with sequence-dependent set-up time and due date restrictions. *J Opl Res Soc* **55**: 1194–1207.

Pearn WL, Chung SH, Chen AY and Yang MH (2004b). A case study on the multistage IC final testing scheduling problem with reentry. *Int J Prod Econ* **88**: 257–267.

Rosenkrantz DJ, Stearns RE and Lewis II PM (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM J Comput* **6**: 563–581.

Shin HJ and Leon VJ (2004). Scheduling with product family set-up times: An application in TFT LCD manufacturing. *Int J Prod Res* **42**: 4235–4248.

So KC (1990). Some heuristics for scheduling jobs on parallel machines with setups. *Mngt Sci* **36**: 467–475.

Tovia F, Mason SJ and Ramasami B (2004). A scheduling heuristic for maximizing wirebonder throughput. *IEEE T Electron Pack* **27**: 145–150.

Appendix

See Tables A1 and A2.

Table A1 Setup times matrix for 26 product types in problem 6 (unit: minutes)

To/ From	*U*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
U	0	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120
1	0	0	15	20	20	360	360	180	30	30	180	360	250	250	180	30	30	150	120	50	50	50	130	20	50	30	20
2	0	15	0	20	20	360	320	180	30	30	180	360	30	30	170	170	250	30	50	50	150	100	100	150	15	250	30
3	0	20	20	0	120	30	30	170	170	250	250	250	130	150	50	260	80	30	50	50	150	100	100	150	15	80	30
4	0	280	280	360	0	130	20	50	260	80	80	280	130	150	50	260	80	30	50	50	150	100	100	150	15	80	30
5	0	280	280	360	15	0	20	50	260	80	80	280	130	150	50	260	80	30	50	50	150	100	100	150	15	80	30
6	0	280	280	360	15	20	0	50	260	80	80	280	15	80	150	150	120	80	250	250	250	100	100	270	30	120	80
7	0	20	20	120	50	15	80	0	150	120	120	30	15	80	150	150	120	100	150	150	150	80	80	30	30	120	100
8	0	20	20	15	320	30	120	120	0	250	250	250	50	150	150	260	80	250	170	170	250	250	250	250	30	80	20
9	0	280	280	200	50	50	150	150	260	0	80	280	250	100	250	150	30	250	50	260	80	80	80	280	30	30	20
10	0	150	150	80	80	250	250	250	270	30	0	20	150	80	280	30	250	15	170	130	250	250	250	250	150	30	50
11	0	270	270	100	100	150	150	280	30	30	100	0	50	180	120	120	250	150	170	130	250	250	250	250	270	50	150
12	0	15	15	30	30	180	180	120	120	50	150	150	0	180	120	120	250	280	250	250	100	30	30	280	250	250	280
13	0	20	20	360	360	260	80	80	280	250	250	100	50	0	120	150	150	15	360	360	320	320	120	15	280	150	15
14	0	280	280	15	15	260	80	80	280	150	150	80	20	15	0	150	208	15	320	320	320	320	120	15	280	30	15
15	0	20	20	50	50	130	250	250	250	150	80	80	150	80	80	0	80	250	50	260	80	80	80	280	30	250	250
16	0	20	20	208	208	280	250	30	30	280	280	280	280	100	30	30	0	130	150	150	120	120	120	30	280	50	150
17	0	280	280	50	50	150	150	80	80	120	250	280	150	80	80	80	80	0	20	20	20	50	50	150	280	250	250
18	0	150	150	80	80	270	270	100	100	50	150	280	270	100	100	100	100	15	0	130	250	50	250	250	150	150	150
19	0	30	50	30	30	15	280	120	120	30	30	15	15	280	280	120	120	30	280	0	250	250	100	30	15	180	180
20	0	80	250	30	280	15	15	120	120	30	30	15	15	360	360	120	120	30	280	280	0	250	100	30	150	30	30
21	0	100	150	30	280	15	50	250	250	80	80	250	250	360	360	120	120	250	250	100	30	0	30	30	30	30	30
22	0	150	150	80	150	50	50	150	150	100	100	150	15	20	20	250	250	250	250	100	30	30	0	250	30	170	250
23	0	270	270	100	150	50	50	50	180	120	120	180	15	20	20	250	250	250	250	100	30	30	30	0	260	260	80
24	0	30	50	30	30	360	280	120	120	30	30	15	250	280	280	120	120	30	280	280	250	250	100	30	0	180	180
25	0	80	250	30	280	360	360	120	120	30	30	15	250	280	280	120	120	30	280	280	250	250	100	30	150	0	180
26	0	80	250	30	280	360	360	120	120	30	30	15	250	280	280	120	120	30	280	280	250	250	100	30	150	0	0

Table A2 The job information of the 120 jobs in LCMSP

Job ID	*Product type*	*Processing time (min)*	*Profit*	*Due date*	*Contract/ Spot*	*Job ID*	*Product type*	*Processing time (min)*	*Profit*	*Due date*	*Contract/ Spot*
1	1	213	50 000	1440	C	61	12	213	75 000	1440	C
2	1	213	50 000	1440	C	62	12	213	75 000	1440	C
3	1	213	50 000	1440	C	63	12	213	65 000	4320	S
4	1	213	40 000	4320	S	64	12	213	65 000	4320	S
5	1	213	40 000	4320	S	65	13	183	72 500	2880	C
6	2	192	60 000	2880	C	66	13	183	72 500	2880	C

Table A2 (Continued)

Job ID	*Product type*	*Processing time (min)*	*Profit*	*Due date*	*Contract/ Spot*	*Job ID*	*Product type*	*Processing time (min)*	*Profit*	*Due date*	*Contract/ Spot*
7	2	192	60 000	2880	C	67	13	183	72 500	4320	C
8	3	213	63 000	1440	C	68	13	183	62 500	4320	S
9	3	213	63 000	1440	C	69	13	183	62 500	4320	S
10	3	213	63 000	1440	C	70	14	213	65 000	2880	C
11	3	213	53 000	4320	S	71	15	183	60 000	2880	C
12	3	213	53 000	4320	S	72	15	183	60 000	2880	C
13	4	175	50 000	4320	C	73	15	183	60 000	2880	C
14	5	183	59 000	1440	C	74	15	183	60 000	4320	C
15	5	183	59 000	1440	C	75	15	183	50 000	4320	S
16	5	183	59 000	1440	C	76	15	183	50 000	4320	S
17	5	183	49 000	4320	S	77	15	183	50 000	4320	S
18	5	183	49 000	4320	S	78	16	175	78 500	4320	C
19	5	183	49 000	4320	S	79	16	175	78 500	4320	C
20	5	183	49 000	4320	S	80	16	175	78 500	4320	C
21	6	213	76 000	2880	C	81	16	175	78 500	4320	C
22	6	213	76 000	2880	C	82	16	175	68 500	4320	S
23	6	213	76 000	2880	C	83	17	213	60 000	2880	C
24	6	213	76 000	2880	C	84	17	213	60 000	2880	C
25	6	213	76 000	2880	C	85	18	183	54 000	4320	C
26	6	213	66 000	4320	S	86	18	183	54 000	4320	C
27	6	213	66 000	4320	S	87	18	183	44 000	4320	S
28	6	213	66 000	4320	S	88	18	183	44 000	4320	S
29	6	213	66 000	4320	S	89	19	200	70 000	2880	C
30	6	213	66 000	4320	S	90	19	200	70 000	2880	C
31	7	183	70 000	1440	C	91	20	200	60 000	2880	C
32	7	183	70 000	1440	C	92	20	200	60 000	4320	C
33	7	183	70 000	1440	C	93	20	200	60 000	4320	C
34	7	183	60 000	4320	S	94	20	200	60 000	4320	C
35	7	183	60 000	4320	S	95	20	200	50 000	4320	S
36	7	183	60 000	4320	S	96	21	192	50 000	4320	C
37	7	183	60 000	4320	S	97	21	192	50 000	4320	C
38	8	167	66 000	2880	C	98	21	192	50 000	4320	C
39	8	167	66 000	2880	C	99	21	192	50 000	4320	C
40	8	167	66 000	4320	C	100	21	192	50 000	4320	C
41	8	167	66 000	4320	C	101	21	192	50 000	4320	C
42	8	167	66 000	4320	C	102	22	213	62 500	2880	C
43	9	192	75 000	1440	C	103	22	213	62 500	2880	C
44	9	192	75 000	1440	C	104	23	213	58 000	4320	S
45	9	192	75 000	1440	C	105	23	213	58 000	4320	S
46	9	192	65 000	4320	S	106	23	213	58 000	4320	S
47	9	192	65 000	4320	S	107	24	200	71 000	2880	C
48	9	192	65 000	4320	S	108	24	200	71 000	2880	C
49	9	192	65 000	4320	S	109	24	200	71 000	2880	C
50	10	183	60 000	4320	C	110	24	200	71 000	2880	C
51	10	183	60 000	4320	C	111	25	183	61 000	2880	C
52	10	183	50 000	4320	S	112	25	183	61 000	2880	C
53	10	183	50 000	4320	S	113	25	183	61 000	4320	C
54	11	167	50 000	2880	C	114	25	183	61 000	4320	C
55	11	167	50 000	2880	C	115	25	183	51 000	4320	S
56	11	167	40 000	4320	S	116	25	183	51 000	4320	S
57	11	167	40 000	4320	S	117	26	192	65 500	1440	C
58	11	167	40 000	4320	S	118	26	192	55 500	4320	S
59	12	213	75 000	1440	C	119	26	192	55 500	4320	S
60	12	213	75 000	1440	C	120	26	192	55 500	4320	S

Received July 2006;
accepted January 2008 after two revisions