Johnson's Rule, Composite Jobs and the Relocation Problem

T.C.E. Cheng[a,*] and B.M.T. Lin[b]

[a]Department of Logistics, The Hong Kong Polytechnic University

Kowloon, Hong Kong

[b]Department of Information and Finance Management

Institute of Information Management

National Chiao Tung University, Hsinchu 300, Taiwan

Abstract: Two-machine flowshop scheduling to minimize makespan is one of the most well-known classical scheduling problems. Johnson's rule for solving this problem has been widely cited in the literature. We introduce in this paper the concept of composite job, which is an artificially constructed job with processing times such that it will incur the same amount of idle time on the second machine as that incurred by a chain of jobs in a given processing sequence. This concept due to Kurisu first appeared in 1976 to deal with the two-machine flowshop scheduling problem involving precedence constraints among the jobs. We show that this concept can be applied to reduce the computational time to solve some related scheduling problems. We also establish a link between solving the two-machine flowshop makespan minimization problem using Johnson's rule and the relocation problem introduced by Kaplan. We present an intuitive interpretation of Johnson's rule in the context of the relocation problem.

Keywords: Flowshop, makespan, Johnson's rule, composite job, relocation problem.

*Corresponding author: Prof. T.C.E. Cheng, E-mail: lgtcheng@inet.polyu.edu.hk

1 Introduction

Johnson's seminal work on makespan minimization in a two-machine flowshop (Johnson 1954) is one of the pioneering papers in the scheduling literature. In the past half a century, the flowshop model and its solution algorithms have been included in most text books on operations management, especially in books on scheduling theory (Brucker 2007; Pinedo 2001). Numerous research papers have been published on extensions of the flowshop model, and on exploration of the computational complexity issues and development of solution algorithms for variants of the flowshop model. The focus of this paper is not to develop new models or algorithms, but to discuss the concept of "composite job", which was proposed by Kurisu (1976). While the concept of composite job is very useful and has pedagogical value, it has not been widely applied in flowshop scheduling. The second goal of this paper is to introduce the relocation problem, which was first studied by Kaplan (1986), and present an intuitive interpretation of Johnson's rule for solving the two-machine flowshop makespan minimization problem in the context of the relocation problem.

The two-machine flowshop to minimize makespan can be formally defined as follows. Let $N = \{J_1, J_2, \ldots, J_n\}$ be a set of n jobs to be processed in a two-machine flowshop. Each job consists of two operations, which must be processed on the first and second machine in that order, respectively. The processing times of job J_i on the two machines are denoted by p_i and q_i, respectively. The problem, denoted by the three-field notation $F2||C_{\max}$ (Graham et al. 1979), is to find a schedule that completes all the jobs in the shortest time, i.e., minimizing the makespan. It can be shown that any schedule can be transformed into a schedule in which the processing sequences on the two machines are identical without increasing the makespan. Consequently, solution schedules are permutation sequences, i.e., the two machines have the same job sequence. As a result, we use *sequence*, instead of schedule, throughout this paper for simplicity of presentation.

To solve the $F2||C_{\max}$ problem, Johnson (1954) gave a decision rule, now popularly called Johnson's rule: *For any two jobs $J_i, J_j \in N$, if $\min\{p_i, q_j\} \leq \min\{q_i, p_j\}$, then schedule job J_i earlier than job J_j.* Based on this rule, a solution algorithm, known as Johnson's algorithm, can be designed, which has a time complexity of $O(n \log n)$. Different forms of Johnson's algorithm have been presented in the literature. The following two forms are probably the most commonly adopted:

- Form 1: Select the shortest processing time amongst all the unscheduled operations. If the operation belongs to machine one, then schedule the job of that operation in the earliest open position; else schedule the job of that operation in the last open position. Remove the job from the job set. Repeat the process until all the jobs are scheduled.

- Form 2: Partition the job set N into $N^+ = \{J_i | p_i \leq q_i, J_i \in N\}$ and $N^- = \{J_i | p_i > q_i, J_i \in N\}$. Schedule the jobs in N^+ in non-decreasing order of p_i, and schedule the jobs in N^- in non-increasing order of q_i. Concatenate the two sequences.

Most operations management textbooks adopt Form 1 of the algorithm. It is clear that the time complexity of Johnson's algorithm is dominated by the sorting procedure because we need to find an ordered list of the jobs or operations. Throughout this paper we call

any algorithm *Johnson's algorithm* that is based on Johnson's rule, regardless of the way in which it is presented, and denote by $J_i \preceq_J J_j$ if job J_i precedes J_j by Johnson's rule.

The rest of this paper is organized into four sections. In Section 2 we introduce the concept of composite job, followed by an illustrative example. We also give examples of applications of the concept of composite job. In Section 3 we discuss the relocation problem, which has been shown to be equivalent to the two-machine flowshop scheduling problem to minimize makespan. We present an intuitive interpretation of Johnson's rule in the context of the relocation problem. We conclude the paper and suggest some directions for future research in Section 4.

2 Composite Jobs

The makespan of a given job sequence for $F2||C_{\max}$ is the sum of all machine-two processing times plus the sum of all idle times incurred on the second machine. Therefore, minimizing the makespan is equivalent to finding a job sequence that minimizes the sum of idle times on the second machine. In studying the two-machine flowshop scheduling problem with precedence constraints in chains among the jobs, Kurisu (1976) introduced the concept of a chain of jobs. That is, given a chain of jobs that will be processed consecutively without interruption, Kurisu defined a composite job as a job that will cause the same amount of idle time as the sum of idle times incurred by the jobs following the processing sequence defined by the chain. The reader is referred to Kurisu (1976) for details of the proof. Kamburowski (2000) established the definition of composite job via considering flowshop scheduling with time lags.

For notational simplicity, consider the natural sequence $S = (J_1, J_2, \cdots, J_n)$ of the jobs. We can replace a chain or a subsequence of jobs by a composite job. Construct composite job $J_{[1:2]}$ from jobs J_1 and J_2 by letting

$$p_{[1:2]} = p_1 + \max\{0, p_2 - q_1\}, \text{ and}$$

$$q_{[1:2]} = \max\{0, q_1 - p_2\} + q_2.$$

It is easy to see that schedule S and schedule $S' = (J_{[1:2]}, J_3, \cdots, J_n)$ have the same total idle time. With $J_{[1:2]}$ and J_3, we can similarly construct composite job $J_{[1:3]}$ by letting $p_{[1:3]} = p_{[1:2]} + \max\{0, p_3 - q_{[1:2]}\}$ and $q_{[1:3]} = \max\{0, q_{[1:2]} - p_3\} + q_3$. Following this line of reasoning, for any $1 < k \leq n$, composite job $J_{[1:k]}$ is recursively defined by $p_{[1:k]} = p_{[1:k-1]} + \max\{0, p_k - q_{[1:k-1]}\}$ and $q_{[1:k]} = \max\{0, q_{[1:(k-1)]} - p_k\} + q_k$. Indeed, $p_{[1:k]}$ can be interpreted as the sum of machine-two idle times in the subsequence $(J_1, J_2, \cdots, J_k)$ (see Figure 1). As shown in Figure 1, a sequence of jobs has overlapping machine-one and machine-two operations. The overlapping intervals can be removed without changing the total idle time on machine two. A composite job is defined by trimming the overlapping intervals. In the backward direction, composite job $J_{[k:n]}$ can be similarly defined for any $k, 1 \leq k < n$. The computational time of the above procedure for constructing composite jobs is $O(n)$.

Figure 1 and Figure 2 about here.

Consider the example shown in Figure 2. The sum of idle times of the optimal schedule is 6. The idle time of composite job $J_{[1:5]}$ is 6, too. If we combine two composite jobs "$J_{[1:3]}$ and $J_{[4:5]}$" or "$J_{[1:2]}$ and $J_{[3:5]}$", then we have the same result.

The above procedure constructs composite jobs $J_{[1:k]}$ and $J_{[k:n]}$ for $1 \leq k \leq n$ in $O(n)$ time. We can extend the procedure to construct $J_{[i:j]}$ for $1 \leq i < j \leq n$ in the following. Composite jobs $J_{[1:2]}, J_{[1:3]}, \ldots, J_{[1:n]}$ are successively determined in $O(n)$ time. Similarly, $J_{[2:3]}, J_{[2:4]}, \ldots, J_{[2:n]}, J_{[3:4]}, \ldots, J_{[3:n]}, \ldots, J_{[n-1:n]}$ are successively determined. The total running time is $O(n^2)$.

In the following, we present some examples to demonstrate applications the concept of composite job. Birman and Mosheiov (2004) studied a two-machine flowshop scheduling problem, which is to determine the due date and schedule the jobs so as to minimize the following objective function: $Z(q,d) = \max\{W^E \max_{1\leq j\leq N} E_j, W^T \max_{1\leq j\leq N} T_j, W_d d\}$, where E_j and T_j are respectively the earliness and tardiness of job J_j; W^E, W^T and W_d are respectively the unit penalty of earliness and tardiness, and for setting the due date to be d. They proposed an $O(n^2 \log n)$ algorithm that can optimally solve the problem. The time complexity results from $O(n)$ repetitions of putting a job in the first position and applying Johnson's algorithm for the remaining jobs. For each job J_k and Johnson's sequence for the remaining $n-1$ jobs, we can compute the makespan of the sequence $(J_k, J_1, J_2, \cdots, J_{k-1}, J_{k+1}, \cdots, J_n)$ by considering job J_k and two composite jobs $J_{[1:k-1]}$ and $J_{[k+1:n]}$. This takes $O(1)$ time. Therefore, the overall time complexity is reduced to $O(n \log n)$ time.

The second example concerns the problems $F2|f|C_{\max}$ and $F2|l|C_{\max}$, which were investigated by Saadani et al. (2005). In the $F2|f|C_{\max}$ problem, there is a proper subset $A \subset N$ of the jobs that cannot be scheduled in the *first* position in a job sequence. The counterpart problem $F2|l|C_{\max}$ has a subset $B \subset N$ of the jobs that cannot be scheduled in the *last* position in a job sequence. Both problems have been shown to be solvable in $O(n^2)$ time by first finding Johnson's sequence for the jobs in N, and then testing each job in $N \backslash A$ (respectively, $N \backslash B$) in the first (respectively, last) position. Because the makespan of each sequence can be calculated in $O(n)$ time, the overall time complexity is $O(n^2)$. To search for a simple rule, like Johnson's rule, Saadani et al. (2005) proposed four intuitive rules, and gave counterexamples of these rules. They also developed several interesting but complicated properties for these two problems. Here we do not provide a simple rule, but show that the two problems can be solved in $O(n \log n)$ time. As in the due-date assignment problem in the first example, we select a job in $N \backslash A$ out of sequence S and schedule it first. The total testing time is thus $O(n)$, which is dominated by the $O(n \log n)$ time required for constructing the initial sequence S. An extended problem is $F2|f,l|C_{\max}$, in which the forbidden constraints are simultaneously applied to the first and last positions. This problem is solvable in polynomial time too, but requires $O(n^3)$ time (Saadani et al. 2005). To apply the concept of composite job, we construct $J_{[i:j]}$ for $1 \leq i < j \leq n$ by a two-level nested loop. Assume jobs $J_i \in N \backslash A$ and $J_j \in N \backslash B$, $i \neq j$, are selected to be positioned in the first and last positions. The makespan of the test sequence can be calculated in constant time using two regular jobs and three composite jobs: $J_i, J_{[1:i-1]}, J_{[i+1:j-1]}, J_{[j+1:n]}, J_j$. To construct all the composite jobs $J_{[i:j]}$ requires $O(n^2)$ time and the above testing procedure takes $O(n^2)$ time, too. Therefore, the overall time complexity reduces from $O(n^3)$ to $O(n^2)$.

Applications in Proofs

To prove the correctness of Johnson's rule, we usually assume that in some optimal schedule there exist two *consecutive* jobs J_i and J_j such that $J_i \preceq_J J_j$ and J_j precedes J_i. Subject to this assumption, we then show that the makespan of the schedule S' obtained by swapping J_i and J_j is no larger than that of the original schedule S. In some flowshop scheduling problems, we however need to establish the validity of the job-interchange argument for two jobs that are not necessarily scheduled in consecutive positions. Consider as an example the following problem (Sekiguchi 1983; Cheng et al. 2000). The jobs are categorized into groups. When the processing of a job follows some job from another group, a setup time depending on the group to which the job belongs is required. In this problem setting, Sekiguchi (1983) showed that jobs from the same group need to be sequenced by Johnson's rule. To prove this property, the job-interchange argument may not be applied for two consecutive jobs in the assumed optimal schedule because the two jobs may belong to different groups. To make the jobs of a group follow the sequence determined by Johnson's rule, we may encounter the situation where two jobs from the same group are not scheduled consecutively and they violate Johnson's rule.

In the following we demonstrate the use of the concept of composite job to show the correctness of Johnson's rule by considering two jobs that are not necessarily consecutive. Assume $J_i \preceq_J J_j$ and J_j precedes J_i in some optimal schedule. Let $(J_{j_1}, J_{j_2}, \ldots, J_{j_k})$ be the sequence of jobs scheduled between J_j and J_i. We show that either moving job J_j forward to the position immediately following J_i or moving job J_i backward to the position immediately preceding J_j will not increase the makespan. For convenience in discussion, we partition the jobs in N into $N^+ = \{J_i | p_i \leq q_i, J_i \in N\}$ and $N^- = \{J_i | p_i > q_i, J_i \in N\}$. Jobs in N^- (respectively, N^+) are called *negative* (respectively, *non-negative*). The meanings of the terms "negative" and "non-negative" will be elaborated in the next section on the relocation problem. Moreover, the following property of the composite job will be used.

Property 1 *Given job sequence $(J_j, J_{j+1}, \ldots, J_i)$, there are the following inequalities: $p_j \leq p_{[j:i]}$ and $q_i \leq q_{[j:i]}$.*

To complete the proof of swapping two jobs that are not necessarily consecutive, we consider the following three cases:

Case 1: $J_j, J_i \in N^+$: Consider the composite job $J_{\tilde{j}}$ formed by the sequence $(J_j, J_{j_1}, \ldots, J_{j_k})$. If $J_{\tilde{j}}$ is negative, then we swap $J_{\tilde{j}}$ and J_i and come up with the sequence $(J_i, J_j, J_{j_1}, \ldots, J_{j_k})$, in which J_i precedes J_j. On the other hand, if $J_{\tilde{j}}$ is non-negative, then by Property 1, $p_{\tilde{j}} \geq p_j \geq p_i$. Therefore, we can also swap $J_{\tilde{j}}$ and J_i, and have the same sequence .

Case 2: $J_j, J_i \in N^-$: If the composite job $J_{\tilde{i}}$ defined by sequence $(J_{j_1}, \ldots, J_{j_k}, J_i)$ is non-negative, then we can swap $J_{\tilde{j}}$ and J_j and come up with the sequence $(J_{j_1}, \ldots, J_{j_k}, J_i, J_j)$, in which J_i precedes J_j. If $J_{\tilde{i}}$ is negative, then Property 1 implies $q_j \leq q_i \leq q_{\tilde{i}}$, and we can swap J_j and $J_{\tilde{i}}$.

Case 3: $J_j \in N^-, J_i \in N^+$: If the composite job defined by $(J_{j_1}, \ldots, J_{j_k})$ is negative, then we swap J_i and the composite job, followed by swapping J_i and J_j. The derived sequence is $(J_i, J_j, J_{j_1}, \ldots, J_{j_k})$. If the composite job is non-negative, then we swap J_j and the composite job, followed by swapping J_j and J_i. The derived sequence is $(J_{j_1}, \ldots, J_{j_k}, J_i, J_j)$. Now, job J_i precedes J_j.

The reasoning used above can be slightly modified (or simplified) to construct a proof of Form 2 of Johnson's algorithm.

Proof of Form 2 of Johnson's algorithm:
Let S be an optimal schedule different from the schedule produced by Form 2 of Johnson's algorithm. Let J_i, J_j be the first pair of jobs in schedule S such that the relation $J_i \preceq_J J_j$ holds but J_j precedes J_i. Recall that $(J_{j_1}, J_{j_2}, \ldots, J_{j_k})$ is the sequence of jobs scheduled between J_j and J_i. We show that eliminating the out-of-order jobs between J_i and J_j will not increase the makespan. Moreover, we need to guarantee that no two jobs that are originally in order will become out of order. Similarly, we consider the same three cases:

Case 1: $J_j, J_i \in N^+$: The analysis of the makespan is the same as above. Because $J_j \preceq J_{j'}$ for $j_1 \leq j' \leq j_k$, by transitivity, $J_i \preceq_J J_j \preceq_J J_{j'}$ for $j_1 \leq j' \leq j_k$. More pairs of out-of-order jobs can possibly be eliminated and no extra pair of jobs become out of order.
Case 2: $J_j, J_i \in N^-$: The analysis of the makespan is the same as above. Transforming the original sequence $(J_j, J_{j_1}, \ldots, J_{j_k}, J_i)$ into $(J_{j_1}, \ldots, J_{j_k}, J_i, J_j)$ makes jobs J_i and J_j in order without causing a new pair of out-of-order jobs.
Case 3: $J_j \in N^-, J_i \in N^+$: The reasoning for the new subsequence $(J_i, J_j, J_{j_1}, \ldots, J_{j_k})$ (respectively, subsequence $(J_{j_1}, \ldots, J_{j_k}, J_i, J_j)$) is the same as that used for proving Case 1 (respectively, Case 2).

Continuing the above swapping procedure, if necessary, we will finally come up with a schedule as specified by Johnson's algorithm. The proof is completed. □

3 The Relocation Problem

Alternative proofs of Johnson's rule have been proposed in the literature in order to simplify presentation, e.g., Kamburowski (1997). In this section we give a proof of Johnson's rule in the context of the relocation problem, which is a resource-constrained single-machine scheduling problem formulated and studied in connection with a redevelopment project in Boston (Kaplan 1986; Kaplan and Birman 1988; PHRG 1986). A pool of v_0 units of a single-type of resource is provided for processing the jobs. Job J_i acquires and consumes α_i units of the resource from the pool for its processing and returns β_i units of the resource to the pool upon its completion. A schedule or sequence is said to be feasible if each job following the sequence can be successfully processed. The objective is to determine the minimum initial resource level v_0 that guarantees the existence of a feasible schedule. Kaplan and Amir (1988) showed that the relocation problem is equivalent to $F2||C_{\max}$. The equivalence is based upon the fact that given a job sequence the sum of idle times on the second machine in $F2||C_{\max}$ is equal to the required initial resource level guaranteeing the feasibility of the processing sequence for the relocation problem. For recent research developments on the relocation problem, the reader is referred to Kononov and Lin (2006; 2007).

If we consider the relocation problem from the perspective of investment planning, then we have the setting in which n projects are required to be finished and each project J_i requires

an investment of α_i units of resources and will provide a return of β_i units of resources. To determine a sequence to execute the projects so as to minimize the initial capital, we need to work out the whole plan as follows: we execute the projects having non-negative profits $\beta_i - \alpha_i$ first and arrange them in non-decreasing order of investments α_i. The projects having negative profits follow and are arranged in non-increasing order of returns β_i. This setting provides an intuitive interpretation of Form 2 of Johnson's algorithm given in Section 1.

We apply the concept of composite job to the relocation problem. Consider the subsequence of jobs $(J_i, J_{i+1}, \ldots, J_j), 1 \leq i < j \leq n$. We make the following definition for composite job $J_{[i:j]}$:

$$\alpha_{[i:j]} = \alpha_{[i:j-1]} + \max\{0, \alpha_j - \beta_{[i:j-1]}\};$$
$$\beta_{[i:j]} = \max\{0, \beta_{[i:j-1]} - \alpha_j\} + \beta_j.$$

In the relocation problem, a job (project) J_i in N^- is called negative because its contribution is $\beta_i - \alpha_i < 0$. On the other hand, non-negative jobs have non-negative contributions, i.e., $\beta_i - \alpha_i \geq 0$. Therefore, the processing of a negative job reduces the resource level in the pool while the processing of a positive job produces extra units of the resource.

Assume S is some optimal sequence and its resource requirement is v_0. Let J_i and J_j be two consecutive jobs in S such that $J_i \preceq_J J_j$, but J_j precedes J_i. We consider the following three cases to show that another schedule that is feasible with respect to v_0 can be obtained by swapping J_j and J_i.

Case 1: $J_j, J_i \in N^+$: Because $\alpha_i \leq \alpha_j$ and $\beta_i - \alpha_i \geq 0$, swapping the two jobs preserves the feasibility of the sequence.
Case 2: $J_j \in N^-, J_i \in N^+$: Because J_j is negative and J_i is non-negative with respect to the resource level at the start of their processing, swapping the two jobs preserves the feasibility of the sequence.
Case 3: $J_j, J_i \in N^-$: Let v_t be the resource level before the start of job J_j in sequence S. Because J_j is negative and consumes the resource, the resource level becomes $v_t - \alpha_j + \beta_j$, which is no less than α_i to ensure the feasibility of job J_i. After the swap, v_t is sufficient for J_i. The remaining issue we need to address is to ensure the feasibility of job J_j after the swap. Consider the following chain of derivations:

$$\begin{aligned} & (v_t - \alpha_i + \beta_i) - \alpha_j \geq (v_t - \alpha_i + \beta_j) - \alpha_j \text{ (because } \beta_i \geq \beta_j) \\ = \; & (v_t - \alpha_j + \beta_j) - \alpha_i \geq 0. \end{aligned}$$

Therefore, job J_j can be processed after the swap.

4 Conclusion

In this paper we introduced the concept of composite job and the relocation problem, which were first proposed in 1976 and 1986, respectively, but have not been widely deployed or studied in the scheduling literature. We demonstrated their potential application in reducing the computational time required to solve related scheduling problems or simplifying the presentation of solution algorithms and their proofs. There is a considerable scope in applying

the concept of composite job in scheduling. Studying the relocation problem with temporal constraints, such as processing times, release dates and due dates, is a worthwhile future research direction. From our experience, theoretical challenges arise whenever temporal parameters are involved in the relocation problem.

Acknowledgements: This research was supported in part by The Hong Kong Polytechnic University under a grant from the *Area of Strategic Development in Chinese Business Services*. The authors are grateful to the anonymous reviewers for their constructive comments.

References

[1] M. Birman and G. Mosheiov (2004) A note on a due-date assignment on a two-machine flow-shop, *Computers and Operations Research*. 31: 473-480.

[2] P. Brucker (2007) *Scheduling Algorithms*, Springer, New York.

[3] T.C.E. Cheng (1993) Efficient implementation of Johnson's rule for the $n/2/F/F_{\max}$ scheduling problem. *Computers and Industrial Engineering*, 22: 495-499.

[4] T.C.E. Cheng, J.N.D. Gupta and G. Wang (2000) A review on flowshop scheduling research with setup times. *Production and Operations Management*, 9: 262-282.

[5] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan (1979) Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287-326.

[6] S.M. Johnson (1954) Optimal two and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1: 61-68.

[7] J. Kamburowski (1997) The nature of simplicity of Johnson's Algorithm. *Omega*, 25: 581-584.

[8] J. Kamburowski (2000) Non-bottleneck machines in three-machine flow shops *Journal of Scheduling*. 3: 209-223.

[9] E.H. Kaplan (1986) Relocation models for public housing redevelopment Programs. *Planning and Design*, 13: 5-19.

[10] E.H. Kaplan and A. Amir (1988) A fast feasibility test for relocation problems. *European Journal of Operational Research*, 35: 201-205.

[11] E.H. Kaplan and O. Berman (1988) Orient Heights housing projects. *Interfaces*, 18: 14-22.

[12] A.V. Kononov and B.M.T. Lin (2006) On the relocation problems with multiple identical working crews. *Discrete Optimization*, 21: 368-381.

[13] A.V. Kononov and B.M.T. Lin (2007) Resource-constrained scheduling to minimize weighted completion time on a single machine, in submission.

[14] T. Kurisu (1976) Two-machine scheduling under required precedence among jobs. *Journal of the Operations Research Society of Japan*, 19: 1-13.

[15] PHRG (1986) *New Lives for Old Buildings: Revitalizing Public Housing Project*, Public Housing Group, Department of Urban Studies and Planning, MIT, Cambridge, Massachusetts.

[16] M. Pinedo (2001) *Scheduling: Theory, Algorithms, and Systems*, Prentice-Hall, New Jersey.

[17] H. Saadani, P. Baptiste and M. Moalla (2005) The simple $F2||C_{\max}$ with forbidden tasks in first or last position: A problem more complex than it seems. *European Journal of Operational Research*, 161: 21-31.

[18] Y. Sekiguchi (1983) Optimal schedule in a GT-type flow-shop under series-parallel precedence constraints. *Journal of the Operations Research Society of Japan*, 26: 226-251.

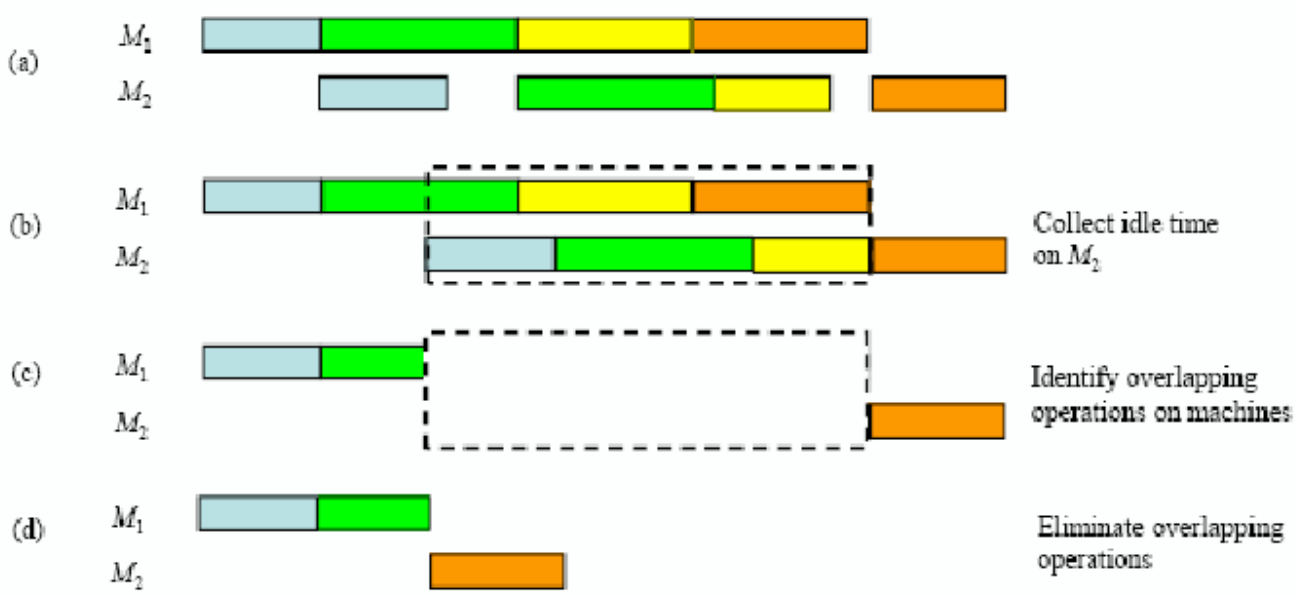

Figure 1: Formation of a composite job.

Jobs	J_1	J_2	J_3	J_4	J_5
p_i	2	5	8	7	3
q_i	3	7	6	4	2

Sum of idle times of sequence $S = J_1, J_2, J_3, J_4, J_5$ is $2 + 2 + 1 + 1 + 0 = 6$.

Composite Jobs	$J_{[1:2]}$	$J_{[1:3]}$	$J_{[1:4]}$	$J_{[1:5]}$
$p_{[1:k]}$	4	5	6	6
$q_{[1:k]}$	7	6	4	3

Composite Jobs	$J_{[4:5]}$	$J_{[3:5]}$	$J_{[2:5]}$	$J_{[1:5]}$
$p_{[k:5]}$	7	9	7	6
$q_{[k:5]}$	3	3	3	3

Figure 2: An illustrative example of composite jobs.